# Planning and Approximate Reasoning

# (MIA- MESIIA) Practical Exercise 2: PAR

# Robot Chef Task

- The deadline for the delivery of this exercise is **November 05th, 2024**
- There is a second submission date set to **January 11st, 2024**, with a maximum grade of **8**.
- A zip file with the source codes in PDDL and a PDF file must be sent.
- The code files and the results should be exported from Visual studio code. If not, include the necessary instructions files to understand the program (documenting the *domain.pddl* and *problem.pddl* file if possible).
- A detailed documentation in PDF is required (see details below)
- The submission of the source and documentation must be done using Moodle.
- This assignment can be achieved by a **group of 2 or 3 persons**.

A Japanese restaurant located in Barcelona has decided to innovate its kitchen operations by employing a **robot chef** to assist with meal preparation. This robot can plan and execute a sequence of actions to accomplish various cooking tasks (e.g., preparing dishes, cooking ingredients, and serving finished meals) as shown in Figure 1.

Figure 1: Robot Chef

**Setup**:

The restaurant kitchen is divided into **7 discrete areas**: the **storage area** (SA), **preparation area** (PA), **cooking area** (CA), **serving area** (SVA), and **dishwashing area** (DWA). Additionally, there are areas for specific ingredient preparation, like **cutting area** (CTA) and **mixing area** (MIXA). The robot chef can move between adjacent areas (e.g., PA and DWA) but cannot move directly between areas blocked by gray walls. For example, the robot can not move directly from **SVA** to **PA**.

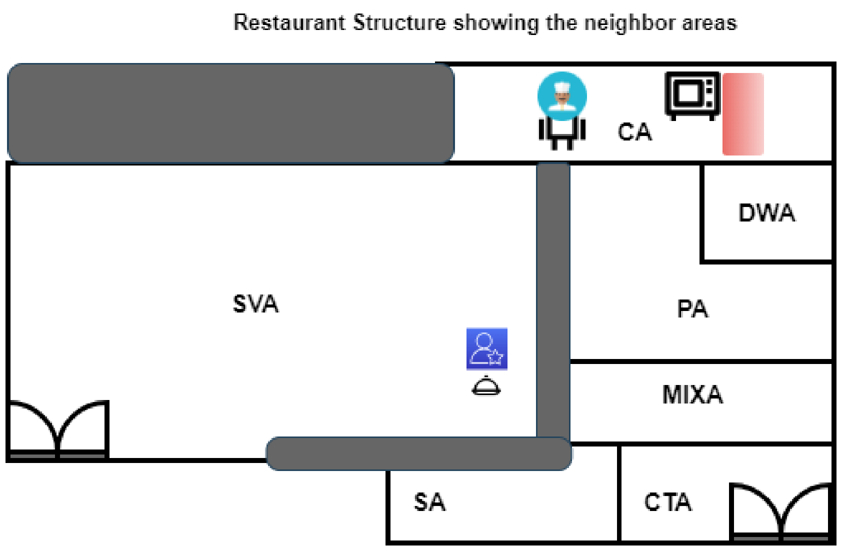

Figure 2: Restaurant Structure, Gray areas means that the robot can not move in these areas.

**Problem Description**:

The **robot chef** receives a list of orders, each specifying a dish that needs to be prepared. For each order, the robot must gather ingredients from the **storage area (SA)**, prepare them in the **preparation area (PA)**, cook them in the **cooking area (CA)**, and then plate the dish in the **serving area (SVA)** for delivery to customers. After cooking, any used cooking tools must be cleaned in the **dishwashing area (DWA)** before being used again.

**Assumptions:**

- Orders are received through a digital system, so the robot knows which dishes to prepare.
- The robot can prepare one dish at a time
- The robot can only carry one item (ingredient or tool) at a time.
- The kitchen must be kept clean, so tools used in one dish need to be washed before being used again.

In the robot cooker task, the types of food and ingredients required can vary from one scenario to another based on the specific dishes ordered by customers. For instance, one scenario might involve preparing SUSHI as illustrated in the next example, requiring ingredients like rice, fish, and seaweed, while another scenario could focus on making RAMEN, which would require noodles, broth, and vegetables.

Thus, the robot must adapt to each case, gathering the appropriate ingredients from storage and following different preparation steps according to the type of dish being prepared. This variability in ingredients and dishes introduces a dynamic aspect to the problem, requiring the robot to plan its actions accordingly to fulfill each unique order.

## Recommendations:

If you want to get more realistic system for robot chef, you can consider more complex aspects of robotic planning and execution within a dynamic and realistic environment, such as:

- **Error Handling and Adaptability**: While the robot's tasks are clearly defined, adding scenarios where the robot must adapt to unexpected situations, such as missing ingredients or equipment failure, could provide depth. Discussing how the robot would detect and handle these issues would be beneficial.
- **Integration with Real-Time Order Systems**: Expanding on how the robot interfaces with the digital order system could clarify operational dynamics, especially how it handles order prioritization and scheduling in real-time.

## Example:

1. The **storage area (SA)** has ingredients like **rice, fish,** and **vegetables**.
2. An order for **sushi** is received.
3. The robot must gather **rice, vegetables** and **fish** from **SA**, **cut** the fish and vegetables in the **cutting area (CTA)**, **mix** the rice in the **mixing area (MIXA)**, and **cook** the rice in the **cooking area (CA)**.
4. The robot then **assembles the sushi** in the **preparation area (PA)** and **plates** it in the **serving area (SVA)**.
5. Any used tools are returned to the **dishwashing area (DWA)** for cleaning.

## Predicates:

you can assume the predicates to solve the robotic cleaner planning problem are:

- *robot-at ?r - robot ?loc - location*: The robot is at a specific location.
- *ingredient-at ?ingredient - ingredient ?loc - location*: An ingredient is at a specific location.
- *tool-at ?tool - tool ?loc - location*: A tool is at a specific location.
- *dish-prepared ?dish - dish*: The dish is fully prepared.
- *tool-clean ?tool - tool*: The tool is clean and ready for use.
- *holding ?item - item*: The robot is holding an item (ingredient or tool).
- *adjacent ?loc1 - location ?loc2 - location*: The locations are adjacent.

Using the predicates you defined and the map as depicted in Figure 2, we assume the initial and goal states of the problem as follows:

### Initial states

- The robot starts at CA.
- Rice and fish are available at SA.
- A clean knife is available at CTA.
- The robot needs to prepare a sushi order.

### Goal State

- The **sushi** is fully prepared and plated in **SVA** (e.g., dish-prepared sushi).
- All used tools are cleaned and returned to their initial locations (e.g., tools-clean knife).
- The robot ends at **SVA** after completing the tasks.

## Actions:

In addition, you can assume that actions, which you need to select from them to update the world state are:

- *pick-up-ingredient*: The robot picks up an ingredient from the **storage area**.
- *prepare-ingredient*: The robot prepares an ingredient (e.g., cuts, mixes) in the appropriate area.
- *cook-ingredient*: The robot cooks an ingredient in the **cooking area**.
- *assemble-dish*: The robot assembles a dish from prepared ingredients in the **preparation area**.
- *plate-dish*: The robot plates the finished dish in the **serving area**.
- *clean-tool*: The robot cleans a tool in the **dishwashing area**.
- *move*: The robot moves from its current location to an adjacent one.

<u>Add more actions if you think that will improve the efficiency and optimality of the planning</u>.

## Modelling the planner:

- Implement the planner in PDDL to solve the problem, indicating the internal representation used to manage the preconditions, to check the applicability of the operators, etc. The <domain.pddl> should define the Robot world actions and predicates. The <problem.pddl> contains the objects, init state, and goal state. The output of your planner should be a sequence of actions that solves the given problem.
- Test your code with a set of testing cases of increasing complexity (**a minimum of 3 scenarios**; one of them is the example of Sushi). For instance, you can assume the agent can start from a different area (e.g., PMA) rather than the BTA. Discuss in the document the solutions your code found for these examples, e.g., if the planner can provide an optimal plan. so if the planner fails to find a plan, there might be something wrong with your PDDL definitions or you need to change the planner.
- The algorithm execution output must be clearly printed out in a pdf text file with an explanation about your code and your representation. This file has to clearly display the states that are being generated and evaluated, and the sequence of the actions.

### Documentation content:

1. Introduction to the problem
2. Analysis of the problem (objects, operators, predicates, etc.)
3. PDDL Implementation (A Domain PDDL file + 3 Problem PDDL files)
4. Testing cases and results (show the important steps to arrive to the solution, the final path as well). Analysis of the results (complexity and number of nodes generated and expanded).

### Evaluation criteria:

| Grade | Item |
|---|---|
| **5** | Implementation |
| **2** | execution of the test suit and additional tests |
| **3** | analysis (of the problem and the results obtained) |
| **10** | Total |