Exercise 2: Automated Planning

Task 1

Task 1.1

- Domain PDDL see: t1_1-domain.pddl
- Problem PDDL see: t1_1-problem.pddl

Task 1.2

- Domain PDDL see: t1_2-domain.pddl
- Problem PDDL see: t1_2-problem.pddl

Task 2

Run the planner with the astar algorithm:

```
python3 plan.py -s astar examples/blocks-world-domain.pddl examples/blocks-world-problem.pddl
```

Task 3

```
python3 plan.py -s bfs task-3/t3-domain.pddl task-3/t3-problem.pddl
```

```
python3 plan.py -s astar task-3/t3-domain.pddl task-3/t3-problem.pddl
```

Both results in:

```
(move reception1 storage1)
(gethoover storage1)
(move storage1 room1)
(cleanroom room1)
```

The solution in t3-solution.soln is:

```
(move reception1 room1)
(move room1 storage1)
(gethoover storage1)
(move storage1 room1)
(cleanroom room1)
```

1. What is the anomaly that occurs when solving the problem in the domain?

The Sussman Anomaly occurs in planning domains where subgoals interact negatively with each other. In this problem we have two subgoals:

- 1: Clean `room1` using a hoover (`clean room1`).
- 2: Get the hoover from `storage1` (`getHoover storage1`).

the planner tries to achieve subgoal 1, it fails because cleaning requires the hoover.

The requirements for achieving subgoal 1 (`clean room1`) are defined in the domain as:

```
(:action cleanRoom
    :parameters (?r - room)
    :precondition (and (at ?r) (haveHoover)) ; Needs hoover and to be in the room
    :effect (clean ?r) ; Marks the room as clean
)
```

Moving back to retrieve the hoover introduces redundant actions.

---

2. Under what circumstances does the anomaly generally occur in classical STRIPS planning?

It occurs in scenarios where subgoal dependencies clash. This happens when:

1. Subgoal interaction:
   - Subgoal 1 creates a condition (`at room1`) that blocks Subgoal 2 (`getting the hoover` from `storage1`).
2. Goal interference:
   - Resolving one subgoal undoes progress toward others (moving to `room1` for cleaning breaks the precondition of being at `storage1` to retrieve the hoover).
3. Sequential goal resolution:
   - Planners tackle subgoals sequentially without considering their dependencies.

---

3. What specifically in the problem and the domain make it susceptible?

The structural features of the domain and problem make it susceptible.

Action preconditions and dependencies `cleanRoom` depends on both `(at room1)` and `(haveHoover)`, requiring careful order (see snipped above).

The agent must explicitly satisfy `(at storage1)` to acquire the hoover, which conflicts with prematurely attempting to clean `room1`.

getHoover:

```
(:action getHoover
    :parameters (?s - storage)
    :precondition (at ?s) ; Must be at the storage location
    :effect (haveHoover) ; Picks up the hoover, enabling cleaning
)
```

Initial problem setup The agent starts in `reception1`: `(:init        (at reception1) ; Initial location of the agent    )` The goal is to clean the room: `(:goal        (clean room1) ; Requires interleaving actions to achieve    )`

Spatial constraints Moving between `reception1`, `room1`, and `storage1` requires efficient planning to minimize redundant moves. If dependencies like `(haveHoover)` and `(at room1)` are resolved in isolation or in the wrong order, they reinforce conflicts.

---

4. Why is the behavior not observable with your planner implementation from Task 2?

The problem doesn't occur due to the BFS or A* search strategy, which examines all possible sequences of actions layer by layer. Thus both BFS and A* are capable of finding the optimal interleaving.

The proper sequence is determined because `(haveHoover)` is identified as a prerequisite for `(clean room1)`. By correctly interleaving `getHoover` before `cleanRoom`, the planner avoids the negative interaction between subgoals.