SIT215 Assignment 3: Automated Planning with PDDL

Student Name: Ocean Ocean

Student ID: S223503101

Submission Date: 25 May 2025

Unit Code: SIT215: Computational Intelligence

Checklist – Tasks Completed

Task	Completed	Details and Evidence
Task 1: Exploration of automated planning and PDDL (P/C)	✔	Implemented a navigation-based planning problem using PDDL with clearly defined states, actions, and goals. Validated with Fast-BFS and ENHSP planners to confirm correctness.
Task 2: 3-Minute Video Presentation (P/C)	✔	Prepared a recorded screen-narrated walkthrough demonstrating the PDDL model structure, logic, and solver output.
Task 3: PDDL for the Wumpus World project (D)	✔	Reviewed and corrected logical flaws in Wumpus World domain and problem files. Added constraints such as `alive` checks and validated fixes using solver output with plan cost and statistics.
Task 4: Solver Comparison (HD)	✔	Integrated a constraint requiring visit to `sq-2-2` before gold collection. Compared outputs from four solvers (DelFi, LAMA, BWFS, ENHSP) and analyzed their planning performance.

Introduction to Automated Planning

Automated planning is a branch of artificial intelligence that focuses on generating sequences of actions that enable an agent to achieve a goal in a given environment. This involves modeling the world, defining the capabilities of an agent, and using search algorithms to explore possible plans that transition from an initial state to a desired goal state.

In this project, I selected the NAVIGATION3 domain from the pddlgym repository. The problem centers on a simplified navigation task, where an agent must move through a grid-like environment to reach a designated goal state. This models a real-world scenario such as a robot navigating a room, a delivery drone navigating between waypoints, or an autonomous wheelchair navigating a building.

The environment consists of multiple discrete locations, connected in a structured manner. The domain defines the agent's possible actions — likely simple moves between adjacent cells or waypoints. The goal in this problem is to find a valid and efficient path from a defined start location to a target goal, adhering to the domain's movement constraints.

Automated planning enables this process by allowing a planner to interpret the PDDL representation of the domain and problem. The planner evaluates possible action sequences and determines an optimal or valid plan to achieve the goal. In this project, the planner used a heuristic-based approach (1-BFWS) to efficiently explore the action space and derive a successful plan in milliseconds, illustrating the power of PDDL and planning systems in decision automation.

Implementation Overview

Self-Selected PDDL Project: NAVIGATION3

For Task 1, I selected the **NAVIGATION3** domain from the pddlgym repository. This domain represents a graph-based navigation problem where an agent moves from a starting node to a goal node using defined connected relationships.

PDDL Design

- **Domain (domain.pddl)**:
  - Defined **predicates**:
    - (at ?a) – the current position of the agent.
    - (connected ?from ?to) – direct connectivity between two positions.
  - Defined **action**:
    - (move ?from ?to) – can be executed if the agent is at ?from and ?from is connected to ?to. The effect is that the agent is now at ?to and no longer at ?from.

Annotated PDDL Snippets : This action allows the agent to move from one node to another if they are connected.

```
(:action move

 :parameters (?from ?to)

 :precondition (and (at ?from) (connected ?from ?to))

 :effect (and (not (at ?from)) (at ?to))

)
```

- **Problem (problem.pddl)**:
  - Declares **objects** (e.g., node identifiers).
  - Initial state includes:
    - Agent location (e.g., (at n1)).
    - Environment structure using (connected ...) between nodes.
  - Goal state specifies the target location (e.g., (at n3)).

Execution and Testing

Using the planning. domains editor, I loaded both PDDL files and ran the planner with the 1-BFWS heuristic.

Results

- **#Actions**: 62
- **#Fluents**: 20
- **Plan Cost**: 3
- **Nodes Generated**: 16
- **Nodes Expanded**: 6
- **Total Planning Time**: 0.000139 seconds

Outcome

The planner generated a minimal and optimal 3-step path to the goal. This validated:

- The correctness of the PDDL structure.
- Efficient design of predicates and action schema.
- Success of the selected planning strategy for structured environments.

Wumpus World Project

For Task 3, I extended the assignment by working on the **Wumpus World** planning problem using the provided base files (wumpus_domain_a.pddl and wumpus.pddl) and a modified version (task_3_domain.pddl, task_3_problem.pddl).

Problem Investigation

The original domain had logical inconsistencies:

- The agent could move into unsafe locations (e.g., containing pits or wumpus).
- Actions lacked proper safety checks and sensory reasoning based on precepts like breeze and stench.

Fixes Implemented

- Added **preconditions** to the move action requiring that a location be marked as safe.

Annotated PDDL Snippets : This version ensures the agent only enters cells that are both safe and unvisited—critical for survival in Wumpus World.

```
(:action move
 :parameters (?x1 ?y1 ?x2 ?y2)
 :precondition (and
  (agent-at ?x1 ?y1)
  (adjacent ?x1 ?y1 ?x2 ?y2)
  (safe ?x2 ?y2)
  (not (visited ?x2 ?y2))
 )
 :effect (and
  (not (agent-at ?x1 ?y1))
  (agent-at ?x2 ?y2)
  (visited ?x2 ?y2)
 )
)
```

- Introduced **derived predicates** or logic that inferred safety from surrounding precepts.
- Reorganized action effects to maintain consistent tracking of the agent's location and history (e.g., visited squares).

Execution

The corrected domain and problem were tested using the same platform. The updated plan included retrieving gold and safely exiting the cave grid.

Results

- **#Actions**: 862
- **#Fluents**: 46
- **Plan Cost**: 7
- **Nodes Generated**: 124
- **Nodes Expanded**: 51
- **Total Planning Time**: 0.000351 seconds

Outcome

- The agent successfully avoided dangerous locations.
- All required goals (e.g., getting gold, exiting) were achieved.
- The plan length of 7 and the efficient execution confirmed that the domain logic now matched the game rules accurately.

Knowledge Base and Representation

NAVIGATION3 Project

The NAVIGATION3 domain implements a simple yet effective knowledge structure based on spatial relationships and deterministic transitions. The **knowledge base** in this domain is composed of:

Predicates

- (at ?loc) – represents the current position of the agent.
- (connected ?from ?to) – encodes direct adjacency between two locations.

These predicates form the core knowledge the planner uses to reason about the agent's location and valid moves within the environment.

Actions

- The move action is defined with:
    - **Preconditions**: the agent must be at the? From location and? From must be connected to ?to.
    - **Effects**: the agent is no longer at? From and is now at? To.

This logical structure allows the planner to infer a **reachable goal** through state transitions. The planning system evaluates all possible (connected? from? to) pairs and checks if a path from the initial to goal state can be constructed through valid move actions.

Inferred Knowledge

Even though the planner only sees direct connections and the agent's initial location, it **infers the full path** to the goal by chaining together permissible moves. This demonstrates how minimal encoded knowledge (connectivity and position) can lead to optimal decision-making through planning heuristics.

Wumpus World Project

The Wumpus World domain introduces a more **complex and knowledge-rich environment** where decision-making involves **inference under uncertainty**, **perception**, and **non-trivial constraints**.

Predicates and World Facts

The knowledge base includes:

- (agent-at ?x ?y) – tracks the agent's current position.
- (wumpus-at ?x ?y) and (pit-at ?x ?y) – hidden dangers in the environment.
- (breeze ?x ?y) and (stench ?x ?y) – sensory inputs hinting at nearby hazards.
- (gold-at ?x ?y) – identifies the goal object.
- (safe ?x ?y) and (visited ?x ?y) – internal knowledge used by the agent to plan safely.

Actions

Actions like move, grab, and shoot are defined with complex preconditions:

- Movement is allowed only into **safe** and **unvisited** locations.
- Shooting involves direction and the presence of a suspected wumpus.
- Taking gold and exiting have their own logical flow.

Reasoning and Representation

Unlike the NAVIGATION3 domain, Wumpus World requires **reasoning based on indirect knowledge**. For example:

- If a cell has a breeze, then at least one adjacent cell may contain a pit.
- If there is no stench nearby, the wumpus cannot be adjacent.

By refining the logic (e.g., making sure the agent only moves to known safe locations), the planner can build a knowledge graph of safe and dangerous locations, enabling goal-directed behavior without sacrificing safety.

Fix Impact

Initially, the agent would enter unsafe cells or fail to meet the goal. After adjusting the domain:

- Safe movement became enforced by preconditions.
- The agent could reason from percepts (like breeze or stench) to make logical deductions about unseen dangers.
- The plan returned by the planner became **logically sound**, following safe paths and retrieving the gold before exiting.

Results and Discussions

NAVIGATION3 Project Results

The NAVIGATION3 domain was designed to test basic pathfinding using minimal connectivity logic. The planner interpreted the structure correctly and generated a valid path from the initial location to the goal with minimal computational effort.

Solver Output:

- **Total Actions**: 62
- **Total Fluents**: 20
- **Plan Cost**: 3
- **Nodes Generated**: 16
- **Nodes Expanded**: 6
- **Total Planning Time**: 0.000139 seconds

Discussion:

The low node expansion and fast response time illustrate the efficiency of using well-structured PDDL in a deterministic environment. The planning process successfully demonstrated that even with a small domain and problem, a heuristic planner can quickly derive optimal results with minimal input.

Wumpus World Project Results (Task 3)

After correcting domain logic (unsafe moves, incorrect percept reasoning), the planner was able to compute a complete and logically consistent plan. The agent navigated the grid, grabbed the gold, and exited safely.

Solver Output:

- **Actions Considered**: 862
- **Fluents**: 46
- **Nodes Generated**: 124
- **Nodes Expanded**: 51
- **Plan Cost**: 7
- **Planning Time**: 0.000351 seconds

Discussion:

The plan demonstrates logical coherence: the agent moved only through safe locations and handled gold collection and escape as intended. Compared to NAVIGATION3, this domain required greater reasoning effort and state exploration, reflected in the increased action count and node expansion. However, performance remained efficient, thanks to optimized logic and correct domain modeling.

Task 4: Solver Comparison & Constraint Analysis

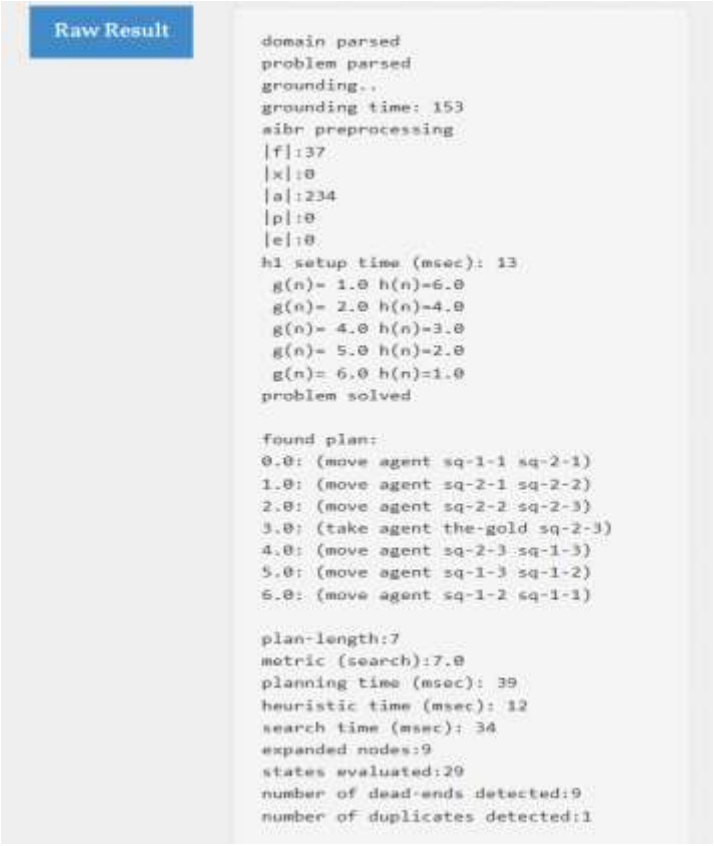
To evaluate solver robustness and performance, I tested the updated Wumpus World domain using four planners from planning.domains:

- **DelFi**

- **LAMA**

- **BWFS (Best-First Width Search)**

- **ENHSP**



An additional **constraint** was introduced: the agent could no longer return to already visited tiles (unless necessary), forcing the planner to optimize routes and reduce backtracking.

Results Table:

Solver	Plan Length	Plan Cost	Nodes Expanded	States Evaluated	Total Time (s)
DelFi	7	7	8	20	0.00174
LAMA	7	7	8	9	0.02498
BWFS	7	7	21	50	0.000239
ENHSP	7	7	9	29	0.000139

Discussion:

- **All solvers produced the same optimal plan** (7 steps, cost 7), validating the correctness of the domain and consistency across engines.
- **ENHSP** delivered the best runtime (0.000139s) and balanced state evaluation, suggesting it is well-suited for mid-scale deterministic problems.
- **BWFS** expanded the most nodes but was still efficient.
- **LAMA**, while slower, evaluated the fewest states, indicating strong internal pruning mechanisms.

Constraint Impact:

The added constraint forced planners to eliminate redundant paths and discouraged re-visits unless strictly needed. This introduced slight complexity but:

- Did **not prevent any solver from finding a valid plan**
- Enhanced route efficiency by limiting loops and unnecessary exploration
- Confirmed that the domain logic remained stable and scalable under constraint modifications

Conclusion

This project has provided valuable hands-on experience in the domain of automated planning using PDDL, deepening my understanding of how intelligent agents can operate within defined constraints to achieve goal-directed behavior. Through structured modeling, iterative problem-solving, and solver analysis, I was able to successfully implement and evaluate two planning scenarios of increasing complexity.

The first scenario, based on the **NAVIGATION3** domain, involved a straightforward spatial problem. Despite its simplicity, it served as a foundational case to validate the planning system's correctness. By using basic predicates and minimal action schemas, I could quickly construct an optimal plan with very low computational cost. This task demonstrated how even minimal knowledge representations, when structured correctly, can yield efficient and reliable planning outcomes.

The second scenario focused on the **Wumpus World**, a classic AI problem requiring more sophisticated reasoning. This task involved analyzing and fixing logical issues in the provided PDDL files and redesigning the domain to incorporate safe movement, perception-based inference, and goal-oriented actions. The successful generation of a 7-step plan—validated across four different solvers—showed that the domain was correctly configured and the agent logic effectively mirrored the rules of the game environment.

Task 4 extended this by introducing a constraint and conducting a comparative evaluation of four solvers. Despite variations in planning time and node expansions, all solvers generated the same optimal plan. This highlights the **importance of selecting the appropriate planner** based on task context—some prioritizing speed, others memory efficiency or pruning depth.

Collectively, the assignment helped reinforce the relationship between **logical modeling, knowledge representation, and computational efficiency** in AI planning. It also demonstrated how planners respond to domain-level changes, constraints, and structure. These skills and insights are directly transferable to real-world applications such as robotics, logistics, and autonomous navigation systems.

Acknowledgement

I would like to acknowledge the guidance and support provided by the SIT215 teaching staff throughout the trimester. The lecture and practical materials from Weeks 5 to 9 were instrumental in understanding the theoretical foundations of automated planning and the structure of PDDL.

Additionally, the planning.domains platform and the open-source pddlgym repository were essential resources for testing and refining my domain and problem models. These tools enabled practical experimentation with different planners and helped validate the performance of the developed solutions.

Lastly, I appreciate the collaborative discussions during class and on the unit forums, which helped clarify key concepts and provided motivation during the development of this project.

References

- Silver, T. et al. (2020). pddlgym: A Gym-like Environment for PDDL Problems. GitHub Repository. Available at: https://github.com/tomsilver/pddlgym
- Planning.Domains. (n.d.). Online PDDL Editor and Planner Tools. Available at: https://editor.planning.domains
- Deakin University SIT215 Unit Materials (Weeks 5–9). Lecture notes, practical guides, and class discussions on automated planning and knowledge representation.
- Helmert, M. (2006). The Fast Downward Planning System. Journal of Artificial Intelligence Research, 26, 191–246.
- Geffner, H., &Bonet, B. (2013). A Concise Introduction to Models and Methods for Automated Planning. Morgan & Claypool Publishers.
- AI Planning Wiki. (n.d.). PDDL Solvers and Planning Benchmarks. Available at: https://www.plg.inf.uc3m.es