K. J. Somaiya College of Engineering, Mumbai-77

Batch: A-4 **Roll No.:** 16010122151
Experiment No. 08
Group No :- 5
Signature of the Staff In-charge with date

Title: Solving planning problems using STRIPS or PDDL tools.

Objective: To write STRIPS scripts to solve planning problem and implement them using PDDL tools

---

Expected Outcome of Experiment:

Course Outcome	After successful completion of the course students should be able to
CO2	Analyze and solve problems for goal based agent architecture (searching and planning algorithms).

Books/ Journals/ Websites referred:

1. **https://planning.wiki/, last retrieved on Feb 27,2025**
2. **https://editor.planning.domains/, last retrieved on Feb 27,2025**
3. **https://www.youtube.com/watch?v=EeQcCs9SnhU, last retrieved on Feb 27,2025**
4. **https://www.youtube.com/watch?v=FS95UjrICy0, last retrieved on Feb 27,2025**
5. **https://nms.kcl.ac.uk/planning/software/optic.html, last retrieved on Feb 27,2025**
6. **https://github.com/yarox/pddl-examples, last retrieved on Feb 27,2025**
7. **https://planning.wiki/_citedpapers/pddl3bnf.pdf , last retrieved on Feb 27,2025**
8. **https://www.youtube.com/watch?v=XW0z8Oik6G8**
9.
10. **https://github.com/potassco/pddl-instances, last retrieved on Feb 27,2025**
11. **"Artificial Intelligence: a Modern Approach" by Russell and Norving, Pearson education Publications**

12. "Artificial Intelligence" By Rich and knight, Tata McGraw Hill Publications

---

Pre Lab/ Prior Concepts:
Goal based agents, searching, uninformed search, informed search

---

Historical Profile: *(Details about planning Vs Searching)*

New Concepts to be learned:

Representing problem as planning problem, ADL, STRIPS, Total order plan, partial order plan

---

Chosen Planning Problem:

The planning problem involves a home-cleaning robot (robot1) that must clean different rooms in a house. The robot begins in the kitchen with a low battery. The tasks it must complete include:

- Vacuuming the living room (which is dirty),
- Mopping the kitchen (which has a spill),
- Taking out the trash from the bedroom,
- And recharging itself to restore battery.

The robot must move between rooms and perform the necessary cleaning actions, considering preconditions like being in the correct room or having sufficient battery.

STRIPS/ADL Script for solving problem:

Domain.pddl

```
(define (domain home-cleaning)
  (:requirements :strips :typing)

  (:types room robot)

  (:predicates
    (at ?r - robot ?rm - room)          ; Robot is in a specific room
    (dirty ?rm - room)                  ; Room is dirty
    (spilled ?rm - room)                ; Room has a spill
    (trash-present ?rm - room)          ; Room has trash
    (battery-low ?r - robot)            ; Robot battery is low
    (at-trash-bin ?r - robot)           ; Robot is at the trash bin
  )

  (:action move
    :parameters (?r - robot ?from - room ?to - room)
    :precondition (at ?r ?from)
    :effect (and (at ?r ?to) (not (at ?r ?from)))
  )

  (:action vacuum-room
    :parameters (?r - robot ?rm - room)
    :precondition (and (at ?r ?rm) (dirty ?rm))
    :effect (not (dirty ?rm))
  )

  (:action mop-room
    :parameters (?r - robot ?rm - room)
    :precondition (and (at ?r ?rm) (spilled ?rm))
    :effect (not (spilled ?rm))
  )

  (:action take-out-trash
    :parameters (?r - robot ?rm - room)
    :precondition (and (at ?r ?rm) (trash-present ?rm))
    :effect (and (not (trash-present ?rm)) (at-trash-bin ?r))
  )

  (:action recharge
    :parameters (?r - robot)
    :precondition (battery-low ?r)
    :effect (not (battery-low ?r))
```

```
  )
)
```

Problem1.pddl : Morning-Routine

```
(define (problem morning-routine)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bedroom)
  (dirty bedroom)
  (spilled bathroom)
  (trash-present kitchen)
)
(:goal
  (and
    (not (dirty bedroom))
    (not (spilled bathroom))
    (not (trash-present kitchen))
  )
)
)
```

Problem2.pddl : Problem Post-Party-Panic

```
(define (problem post-party-panic)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bathroom)
  (dirty living-room)
  (spilled kitchen)
  (battery-low robot1)
)
(:goal
  (and
    (not (dirty living-room))
```

```
    (not (spilled kitchen))
    (not (battery-low robot1))
  )
)
)
```

Problem3.pddl : Clean-Nearby-First

```
(define (problem clean-nearby-first)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bathroom)
  (dirty living-room)
  (spilled kitchen)
  (trash-present bathroom)
)
(:goal
  (and
    (not (dirty living-room))
    (not (spilled kitchen))
    (not (trash-present bathroom))
  )
)
)
```

Problem4.pddl : One-Room-Mess

```
(define (problem one-room-mess)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bedroom)
  (dirty bedroom)
```

```
  (spilled bedroom)
  (trash-present bedroom)
)
(:goal
  (and
    (not (dirty bedroom))
    (not (spilled bedroom))
    (not (trash-present bedroom))
  )
)
)
```

Problem5.pddl : Clean-Entire-Home

```
(define (problem clean-entire-home)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 living-room)
  (dirty living-room)
  (spilled kitchen)
  (trash-present bedroom)
  (battery-low robot1)
)
(:goal
  (and
    (not (dirty living-room))
    (not (spilled kitchen))
    (not (trash-present bedroom))
    (not (battery-low robot1))
  )
)
)
```

PDDL Script for solving problem:

Domain.pddl

Domain.pddl
Problem1.pddl
Plan (1)

```
(define (domain home-cleaning)
(:requirements :strips :typing)
(:types room robot)
(:predicates
(at ?r - robot ?rm - room) ; Robot is in a specific room
(dirty ?rm - room) ; Room is dirty
(spilled ?rm - room) ; Room has a spill
(trash-present ?rm - room) ; Room has trash
(battery-low ?r - robot) ; Robot battery is low
(at-trash-bin ?r - robot) ; Robot is at the trash bin
)
(:action move
:parameters (?r - robot ?from - room ?to - room)
:precondition (at ?r ?from)
:effect (and (at ?r ?to) (not (at ?r ?from)))
)
(:action vacuum-room
:parameters (?r - robot ?rm - room)
:precondition (and (at ?r ?rm) (dirty ?rm))
:effect (not (dirty ?rm))
)
(:action mop-room
:parameters (?r - robot ?rm - room)
:precondition (and (at ?r ?rm) (spilled ?rm))
:effect (not (spilled ?rm))
)
(:action take-out-trash
:parameters (?r - robot ?rm - room)
:precondition (and (at ?r ?rm) (trash-present ?rm))
:effect (and (not (trash-present ?rm)) (at-trash-bin ?r))
)
(:action recharge
:parameters (?r - robot)
:precondition (battery-low ?r)
:effect (not (battery-low ?r))
)
)
```

Problem1.pddl : Morning-Routine

Domain.pddl
Problem1.pddl
Plan (1)

```
(define (problem morning-routine)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bedroom)
  (dirty bedroom)
  (spilled bathroom)
  (trash-present kitchen)
)
(:goal
  (and
    (not (dirty bedroom))
    (not (spilled bathroom))
    (not (trash-present kitchen))
  )
)
)
```

Domain.pddl
Problem1.pddl
Plan (1)

Found Plan (output)

- (vacuum-room robot1 bedroom)
- (move robot1 bedroom bathroom)
- (mop-room robot1 bathroom)
- (move robot1 bathroom kitchen)
- (take-out-trash robot1 kitchen)

```
(:action vacuum-room
  :parameters (robot1 bedroom)
  :precondition
    (and
      (at robot1 bedroom)
      (dirty bedroom)
    )
  :effect
    (not
      (dirty bedroom)
    )
)
```

Problem2.pddl : Problem Post-Party-Panic

Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)

```
(define (problem post-party-panic)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bathroom)
  (dirty living-room)
  (spilled kitchen)
  (battery-low robot1)
)
(:goal
  (and
    (not (dirty living-room))
    (not (spilled kitchen))
    (not (battery-low robot1))
  )
)
)
```

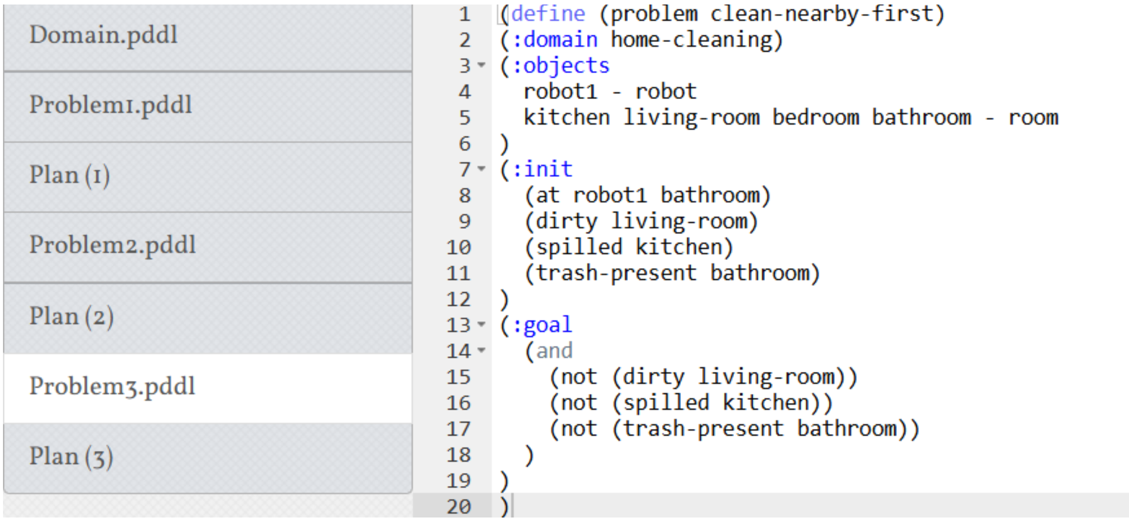
Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)

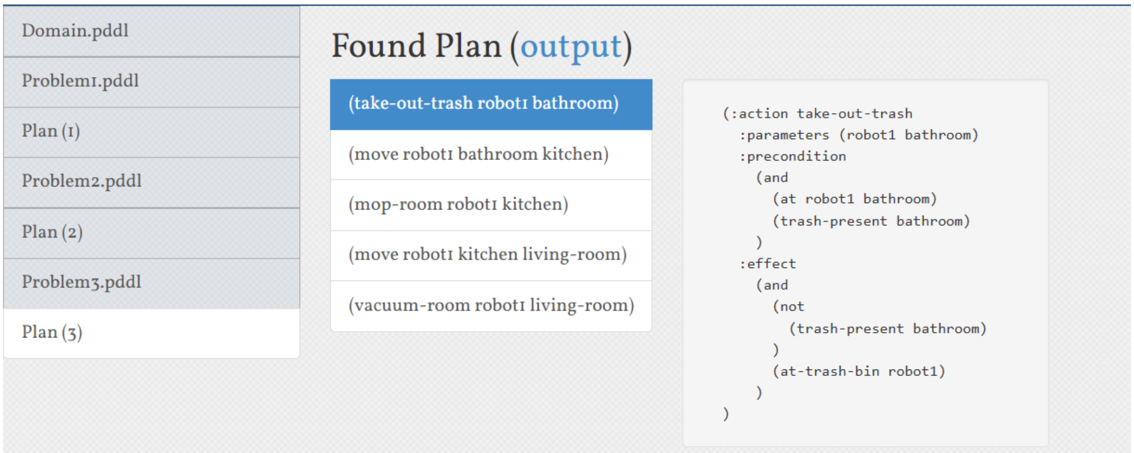
Found Plan (output)

- (recharge robot1)
- (move robot1 bathroom kitchen)
- (mop-room robot1 kitchen)
- (move robot1 kitchen living-room)
- (vacuum-room robot1 living-room)

```
(:action recharge
  :parameters (robot1)
  :precondition
    (battery-low robot1)
  :effect
    (not
      (battery-low robot1)
    )
)
```

Problem3.pddl : Clean-Nearby-First





Problem4.pddl : One-Room-Mess

Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)
Problem3.pddl
Plan (3)
Problem4.pddl
Plan (4)

```
(define (problem one-room-mess)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 bedroom)
  (dirty bedroom)
  (spilled bedroom)
  (trash-present bedroom)
)
(:goal
  (and
    (not (dirty bedroom))
    (not (spilled bedroom))
    (not (trash-present bedroom))
  )
)
)
```

Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)
Problem3.pddl
Plan (3)
Problem4.pddl
Plan (4)

Found Plan (output)

(take-out-trash robot1 bedroom)
(mop-room robot1 bedroom)
(vacuum-room robot1 bedroom)

```
(:action take-out-trash
  :parameters (robot1 bedroom)
  :precondition
    (and
      (at robot1 bedroom)
      (trash-present bedroom)
    )
  :effect
    (and
      (not
        (trash-present bedroom)
      )
      (at-trash-bin robot1)
    )
)
```

Problem5.pddl : Clean-Entire-Home

Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)
Problem3.pddl
Plan (3)
Problem4.pddl
Plan (4)
Problem5.pddl
Plan (5)

```
(define (problem clean-entire-home)
(:domain home-cleaning)
(:objects
  robot1 - robot
  kitchen living-room bedroom bathroom - room
)
(:init
  (at robot1 living-room)
  (dirty living-room)
  (spilled kitchen)
  (trash-present bedroom)
  (battery-low robot1)
)
(:goal
  (and
    (not (dirty living-room))
    (not (spilled kitchen))
    (not (trash-present bedroom))
    (not (battery-low robot1))
  )
)
)
```

Domain.pddl
Problem1.pddl
Plan (1)
Problem2.pddl
Plan (2)
Problem3.pddl
Plan (3)
Problem4.pddl
Plan (4)
Problem5.pddl
Plan (5)

Found Plan (output)

(vacuum-room robot1 living-room)
(recharge robot1)
(move robot1 living-room bedroom)
(take-out-trash robot1 bedroom)
(move robot1 bedroom kitchen)
(mop-room robot1 kitchen)

```
(:action vacuum-room
  :parameters (robot1 living-room)
  :precondition
    (and
      (at robot1 living-room)
      (dirty living-room)
    )
  :effect
    (not
      (dirty living-room)
    )
)
```

Explanation of PDDL Model:

Domain.pddl

The domain defines the types, predicates, and actions available to the robot:

- Types:
  - room: Represents rooms in the house.
  - robot: Represents the cleaning robot.
- Predicates:
  - (at ?r ?rm): The robot ?r is in room ?rm.
  - (dirty ?rm): Room ?rm is dirty.
  - (spilled ?rm): Room ?rm has a liquid spill.
  - (trash-present ?rm): Room ?rm has trash.
  - (battery-low ?r): Robot ?r has low battery.
  - (at-trash-bin ?r): Robot ?r is at the trash bin.
- Actions:
  - move: Allows the robot to move from one room to another.
  - vacuum-room: Cleans dirt from a room.
  - mop-room: Mops a room with a spill.
  - take-out-trash: Removes trash from a room and takes it to the trash bin.
  - recharge: Recharges the robot when its battery is low.

Each action has preconditions (what must be true to execute the action) and effects (how the world changes after the action).

Problem.pddl

Defines a specific scenario using the domain:

- Objects:
  - robot1 is the only robot.
  - kitchen, living-room, bedroom, and bathroom are the rooms.
- Initial State:
  - robot1 is in the kitchen.
  - The living room is dirty.
  - The kitchen has a spill.
  - The bedroom has trash.
  - The robot has low battery.
- Goal: The robot must:
  - Clean the living room.
  - Mop the kitchen.
  - Remove the trash from the bedroom.
  - Recharge itself.

Post Lab Descriptive Questions:

1. **How does ADL (Action Description Language) extend STRIPS?**

ADL (Action Description Language) extends STRIPS by providing greater expressive power in representing planning problems. While STRIPS restricts actions to having simple preconditions and effects, ADL allows:

- Conjunctive, disjunctive, and negated conditions in both preconditions and effects.
- Quantified statements (using `forall`, `exists`) in action definitions.
- Conditional effects—effects that occur only if certain conditions hold.
- Typing of variables to restrict their domain (also added to later STRIPS variants).

This makes ADL more suitable for complex, real-world domains where conditional

logic, nested reasoning, and variable constraints are essential.

2. Define **Partial Order Planning (POP)** and **Total Order Planning (TOP)**. How do they differ?

Partial Order Planning (POP):
In POP, actions are partially ordered, meaning only the necessary orderings between actions are enforced. The planner allows flexibility in the execution sequence, as long as causal dependencies and constraints are preserved. This leads to more efficient and parallelizable plans.

Total Order Planning (TOP):
In TOP, all actions are strictly ordered—each step follows a fixed sequence. While simpler to implement, it can be less flexible and sometimes less efficient, especially in domains where multiple actions are independent and could be done in parallel.

Aspect	**POP**	**TOP**
Action sequence	Partially ordered	Totally ordered
Flexibility	High	Low
Parallelism	Supports parallel actions	Sequential only

Plan representation	Graph-based (steps, causal links)	Linear sequence

3. Would **Partial Order Planning** be beneficial in this problem? Why or why not?

Yes, **Partial Order Planning (POP)** would be beneficial in this problem. Here's why:

4. Many cleaning actions (like mopping the kitchen, vacuuming the living room, and taking out the trash) are **independent** and can be done in **any order**, provided the robot is in the correct room and meets preconditions.

5. POP allows the planner to **defer decisions** about the order of actions, leading to a **more efficient plan** with possible **parallel execution** (if multiple robots were added later).

6. This flexibility could also help the planner **reorder actions dynamically** if unexpected situations arise (like battery draining sooner).

7. Correlate the knowledge engineerings steps with PDDL scripts

The process of knowledge engineering in AI planning maps directly to how we write PDDL scripts. Here's how:

1. Domain Understanding and Problem Modeling
   Identify key objects, actions, and goals in the real world.
   In PDDL: Reflected in the domain's :types, :predicates, and :actions.

2. Defining Action Semantics
   Specify what each action requires (preconditions) and how it changes the world (effects).
   In PDDL: Encoded using :action blocks in the domain file.

3. Environment Initialization
   Describe the starting state of the world.
   In PDDL: Done in the (:init ...) section of the problem file.

4. Goal Specification
   Clearly define what constitutes a successful state.
   In PDDL: Specified in the (:goal ...) block.

5. Validation and Iteration
   Simulate, test, and refine the model to ensure correctness and efficiency.
   In practice: Use planners to run the PDDL and revise as needed.