COMP9414 Artificial Intelligence

Tutorial week 1 – Knowledge representation and reasoning

1 Background

Knowledge Representation and Reasoning (KRR) is a fundamental area of AI that focuses on how knowledge about the world can be formally represented and used by a machine to solve complex tasks. This discipline intertwines the cognitive aspects of how humans understand and process information with the computational methods necessary for enabling machines to exhibit intelligent behaviour.

In AI systems, the integration of knowledge representation and reasoning capabilities allows for the development of intelligent agents that can operate autonomously in complex environments. For instance, KRR is essential in natural language processing (NLP) for understanding and generating human language by associating linguistic input with world knowledge.

1.1 Rule-based Systems

A rule-based system is a type of computer system that leverages domain-specific knowledge in the form of predefined rules. In any field or domain, there exists a body of specialised knowledge that experts use to solve problems or make decisions. This knowledge encompasses facts, relationships, constraints, and patterns relevant to that domain. In a rule-based system, this domain-specific knowledge is captured and represented in the form of rules. For example, in the medical domain, knowledge about symptoms, diseases, and their relationships might be represented.

Rules form the backbone of a rule-based system. These rules are expressed in the form of "if-then" statements, also known as production rules. Each rule consists of two parts: the antecedent (the "if" part) and the consequent (the "then" part). The antecedent specifies the conditions or criteria that must be met for the rule to be applied, while the consequent specifies the action or conclusion to be taken if the conditions are met. Rules encode the logical relationships, decision criteria, or problem-solving strategies relevant to the domain.

Rule-based systems can effectively solve problems, make decisions, or provide recommendations within their designated domain. These systems are particularly useful in domains where expertise can be codified into explicit rules, such as expert systems, diagnostic systems, decision support systems, and natural language processing applications. For instance, a rule-based system might assist a doctor in choosing a diagnosis based on symptoms, or select tactical moves to play a game.

1.2 Knowledge Representation

At its core, knowledge representation is about encoding information about the world in a form that a computer system can utilise to solve complex tasks such as diagnosing a medical condition, carrying out natural language conversations, or navigating in a dynamic environment. It involves **defining the structure and semantics of information**, such as objects, events, and their relationships, using languages and frameworks like logic, semantic networks, and frames [1].

The choice of representation significantly impacts a system's ability to reason about the world. For instance, logical representations, such as first-order logic, provide a powerful and expressive language for capturing facts and rules about the world but may be computationally intensive [2]. Alternatively, more heuristic-based representations might offer computational efficiency at the expense of expressiveness.

1.3 Reasoning

Reasoning is the process by which a system derives conclusions or makes decisions based on the knowledge it possesses. In KRR, reasoning typically involves inference mechanisms that apply logical rules to the knowledge base to generate new knowledge or verify the consistency of the existing knowledge.

Various reasoning techniques exist, including deductive reasoning, where conclusions are drawn from general rules; inductive reasoning, which involves generalising from specific instances; and abductive reasoning, which infers the most likely explanations for given data [3]. Each of these reasoning strategies can be applied depending on the nature of the problem and the available knowledge.

2 Environments

2.1 Mountain Car Environment

For the first part of the tutorial, we will use the gymnasium library [1]. This is a well-known library regularly used for reinforcement learning environments. In particular, we will use the Mountain Car [2] environment (see Fig. 1) and create a simple rule-based system to control car movement.

The Mountain Car is a control problem in which a car is located on a unidimensional track between two steep hills. The car starts at a random position at the bottom of the valley ($-0.6 < x < -0.4$) with no velocity ($v = 0$). The

[1] `https://gymnasium.farama.org/`

[2] `https://gymnasium.farama.org/environments/classic_control/mountain_car/`

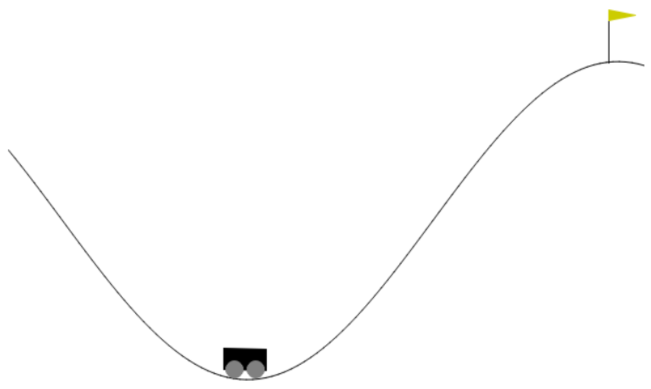


Figure 1: Mountain Car environment

aim is to reach the top of the right hill. However, the car engine does not have enough power to claim to the top directly and, therefore, needs to build momentum moving toward the left hill first. An agent controlling the car movements observes two state variables, namely, the position x and the velocity v. The position x varies between -1.2 and 0.6 in the x-axis (with $x \approx -0.53$ the lowest height) and the velocity v between -0.07 and 0.07. The agent can take three different actions: accelerate the car to the left (0), do not accelerate (1), and accelerate the car to the right (2). The task is completed in case the top of the right hill is climbed ($x \geq 0.5$) or if the length of the episode is 200 iterations in which case the episode is forcibly terminated.

Gymnasium environments provide access to the environment action and observation spaces. The following code initialises an environment while visually rendering the output (be aware that rendering the output might not be available on platforms such as Colab). Next, a random action is selected and performed in the environment for a predetermined number of iterations.

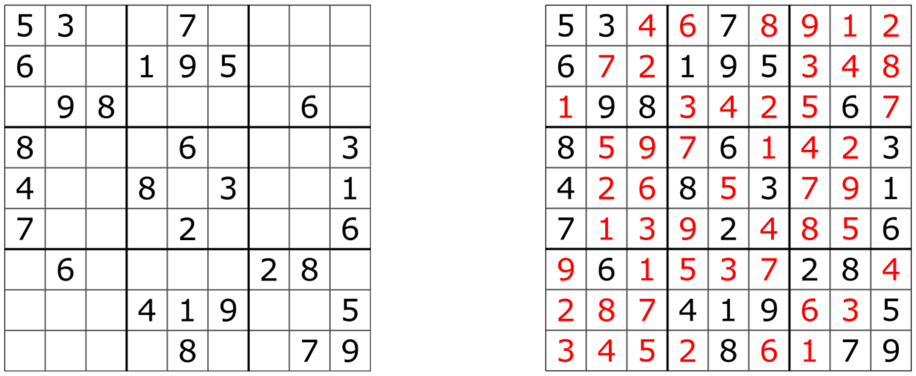
```
import gymnasium as gym
env = gym.make("MountainCar-v0", render_mode="human")
observation, info = env.reset()

for _ in range(1000):
    action = env.action_space.sample()
    observation, reward, terminated, truncated, info = env.step(action)

    if terminated or truncated:
        observation, info = env.reset()
```

```
env.close()
```

2.2 Sudoku

Sudoku is a popular logic-based number puzzle that challenges individuals to fill a 9×9 grid such that each row, column, and 3×3 subgrid contains all digits from 1 to 9 without repetition. The puzzle, which draws on the principles of Latin squares developed by Leonhard Euler, begins with a partially completed grid, and the task is to logically deduce the placement of the remaining numbers. An example of Sudoku is presented in Fig. 2.

Sudoku's appeal and complexity have garnered attention in various academic fields. In mathematics, the puzzle is often explored within combinatorial design theory, focusing on its structural properties and the uniqueness of solutions [2]. In computer science, Sudoku is widely used as a test case for algorithms addressing constraint satisfaction problems, and KRR [4]. Additionally, cognitive scientists study Sudoku to gain insights into human problem-solving behaviour, particularly in understanding how people approach and resolve complex logical challenges [5].



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Figure 2: A typical Sudoku puzzle (left), and the solution to the puzzle (right).

3 Experiments

3.1 Part 1: RBS with Mountain Car

(a) Using Python, create the Mountain Car environment using Gymnasium, then start resetting the environment. Gymnasium needs `swig` and `box2d` packages, therefore, before creating the environment you should run the following:

```
pip install swig
pip install gymnasium[box2d]
```

(b) Create a loop with an adequate number of iterations to run the simulation.

(c) Select a random action and execute it in the environment. Observe how the car position varies over time.

(d) Instead of selecting a random action, select accelerating the car to the right at each time step.

(e) As the car needs to build momentum, let's create two simple rules as follows:

   i Accelerate the car to the right if it is climbing the hill to the right while increasing the velocity.

   ii. Accelerate the car to the left if it is climbing the hill to the left while decreasing the velocity.

(f) Let's expand our knowledge base by adding two more rules, as follows:

   i Accelerate the car to the right if it is climbing the hill to the right while increasing the velocity.

   ii. Accelerate the car to the left if it is climbing the hill to the left while decreasing the velocity.

   iii. Accelerate the car to the right if it is descending the hill to the left while increasing the velocity.

   iv. Accelerate the car to the left if it is descending the hill to the right while decreasing the velocity.

(g) For the previous setups, i.e., always accelerating to the right, two rules knowledge base, and four rules knowledge base, run the experiment 100 times and show the results using boxplots.

(h) **Challenge:** is there any better set of rules you can define for the Mountain Car environment?

3.2 Part 2: KRR with Sudoku

1. Implement the Sudoku presented in Fig. 2 in Python considering as the initial state the figure shown to the left.
2. Can you explain all the knowledge and constraints in Sudoku?
3. Encode all conditions in Sudoku one by one.
4. Can you indicate whether each condition in Sudoku is deductive, inductive, or abductive?
5. Develop an algorithm that can solve Sudoku based on KRR.
6. Print the final solved puzzle.

7. Under what conditions there is no solution for the problem?
8. Can you come up with a Sudoku configuration that is impossible to be solved?

References

[1] J. F. Sowa, *Knowledge representation: logical, philosophical and computational foundations.* Brooks/Cole Publishing Co., 1999.

[2] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach.* Pearson, 2016.

[3] R. Brachman and H. Levesque, *Knowledge representation and reasoning.* Morgan Kaufmann, 2004.

[4] H. Simonis, “Sudoku as a constraint problem,” in *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, vol. 12, pp. 13–27, Citeseer, 2005.

[5] S. I. Robertson, *Problem solving: Perspectives from cognition and neuroscience.* Psychology Press, 2016.