

The *Wumpus* World

by **MiMaCh System** – *Course of Intelligent Systems a.a. 2021/2022*

Contents



Wumpus World

- Description
- PEAS Description
- Environment Properties



Logic and Knowledge

- Logical Agents
- Fundamentals and Background



Logical Formalisms

- A Graphic Model
- ASP Implementation
- MiniZinc Program



Conclusion

- Uncertainty of the *Wumpus World*
- Further Improvement for our Models

Wumpus World

- Description

- The **Wumpus world** is a simple world example to illustrate the importance of a **knowledge-based agent**.
- It was inspired by a video game **Hunt the Wumpus** by *Gregory Yob* in 1973.
- The Wumpus World is a **cave** composed by 16 **rooms** connected with passageways.
- The cave has a room with a **beast** called *Wumpus* who eats anyone in the room. (*Note: the Wumpus is static*)
- Other rooms can contain a **bottomless Pit**.

Wumpus World

- PEAS Description

Performance Measure:

- +1000 points if the agent takes the gold
- -1000 points if the agent dies (eaten by the *Wumpus* or falling into a pit)
- -1 for each action
- -10 for using its arrow

Environment:

- A matrix 4x4
- The agent starts from position [1,1]
- Location of *Wumpus* and gold are **chosen randomly** (they can be in the same position)
- Each room of the cave can be a **pit** with $p = 0.2$

Wumpus World

- PEAS Description

Actuators:

- Turn Left
- Turn Right
- Move
- Grab
- Release
- Shoot

Sensors:

- The agent will perceive:
 - **Stench**
 - **Breeze**
 - **Glitter**
 - **Bump**
- When the *Wumpus* is shot, its horrible **scream** can be perceived everywhere.

Goal:

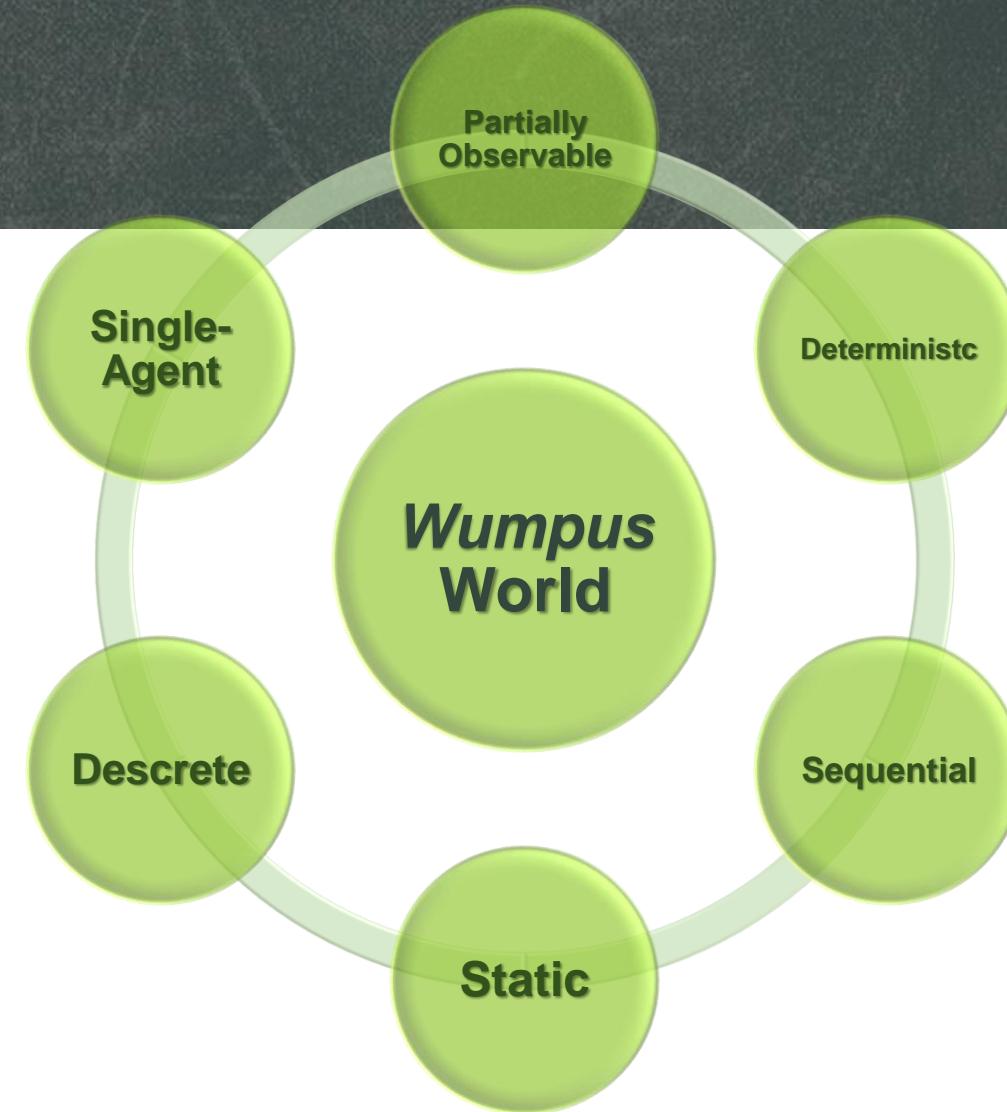
- The **game ends** if:
 - The agent dies
 - Came out of the cave with gold

Wumpus World

- Environment Properties

The *Wumpus world* properties:

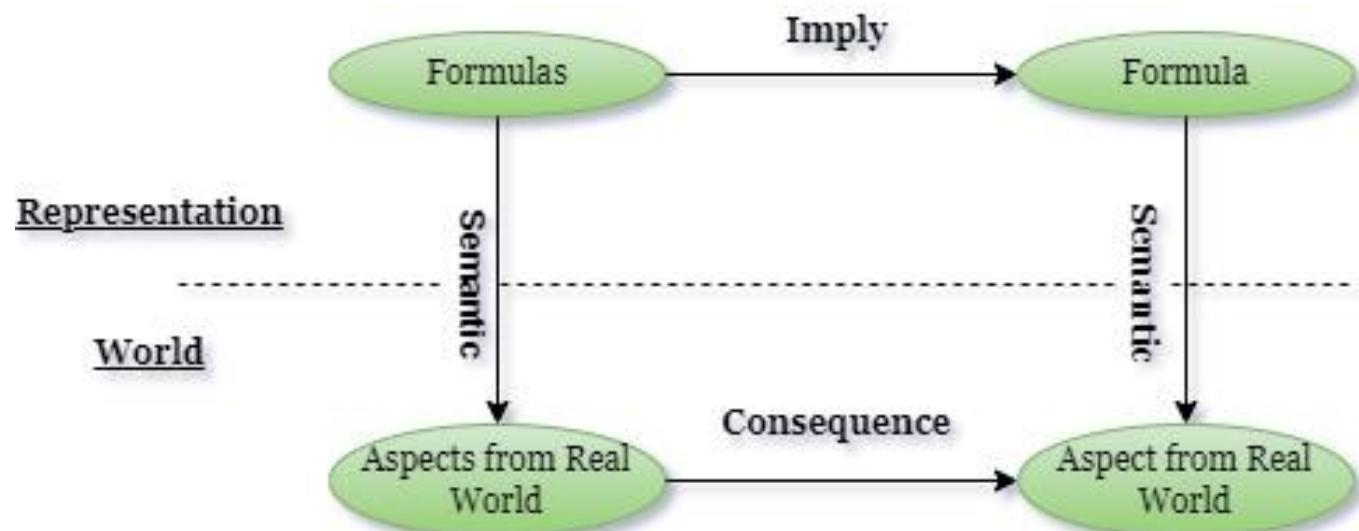
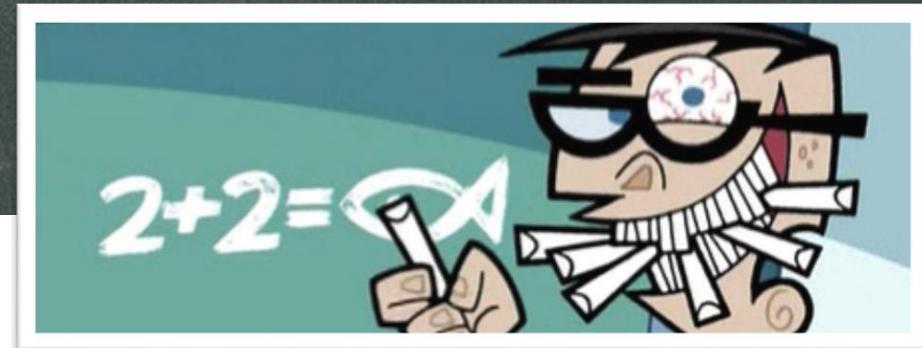
- **Partially Observable**: the agent has a local perception.
- **Deterministic**: the result and outcome of the world are already known.
- **Sequential**: the order is important.
- **Static**: *Wumpus* and *Pits* are not moving.
- **Discrete**: the environment is discrete.
- **Single-Agent**: there is only one agent and *Wumpus* is not considered as an agent.



Logic and Knowledge

- Logical Agents

- **Knowledge-based agents** use their knowledge about the world to generate **good decisions**.
- Knowledge is represented as a **set of sentences** in a formal language. The language defines the truth value of a formula given a specific world.
- A Logical Agent is composed by a **knowledge base** and an **inferential procedures**.



Logic and Knowledge

- Fundamentals and Background

- In AI **Inference** is the process that **generates conclusions** from **evidence and facts**.
 - **Inference rules** are the templates for generating **valid arguments**.
 - The **implication** among all the connectives plays an important role.
 - There are different types of Inference rules.
 - They are used in AI to derive **proofs** (a sequence of «*actions*» to reach the goal).
 - **Propositional Logic** and **First-Order Logic** are the core of Logical Agents.
 - Basis for **ASP** and **CSP**.
- $A \rightarrow B$

Logical Formalisms

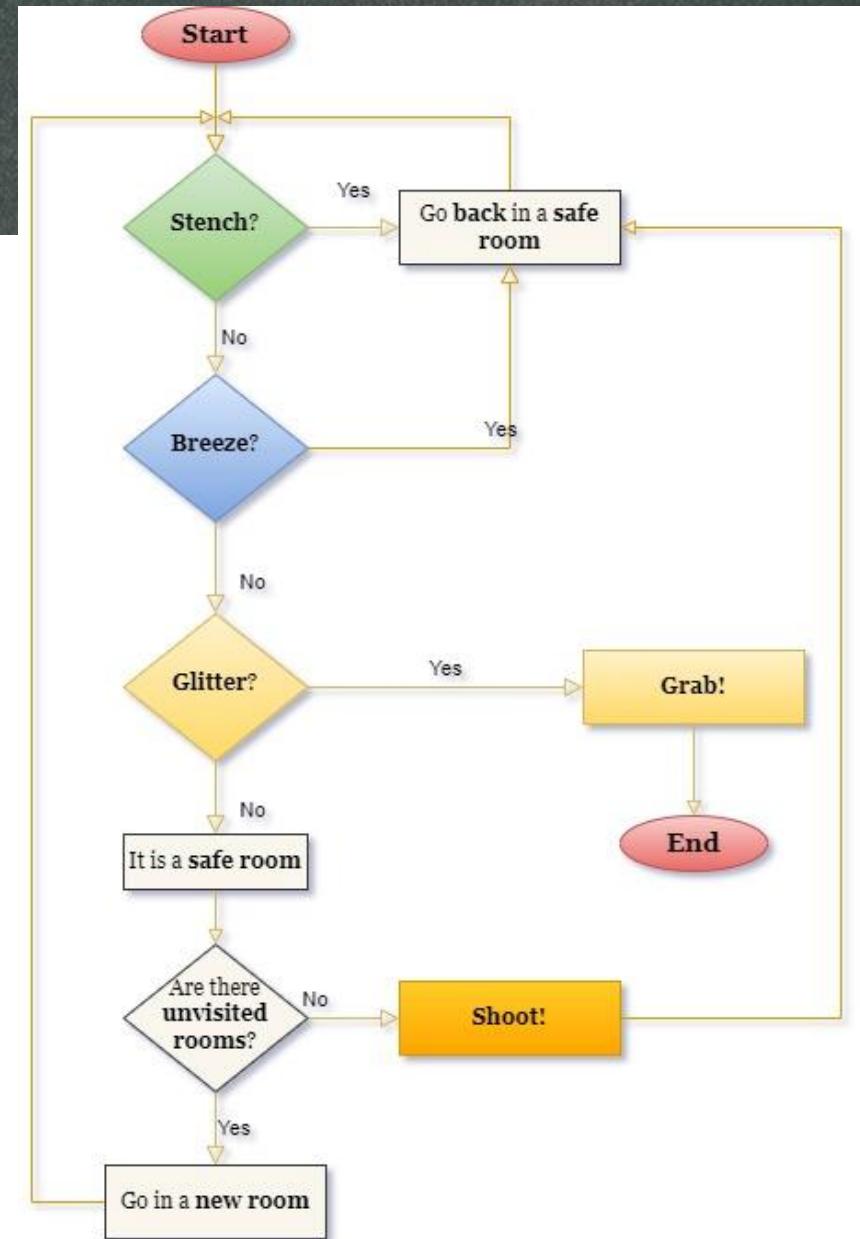
- Flow Chart
- ASP Implementation
- MiniZinc Program

Logical Formalism

- A Graphic Model

Flow Chart

- A flowchart is a **diagram** that describes a **process, system or algorithm**
- Simple and **easy** to understand
- There is a specific set of **flowchart symbols** ([see more here](#))



Logical Formalism

- ASP Implementation

Answer Set Programming

- Answer set programming (**ASP**) is a form of declarative programming oriented towards difficult search problems.
- It is based on the stable model (**answer set**) semantics of **logic programming**.

Logical Formalism

- ASP Implementation

ASP Implementation

Logical Formalism

- MiniZinc Program

Constraint Satisfaction Problems

- Many real-life problems of AI and OR can be formulated as a **Constraint Satisfaction Problem (CSP)**
- CSP is a powerful tool for **knowledge representation**
- **MiniZinc** is a free and open-source **constraint modeling language**.
- Use MiniZinc to **model constraint satisfaction** and **optimization problems**

Logical Formalism

- MiniZinc Program

MiniZinc Implementation

- Data File
- Elements = {Ok=0, Stench=2, Breeze=3, Glitter=4, Pit=5, Wumpus=6};
- We assume that the input instance is correct and there is only one Wumpus and only one room with the Gold

```
wumpus_world_instance.dzn X
1 size  = 4;
2 cave  = [
3   2, 0, 3, 5|
4   6, 4, 5, 3|
5   2, 0, 3, 0|
6   0, 3, 5, 3| ];
```

Logical Formalism

- MiniZinc Program

MiniZinc Implementation

- Model File

```
3 %input matrix 4x4
4 int: size;
5 set of int: DIM = 1..size;
6 array[DIM, DIM] of int: cave;
7 output ["Cave: \n", show2d(cave)];
```

```
8 % as result we want a 0/1 matrix that encodes
9 % points on a path in the cave.
10 % 1 means the point is on the path and 0 means it's not.
11 array[DIM, DIM] of var 0..1: path;
12
13 %Archer starts from path[4,1]
14 %The Game ends when the Archer reaches the gold
15 var int: gold_row;
16 var int: gold_col;
17 constraint forall(i in DIM, j in DIM where cave[i, j]=4)
18 (
19   gold_row=i /\ gold_col=j
20 );
21 %So we fix start room and end room
22 constraint path[size,1]=1 /\ path[gold_row, gold_col]=1;
```

Logical Formalism

- MiniZinc Program

- Model File

```
Cave:  
[| 2, 0, 3, 5  
| 6, 4, 5, 3  
| 2, 0, 3, 0  
| 0, 3, 5, 3  
|]  
  
Path:  
[| 1, 1, 1, 0  
| 0, 1, 0, 1  
| 1, 1, 1, 1  
| 1, 1, 0, 1  
|]  
  
Performance Measure: 988.  
-----  
-----
```

```
26 %Pits can not be in the path  
27 constraint forall(i in DIM, j in DIM where cave[i, j] = 5)(  
28   path[i, j] = 0  
29 );  
30 %Also the room with the Wumpus can not be in the path  
31 constraint forall(i in DIM, j in DIM where cave[i, j] = 6)(  
32   path[i, j] = 0  
33 );  
34  
35 %check the room that can be reached  
36 constraint forall(i in DIM, j in DIM where cave[i, j]=0 \vee cave[i,j]=2 \vee cave[i,j]=3)  
37 (  
38   path[i, j] = 1  
39 );  
  
39 output ["\n Path: \n", show2d(path)];  
40  
41 var int: pathCost = sum(i in DIM, j in DIM where path[i, j] = 1)(1);  
42 output ["\n Performance Measure: \$(1000-pathCost)."];  
43 solve minimize pathCost;
```

Conclusion

- Uncertainty of the *Wumpus World*
- Improvements for our models

Conclusion

- Uncertainty of the *Wumpus World*

- Logic has the advantage of being an easy way to represent the **knowledge base** of the logical agents.
- In a **partially observable** environment it is important to manage also the **uncertain knowledge**.
- **Probability** and **statistics** represent in an efficient way the **uncertainty**.
- These are the basis of the nowadays **Machine Learning**.

Conclusion

- Uncertainty of the *Wumpus* World

What we know

- $p(P[i,j] = \text{true}) = 0.2$
the probability p that in position $[i,j]$ there is a pit P is equal to 0.2
- Positions for the Wumpus and the Gold are random
 - $p(W[i,j] = \text{true}) = \frac{1}{n \times n} = \frac{1}{4 \times 4}$
 - $p(G[i,j] = \text{true}) = \frac{1}{16}$

What we can derive

- Use properties of **conditional probability** to derive new knowledge
 - $p(\text{Effect} \mid \text{Cause})$
 - $p(A|B) = \frac{p(A \cap B)}{p(B)}$
- Apply the Bayes Rule in the *Wumpus* world
 - If there is Breeze in [1,2] [2,1], not in [1,1]
 - $p(B[1,1], B[1,2], B[2,1] \mid P[1,1], \dots, P[4,4]) \cdot p(P[1,1], \dots, P[4,4])$

Conclusion

- Further Improvement for our Models

- Make the implementations more generic, working for $n \times n$ matrices.
- Find a better way to express the game in CSP:
 - Minimize the number of cells in $\text{path}[i,j]=1$
 - In a way that the agent can move in adjacent cells
- Combine the two implementations with a nice and easy online GUI.
- Try to explore more logical formalisms (such as Propositional Logic and First-Order Logic)
- Make a more detailed probability model.

Resources

- **Text Book:**
[Intelligenza Artificiale. Un Approccio Moderno. Stuart J Russell, Peter Norvig. Pearson, 2021.](#)
- **Online Sources:**
[javatpoint.com](#)
[logical agents for wumpus world](#)
[diagrams.net](#)
[what is a flowchart](#)
- **ASP Programming:**
- **CSP Programming and MiniZinc:**
[Constraint Satisfaction Problems](#)
[minizinc.org](#)



Thanks for your attention

MiMaCh System

- Canonaco Martina [231874]
- Morello Michele [223953]
- Passarelli Chiara [223971]