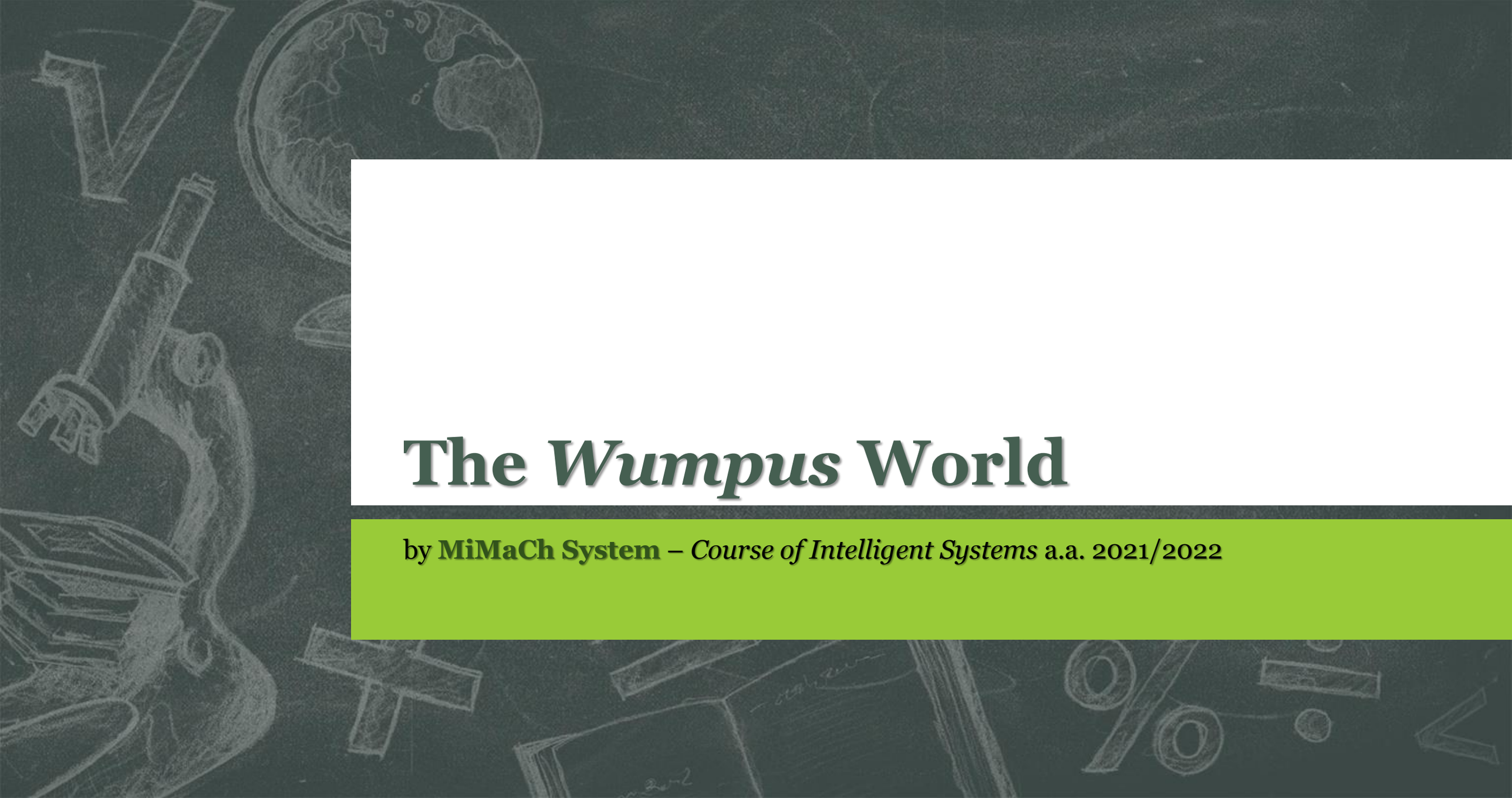


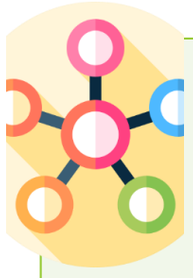
The *Wumpus* World

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 but not in [1,1])
- Each room (except [1,1]) 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.

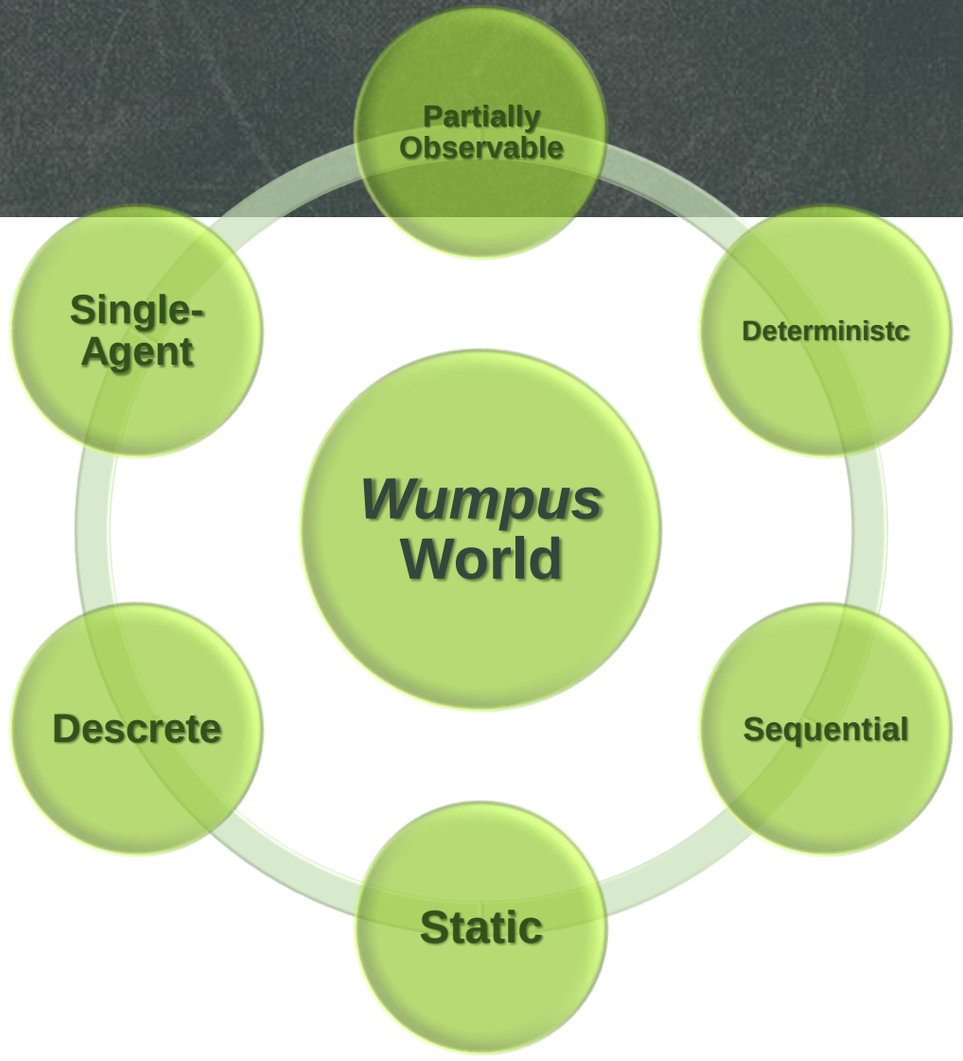
Goal:

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

Wumpus World
- Environment Properties

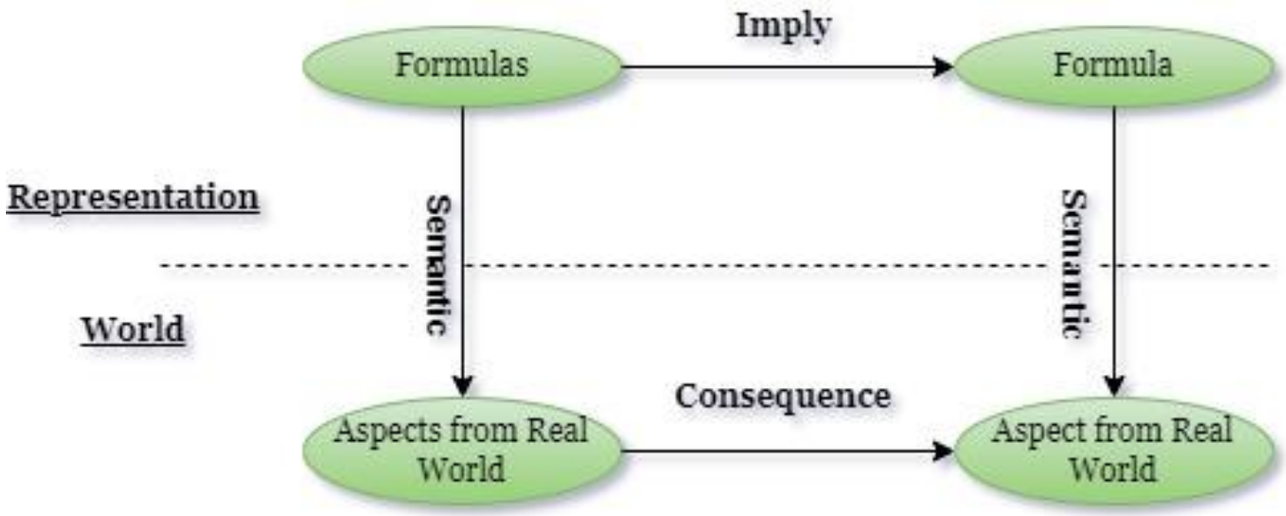
The *Wumpus* world properties:

- **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.

$$A \rightarrow B$$

- 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**.

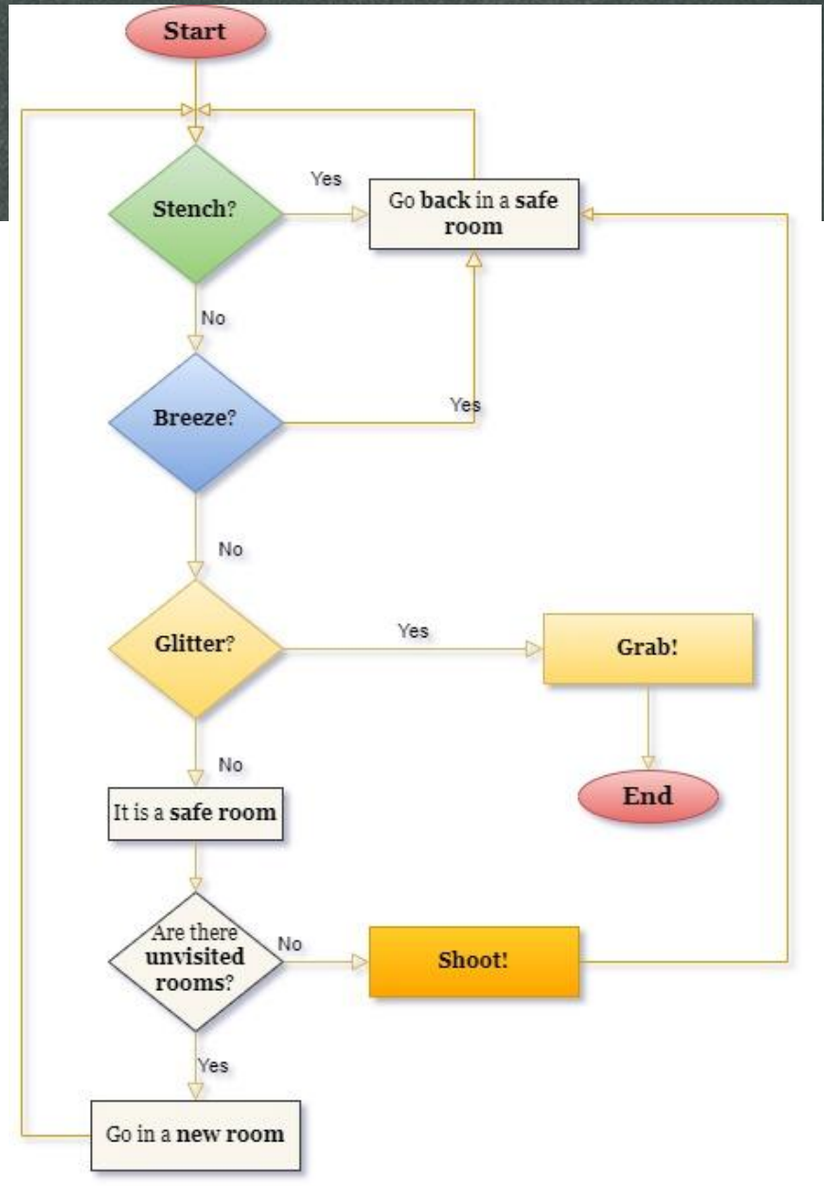
Logical Formalisms

- 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**.

- The implementation is based on the previous **flow chart**

Logical Formalism
- ASP Implementation

```
playerPos(1,1).
visited(X,Y):-playerPos(X,Y).
possibleChoice(X,Y):-playerPos(X,Q), Y=Q+1, Y<=4, not visited(X,Y).
possibleChoice(X,Y):-playerPos(X,Q), Y=Q-1, Y>=1, not visited(X,Y).
possibleChoice(X,Y):-playerPos(Q,Y), X=Q+1, X<=4, not visited(X,Y).
possibleChoice(X,Y):-playerPos(Q,Y), X=Q-1, X>=1, not visited(X,Y).
playerPos(X,Y) | out(X,Y):- possibleChoice(X,Y).
```

```
safe(X,Y):-visited(X,Y), not stench(X,Y), not breeze(X,Y).
:~#count{X,Y: playerPos(X,Y)}!=1. [1@2, X,Y]
:~playerPos(X,Y), not safe(X,Y). [1@1, X,Y]
gold(X,Y):-glitter(X,Y), playerPos(X,Y).
```

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

```
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**

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

```
% as result we want a 0/1 matrix that encodes
% points on a path in the cave.
% 1 means the point is on the path and 0 means it's not.
array[DIM, DIM] of var 0..1: path;

%Archer starts from path[4,1]
%The Game ends when the Archer reaches the gold
var int: gold_row;
var int: gold_col;
constraint forall(i in DIM, j in DIM where cave[i, j]=4)
(
  gold_row=i /\ gold_col=j
);
%So we fix start room and end room
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.
----------
----------
```

```
%Pits can not be in the path
constraint forall(i in DIM, j in DIM where cave[i, j] = 5)(
  path[i, j] = 0
);
%Also the room with the Wumpus can not be in the path
constraint forall(i in DIM, j in DIM where cave[i, j] = 6)(
  path[i, j] = 0
);

%check the room that can be reached
constraint forall(i in DIM, j in DIM where cave[i, j]=0 \/ cave[i,j]=2 \/ cave[i,j]=3)
(
  path[i, j] = 1
);
```

```
output ["\n Path: \n", show2d(path)];

var int: pathCost = sum(i in DIM, j in DIM where path[i, j] = 1)(1);
output ["\n Performance Measure: \(1000-pathCost)."];
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

- $p(P[i,j] = true) = 0.2$
  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
  - $p(W[i,j] = true) = \frac{1}{(n \times n)-1} = \frac{1}{(4 \times 4)-1} = \frac{1}{15}$
  - $p(G[i,j] = true) = \frac{1}{15}$

What we can derive

- Use properties of **conditional probability** to derive new knowledge
  - $p(Effect \mid Cause)$
  - $p(A|B) = \frac{p(A \cap B)}{p(B)}$
- 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 espress the game in CSP:
  - **Minimize** the number of cells in 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:
  Answer Set Programming
- CSP and MiniZinc:
  Constraint Satisfaction Problems
  minizinc.org

Thanks for your attention

MiMaCh System

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