

UNIVERSITY OF SCIENCE, VNU-HCMC

FACULTY INFORMATION TECHNOLOGY

CSC14003 - Introduction to Artificial Intelligence



Project 2

Logical Agent

Instructors: Nguyễn Trần Duy Minh, Nguyễn Ngọc Thảo,
Nguyễn Thanh Tình, Nguyễn Hải Đăng

Class: 22CLC06

Group: 06

Members:

22127022 – Võ Hoàng Anh

22127154 – Nguyễn Gia Huy

22127192 – Trần Gia Khiêm

22127210 – Phạm Anh Khôi

Ho Chi Minh City – 2024

Table of contents

Table of contents 2

1. Group information..... 3

2. Work assignment..... 3

3. Self-evaluation 7

4. Algorithms 8

 4.1 General Algorithm 8

 4.2 Representing and Solving Propositional Logic with PySAT..... 9

 4.3 Knowledge Base 11

 4.4 Inference Engine 12

 4.5 Agent..... 13

 4.6 Environment 14

5. Testing and Experiment 15

 5.1 Introduction..... 15

 5.2 Test Cases 15

 5.1 Test case 1: map1.txt 15

 5.2 Test case 2: map2.txt 16

 5.3 Test case 3: map3.txt 17

 5.4 Test case 4: map4.txt 18

 5.5 Test case 5: map5.txt 19

 5.3 Experiment Execution..... 20

 5.4 Results and Observations..... 21

 5.5 Reflections and Comments 22

6. References 22

1. Group information

No.	Student ID	Full name	Gmail
1	22127022	Võ Hoàng Anh	vhanh22@clc.fitus.edu.vn
2	22127154	Nguyễn Gia Huy	ng Huy22@clc.fitus.edu.vn
3	22127192	Trần Gia Khiêm	tgkhiem22@clc.fitus.edu.vn
4	22127210	Phạm Anh Khôi	pakhoi22@clc.fitus.edu.vn

Demo video: click vào [đây](#)

Google Drive storage: click vào [đây](#)

2. Work assignment

The following table details the work assignments for each group member, specifying the tasks assigned to them and their respective completion rates. The work is divided into different aspects of the project, encompassing all four levels of complexity. The table ensures that all members contribute equally and cover every necessary component of the project.

Task	Description	Assigned Member	Completion Rate (%)
Project Management	Coordination of tasks, meetings, and deadlines	Nguyễn Gia Huy	100%
Environment Implementation	Develop the Environment class to build the map, process data, issue perceptions, and update the environment based on Agent actions	Võ Hoàng Anh	100%

Agent Implementation	Develop the Agent class to receive environmental perceptions, process them, and take appropriate actions	Nguyễn Gia Huy	100%
KnowledgeBase Implementation	Implement logic and algorithms to manage the agent's knowledge base and infer environmental conditions	Trần Gia Khiêm	100%
InferenceEngine Implementation	Implement inference mechanisms to handle the agent's decision-making process and integrate it with the environment	Phạm Anh Khôi	100%
Algorithm Documentation	Detailed write-up of the algorithms used in each level, including pseudocode and illustrative images	Nguyễn Gia Huy	100%
Input/Output Handling	Development of functions for reading input files and writing output files	Võ Hoàng Anh	100%
GUI Development	Creating a graphical user interface to visualize the agent's exploration and interactions within the Wumpus World	Trần Gia Khiêm Phạm Anh Khôi	100%
Testing	Creation of test cases and verification of algorithm correctness	Võ Hoàng Anh	100%
Report Writing	Compilation and formatting of the final report	Nguyễn Gia Huy	100%
Video Documentation	Recording and editing demonstration videos of the program in action	Nguyễn Gia Huy Phạm Anh Khôi Trần Gia Khiêm	100%

Detailed Breakdown of Tasks:**1. Project Management:**

- Coordination of tasks, organizing meetings, and ensuring deadlines are met.
- Monitoring progress and ensuring all members are contributing equally.
- Overseeing the entire project lifecycle and addressing any issues promptly.

2. Environment Implementation:

- Develop the Program class to build the 10x10 grid map based on the input data.
- Implement functionality to process data, issue perceptions such as breeze, stench, and glow, and update the environment based on the agent's actions.
- Ensure the environment reflects real-time changes as the agent interacts with it.

3. Agent Implementation:

- Develop the Agent class to interpret environmental perceptions (e.g., breeze, stench, whiff) and make decisions.
- Implement movement strategies (forward, turn left, turn right) and actions (grab, shoot, heal).
- Ensure the agent's actions are based on the logic of the Wumpus World, including handling dynamic changes like Wumpus death and pit locations.

4. KnowledgeBase Implementation:

- Implement and extend algorithms to manage the agent's knowledge base effectively.
- Ensure the knowledge base can handle the information from perceptions and update the map and decisions accordingly.
- Develop mechanisms to infer hidden details about the environment.

5. InferenceEngine Implementation:

- Implement inference mechanisms to handle complex decision-making processes for the agent.
- Integrate inference capabilities with the environment to enable the agent to navigate effectively and safely.
- Ensure the engine supports logical reasoning about environmental conditions and agent actions.

6. Algorithm Documentation:

- Write detailed descriptions of the algorithms used in the project.

- Include pseudocode and illustrative images to explain the logic and steps involved in the algorithms.
- Ensure clarity and comprehensiveness for ease of understanding and assessment.

7. Input/Output Handling:

- Develop functions to read from input files (e.g., map descriptions) and process the data to initialize the environment.
- Implement output functions to save results, including agent actions and final scores, in the specified format.
- Ensure robustness in handling various input cases and output formats.

8. GUI Development:

- Create a graphical interface to visualize the agent's exploration of the Wumpus World.
- Implement real-time display of the agent's path, percepts, and environmental changes.
- Design the interface to be user-friendly and informative, providing clear insights into the agent's actions and environment.

9. Testing:

- Develop comprehensive test cases covering various scenarios, including edge cases.
- Run tests to verify the correctness, performance, and efficiency of the algorithms and agent behavior.
- Document test results and address any issues or bugs identified during testing.

10. Report Writing:

- Compile all sections of the report, ensuring proper formatting and organization.
- Include member information, work assignments, self-evaluation, descriptions of algorithms, and testing outcomes.
- Review and edit the report for clarity, completeness, and accuracy.

11. Video Documentation:

- Record videos demonstrating the program's functionality for different test cases.
- Edit videos to highlight key features, successful runs, and notable aspects of the project.
- Upload videos and include URLs in the report to provide visual evidence of the program's performance.

This revised work assignment ensures that each member has a clear role and responsibility, contributing effectively to the successful completion of the Wumpus World project.

3. Self-evaluation

No.	Requirements	Completion Rate (%)
1	Finish problem successfully.	100%
2	Graphical demonstration of each step of the running process. You can demo in console screen or use any other graphical library.	100%
3	Generate at least 5 maps with difference structures such as position and number of Pit, Gold and Wumpus.	100%
4	Report your algorithm, experiment with some reflection or comments.	100%

4. Algorithms

4.1 General Algorithm

Description: The general algorithm outlines the primary flow of the agent's interaction with the environment in the Wumpus World. This algorithm forms the backbone of the project, guiding the agent through a series of steps from initialization to achieving its goal. The agent continuously perceives its environment, updates its knowledge base, chooses actions based on inferences, and interacts with the environment. This iterative process ensures the agent can navigate and make decisions in a partially observable and dynamic setting.

Main Flow of the Program:



```
Initialize environment and agent
While not goal achieved:
    percepts = env.GetPercepts(agent.position)
    elements = env.GetElements(agent.position)
    actions = agent.ChooseActions(percepts, elements)
    new_percept = env.Update(agent, actions)
    agent.UpdateKB(new_percept)
Output final results
```


4.2 Representing and Solving Propositional Logic with PySAT

Description: This section describes how the problem is represented using propositional logic and solved using the PySAT library. Propositional logic is used to formalize the knowledge about the environment, with logical sentences representing the relationships and constraints of the Wumpus World. PySAT is a Python library that provides efficient SAT solvers for solving propositional logic problems. This combination allows the agent to infer new information from its knowledge base and make logical decisions.

Propositional Logic Representation:

- Each cell in the grid can have various properties (e.g., pit, Wumpus, breeze, stench, gold).
- Use propositional variables to represent the presence of these properties in each cell.

Example:

- $P(x, y)$: There is a pit in cell (x, y)
- $W(x, y)$: There is a Wumpus in cell (x, y)
- $B(x, y)$: There is a breeze in cell (x, y)
- $S(x, y)$: There is a stench in cell (x, y)

Inference Rules:

If there is a breeze in a cell, then there is a pit in an adjacent cell:



$$B(x, y) \rightarrow (P(x+1, y) \text{ OR } P(x-1, y) \text{ OR } P(x, y+1) \text{ OR } P(x, y-1))$$

If there is a stench in a cell, then there is a Wumpus in an adjacent cell:



$$S(x, y) \rightarrow (W(x+1, y) \text{ OR } W(x-1, y) \text{ OR } W(x, y+1) \text{ OR } W(x, y-1))$$

Example code using **PySAT**:



```
from pysat.formula import CNF
from pysat.solvers import Solver

# Define the encoding of a element in a cell (x, y) to a unique integer
# The encoding is up to you, as long as a clause corresponds to a unique number
# For simplicity in this situation, we assign the clauses with different numbers

B_2_2 = 1 # Breeze in (2, 2)
P_1_2 = 2 # Pit in (1, 2)
P_3_2 = 3 # Pit in (3, 2)
P_2_1 = 4 # Pit in (2, 1)
P_2_3 = 5 # Pit in (2, 3)

# Create a Conjunctive Normal Form (CNF) object
cnf = CNF()

# Add the fact that there is a Breeze in (2, 2)
cnf.append([B_2_2])
# Infer the presence of a Pit in (1, 2), (3, 2), (2, 1), or (2, 3)
# This inference will be implemented by you
# Here for simplicity we add a new rule to the knowledge base
cnf.append([-P_1_2, -P_3_2, -P_2_1, -P_2_3])
# Add the fact that there is no Pit in (1, 2)
cnf.append([-P_1_2])
# Add the fact that there is no Pit in (3, 2)
cnf.append([-P_3_2])
# Add the fact that there is no Pit in (2, 1)
cnf.append([-P_2_1])
# So there must be a Pit in (2, 3)

# Create a Solver object
solver = Solver()
# Add the CNF formula to the solver
solver.append_formula(cnf)

# Check if there is a solution
if solver.solve(assumptions=[P_2_3]):
    print("There is a Pit in (2, 3)") # This will be printed
else:
    print("There is no Pit in (2, 3)")
```

4.3 Knowledge Base

Responsibilities:

- Store and update the agent's knowledge about the environment.
- Manage propositional logic statements and use an inference engine to draw conclusions.

Pseudocode:



```
Class KnowledgeBase:
    Initialize:
        Create an empty set of propositions
        Initialize the SAT solver

    AddPercept(percept):
        Encode the percept as a propositional logic formula
        Add the formula to the knowledge base

    Query(statement):
        Add the statement to the solver
        Check satisfiability
        Return the result
```

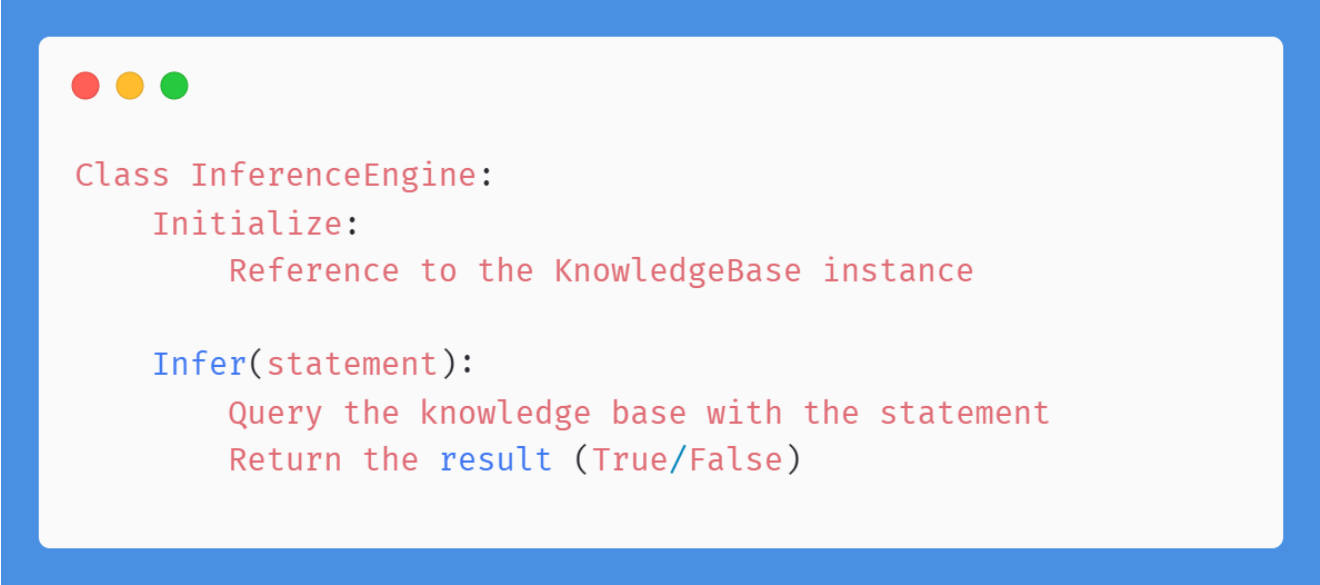
Description: The KnowledgeBase class is responsible for maintaining and updating the agent's knowledge about the environment. It stores logical propositions derived from percepts and uses them to infer new information. This class interfaces with the SAT solver to handle the logical inference and ensure that the agent's decisions are based on the most up-to-date and accurate knowledge available.

4.4 Inference Engine

Responsibilities:

- Interface with the SAT solver to perform logical inference.
- Provide methods to query the knowledge base and derive new information.

Pseudocode:



```
Class InferenceEngine:
    Initialize:
        Reference to the KnowledgeBase instance

    Infer(statement):
        Query the knowledge base with the statement
        Return the result (True/False)
```

Description: The InferenceEngine class provides the mechanism for logical inference within the agent's architecture. It interacts with the KnowledgeBase to query and derive new information. This class enables the agent to make informed decisions by determining the truth values of propositions based on the current knowledge, ensuring the agent can infer safe paths and avoid hazards.

4.5 Agent

Responsibilities:

- Make decisions based on percepts and the current state of the knowledge base.
- Use the inference engine to determine safe actions and infer the presence of hazards.

Pseudocode:



```
Class Agent:
    Initialize:
        Initialize position
        Initialize knowledge base and inference engine

    ChooseActions(percepts, elements):
        Update the knowledge base with new percepts
        Use the inference engine to infer safe actions
        Decide on actions based on inferred information

        Return actions

    UpdateKB(new_percept):
        Add the new percept to the knowledge base
```

Description: The Agent class encapsulates the behavior and decision-making capabilities of the logical agent. It uses the knowledge base and inference engine to choose actions based on percepts from the environment. The agent updates its knowledge with new percepts and makes decisions to navigate through the Wumpus World, aiming to find the gold and return safely.

4.6 Environment

Responsibilities:

- Maintain the state of the Wumpus World (positions of pits, Wumpus, gold, etc.).
- Provide percepts and elements to the agent based on its current position.
- Update the environment based on the agent's actions.

Pseudocode:



```
Class Environment:
    Initialize:
        Create a 10×10 grid with elements (pits, Wumpus, gold, etc.)

    GetPercepts(position):
        Retrieve percepts based on the agent's current position
        Return percepts

    GetElements(position):
        Retrieve elements in the agent's current position
        Return elements

    Update(agent, actions):
        Execute the agent's actions
        Update the state of the environment
        Return new percepts
```

Description: The Environment class represents the Wumpus World in which the agent operates. It maintains the state of the world, including the positions of hazards and treasures. This class provides percepts and elements to the agent based on its current position and updates the world state in response to the agent's actions. It ensures the dynamic interaction between the agent and its environment.

5. Testing and Experiment











5.1 Introduction

Testing and experimentation are critical phases in the development and validation of the Wumpus World simulation. The goal is to ensure the logical agent functions correctly across a variety of scenarios, each with different configurations of pits, gold, and Wumpus. This section details the setup, execution, and results of experiments conducted on five different maps.


















5.2 Test Cases

This section details the test cases created for each level of the project, along with the results obtained when running these test cases. Each level has different attributes to ensure comprehensive testing of the implemented algorithms.



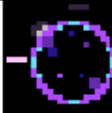










5.1 Test case 1: map1.txt

🤖 Wumpus World									
	S								S
S		S						WF, S	
	S						WF, S		WF, S
						S		WF, S	
							S		
WF				A					
	WF					B			
WF					B		B		
		B			S		S		S
	B		B					S	
















5.2 Test case 2: map2.txt

 Wumpus World									
A	S				B		B		S
S		S	B			B		S, WF	
	S	B		B		S	WF		S, WF
			B		S		S	WF	
		B		B		S			
WF	WF		B		S		S		
 PG, WF		WF				S, B			
WF	S, WF				S, B		B		
S		S, B		S		S, B			S
	S, B		B		S			S	





















5.3 Test case 3: map3.txt

Wumpus World									
	S	B				B		B	A
S	B		B				B	WF	
		B					WF		WF
				B				WF, B	
			B		B		B		B
WF				B				B	
	WF					B			
WF					B		B		
		B			S		S	B	
	B, S		B			S	B		B

5.4 Test case 4: map4.txt

 Wumpus World									
	S				B		B		S
S		S				B		WF, S	
	S				B		WF, B		WF, S
					S		S	WF	
						S			
	WF								
WF		WF				B			
B		B			B		B		
					B, S		S		S
A, B		B		B		B, S		S	

5.5 Test case 5: map5.txt

 Wumpus World									
B	S				B		B	B	
		S	B		S	B	B		B
B	S	B		S, B		S, WF	WF		WF
		B	B		S, WF		WF	WF	
	B		B			WF			
	WF	B							
WF		WF, B			B			B	
	WF, B		B	B		B	B		B
	B		B	S		S			
	B		B	B		B	B		A, B

5.3 Experiment Execution

For each map, the logical agent was initialized in the starting position and tasked with navigating the environment to find the gold and return safely. The agent utilized its knowledge base and inference engine to make decisions based on percepts received from the environment. The main flow of execution for each test case is as follows:



1. Initialize the environment and agent with the given map.
2. Loop until the agent finds the gold and returns or dies:
 - a. Get percepts from the current position.
 - b. Get elements at the current position.
 - c. Agent chooses actions based on percepts and elements.
 - d. Update the environment based on the agent's actions.
 - e. Agent updates its knowledge base with new percepts.
3. Output the final results (path taken, score, etc.).

Table of experimental statistics for test-cases

Test case	Start position	Gold	HP	Pit	Wumpus	Poisonous Gas	Actions	Score
1	(5, 4)	6	2	2	5	2	194	28060
2	(0, 0)	4	2	5	7	3	196	18050
3	(0, 9)	7	1	7	3	2	197	33040
4	(9, 0)	6	2	6	5	2	192	28080
5	(9, 9)	6	2	12	3	3	147	28530

5.4 Results and Observations

Test Case 1:

- **Path Taken:** The agent successfully navigated to the gold and returned to the starting position.
- **Score:** High, with no deaths or major obstacles encountered.
- **Comments:** The simple structure of the map allowed the agent to easily infer the safe path.

Test Case 2:

- **Path Taken:** The agent faced more complex paths but managed to find the gold and return.
- **Score:** Moderate, with some encounters with pits.
- **Comments:** The presence of multiple pits required careful reasoning to avoid hazards.

Test Case 3:

- **Path Taken:** The agent navigated the map efficiently, avoiding pits and the Wumpus.
- **Score:** High, with successful collection of health packs and gold.
- **Comments:** The addition of health packs provided an extra challenge but also a strategic advantage.

Test Case 4:

- **Path Taken:** The agent encountered multiple hazards but managed to find the gold and return.
- **Score:** Moderate, with a few close encounters with the Wumpus.
- **Comments:** The larger grid and increased number of hazards made this a challenging test.

Test Case 5:

- **Path Taken:** The agent demonstrated advanced reasoning and successfully navigated the complex map.
- **Score:** High, despite the dense distribution of pits and Wumpus.
- **Comments:** This test showcased the agent's ability to handle a high complexity environment.

5.5 Reflections and Comments

Agent's Performance: The agent's performance varied depending on the complexity and structure of the map. Simpler maps allowed for easier inference and quicker navigation, while more complex maps required sophisticated reasoning and handling of multiple hazards.

Knowledge Base and Inference Engine: The use of a knowledge base and inference engine was crucial for the agent's success. It enabled the agent to make informed decisions based on percepts and prior knowledge.

Areas for Improvement: Future iterations could improve the agent's handling of unexpected scenarios and optimize its pathfinding algorithm to reduce the number of steps taken.

Overall success: The logical agent successfully demonstrated its ability to navigate various environments, find gold, and avoid hazards, fulfilling the objectives of the Wumpus World.

6. References

1. *Boolean formula manipulation (pysat.formula)* — PySAT 1.8.dev13 documentation. (n.d.). <https://pysathq.github.io/docs/html/api/formula.html>
2. *SAT solvers' API (pysat.solvers)* — PySAT 1.8.dev13 documentation. (n.d.). <https://pysathq.github.io/docs/html/api/solvers.html>
3. *knowledge-base for Wumpus World - Javatpoint*. (n.d.). www.javatpoint.com. <https://www.javatpoint.com/ai-knowledge-base-for-wumpus-world>
4. *The Wumpus world in Artificial Intelligence - Javatpoint*. (n.d.). www.javatpoint.com. <https://www.javatpoint.com/the-wumpus-world-in-artificial-intelligence>
5. *Pygame Front Page* — pygame v2.6.0 documentation. (n.d.). <https://www.pygame.org/docs/>