Robot Chef Task

PLANNING AND APPROXIMATE REASONING

Joan Caballero Castro
Marc Gonzalez Vidal

Master in Artificial Intelligence
Universitat Politècnica de Catalunya (UPC)
Universitat Rovira i Virgili (URV)

Quadrimester 1, 2024/25

Contents

1 Introduction to the Problem

In this work, we developed a robot chef capable of independently managing a sushi restaurant. This includes tasks such as cooking, cleaning utensils, and delivering dishes. The primary goal is to fulfill and deliver all customer orders efficiently within the restaurant. The robot will receive a list of orders, with each entry detailing the specific dish to be prepared. The restaurant layout is divided into distinct rooms, each assigned a specific function consistent with the roles outlined in the assignment statement.

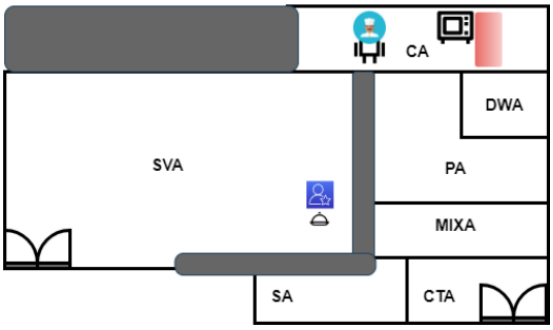

Figure 1: Restaurant Structure

As specified, the robot can move only between adjacent, unblocked areas, meaning there must be no wall separating them. Due to the unclear image, we determined that the only connection to SVA is through CA; no other rooms are linked directly to SVA.

The robot is equipped to pick up, utilize, and transport both ingredients and tools necessary for preparing the dishes. Once a dish is complete, the robot serves it to the customers, at which point the order is considered fulfilled.

1.1 Assumptions

During the programming of this assignment, several assumptions were made, some of which are explicitly stated in the task requirements:

- Orders are received through a digital system, enabling the robot to determine which dishes to prepare.
- The robot is capable of preparing only one dish at a time.
- The robot can carry only one item at a time.
- Tools must be cleaned prior to being used for a different dish.
- All tools must be returned to their original positions upon the completion of all orders.

- In order to manipulate an ingredient, both the ingredient and the necessary tool must be present in the same location as the robot.
- To cut and mix ingredients, the robot must have the appropriate tool in hand.
- For both cooking and assembling dishes, the robot cannot be holding any items.
- There is no order between cutting, mixing, and cooking the ingredients.
- To wash a tool, the robot has to drop it in the washing area and clean it with its hands empty.

2 Analysis of the Problem

2.1 Objects

The following objects represent the entities in the environment that are essential for completing the tasks.

- Location: Represents the distinct areas within the restaurant, each with a unique purpose.
  - Cooking Area (CA): Area designated for cooking ingredients.
  - Serving Area (SVA): Area where customers wait, where completed dishes must be served.
  - Dish-washing Area (DWA): Area for cleaning used tools.
  - Preparation Area (PA): Area for assembling dishes, readying them for delivery.
  - Mixing Area (MIXA): Area designated for mixing ingredients.
  - Cutting Area (CTA): Area where ingredients are cut.
  - Storage Area (SA): Area for storing ingredients.
- Item: Represents any tangible object that can be picked up at some point during the process.
  - Ingredient: Represents a food item used in dish preparation.
  - Dish: A combination of ingredients, prepared according to a specific process, which can be served to a customer once ready.
  - Tool: Represents an implement used to alter ingredients during preparation, with a state that can be either dirty or clean.
    - Cutter: Tool that can be used for cutting ingredients.
    - Mixer: Tool that can be used for mixing ingredients.
    - Cooker: Tool that can be used for cooking ingredients.

2.2 Search Space

To accurately calculate the search space, we need to consider the various combinations of possible states that can arise based on our problem representation. Each state in our problem is defined by the following factors: the total number of locations (L), the total number of ingredients (ING), the number of tools (T), the number of dishes (D), and the number of items ($I = ING + T + D$).

We will analyze these components step by step to define the final search space formula, which provides the total number of states our system can have.

First, the robot can be located at any of the locations. This means the robot can occupy any of the L locations. Thus, the number of possible states for the robot's location is:

$$L$$

Next, the robot can be holding any one of the items (ingredient, tool, or dish) or nothing at all. Since the robot can hold at most one item at a time, there are $I + 1$ possibilities (with the extra 1 representing the state where the robot is not holding anything). Therefore, the number of possible holding states for the robot is:

$$I + 1$$

Now, we need to consider the possible locations of each item. Each item can either be at one of the locations or be held by the robot. Since the robot can hold at most one item at a time, we must account for this constraint in our calculation. When the robot is not holding any item, all I items are at one of the L locations, leading to L^I. When the robot is holding one item, there are I choices for which item is being held, and the remaining $I - 1$ items can be at any of the L locations. This results in $I \times L^{I-1}$. Thus, the total number of possible configurations for the items is:

$$L^I + I \times L^{I-1}$$

Next, each tool can be either clean or not clean. Since there are T tools, each with two possible states, the total number of possible tool status configurations is:

$$2^T$$

For the ingredients, each has four status variables: cut, mixed, cooked, and ready. However, the state where an ingredient is cut, mixed, and cooked but not ready is not possible, as an ingredient becomes ready when all required processes are completed. This gives each ingredient $2^4 - 1 = 15$ possible states. Therefore, for all ING ingredients, the total number of ingredient status combinations is:

$$15^{ING}$$

Similarly, each dish has two status variables: prepared and served. A dish cannot be served without being prepared first, so the state where a dish is served but not prepared is impossible. This reduces the total possible states per dish to 3 valid states. Therefore, for all D dishes, the total number of dish status combinations is:

$$3^D$$

Regarding orders, since only one order can be processed at a time, there are $D + 1$ possibilities (either no order is being processed or one of the D dishes

is being processed).Additionally, each dish can have its order processed or not, resulting in 2^D combinations for the order-processed status. Therefore, the total number of order status configurations is:

$$(D+1) \times 2^D$$

Combining all these components, the total search space is the product of these factors, representing the robot's location, the possible configurations of items, the statuses of tools, ingredients, and dishes. The complete formula for the search space is:

$$S = L \times (I+1) \times (L^I + I \times L^{I-1}) \times 2^T \times 15^{ING} \times 3^D \times (D+1) \times 2^D$$

2.3 Operators

The following operators define the actions that the robot can perform to accomplish the task of preparing and serving dishes.

- **move (?from - Location ?to - Location):** This action moves the robot from its current location (`from`) to an adjacent location (`to`). The robot must be at the starting location and the target location must be adjacent.
- **pick-up-item (?i - Item ?l - Location):** This action allows the robot to pick up an item (`i`) from a specific location (`l`). The robot and the item must be at the same location, and the robot must not already be holding an item. After execution, the item is no longer at the location, and the robot is holding it.
- **drop-item (?i - Item ?l - Location):** This action allows the robot to drop an item (`i`) at a specified location (`l`). The robot must be holding the item and be at the location. After dropping the item, the robot is no longer holding it, and the item is marked as present at the location.
- **clean-tool (?t - Tool ?l - DWA):** This action cleans a tool (`t`) at a designated work area (`l`). The robot must be at the tool's location. After execution, the tool is marked as clean.
- **cut-ingredient (?i - Ingredient ?t - Cutter ?l - CTA):** This action allows the robot to cut an ingredient (`i`) using a cutting tool (`t`) at a cutting area (`l`). The robot must be holding the cutter, be at the location, and the ingredient must be in need of cutting. After execution, the ingredient is marked as cut and no longer in need of cutting. Additionally, the tool is marked as needing cleaning.
- **mix-ingredient (?i - Ingredient ?t - Mixer ?l - MIXA):** This action allows the robot to mix an ingredient (`i`) using a mixing tool (`t`) at a mixing area (`l`). The robot must be holding the mixer, be at the location, and the ingredient must need mixing. After execution, the ingredient is

marked as mixed and no longer in need of mixing, and the tool requires cleaning.

- **cook-ingredient (?i - Ingredient ?t - Cooker ?l - CA):** This action allows the robot to cook an ingredient (`i`) using a cooking appliance (`t`) at a cooking area (`l`). The robot must not be holding any items, the ingredient and cooker must be at the location, and the ingredient must need cooking. After execution, the ingredient is marked as cooked, no longer needing cooking, and the tool is marked as dirty.
- **check-prepared (?i - Ingredient ?d - Dish):** This action verifies that an ingredient (`i`) in a dish (`d`) is ready for use. The ingredient is ready if it does not need cutting, mixing, or cooking, or if those steps have already been completed.
- **assemble-dish (?d - Dish ?l - PA):** This action assembles a dish (`d`) at the preparation area (`l`). The robot must be at the location and not holding any item. All ingredients required for the dish must be ready and present at the location. After assembly, the dish is marked as prepared and present at the location, and any ingredients are no longer marked at the location, in other words, they are not longer to be used again.
- **serve-dish (?d - Dish ?l - SVA):** This action serves a prepared dish (`d`) at the serving area (`l`). The robot must be at the location, holding the dish, and the dish must be prepared. After execution, the dish is marked as served, and the robot is no longer holding it.
- **start-order (?d - Dish):** This action initiates the preparation process for a dish (`d`). No other orders should be in progress. After execution, the system is marked as processing the specified order.
- **end-order (?d - Dish):** This action completes an order for a dish (`d`). The dish must have been served, and all tools involved must be clean and at their initial locations. After execution, the order is marked as processed, and the system is no longer processing any order.

2.4 Predicates

The following predicates represent the various states and requirements of the robot, items, tools, and ingredients, as well as the relationships between locations and dish preparation.

- **robot-at (?l - Location):** Indicates that the robot is currently located at `l`.
- **holding (?i - Item):** Specifies that the robot is holding item `i`.
- **holding-any:** Denotes that the robot is currently holding any item.

- **item-at (?i - Item ?l - Location):** States that item `i` is located at location `l`.
- **tool-clean (?t - Tool):** Indicates that tool `t` is clean.
- **initial-tool-loc (?t - Tool ?l - Location):** Specifies the initial location `l` of tool `t`.
- **processing-order (?d - Dish):** Indicates that an order for dish `d` is currently being processed.
- **processing-any-order:** Denotes that any order is currently being processed.
- **order-processed (?d - Dish):** Specifies that the order for dish `d` has been completed.
- **dish-prepared (?d - Dish):** Indicates that dish `d` is prepared and ready to be served.
- **dish-served (?d - Dish):** Denotes that dish `d` has been served.
- **ingredient-in-dish (?d - Dish ?i - Ingredient):** Specifies that ingredient `i` is part of dish `d`.
- **adjacent (?l1 - Location ?l2 - Location):** States that locations `l1` and `l2` are adjacent to each other, allowing movement between them.
- **ingredient-cut (?i - Ingredient):** Denotes that ingredient `i` has been cut.
- **ingredient-mixed (?i - Ingredient):** Indicates that ingredient `i` has been mixed.
- **ingredient-cooked (?i - Ingredient):** Specifies that ingredient `i` has been cooked.
- **ingredient-ready (?i - Ingredient):** Denotes that ingredient `i` is ready for use, having met all preparation requirements.
- **needs-cutting (?i - Ingredient):** States that ingredient `i` needs to be cut as part of its preparation.
- **needs-mixing (?i - Ingredient):** Specifies that ingredient `i` needs to be mixed.
- **needs-cooking (?i - Ingredient):** Indicates that ingredient `i` requires cooking.

3 PDDL Implementation

In this section, we present the PDDL code for the domain and problem files. We implemented a base domain file along with the first three corresponding problems, and an extended domain file along with problems 4 and 5.

3.1 Domain

This domain implements all the core functionalities that we have described throughout the document.

```
(define (domain robot-chef)

(:requirements :typing :strips :fluents :adl)

(:types
    Location Item - Object
    Ingredient Tool Dish - Item
    Cutter Mixer Cooker - Tool
    CA SVA DWA PA MIXA CTA SA - Location
)

(:predicates
    ;; Robot Status
    (robot-at ?l - Location)
    (holding ?i - Item)
    (holding-any)

    ;; Item Locations
    (item-at ?i - Item ?l - Location)

    ;; Tool Status & Locations
    (tool-clean ?t - Tool)
    (initial-tool-loc ?t - Tool ?l - Location)

    ;; Order Status
    (processing-order ?d - Dish)
    (processing-any-order)
    (order-processed ?d - Dish)

    ;; Dish Status & Requirements
    (dish-prepared ?d - Dish)
    (dish-served ?d - Dish)
    (ingredient-in-dish ?d - Dish ?i - Ingredient)
```

```
    ;; Adjacent Locations
    (adjacent ?l1 - Location ?l2 - Location)

    ;; Ingredient States
    (ingredient-cut ?i - Ingredient)
    (ingredient-mixed ?i - Ingredient)
    (ingredient-cooked ?i - Ingredient)
    (ingredient-ready ?i - Ingredient)

    ;; Ingredient Needs
    (needs-cutting ?i - Ingredient)
    (needs-mixing ?i - Ingredient)
    (needs-cooking ?i - Ingredient)
)

(:action move
    :parameters (?from - Location ?to - Location)
    :precondition (and (robot-at ?from) (adjacent ?from ?to))
    :effect (and (not (robot-at ?from)) (robot-at ?to))
)

(:action pick-up-item
    :parameters (?i - Item ?l - Location)
    :precondition (and (robot-at ?l) (item-at ?i ?l) (not
        (holding-any)))
    :effect (and (not (item-at ?i ?l)) (holding ?i) (holding-any))
)

(:action drop-item
    :parameters (?i - Item ?l - Location)
    :precondition (and (robot-at ?l) (holding ?i))
    :effect (and (item-at ?i ?l) (not (holding ?i)) (not
        (holding-any)))
)

(:action clean-tool
    :parameters (?t - Tool ?l - DWA)
    :precondition (and (robot-at ?l) (item-at ?t ?l))
    :effect (tool-clean ?t)
)

(:action cut-ingredient
    :parameters (?i - Ingredient ?t - Cutter ?l - CTA)
    :precondition (and (robot-at ?l) (holding ?t) (item-at ?i ?l)
        (needs-cutting ?i))
    :effect (and (ingredient-cut ?i) (not (needs-cutting ?i)) (not
        (tool-clean ?t)))
)
```

```
(:action mix-ingredient
    :parameters (?i - Ingredient ?t - Mixer ?l - MIXA)
    :precondition (and (robot-at ?l) (holding ?t) (item-at ?i ?l)
        (needs-mixing ?i))
    :effect (and (ingredient-mixed ?i) (not (needs-mixing ?i)) (not
        (tool-clean ?t)))
)

(:action cook-ingredient
    :parameters (?i - Ingredient ?t - Cooker ?l - CA)
    :precondition (and (robot-at ?l) (not (holding-any)) (item-at ?i
        ?l) (item-at ?t ?l) (needs-cooking ?i))
    :effect (and (ingredient-cooked ?i) (not (needs-cooking ?i))
        (not (tool-clean ?t)))
)

(:action check-prepared
    :parameters (?i - Ingredient ?d - Dish)
    :precondition (and (ingredient-in-dish ?d ?i)
       (or (not (needs-cutting ?i)) (ingredient-cut ?i))
       (or (not (needs-mixing ?i)) (ingredient-mixed ?i))
       (or (not (needs-cooking ?i)) (ingredient-cooked ?i))
    )
    :effect (ingredient-ready ?i)
)

(:action assemble-dish
    :parameters (?d - Dish ?l - PA)
    :precondition (and (robot-at ?l) (not (holding-any))
                  (forall (?i - Ingredient)
                      (imply
                          (ingredient-in-dish ?d ?i)
                          (and (ingredient-ready ?i) (item-at ?i ?l))
                      )
                  )
    )
    :effect (and (dish-prepared ?d) (item-at ?d ?l)
           (forall (?i - Ingredient)
               (when
                   (ingredient-in-dish ?d ?i)
                   (not (item-at ?i ?l))
               )
           )
    )
)
```

```
(:action serve-dish
    :parameters (?d - Dish ?l - SVA)
    :precondition (and (robot-at ?l) (holding ?d) (dish-prepared ?d))
    :effect (and (dish-served ?d) (not (holding ?d)) (not
        (holding-any)))
)

(:action start-order
    :parameters (?d - Dish)
    :precondition (and (not (processing-any-order)))
    :effect (and (processing-any-order) (processing-order ?d))
)

(:action end-order
    :parameters (?d - Dish)
    :precondition (and (processing-order ?d) (dish-served ?d)
       (forall (?t - Tool)
           (and (tool-clean ?t)
               (exists (?l - Location)
                   (and (initial-tool-loc ?t ?l) (item-at ?t ?l))
               )
           )
       )
    )
    :effect (and (not (processing-order ?d)) (not
        (processing-any-order)) (order-processed ?d))
)
)
```

3.2 Problem 1

```
(define (problem problem1) (:domain robot-chef)

(:objects
    sushi - Dish
    fish seaweed rice - Ingredient
    knife - Cutter
    gloves - Mixer
    pot - Cooker

    ca - CA
    sva - SVA
    dwa - DWA
    pa - PA
```

```
    mixa - MIXA
    cta - CTA
    sa - SA
)

(:init
    (robot-at ca)

    (adjacent ca sva) (adjacent sva ca) (adjacent ca dwa) (adjacent
        dwa ca)
    (adjacent ca pa) (adjacent pa ca) (adjacent pa dwa) (adjacent
        dwa pa)
    (adjacent pa mixa) (adjacent mixa pa) (adjacent mixa cta)
        (adjacent cta mixa)
    (adjacent sa cta) (adjacent cta sa) (adjacent mixa sa) (adjacent
        sa mixa)

    ; -- Tools --
    (item-at knife cta) (item-at gloves mixa) (item-at pot ca)
    (initial-tool-loc knife cta) (initial-tool-loc gloves mixa)
        (initial-tool-loc pot ca)
    (tool-clean knife) (tool-clean gloves) (tool-clean pot)

    ; -- Sushi --
    (item-at fish sa) (item-at seaweed sa) (item-at rice sa)

    (ingredient-in-dish sushi fish)
    (ingredient-in-dish sushi seaweed)
    (ingredient-in-dish sushi rice)

    (needs-cutting fish)
    (needs-cutting seaweed)
    (needs-mixing rice) (needs-cooking rice)
)

(:goal (forall (?d - Dish) (order-processed ?d)))
)
```

3.3 Problem 2

```
(define (problem problem2) (:domain robot-chef)

(:objects
    sushi ramen - Dish
    fish seaweed rice noodles broth vegetables meat milk eggs -
        Ingredient
    knife - Cutter
    gloves - Mixer
    pot - Cooker

    ca - CA
    sva - SVA
    dwa - DWA
    pa - PA
    mixa - MIXA
    cta - CTA
    sa - SA
)

(:init
    (robot-at ca)

    (adjacent ca sva) (adjacent sva ca) (adjacent ca dwa) (adjacent
        dwa ca)
    (adjacent ca pa) (adjacent pa ca) (adjacent pa dwa) (adjacent
        dwa pa)
    (adjacent pa mixa) (adjacent mixa pa) (adjacent mixa cta)
        (adjacent cta mixa)
    (adjacent sa cta) (adjacent cta sa) (adjacent mixa sa) (adjacent
        sa mixa)

    ; -- Tools --
    (item-at knife cta) (item-at gloves mixa) (item-at pot ca)
    (initial-tool-loc knife cta) (initial-tool-loc gloves mixa)
        (initial-tool-loc pot ca)
    (tool-clean knife) (tool-clean gloves) (tool-clean pot)

    ; -- Sushi --
    (item-at fish sa) (item-at seaweed sa) (item-at rice sa)

    (ingredient-in-dish sushi fish)
    (ingredient-in-dish sushi seaweed)
    (ingredient-in-dish sushi rice)

    (needs-cutting fish)
```

```
    (needs-cutting seaweed)
    (needs-mixing rice) (needs-cooking rice)

    ; -- Ramen --
    (item-at noodles sa) (item-at broth sa) (item-at vegetables sa)

    (ingredient-in-dish ramen noodles)
    (ingredient-in-dish ramen broth)
    (ingredient-in-dish ramen vegetables)

    (needs-cooking noodles)
    (needs-cooking broth)
    (needs-cutting vegetables)

    ; -- Other Ingredients --
    (item-at meat sa) (item-at milk sa) (item-at eggs sa)
)

(:goal (forall (?d - Dish) (order-processed ?d)))
)
```

3.4 Problem 3

```
(define (problem problem3) (:domain robot-chef)

(:objects
    sushi ramen curry_rice - Dish
    fish seaweed rice1 rice2 chicken curry noodles broth vegetables
        meat milk eggs - Ingredient
    knife - Cutter
    gloves - Mixer
    pot - Cooker

    ca - CA
    sva - SVA
    dwa - DWA
    pa - PA
    mixa - MIXA
    cta - CTA
    sa - SA
)

(:init
    (robot-at ca)
```

```
    (adjacent ca sva) (adjacent sva ca) (adjacent ca dwa) (adjacent
        dwa ca)
    (adjacent ca pa) (adjacent pa ca) (adjacent pa dwa) (adjacent
        dwa pa)
    (adjacent pa mixa) (adjacent mixa pa) (adjacent mixa cta)
        (adjacent cta mixa)
    (adjacent sa cta) (adjacent cta sa) (adjacent mixa sa) (adjacent
        sa mixa)

    ; -- Tools --
    (item-at knife cta) (item-at gloves mixa) (item-at pot ca)
    (initial-tool-loc knife cta) (initial-tool-loc gloves mixa)
        (initial-tool-loc pot ca)
    (tool-clean knife) (tool-clean gloves) (tool-clean pot)

    ; -- Sushi --
    (item-at fish sa) (item-at seaweed sa) (item-at rice1 sa)

    (ingredient-in-dish sushi fish)
    (ingredient-in-dish sushi seaweed)
    (ingredient-in-dish sushi rice1)

    (needs-cutting fish)
    (needs-cutting seaweed)
    (needs-mixing rice1) (needs-cooking rice1)

    ; -- Ramen --
    (item-at noodles sa) (item-at broth sa) (item-at vegetables sa)

    (ingredient-in-dish ramen noodles)
    (ingredient-in-dish ramen broth)
    (ingredient-in-dish ramen vegetables)

    (needs-cooking noodles)
    (needs-cooking broth)
    (needs-cutting vegetables)

    ; -- Curry Rice --
    (item-at chicken sa) (item-at rice2 sa) (item-at curry sa)

    (ingredient-in-dish curry_rice chicken)
    (ingredient-in-dish curry_rice rice2)
    (ingredient-in-dish curry_rice curry)

    (needs-cutting chicken) (needs-cooking chicken)
    (needs-mixing rice2) (needs-cooking rice2)
    (needs-cooking curry)

    ; -- Other Ingredients --
    (item-at meat sa) (item-at milk sa) (item-at eggs sa)
)
(:goal (forall (?d - Dish) (order-processed ?d)))
)
```

3.5 Domain (Extra)

This domain implements additional functionalities, including multiple robots and tool durability. The sushi restaurant now operates with several robots, but they must be cautious, as each tool has a set durability that once reached, causes the tool to break and become unusable.

To implement these features, we updated the base domain by adding and modifying several predicates, functions, and actions.

```
(define (domain robot-chef)

(:requirements :typing :strips :fluents :adl)

(:types
    Location Item Robot - Object
    ...
)

(:functions
    (tool-durability ?t - Tool) ; Number of uses remaining for tool
        ?t
)

(:predicates
    ;; Robot Status
    (robot-at ?r - Robot ?l - Location)
    (holding ?r - Robot ?i - Item)
    (holding-any ?r - Robot)
    ...

    ;; Tool Status & Locations
    ...
    (tool-discarded ?t - Tool) ; Tool ?t has been discarded
    (tool-used-by ?t - Tool ?d - Dish) ; Tool ?t has been used by
        dish ?d

    ...
)

(:action move
    :parameters (?r - Robot ?from - Location ?to - Location)
    :precondition (and (robot-at ?r ?from) (adjacent ?from ?to))
    :effect (and (not (robot-at ?r ?from)) (robot-at ?r ?to))
)
```

```
(:action pick-up-item
    :parameters (?r - Robot ?i - Item ?l - Location)
    :precondition (and (robot-at ?r ?l) (item-at ?i ?l) (not
        (holding-any ?r)))
    :effect (and (not (item-at ?i ?l)) (holding ?r ?i) (holding-any
        ?r))
)

(:action drop-item
    :parameters (?r - Robot ?i - Item ?l - Location)
    :precondition (and (robot-at ?r ?l) (holding ?r ?i))
    :effect (and (item-at ?i ?l) (not (holding ?r ?i)) (not
        (holding-any ?r)))
)

(:action clean-tool
    :parameters (?r - Robot ?t - Tool ?l - DWA)
    :precondition (and (robot-at ?r ?l) (item-at ?t ?l))
    :effect (and (tool-clean ?t) (forall (?d - Dish) (not
        (tool-used-by ?t ?d))))
)

(:action cut-ingredient
    :parameters (?r - Robot ?i - Ingredient ?t - Cutter ?d - Dish ?l
        - CTA)
    :precondition (and (robot-at ?r ?l) (holding ?r ?t) (item-at ?i
        ?l) (needs-cutting ?i) (> (tool-durability ?t) 0)
        (ingredient-in-dish ?d ?i) (or (tool-clean ?t)
        (tool-used-by ?t ?d)))
    :effect (and (ingredient-cut ?i) (not (needs-cutting ?i)) (not
        (tool-clean ?t)) (tool-used-by ?t ?d) (decrease
        (tool-durability ?t) 1))
)

(:action mix-ingredient
    :parameters (?r - Robot ?i - Ingredient ?t - Mixer ?d - Dish ?l
        - MIXA)
    :precondition (and (robot-at ?r ?l) (holding ?r ?t) (item-at ?i
        ?l) (needs-mixing ?i) (> (tool-durability ?t) 0)
        (ingredient-in-dish ?d ?i) (or (tool-clean ?t)
        (tool-used-by ?t ?d)))
    :effect (and (ingredient-mixed ?i) (not (needs-mixing ?i)) (not
        (tool-clean ?t)) (tool-used-by ?t ?d) (decrease
        (tool-durability ?t) 1))
)
```

```
(:action cook-ingredient
    :parameters (?r - Robot ?i - Ingredient ?t - Cooker ?d - Dish ?l
        - CA)
    :precondition (and (robot-at ?r ?l) (not (holding-any ?r))
        (item-at ?i ?l) (item-at ?t ?l) (needs-cooking ?i) (>
        (tool-durability ?t) 0) (ingredient-in-dish ?d ?i) (or
        (tool-clean ?t) (tool-used-by ?t ?d)))
    :effect (and (ingredient-cooked ?i) (not (needs-cooking ?i))
        (not (tool-clean ?t)) (tool-used-by ?t ?d) (decrease
        (tool-durability ?t) 1))
)

(:action check-prepared
    :parameters (?i - Ingredient ?d - Dish)
    :precondition (and (ingredient-in-dish ?d ?i)
       (or (not (needs-cutting ?i)) (ingredient-cut ?i))
       (or (not (needs-mixing ?i)) (ingredient-mixed ?i))
       (or (not (needs-cooking ?i)) (ingredient-cooked ?i))
    )
    :effect (ingredient-ready ?i)
)

(:action assemble-dish
    :parameters (?r - Robot ?d - Dish ?l - PA)
    :precondition (and (robot-at ?r ?l) (not (holding-any ?r))
                  (forall (?i - Ingredient)
                      (imply
                          (ingredient-in-dish ?d ?i)
                          (and (ingredient-ready ?i) (item-at ?i ?l))
                      )
                  )
    )
    :effect (and (dish-prepared ?d) (item-at ?d ?l)
           (forall (?i - Ingredient)
               (when
                   (ingredient-in-dish ?d ?i)
                   (not (item-at ?i ?l))
               )
           )
    )
)

(:action serve-dish
    :parameters (?r - Robot ?d - Dish ?l - SVA)
    :precondition (and (robot-at ?r ?l) (holding ?r ?d)
        (dish-prepared ?d))
    :effect (and (dish-served ?d) (not (holding ?r ?d)) (not
        (holding-any ?r)))
)
```

```
(:action start-order
    :parameters (?d - Dish)
    :effect (processing-order ?d)
)

(:action end-order
    :parameters (?d - Dish)
    :precondition (and (processing-order ?d) (dish-served ?d)
        (forall (?t - Tool)
            (or (tool-discarded ?t)
                (and (tool-clean ?t)
                    (exists (?l - Location)
                        (and (initial-tool-loc ?t ?l) (item-at ?t ?l))
                    )
                )
            )
        )
    )
    :effect (and (not (processing-order ?d)) (order-processed ?d))
)

(:action throw-tool
    :parameters (?r - Robot ?t - Tool)
    :precondition (and (holding ?r ?t) (= (tool-durability ?t) 0))
    :effect (and (not (holding ?r ?t)) (not (holding-any ?r))
        (tool-discarded ?t))
)
)
```

3.6 Problem 4 (Extra)

```
(define (problem problem4) (:domain robot-chef-plus)

(:objects
    r1 r2 - Robot
    sushi ramen - Dish
    fish seaweed rice noodles broth vegetables meat milk eggs -
        Ingredient
    knife1 knife2 - Cutter
    gloves - Mixer
    pot1 pot2 - Cooker

    ca - CA
    sva - SVA
```

```
    dwa - DWA
    pa - PA
    mixa - MIXA
    cta - CTA
    sa - SA
)

(:init
    (robot-at r1 ca) (robot-at r2 sa)

    (adjacent ca sva) (adjacent sva ca) (adjacent ca dwa) (adjacent
        dwa ca)
    (adjacent ca pa) (adjacent pa ca) (adjacent pa dwa) (adjacent
        dwa pa)
    (adjacent pa mixa) (adjacent mixa pa) (adjacent mixa cta)
        (adjacent cta mixa)
    (adjacent sa cta) (adjacent cta sa) (adjacent mixa sa) (adjacent
        sa mixa)

    ; -- Tools --
    (item-at knife1 cta) (item-at knife2 cta) (item-at gloves mixa)
        (item-at pot1 ca) (item-at pot2 ca)
    (initial-tool-loc knife1 cta) (initial-tool-loc knife2 cta)
        (initial-tool-loc gloves mixa) (initial-tool-loc pot1 ca)
        (initial-tool-loc pot2 ca)
    (tool-clean knife1) (tool-clean knife2) (tool-clean gloves)
        (tool-clean pot1) (tool-clean pot2)

    (= (tool-durability knife1) 2) (= (tool-durability knife2) 2) (=
        (tool-durability gloves) 3)
    (= (tool-durability pot1) 2) (= (tool-durability pot2) 2)

    ; -- Sushi --
    (item-at fish sa) (item-at seaweed sa) (item-at rice sa)

    (ingredient-in-dish sushi fish)
    (ingredient-in-dish sushi seaweed)
    (ingredient-in-dish sushi rice)

    (needs-cutting fish)
    (needs-cutting seaweed)
    (needs-mixing rice) (needs-cooking rice)

    ; -- Ramen --
    (item-at noodles sa) (item-at broth sa) (item-at vegetables sa)

    (ingredient-in-dish ramen noodles)
    (ingredient-in-dish ramen broth)
    (ingredient-in-dish ramen vegetables)
```

```
    (needs-cooking noodles)
    (needs-cooking broth)
    (needs-cutting vegetables)

    ; -- Other Ingredients --
    (item-at meat sa) (item-at milk sa) (item-at eggs sa)
)

(:goal (forall (?d - Dish) (order-processed ?d)))
)
```

3.7 Problem 5 (Extra)

```
(define (problem problem5) (:domain robot-chef-plus)

(:objects
    r1 r2 r3 - Robot
    sushi ramen curry_rice - Dish
    fish seaweed rice1 rice2 chicken curry noodles broth vegetables
        meat milk eggs - Ingredient
    knife1 knife2 knife3 - Cutter
    gloves - Mixer
    pot1 pot2 - Cooker

    ca - CA
    sva - SVA
    dwa - DWA
    pa - PA
    mixa - MIXA
    cta - CTA
    sa - SA
)

(:init
    (robot-at r1 ca) (robot-at r2 pa) (robot-at r3 sa)
    (adjacent ca sva) (adjacent sva ca) (adjacent ca dwa) (adjacent
        dwa ca)
    (adjacent ca pa) (adjacent pa ca) (adjacent pa dwa) (adjacent
        dwa pa)
    (adjacent pa mixa) (adjacent mixa pa) (adjacent mixa cta)
        (adjacent cta mixa)
    (adjacent sa cta) (adjacent cta sa) (adjacent mixa sa) (adjacent
        sa mixa)
```

```
    ; -- Tools --
    (item-at knife1 cta) (item-at knife2 cta) (item-at knife3 cta)
        (item-at gloves mixa) (item-at pot1 ca) (item-at pot2 ca)
    (initial-tool-loc knife1 cta) (initial-tool-loc knife2 cta)
        (initial-tool-loc knife3 cta) (initial-tool-loc gloves
        mixa) (initial-tool-loc pot1 ca) (initial-tool-loc pot2 ca)
    (tool-clean knife1) (tool-clean knife2) (tool-clean knife3)
        (tool-clean gloves) (tool-clean pot1) (tool-clean pot2)

    (= (tool-durability knife1) 2) (= (tool-durability knife2) 1) (=
        (tool-durability knife3) 1)
    (= (tool-durability gloves) 2)
    (= (tool-durability pot1) 4) (= (tool-durability pot2) 3)

    ; -- Sushi --
    (item-at fish sa) (item-at seaweed sa) (item-at rice1 sa)

    (ingredient-in-dish sushi fish)
    (ingredient-in-dish sushi seaweed)
    (ingredient-in-dish sushi rice1)

    (needs-cutting fish)
    (needs-cutting seaweed)
    (needs-mixing rice1) (needs-cooking rice1)

    ; -- Ramen --
    (item-at noodles sa) (item-at broth sa) (item-at vegetables sa)

    (ingredient-in-dish ramen noodles)
    (ingredient-in-dish ramen broth)
    (ingredient-in-dish ramen vegetables)

    (needs-cooking noodles)
    (needs-cooking broth)
    (needs-cutting vegetables)

    ; -- Curry Rice --
    (item-at chicken sa) (item-at rice2 sa) (item-at curry sa)

    (ingredient-in-dish curry_rice chicken)
    (ingredient-in-dish curry_rice rice2)
    (ingredient-in-dish curry_rice curry)

    (needs-cutting chicken) (needs-cooking chicken)
    (needs-mixing rice2) (needs-cooking rice2)
    (needs-cooking curry)

    ; -- Other Ingredients --
    (item-at meat sa) (item-at milk sa) (item-at eggs sa)
)

(:goal (forall (?d - Dish) (order-processed ?d)))
```

4 Testing Cases

In this section, we will outline all test cases conducted. It is important to note that the layout of the areas is fixed; the restaurant's architecture remains consistent as described in the problem introduction.

4.1 Test Case 1

Description

In this initial test case, we aim to evaluate the simplest possible scenario: preparing a single dish, specifically the standard dish served in the restaurant—a sushi dish.

To prepare sushi, the ingredients required are fish, seaweed, and rice, each needing specific preparation steps. The fish and seaweed must be cut, while the rice needs to be cooked and mixed. For cutting, the robot requires a knife; for mixing, gloves are needed; and for cooking the rice, a pot is necessary. Each tool will be located in its designated area (e.g., the knife in the cutting area).

The objective of this test case is to verify that all constraints in our problem are respected, that actions are executed in the correct sequence, and that each action is carried out in its designated area.

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the robot to achieve the goal.

```
step  0: START-ORDER SUSHI
      1: MOVE CA PA
      2: MOVE PA MIXA
      3: MOVE MIXA SA
      4: PICK-UP-ITEM FISH SA
      5: MOVE SA CTA
      6: DROP-ITEM FISH CTA
      7: MOVE CTA SA
      8: PICK-UP-ITEM SEAWEED SA
      9: MOVE SA CTA
     10: DROP-ITEM SEAWEED CTA
     11: MOVE CTA SA
     12: PICK-UP-ITEM RICE SA
     13: MOVE SA CTA
     14: MOVE CTA MIXA
     15: DROP-ITEM RICE MIXA
     16: MOVE MIXA CTA
```

```
17: PICK-UP-ITEM KNIFE CTA
18: CUT-INGREDIENT FISH KNIFE
    SUSHI CTA
19: CUT-INGREDIENT SEAWEED
    KNIFE SUSHI CTA
20: CHECK-PREPARED FISH SUSHI
21: CHECK-PREPARED SEAWEED
    SUSHI
22: DROP-ITEM KNIFE CTA
23: PICK-UP-ITEM FISH CTA
24: MOVE CTA MIXA
25: MOVE MIXA PA
26: DROP-ITEM FISH PA
27: MOVE PA MIXA
28: MOVE MIXA CTA
29: PICK-UP-ITEM SEAWEED CTA
30: MOVE CTA MIXA
31: MOVE MIXA PA
32: DROP-ITEM SEAWEED PA
33: MOVE PA MIXA
34: MOVE MIXA CTA
35: PICK-UP-ITEM KNIFE CTA
36: MOVE CTA MIXA
37: MOVE MIXA PA
38: MOVE PA DWA
39: DROP-ITEM KNIFE DWA
40: CLEAN-TOOL KNIFE DWA
41: PICK-UP-ITEM KNIFE DWA
42: MOVE DWA CA
43: MOVE CA PA
44: MOVE PA MIXA
45: MOVE MIXA CTA
46: DROP-ITEM KNIFE CTA
47: MOVE CTA MIXA
48: PICK-UP-ITEM GLOVES MIXA
```

```
49: MIX-INGREDIENT RICE
    GLOVES SUSHI MIXA
50: MOVE MIXA PA
51: MOVE PA DWA
52: DROP-ITEM GLOVES DWA
53: CLEAN-TOOL GLOVES DWA
54: PICK-UP-ITEM GLOVES DWA
55: MOVE DWA PA
56: MOVE PA MIXA
57: DROP-ITEM GLOVES MIXA
58: PICK-UP-ITEM RICE MIXA
59: MOVE MIXA PA
60: MOVE PA CA
61: DROP-ITEM RICE CA
62: COOK-INGREDIENT RICE POT
    SUSHI CA
63: PICK-UP-ITEM RICE CA
64: CHECK-PREPARED RICE SUSHI
65: MOVE CA PA
66: DROP-ITEM RICE PA
67: ASSEMBLE-DISH SUSHI PA
68: PICK-UP-ITEM SUSHI PA
69: MOVE PA CA
70: MOVE CA SVA
71: SERVE-DISH SUSHI SVA
72: MOVE SVA CA
73: PICK-UP-ITEM POT CA
74: MOVE CA DWA
75: DROP-ITEM POT DWA
76: CLEAN-TOOL POT DWA
77: PICK-UP-ITEM POT DWA
78: MOVE DWA CA
79: DROP-ITEM POT CA
80: END-ORDER SUSHI
```

Analysis

As demonstrated, the plan adheres to all specified descriptions, indicating that our domain is well-defined. The robot begins by retrieving the fish, which is placed in the cutting area. It then follows the same procedure for the seaweed and rice, with the rice being handled in the mixing area. The initial step involves placing all ingredients in their respective preparation zones.

Subsequently, the robot picks up the knife and proceeds to cut both the fish and the seaweed, which can be done without cleaning the tool, as both ingredients

are part of the same dish. After cutting, the fish and seaweed are moved to the preparation area. The robot then cleans the knife and returns it to the cutting area, its initial location.

Next, the robot puts on gloves to mix the rice; once mixing is complete, the gloves are washed and returned to their original place. The rice is then cooked and transferred to the preparation area. Once all ingredients are gathered, the robot assembles the dish and serves it in the serving area. Finally, the robot cleans the pot and returns it to its designated area, completing the order.

4.2 Test Case 2

Description

In this second scenario, we will keep the core aspects of the first test case but add some extra complexity. Here, customers are ordering both a sushi and a ramen dish. For the ramen, additional ingredients will be required. The robot will need to manage the preparation of both dishes in sequence, ensuring that all steps are followed for each order while maintaining cleanliness and proper organization of tools and ingredients.

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the robot to achieve the goal.

```
step  0: START-ORDER RAMEN
      1: MOVE CA PA
      2: MOVE PA MIXA
      3: MOVE MIXA SA
      4: PICK-UP-ITEM VEGETABLES SA
      5: MOVE SA CTA
      6: DROP-ITEM VEGETABLES CTA
      7: MOVE CTA SA
      8: PICK-UP-ITEM FISH SA
      9: MOVE SA CTA
     10: DROP-ITEM FISH CTA
     11: MOVE CTA SA
     12: PICK-UP-ITEM SEAWEED SA
     13: MOVE SA CTA
     14: DROP-ITEM SEAWEED CTA
     15: MOVE CTA SA
     16: PICK-UP-ITEM RICE SA
     17: MOVE SA MIXA
     18: DROP-ITEM RICE MIXA
     19: MOVE MIXA SA
```

```
20: PICK-UP-ITEM NOODLES SA
21: MOVE SA MIXA
22: MOVE MIXA PA
23: DROP-ITEM NOODLES PA
24: MOVE PA MIXA
25: MOVE MIXA SA
26: PICK-UP-ITEM BROTH SA
27: MOVE SA CTA
28: MOVE CTA MIXA
29: MOVE MIXA PA
30: MOVE PA CA
31: DROP-ITEM BROTH CA
32: COOK-INGREDIENT BROTH POT
    RAMEN CA
33: PICK-UP-ITEM POT CA
34: MOVE CA DWA
35: DROP-ITEM POT DWA
36: CHECK-PREPARED BROTH RAMEN
37: CLEAN-TOOL POT DWA
38: PICK-UP-ITEM POT DWA
39: MOVE DWA CA
40: DROP-ITEM POT CA
41: PICK-UP-ITEM BROTH CA
42: MOVE CA PA
43: DROP-ITEM BROTH PA
44: MOVE PA MIXA
45: MOVE MIXA CTA
46: PICK-UP-ITEM KNIFE CTA
47: CUT-INGREDIENT VEGETABLES
    KNIFE RAMEN CTA
48: CHECK-PREPARED VEGETABLES
    RAMEN
49: DROP-ITEM KNIFE CTA
50: PICK-UP-ITEM VEGETABLES
    CTA
51: MOVE CTA MIXA
52: MOVE MIXA PA
53: DROP-ITEM VEGETABLES PA
54: PICK-UP-ITEM NOODLES PA
55: MOVE PA CA
56: DROP-ITEM NOODLES CA
57: COOK-INGREDIENT NOODLES
    POT RAMEN CA
58: PICK-UP-ITEM NOODLES CA
59: CHECK-PREPARED NOODLES
    RAMEN
60: MOVE CA PA
61: DROP-ITEM NOODLES PA
62: ASSEMBLE-DISH RAMEN PA
```

```
63: PICK-UP-ITEM RAMEN PA
64: MOVE PA CA
65: MOVE CA SVA
66: SERVE-DISH RAMEN SVA
67: MOVE SVA CA
68: PICK-UP-ITEM POT CA
69: MOVE CA DWA
70: DROP-ITEM POT DWA
71: CLEAN-TOOL POT DWA
72: PICK-UP-ITEM POT DWA
73: MOVE DWA CA
74: DROP-ITEM POT CA
75: MOVE CA PA
76: MOVE PA MIXA
77: MOVE MIXA CTA
78: PICK-UP-ITEM KNIFE CTA
79: MOVE CTA MIXA
80: MOVE MIXA PA
81: MOVE PA DWA
82: DROP-ITEM KNIFE DWA
83: CLEAN-TOOL KNIFE DWA
84: PICK-UP-ITEM KNIFE DWA
85: MOVE DWA CA
86: MOVE CA PA
87: MOVE PA MIXA
88: MOVE MIXA CTA
89: DROP-ITEM KNIFE CTA
90: END-ORDER RAMEN
91: START-ORDER SUSHI
92: PICK-UP-ITEM KNIFE CTA
93: CUT-INGREDIENT FISH KNIFE
    SUSHI CTA
94: CUT-INGREDIENT SEAWEED
    KNIFE SUSHI CTA
95: CHECK-PREPARED FISH SUSHI
96: CHECK-PREPARED SEAWEED
    SUSHI
97: DROP-ITEM KNIFE CTA
98: PICK-UP-ITEM FISH CTA
99: MOVE CTA MIXA
100: MOVE MIXA PA
101: DROP-ITEM FISH PA
102: MOVE PA MIXA
103: MOVE MIXA CTA
104: PICK-UP-ITEM SEAWEED CTA
105: MOVE CTA MIXA
106: MOVE MIXA PA
107: DROP-ITEM SEAWEED PA
108: MOVE PA MIXA
```

```
109: MOVE MIXA CTA
110: PICK-UP-ITEM KNIFE CTA
111: MOVE CTA MIXA
112: MOVE MIXA PA
113: MOVE PA DWA
114: DROP-ITEM KNIFE DWA
115: CLEAN-TOOL KNIFE DWA
116: PICK-UP-ITEM KNIFE DWA
117: MOVE DWA CA
118: MOVE CA PA
119: MOVE PA MIXA
120: MOVE MIXA CTA
121: DROP-ITEM KNIFE CTA
122: MOVE CTA MIXA
123: PICK-UP-ITEM GLOVES MIXA
124: MIX-INGREDIENT RICE
    GLOVES SUSHI MIXA
125: MOVE MIXA PA
126: MOVE PA DWA
127: DROP-ITEM GLOVES DWA
128: CLEAN-TOOL GLOVES DWA
129: PICK-UP-ITEM GLOVES DWA
130: MOVE DWA PA
131: MOVE PA MIXA
132: DROP-ITEM GLOVES MIXA
```

```
133: PICK-UP-ITEM RICE MIXA
134: MOVE MIXA PA
135: MOVE PA CA
136: DROP-ITEM RICE CA
137: COOK-INGREDIENT RICE POT
    SUSHI CA
138: PICK-UP-ITEM RICE CA
139: CHECK-PREPARED RICE SUSHI
140: MOVE CA PA
141: DROP-ITEM RICE PA
142: ASSEMBLE-DISH SUSHI PA
143: PICK-UP-ITEM SUSHI PA
144: MOVE PA CA
145: MOVE CA SVA
146: SERVE-DISH SUSHI SVA
147: MOVE SVA CA
148: PICK-UP-ITEM POT CA
149: MOVE CA DWA
150: DROP-ITEM POT DWA
151: CLEAN-TOOL POT DWA
152: PICK-UP-ITEM POT DWA
153: MOVE DWA CA
154: DROP-ITEM POT CA
155: END-ORDER SUSHI
```

Analysis

The robot begins by preparing the ramen, gathering and positioning all necessary ingredients in their designated areas. Notably, it organizes ingredients required for both the ramen and sushi orders ahead of time, anticipating future steps. With everything in place, it starts the ramen preparation, first cooking the broth, then cleaning the pot immediately afterward to maintain a sanitary workspace. The broth is then set aside in the preparation area. Next, the robot cuts the vegetables and places them, along with the cooked noodles, in the same preparation zone. When all components are ready, the ramen is assembled and promptly served.

Following the ramen, the robot cleans each tool used, ensuring the kitchen is reset and ready to proceed with the next task. The sushi order then begins, with ingredients already positioned for efficient access. The robot takes the knife to cut the fish and seaweed, moving them to the preparation area once complete. After this, it cleans the knife, then heads to the mixing area to put on gloves and mix the rice. Following this, it cleans the gloves and leaves them in the mixing zone before taking the rice to the cooking area. Once the rice is

cooked, it joins the other sushi ingredients in the preparation area, and the dish is assembled. Finally, the sushi is served, and the pot is cleaned and returned to its place, marking the completion of the sushi order and resolution of the task.

4.3 Test Case 3

Description

Following the same approach, let's work on another problem where we will add a new dish that will be ordered by the restaurant's customers. In this case, it will be a curry rice dish. That means in this problem, we will now be combining three different dishes (ramen, sushi, curry rice), with some ingredients, such as rice, being part of the recipes for multiple dishes. We will analyze whether our robot can handle this situation.

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the robot to achieve the goal.

```
step  0: START-ORDER CURRY_RICE
      1: MOVE CA PA
      2: MOVE PA MIXA
      3: MOVE MIXA SA
      4: PICK-UP-ITEM
          VEGETABLES SA
      5: MOVE SA CTA
      6: DROP-ITEM VEGETABLES
          CTA
      7: MOVE CTA SA
      8: PICK-UP-ITEM FISH SA
      9: MOVE SA CTA
     10: DROP-ITEM FISH CTA
     11: MOVE CTA SA
     12: PICK-UP-ITEM SEAWEED
         SA
     13: MOVE SA CTA
     14: DROP-ITEM SEAWEED CTA
     15: MOVE CTA SA
     16: PICK-UP-ITEM RICE2 SA
     17: MOVE SA MIXA
     18: DROP-ITEM RICE2 MIXA
     19: MOVE MIXA SA
     20: PICK-UP-ITEM CHICKEN
         SA
     21: MOVE SA CTA
     22: DROP-ITEM CHICKEN CTA
     23: MOVE CTA SA
     24: PICK-UP-ITEM RICE1 SA
     25: MOVE SA MIXA
     26: DROP-ITEM RICE1 MIXA
     27: MOVE MIXA SA
     28: PICK-UP-ITEM CURRY SA
     29: MOVE SA MIXA
     30: MOVE MIXA PA
     31: DROP-ITEM CURRY PA
     32: MOVE PA MIXA
     33: MOVE MIXA SA
     34: PICK-UP-ITEM NOODLES SA
     35: MOVE SA MIXA
     36: MOVE MIXA PA
     37: DROP-ITEM NOODLES PA
     38: MOVE PA MIXA
     39: MOVE MIXA SA
     40: PICK-UP-ITEM BROTH SA
     41: MOVE SA CTA
     42: MOVE CTA MIXA
     43: MOVE MIXA PA
     44: DROP-ITEM BROTH PA
     45: MOVE PA MIXA
```

```
46: PICK-UP-ITEM GLOVES MIXA
47: MIX-INGREDIENT RICE2
    GLOVES CURRY_RICE MIXA
48: MOVE MIXA PA
49: MOVE PA DWA
50: DROP-ITEM GLOVES DWA
51: CLEAN-TOOL GLOVES DWA
52: PICK-UP-ITEM GLOVES DWA
53: MOVE DWA PA
54: MOVE PA MIXA
55: DROP-ITEM GLOVES MIXA
56: PICK-UP-ITEM RICE2 MIXA
57: MOVE MIXA PA
58: MOVE PA CA
59: DROP-ITEM RICE2 CA
60: COOK-INGREDIENT RICE2 POT
    CURRY_RICE CA
61: PICK-UP-ITEM POT CA
62: MOVE CA DWA
63: DROP-ITEM POT DWA
64: CHECK-PREPARED RICE2
    CURRY_RICE
65: CLEAN-TOOL POT DWA
66: PICK-UP-ITEM POT DWA
67: MOVE DWA CA
68: DROP-ITEM POT CA
69: PICK-UP-ITEM RICE2 CA
70: MOVE CA PA
71: DROP-ITEM RICE2 PA
72: PICK-UP-ITEM CURRY PA
73: MOVE PA CA
74: DROP-ITEM CURRY CA
75: COOK-INGREDIENT CURRY POT
    CURRY_RICE CA
76: PICK-UP-ITEM POT CA
77: MOVE CA DWA
78: DROP-ITEM POT DWA
79: CHECK-PREPARED CURRY
    CURRY_RICE
80: CLEAN-TOOL POT DWA
81: PICK-UP-ITEM POT DWA
82: MOVE DWA CA
83: DROP-ITEM POT CA
84: PICK-UP-ITEM CURRY CA
85: MOVE CA PA
86: DROP-ITEM CURRY PA
87: MOVE PA MIXA
88: MOVE MIXA CTA
```

```
89: PICK-UP-ITEM KNIFE CTA
90: CUT-INGREDIENT CHICKEN
    KNIFE CURRY_RICE CTA
91: MOVE CTA MIXA
92: MOVE MIXA PA
93: MOVE PA DWA
94: DROP-ITEM KNIFE DWA
95: CLEAN-TOOL KNIFE DWA
96: PICK-UP-ITEM KNIFE DWA
97: MOVE DWA PA
98: MOVE PA MIXA
99: MOVE MIXA CTA
100: DROP-ITEM KNIFE CTA
101: PICK-UP-ITEM CHICKEN CTA
102: MOVE CTA MIXA
103: MOVE MIXA PA
104: MOVE PA CA
105: DROP-ITEM CHICKEN CA
106: COOK-INGREDIENT CHICKEN
    POT CURRY_RICE CA
107: PICK-UP-ITEM CHICKEN CA
108: CHECK-PREPARED CHICKEN
    CURRY_RICE
109: MOVE CA PA
110: DROP-ITEM CHICKEN PA
111: ASSEMBLE-DISH CURRY_RICE
    PA
112: PICK-UP-ITEM CURRY_RICE
    PA
113: MOVE PA CA
114: MOVE CA SVA
115: SERVE-DISH CURRY_RICE SVA
116: MOVE SVA CA
117: PICK-UP-ITEM POT CA
118: MOVE CA DWA
119: DROP-ITEM POT DWA
120: CLEAN-TOOL POT DWA
121: PICK-UP-ITEM POT DWA
122: MOVE DWA CA
123: DROP-ITEM POT CA
124: END-ORDER CURRY_RICE
125: START-ORDER RAMEN
126: MOVE CA PA
127: MOVE PA MIXA
128: MOVE MIXA CTA
129: PICK-UP-ITEM KNIFE CTA
130: CUT-INGREDIENT
    VEGETABLES KNIFE RAMEN CTA
```

```
131: CHECK-PREPARED
    VEGETABLES RAMEN
132: DROP-ITEM KNIFE CTA
133: PICK-UP-ITEM VEGETABLES
    CTA
134: MOVE CTA MIXA
135: MOVE MIXA PA
136: DROP-ITEM VEGETABLES PA
137: PICK-UP-ITEM NOODLES PA
138: MOVE PA CA
139: DROP-ITEM NOODLES CA
140: COOK-INGREDIENT NOODLES
    POT RAMEN CA
141: PICK-UP-ITEM NOODLES CA
142: CHECK-PREPARED NOODLES
    RAMEN
143: MOVE CA PA
144: DROP-ITEM NOODLES PA
145: PICK-UP-ITEM BROTH PA
146: MOVE PA CA
147: DROP-ITEM BROTH CA
148: COOK-INGREDIENT BROTH
    POT RAMEN CA
149: CHECK-PREPARED BROTH
    RAMEN
150: PICK-UP-ITEM BROTH CA
151: MOVE CA PA
152: DROP-ITEM BROTH PA
153: ASSEMBLE-DISH RAMEN PA
154: PICK-UP-ITEM RAMEN PA
155: MOVE PA CA
156: MOVE CA SVA
157: SERVE-DISH RAMEN SVA
158: MOVE SVA CA
159: PICK-UP-ITEM POT CA
160: MOVE CA DWA
161: DROP-ITEM POT DWA
162: CLEAN-TOOL POT DWA
163: PICK-UP-ITEM POT DWA
164: MOVE DWA CA
165: DROP-ITEM POT CA
166: MOVE CA PA
167: MOVE PA MIXA
168: MOVE MIXA CTA
169: PICK-UP-ITEM KNIFE CTA
170: MOVE CTA MIXA
171: MOVE MIXA PA
172: MOVE PA DWA
173: DROP-ITEM KNIFE DWA
```

```
174: CLEAN-TOOL KNIFE DWA
175: PICK-UP-ITEM KNIFE DWA
176: MOVE DWA CA
177: MOVE CA PA
178: MOVE PA MIXA
179: MOVE MIXA CTA
180: DROP-ITEM KNIFE CTA
181: END-ORDER RAMEN
182: START-ORDER SUSHI
183: PICK-UP-ITEM KNIFE CTA
184: CUT-INGREDIENT FISH
    KNIFE SUSHI CTA
185: CUT-INGREDIENT SEAWEED
    KNIFE SUSHI CTA
186: CHECK-PREPARED FISH SUSHI
187: CHECK-PREPARED SEAWEED
    SUSHI
188: DROP-ITEM KNIFE CTA
189: PICK-UP-ITEM FISH CTA
190: MOVE CTA MIXA
191: MOVE MIXA PA
192: DROP-ITEM FISH PA
193: MOVE PA MIXA
194: MOVE MIXA CTA
195: PICK-UP-ITEM SEAWEED CTA
196: MOVE CTA MIXA
197: MOVE MIXA PA
198: DROP-ITEM SEAWEED PA
199: MOVE PA MIXA
200: MOVE MIXA CTA
201: PICK-UP-ITEM KNIFE CTA
202: MOVE CTA MIXA
203: MOVE MIXA PA
204: MOVE PA DWA
205: DROP-ITEM KNIFE DWA
206: CLEAN-TOOL KNIFE DWA
207: PICK-UP-ITEM KNIFE DWA
208: MOVE DWA CA
209: MOVE CA PA
210: MOVE PA MIXA
211: MOVE MIXA CTA
212: DROP-ITEM KNIFE CTA
213: MOVE CTA MIXA
214: PICK-UP-ITEM GLOVES MIXA
215: MIX-INGREDIENT RICE1
    GLOVES SUSHI MIXA
216: MOVE MIXA PA
217: MOVE PA DWA
218: DROP-ITEM GLOVES DWA
```

```
219: CLEAN-TOOL GLOVES DWA
220: PICK-UP-ITEM GLOVES DWA
221: MOVE DWA PA
222: MOVE PA MIXA
223: DROP-ITEM GLOVES MIXA
224: PICK-UP-ITEM RICE1 MIXA
225: MOVE MIXA PA
226: MOVE PA CA
227: DROP-ITEM RICE1 CA
228: COOK-INGREDIENT RICE1
    POT SUSHI CA
229: PICK-UP-ITEM RICE1 CA
230: CHECK-PREPARED RICE1
    SUSHI
231: MOVE CA PA
```

```
232: DROP-ITEM RICE1 PA
233: ASSEMBLE-DISH SUSHI PA
234: PICK-UP-ITEM SUSHI PA
235: MOVE PA CA
236: MOVE CA SVA
237: SERVE-DISH SUSHI SVA
238: MOVE SVA CA
239: PICK-UP-ITEM POT CA
240: MOVE CA DWA
241: DROP-ITEM POT DWA
242: CLEAN-TOOL POT DWA
243: PICK-UP-ITEM POT DWA
244: MOVE DWA CA
245: DROP-ITEM POT CA
246: END-ORDER SUSHI
```

Analysis

Given the extensive nature of this plan, we will refrain from delving into the finer details and instead provide a general overview of its execution.

As anticipated, the robot adheres to the same strategy as in previous tasks. It begins by preparing the curry rice, first retrieving all the necessary ingredients from storage and placing them in their designated locations—this includes not only the ingredients required for the curry rice but also others that might be used later. Once everything is in place, the robot proceeds to prepare the curry rice. Before moving on to the next dish, it ensures that all tools are cleaned, maintaining hygiene throughout the process.

Next, the robot begins preparing the ramen, following the same method. It gathers and prepares the ingredients, assembles the dish, and delivers it to the serving area. Once the ramen is served, the robot cleans the used tools before proceeding to the final dish, the sushi. Similar to the previous tasks, it prepares the ingredients, assembles the dish, and serves it, ensuring that tools are cleaned and the work area is sanitized before concluding the order.

4.4 Test Case 4 (Expanded Domain)

These two tests represent additional cases in which we introduced new elements to the domain, as the problem statement indicated that the system should be capable of handling unexpected scenarios. To address this, we implemented a tool durability feature, where each tool can be used a specified number of times before it becomes unusable and must be replaced. Furthermore, we expanded the scenario to include the involvement of multiple robots in the restaurant's operations.

Description

In this scenario, we have two robots operating within the same restaurant layout. The setup includes two knives and pots, each with a limited durability—each knife and pot can be used twice before they are considered unusable. Additionally, there is a pair of gloves with a maximum of three uses. The customers place orders for sushi and ramen, and we introduce extra ingredients that are not required for the preparation of these dishes, to test if the robots encounter any conflicts or issues in handling unnecessary items.

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the robot to achieve the goal.

```
step  0: MOVE R2 SA CTA
      1: START-ORDER RAMEN
      2: MOVE R2 CTA SA
      3: PICK-UP-ITEM R2
          VEGETABLES SA
      4: MOVE R2 SA CTA
      5: DROP-ITEM R2
          VEGETABLES CTA
      6: MOVE R2 CTA SA
      7: PICK-UP-ITEM R2 FISH
          SA
      8: MOVE R2 SA CTA
      9: DROP-ITEM R2 FISH CTA
     10: MOVE R2 CTA SA
     11: PICK-UP-ITEM R2
         SEAWEED SA
     12: MOVE R2 SA CTA
     13: DROP-ITEM R2 SEAWEED
         CTA
     14: MOVE R1 CA PA
     15: MOVE R1 PA MIXA
     16: MOVE R1 MIXA SA
     17: PICK-UP-ITEM R1 RICE SA
     18: PICK-UP-ITEM R2 KNIFE2
         CTA
     19: MOVE R1 SA MIXA
     20: DROP-ITEM R1 RICE MIXA
     21: MOVE R1 MIXA SA
     22: PICK-UP-ITEM R1
          NOODLES SA
     23: MOVE R1 SA MIXA
     24: MOVE R1 MIXA PA
     25: DROP-ITEM R1 NOODLES PA
     26: MOVE R1 PA MIXA
     27: MOVE R1 MIXA SA
     28: PICK-UP-ITEM R1 BROTH SA
     29: MOVE R1 SA MIXA
     30: MOVE R1 MIXA PA
     31: DROP-ITEM R1 BROTH PA
     32: PICK-UP-ITEM R1 NOODLES PA
     33: MOVE R1 PA CA
     34: DROP-ITEM R1 NOODLES CA
     35: COOK-INGREDIENT R1
         NOODLES POT2 RAMEN CA
     36: PICK-UP-ITEM R1 NOODLES CA
     37: CHECK-PREPARED NOODLES
         RAMEN
     38: MOVE R1 CA PA
     39: DROP-ITEM R1 NOODLES PA
     40: MOVE R2 CTA MIXA
     41: MOVE R1 PA MIXA
     42: DROP-ITEM R2 KNIFE2 MIXA
     43: PICK-UP-ITEM R1 KNIFE2
         MIXA
     44: MOVE R1 MIXA CTA
     45: MOVE R2 MIXA PA
     46: PICK-UP-ITEM R2 BROTH PA
     47: MOVE R2 PA CA
     48: DROP-ITEM R2 BROTH CA
     49: COOK-INGREDIENT R2 BROTH
         POT2 RAMEN CA
     50: PICK-UP-ITEM R2 BROTH CA
     51: MOVE R2 CA PA
     52: CHECK-PREPARED BROTH RAMEN
```

```
53: DROP-ITEM R2 BROTH PA
54: MOVE R2 PA CA
55: PICK-UP-ITEM R2 POT2 CA
56: MOVE R2 CA PA
57: MOVE R2 PA MIXA
58: THROW-TOOL R2 POT2
59: MOVE R2 MIXA CTA
60: CUT-INGREDIENT R1
    VEGETABLES KNIFE2 RAMEN
    CTA
61: DROP-ITEM R1 KNIFE2 CTA
62: PICK-UP-ITEM R1
    VEGETABLES CTA
63: CHECK-PREPARED VEGETABLES
    RAMEN
64: MOVE R1 CTA MIXA
65: MOVE R1 MIXA PA
66: DROP-ITEM R1 VEGETABLES PA
67: ASSEMBLE-DISH R1 RAMEN PA
68: MOVE R2 CTA MIXA
69: PICK-UP-ITEM R1 RAMEN PA
70: MOVE R1 PA CA
71: MOVE R1 CA SVA
72: SERVE-DISH R1 RAMEN SVA
73: MOVE R1 SVA CA
74: MOVE R1 CA DWA
75: MOVE R2 MIXA CTA
76: PICK-UP-ITEM R2 KNIFE2 CTA
77: MOVE R2 CTA MIXA
78: MOVE R2 MIXA PA
79: MOVE R2 PA DWA
80: DROP-ITEM R2 KNIFE2 DWA
81: MOVE R1 DWA CA
82: MOVE R1 CA SVA
83: CLEAN-TOOL R2 KNIFE2 DWA
84: PICK-UP-ITEM R2 KNIFE2 DWA
85: MOVE R2 DWA CA
86: MOVE R2 CA PA
87: MOVE R2 PA MIXA
88: MOVE R2 MIXA CTA
89: DROP-ITEM R2 KNIFE2 CTA
90: END-ORDER RAMEN
91: START-ORDER SUSHI
92: PICK-UP-ITEM R2 KNIFE2 CTA
93: CUT-INGREDIENT R2 FISH
    KNIFE2 SUSHI CTA
94: THROW-TOOL R2 KNIFE2
95: CHECK-PREPARED FISH SUSHI
```

```
96: PICK-UP-ITEM R2 FISH CTA
97: MOVE R2 CTA MIXA
98: MOVE R2 MIXA PA
99: DROP-ITEM R2 FISH PA
100: MOVE R1 SVA CA
101: MOVE R2 PA MIXA
102: MOVE R2 MIXA CTA
103: MOVE R1 CA PA
104: MOVE R1 PA MIXA
105: PICK-UP-ITEM R2 KNIFE1
    CTA
106: MOVE R1 MIXA CTA
107: CUT-INGREDIENT R2
    SEAWEED KNIFE1 SUSHI CTA
108: PICK-UP-ITEM R1 SEAWEED
    CTA
109: MOVE R1 CTA MIXA
110: CHECK-PREPARED SEAWEED
    SUSHI
111: MOVE R1 MIXA PA
112: DROP-ITEM R1 SEAWEED PA
113: DROP-ITEM R2 KNIFE1 CTA
114: MOVE R1 PA CA
115: PICK-UP-ITEM R2 KNIFE1
    CTA
116: MOVE R2 CTA MIXA
117: MOVE R2 MIXA PA
118: MOVE R2 PA DWA
119: DROP-ITEM R2 KNIFE1 DWA
120: CLEAN-TOOL R2 KNIFE1 DWA
121: PICK-UP-ITEM R2 KNIFE1
    DWA
122: MOVE R2 DWA PA
123: MOVE R2 PA MIXA
124: MOVE R1 CA PA
125: MOVE R2 MIXA CTA
126: DROP-ITEM R2 KNIFE1 CTA
127: MOVE R1 PA MIXA
128: PICK-UP-ITEM R1 GLOVES
    MIXA
129: MOVE R2 CTA MIXA
130: MIX-INGREDIENT R1 RICE
    GLOVES SUSHI MIXA
131: MOVE R1 MIXA PA
132: PICK-UP-ITEM R2 RICE MIXA
133: MOVE R2 MIXA PA
134: MOVE R1 PA DWA
135: DROP-ITEM R1 GLOVES DWA
```

```
136: CLEAN-TOOL R1 GLOVES DWA
137: PICK-UP-ITEM R1 GLOVES
    DWA
138: MOVE R1 DWA PA
139: MOVE R1 PA MIXA
140: DROP-ITEM R1 GLOVES MIXA
141: MOVE R2 PA CA
142: DROP-ITEM R2 RICE CA
143: COOK-INGREDIENT R2 RICE
    POT1 SUSHI CA
144: PICK-UP-ITEM R2 RICE CA
145: CHECK-PREPARED RICE SUSHI
146: MOVE R2 CA PA
147: DROP-ITEM R2 RICE PA
```

```
148: ASSEMBLE-DISH R2 SUSHI PA
149: PICK-UP-ITEM R2 SUSHI PA
150: MOVE R2 PA CA
151: MOVE R2 CA SVA
152: SERVE-DISH R2 SUSHI SVA
153: MOVE R2 SVA CA
154: PICK-UP-ITEM R2 POT1 CA
155: MOVE R2 CA DWA
156: DROP-ITEM R2 POT1 DWA
157: CLEAN-TOOL R2 POT1 DWA
158: PICK-UP-ITEM R2 POT1 DWA
159: MOVE R2 DWA CA
160: DROP-ITEM R2 POT1 CA
161: END-ORDER SUSHI
```

Analysis

It is interesting to observe that, despite having two robots, the strategy employed remains the same. Initially, the robots place all the required ingredients (excluding the extra, non-essential ones) in their designated zones for preparation. Following this, the robots cooperate by delegating tasks to efficiently complete the first order, in this case, ramen. For instance, Robot 1 is seen cutting the vegetables while simultaneously cooking the broth. They follow the same approach for preparing the sushi. Notably, during the sushi preparation, the knife reaches the end of its usability, prompting the robot to plan for a replacement. It proceeds by retrieving a new knife and continues cutting the remaining ingredients, ensuring the completion of the order. After that the sushi order is ended and the problem is solved correctly.

4.5 Test Case 5 (Expanded Domain)

Description

This scenario presents the most complex planning task among all the ones tested so far. In this case, the restaurant is managed by three robots, and the customers place orders for three different dishes: sushi, ramen, and curry rice. As before, the tools have limited durability, with each tool having a set number of uses before it becomes unusable. To prevent the problem from becoming unsolvable, additional tools are available for the robots to use when one breaks.

What makes this situation more challenging is the fact that the tools have varying levels of durability. For example, the knives may have different numbers of uses left, so the robots must strategically decide which knife to use first, opting for the one with more durability or the one that still has uses remaining. This introduces an added layer of complexity to the planning process, as the robots

need to consider the durability of the tools while planning the tasks for preparing and assembling the dishes.

In this case, the robots are tasked with preparing sushi, ramen, and curry rice, and they must coordinate their efforts to ensure that all orders are completed successfully while also managing tool durability and potential failures. The need for precise tool management, cooperation between robots, and adaptability to tool limitations makes this scenario significantly more intricate than the previous ones.

Results

The following plan was generated by the planner for this test case. Each step represents an action taken by the robot to achieve the goal.

```
step  0: MOVE R3 SA CTA
      1: MOVE R2 PA MIXA
      2: START-ORDER CURRY_RICE
      3: MOVE R3 CTA SA
      4: PICK-UP-ITEM R3 RICE2
          SA
      5: MOVE R3 SA MIXA
      6: DROP-ITEM R3 RICE2
          MIXA
      7: PICK-UP-ITEM R3
          GLOVES MIXA
      8: MOVE R2 MIXA SA
      9: PICK-UP-ITEM R2 RICE1
          SA
     10: MOVE R2 SA MIXA
     11: DROP-ITEM R2 RICE1 MIXA
     12: MOVE R2 MIXA SA
     13: PICK-UP-ITEM R2
         CHICKEN SA
     14: MOVE R2 SA CTA
     15: DROP-ITEM R2 CHICKEN
         CTA
     16: MOVE R2 CTA MIXA
     17: MOVE R2 MIXA SA
     18: PICK-UP-ITEM R2
         VEGETABLES SA
     19: MOVE R2 SA CTA
     20: DROP-ITEM R2
         VEGETABLES CTA
     21: MOVE R2 CTA MIXA
     22: MOVE R2 MIXA SA
     23: PICK-UP-ITEM R2 FISH SA
     24: MOVE R2 SA CTA
     25: DROP-ITEM R2 FISH CTA
     26: MOVE R2 CTA MIXA
     27: MOVE R2 MIXA SA
     28: PICK-UP-ITEM R2 SEAWEED SA
     29: MOVE R2 SA CTA
     30: DROP-ITEM R2 SEAWEED CTA
     31: PICK-UP-ITEM R2 KNIFE3 CTA
     32: CUT-INGREDIENT R2 CHICKEN
         KNIFE3 CURRY_RICE CTA
     33: THROW-TOOL R2 KNIFE3
     34: MOVE R2 CTA MIXA
     35: MOVE R2 MIXA SA
     36: DROP-ITEM R3 GLOVES MIXA
     37: MOVE R3 MIXA CTA
     38: MOVE R1 CA PA
     39: MOVE R1 PA MIXA
     40: PICK-UP-ITEM R2 CURRY SA
     41: MOVE R2 SA MIXA
     42: MOVE R2 MIXA PA
     43: DROP-ITEM R2 CURRY PA
     44: MOVE R2 PA MIXA
     45: MOVE R2 MIXA SA
     46: PICK-UP-ITEM R2 NOODLES SA
     47: MOVE R2 SA MIXA
     48: MOVE R2 MIXA PA
     49: DROP-ITEM R2 NOODLES PA
     50: MOVE R2 PA MIXA
     51: MOVE R2 MIXA SA
     52: PICK-UP-ITEM R2 BROTH SA
```

```
53: MOVE R2 SA MIXA
54: MOVE R2 MIXA PA
55: DROP-ITEM R2 BROTH PA
56: MOVE R2 PA MIXA
57: MOVE R2 MIXA CTA
58: PICK-UP-ITEM R2 CHICKEN
    CTA
59: MOVE R2 CTA MIXA
60: MOVE R2 MIXA PA
61: DROP-ITEM R2 CHICKEN PA
62: MOVE R2 PA MIXA
63: PICK-UP-ITEM R1 GLOVES
    MIXA
64: MIX-INGREDIENT R1 RICE2
    GLOVES CURRY_RICE MIXA
65: MOVE R1 MIXA PA
66: MOVE R1 PA DWA
67: DROP-ITEM R1 GLOVES DWA
68: CLEAN-TOOL R1 GLOVES DWA
69: PICK-UP-ITEM R1 GLOVES DWA
70: MOVE R1 DWA PA
71: MOVE R1 PA MIXA
72: PICK-UP-ITEM R2 RICE2 MIXA
73: MOVE R2 MIXA PA
74: MOVE R2 PA CA
75: DROP-ITEM R2 RICE2 CA
76: COOK-INGREDIENT R2 RICE2
    POT2 CURRY_RICE CA
77: PICK-UP-ITEM R2 RICE2 CA
78: CHECK-PREPARED RICE2
    CURRY_RICE
79: MOVE R2 CA PA
80: DROP-ITEM R2 RICE2 PA
81: PICK-UP-ITEM R2 CHICKEN PA
82: MOVE R2 PA CA
83: DROP-ITEM R2 CHICKEN CA
84: COOK-INGREDIENT R2
    CHICKEN POT2 CURRY_RICE CA
85: MOVE R2 CA PA
86: CHECK-PREPARED CHICKEN
    CURRY_RICE
87: PICK-UP-ITEM R2 CURRY PA
88: MOVE R2 PA CA
89: DROP-ITEM R2 CURRY CA
90: COOK-INGREDIENT R2 CURRY
    POT2 CURRY_RICE CA
91: CHECK-PREPARED CURRY
    CURRY_RICE
```

```
92: PICK-UP-ITEM R2 CHICKEN CA
93: MOVE R2 CA PA
94: DROP-ITEM R2 CHICKEN PA
95: MOVE R2 PA CA
96: PICK-UP-ITEM R2 POT2 CA
97: THROW-TOOL R2 POT2
98: PICK-UP-ITEM R2 CURRY CA
99: MOVE R2 CA PA
100: DROP-ITEM R2 CURRY PA
101: ASSEMBLE-DISH R2
    CURRY_RICE PA
102: MOVE R2 PA MIXA
103: PICK-UP-ITEM R3 KNIFE2
    CTA
104: MOVE R2 MIXA PA
105: PICK-UP-ITEM R2
    CURRY_RICE PA
106: MOVE R2 PA CA
107: MOVE R2 CA SVA
108: SERVE-DISH R2 CURRY_RICE
    SVA
109: DROP-ITEM R1 GLOVES MIXA
110: DROP-ITEM R3 KNIFE2 CTA
111: END-ORDER CURRY_RICE
112: START-ORDER RAMEN
113: PICK-UP-ITEM R3 KNIFE2
    CTA
114: CUT-INGREDIENT R3
    VEGETABLES KNIFE2 RAMEN
    CTA
115: THROW-TOOL R3 KNIFE2
116: CHECK-PREPARED
    VEGETABLES RAMEN
117: MOVE R2 SVA CA
118: MOVE R2 CA PA
119: MOVE R2 PA MIXA
120: MOVE R2 MIXA CTA
121: PICK-UP-ITEM R2
    VEGETABLES CTA
122: MOVE R2 CTA MIXA
123: PICK-UP-ITEM R1 RICE1
    MIXA
124: MOVE R2 MIXA PA
125: DROP-ITEM R2 VEGETABLES
    PA
126: DROP-ITEM R1 RICE1 MIXA
127: PICK-UP-ITEM R2 NOODLES
    PA
```

```
128: MOVE R2 PA CA
129: DROP-ITEM R2 NOODLES CA
130: COOK-INGREDIENT R2
    NOODLES POT1 RAMEN CA
131: PICK-UP-ITEM R2 POT1 CA
132: MOVE R2 CA DWA
133: DROP-ITEM R2 POT1 DWA
134: CHECK-PREPARED NOODLES
    RAMEN
135: CLEAN-TOOL R2 POT1 DWA
136: PICK-UP-ITEM R2 POT1 DWA
137: MOVE R2 DWA CA
138: DROP-ITEM R2 POT1 CA
139: MOVE R2 CA PA
140: MOVE R2 PA MIXA
141: MOVE R2 MIXA CTA
142: MOVE R1 MIXA PA
143: MOVE R1 PA CA
144: PICK-UP-ITEM R1 NOODLES
    CA
145: MOVE R1 CA PA
146: DROP-ITEM R1 NOODLES PA
147: PICK-UP-ITEM R2 KNIFE1
    CTA
148: PICK-UP-ITEM R1 BROTH PA
149: MOVE R1 PA CA
150: DROP-ITEM R1 BROTH CA
151: COOK-INGREDIENT R1 BROTH
    POT1 RAMEN CA
152: PICK-UP-ITEM R1 BROTH CA
153: CHECK-PREPARED BROTH
    RAMEN
154: MOVE R1 CA PA
155: DROP-ITEM R1 BROTH PA
156: ASSEMBLE-DISH R1 RAMEN PA
157: MOVE R1 PA MIXA
158: PICK-UP-ITEM R1 RICE1
    MIXA
159: MOVE R3 CTA MIXA
160: MOVE R3 MIXA PA
161: PICK-UP-ITEM R3 RAMEN PA
162: MOVE R3 PA CA
163: MOVE R3 CA SVA
164: SERVE-DISH R3 RAMEN SVA
165: MOVE R3 SVA CA
166: PICK-UP-ITEM R3 POT1 CA
167: MOVE R3 CA DWA
168: DROP-ITEM R3 POT1 DWA
169: CLEAN-TOOL R3 POT1 DWA
170: PICK-UP-ITEM R3 POT1 DWA
171: MOVE R3 DWA CA
172: DROP-ITEM R3 POT1 CA
173: DROP-ITEM R2 KNIFE1 CTA
174: END-ORDER RAMEN
```

```
175: PICK-UP-ITEM R2 KNIFE1
    CTA
176: START-ORDER SUSHI
177: CUT-INGREDIENT R2 FISH
    KNIFE1 SUSHI CTA
178: CUT-INGREDIENT R2
    SEAWEED KNIFE1 SUSHI CTA
179: CHECK-PREPARED FISH SUSHI
180: CHECK-PREPARED SEAWEED
    SUSHI
181: DROP-ITEM R1 RICE1 MIXA
182: PICK-UP-ITEM R1 GLOVES
    MIXA
183: MIX-INGREDIENT R1 RICE1
    GLOVES SUSHI MIXA
184: THROW-TOOL R1 GLOVES
185: PICK-UP-ITEM R1 RICE1
    MIXA
186: MOVE R1 MIXA PA
187: THROW-TOOL R2 KNIFE1
188: MOVE R1 PA CA
189: DROP-ITEM R1 RICE1 CA
190: MOVE R1 CA PA
191: PICK-UP-ITEM R2 FISH CTA
192: MOVE R2 CTA MIXA
193: MOVE R2 MIXA PA
194: DROP-ITEM R2 FISH PA
195: MOVE R2 PA MIXA
196: MOVE R2 MIXA CTA
197: PICK-UP-ITEM R2 SEAWEED
    CTA
198: MOVE R2 CTA MIXA
199: MOVE R2 MIXA PA
200: DROP-ITEM R2 SEAWEED PA
201: COOK-INGREDIENT R3 RICE1
    POT1 SUSHI CA
202: PICK-UP-ITEM R3 RICE1 CA
203: CHECK-PREPARED RICE1
    SUSHI
204: MOVE R3 CA PA
205: DROP-ITEM R3 RICE1 PA
206: ASSEMBLE-DISH R3 SUSHI PA
207: PICK-UP-ITEM R3 SUSHI PA
208: MOVE R3 PA CA
209: MOVE R3 CA SVA
210: MOVE R2 PA CA
211: SERVE-DISH R3 SUSHI SVA
212: PICK-UP-ITEM R2 POT1 CA
213: MOVE R2 CA DWA
214: DROP-ITEM R2 POT1 DWA
215: CLEAN-TOOL R2 POT1 DWA
216: PICK-UP-ITEM R2 POT1 DWA
217: MOVE R2 DWA CA
218: DROP-ITEM R2 POT1 CA
219: END-ORDER SUSHI
```

Analysis

The strategy appears to remain consistent. Although the first task is to prepare the curry rice, all necessary ingredients are gathered and placed in their designated spots. It is also notable that the planner occasionally experiences some confusion; for instance, robot 3 retrieves the gloves, only to later drop them in the same location without using them after a few movements by the other robots. This suggests that the generated plan is not fully optimized, as these actions do not contribute toward achieving the overall objective. Nevertheless, all constraints are adhered to, including those related to handling broken tools. Additionally, it is interesting to observe that robot 3 is used far less frequently than robots 1 and 2, indicating that the addition of more robots simply adds complexity without improving efficiency. In fact, this is the case because, at any given moment, only a single action can be performed, meaning that increased robot numbers do not result in a more effective plan due to the lack of possible parallelism.

5 Analysis of the Results

This section provides a detailed analysis of the results, focusing on key metrics affected by various aspects of the problem. Since our solution involves numeric fluents and types, we utilized the Metric-FF planner, which supports these features. The experiments were conducted locally on an Ubuntu virtual machine.

5.1 Complexity Analysis

This subsection presents the complexity of the solutions found by the planner for each test case. The table below summarizes how key factors such as the number of robots and dishes impact the number of nodes expanded, plan length, and computational time. These factors provide deeper insights into the complexity of each test case and how the planner's performance scales with these variables.

Table 1: Comparison of Test Case Characteristics and Complexity Metrics

Problem	**Robots**	**Dishes**	**Actions**	**States**	**Time (s)**
problem1	1	1	81	950	0.02
problem2	1	2	156	8,341	0.49
problem3	1	3	247	68,155	2.68
problem4	2	2	162	100,418	22.48
problem5	3	3	220	1,935,327	1,325.83

In **Problem 1**, the simplest scenario with one robot and one dish results in a minimal number of actions (81) and a relatively low number of states generated (950). The computational time is negligible (0.02 seconds), reflecting the straightforward planning complexity.

Problem 2 introduces an additional dish while maintaining a single robot. This increase in complexity leads to a rise in the number of actions (156), states generated (8,341), and computational time (0.49 seconds). **Problem 3** further increases the complexity by adding one more dish. The number of actions rises to 247, states generated to 68,155, and computational time to 2.68 seconds. These examples demonstrate that while the number of actions increases nearly linearly (adding approximately 80 actions per additional dish), the number of states generated and the computational time grow at a more substantial rate.

Problem 4 maintains the same configuration as Problem 2 but introduces an additional robot. This setup results in 162 actions, 100,418 states, and a computational time of 22.48 seconds. Interestingly, adding a second robot leads to a decline in performance across all metrics compared to Problem 2: there are 6 more actions despite having an extra robot, the number of states generated increases by over tenfold, and the computational time nearly quintuples. These observations suggest that increasing the number of robots, while seem-

ingly beneficial for reducing the total number of actions, can ultimately be counterproductive due to the significant escalation in state space and computational demands.

Problem 5 represents the most complex scenario with three robots managing three dishes. Compared to Problem 3, which involved two robots and three dishes, this configuration results in a slight reduction in the number of actions to 220, which is 25 actions fewer than in Problem 3. However, the number of states generated increases dramatically by nearly thirty times to 1,935,327, and the computational time escalates substantially to 1,325.83 seconds. This contrasts with Problem 4 where adding an extra robot slightly increased the number of actions. These results indicate that while increasing the number of robots may offer a marginal decrease in actions at a larger scale, it can lead to an impractical surge in the number of states and the required computational time, ultimately hindering planner performance in more complex scenarios.

5.2 Impact of Number of Robots on Solution Complexity

In this subsection, we explore how varying the number of robots impacts the complexity of the solutions generated by the planner. We conducted experiments ranging from 1 robot and 1 dish to 3 robots and 3 dishes. For each configuration, we recorded the number of actions, number of states, and computation time. Table 2 summarizes the collected data.

Table 2: Impact of Number of Robots on Solution Complexity Metrics

Robots	Dishes	Actions	States	Time (s)
1	1	81	950	0,02
1	2	156	8,341	0,49
1	3	247	68,155	2,68
2	1	67	2,674	1,08
2	2	162	100,418	22,48
2	3	204	562,221	110,11
3	1	78	22,129	1,55
3	2	191	5,929,945	3556,07
3	3	220	1,935,327	1325,83

In Table 2 are presented the metrics for all combinations of robots and dishes. In this analysis, we focus on grouping the results based on the number of robots, creating three distinct groups corresponding to configurations with 1, 2, and 3 robots, as highlighted in the table. Following this, we will present and discuss the corresponding graphs for each complexity metric.

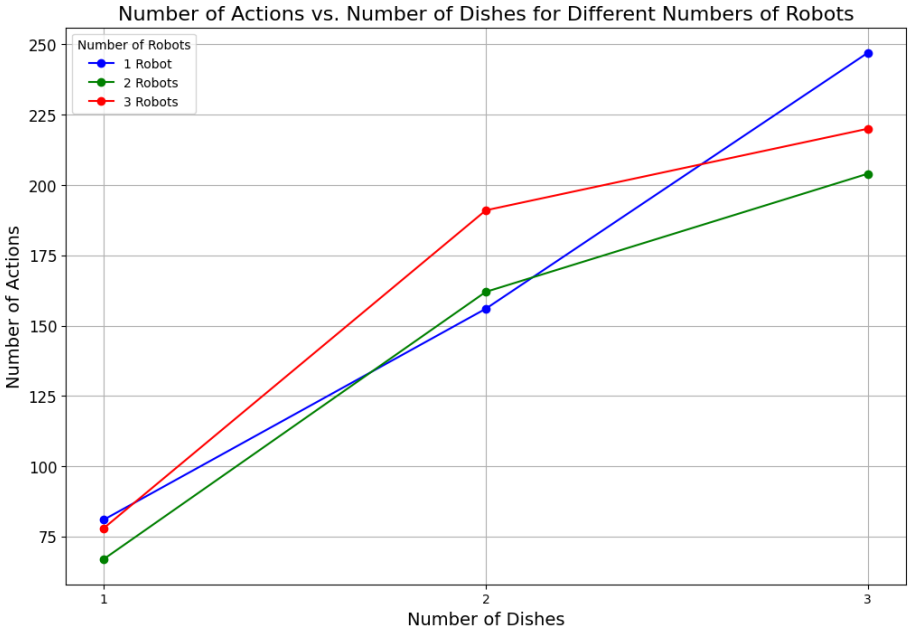

Figure 2: Number of Actions vs. Number of Dishes for Different Numbers of Robots

Figure 2 illustrates the number of actions required as the number of dishes increases for varying numbers of robots. The graph indicates a positive correlation between the number of dishes and the number of actions, with each additional dish linearly increasing the number of actions. However, this metric is not as straightforward as it initially appears. For instance, with 1 dish, the configuration with 3 robots require fewer actions than with 1 robot, with 2 robots being the optimal configuration; with 2 dishes, the optimal number of robots is 1, followed by 2 robots and then 3 robots by a large margin; and with 3 dishes, the optimal configuration is 2 robots again, followed by 3 robots and 1 robot resulting in the highest number of actions. Moreover, it is important to note that the solutions generated by the planner are not necessarily the most optimal in terms of minimizing the number of actions.

These observations imply that the relationship between the number of robots and the number of actions is influenced by the specific number of dishes, which reflects the complexity of the problem. For scenarios with fewer dishes, the difference in the number of actions across different number of robots configurations is minimal. However, as the number of dishes increases, the optimal number of robots for minimizing actions becomes less predictable, indicating that neither increasing nor decreasing the number of robots consistently leads to more optimal plans. We wanted to test configurations with additional dishes, but the computational time requirements proved prohibitively large.

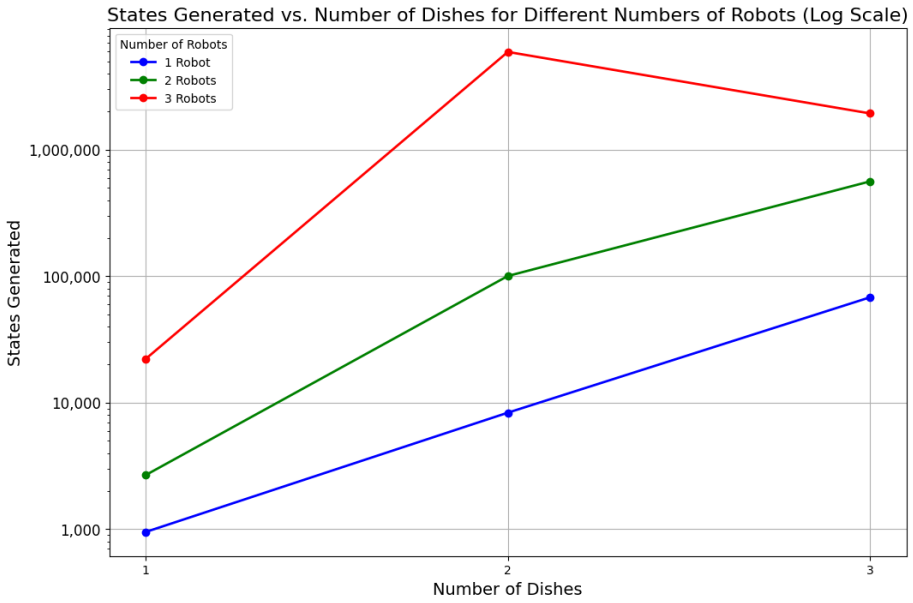

Figure 3: States Generated vs. Number of Dishes for Different Numbers of Robots

Figure 3 displays the relationship between the number of states generated as the number of dishes increases for varying numbers of robots. Firstly, we must observe that the scale of the graph is logarithmic. Secondly, this metric does not indicate complexity: across all numbers of dishes, having fewer robots results in fewer states. This is logical, as fewer robots imply fewer possible state combinations.

In terms of state generation, it is more advantageous to have fewer robots. Observing Figure 2, where the differences in the number of actions are minimal, we can argue that using a smaller number of robots is preferable given the substantial reduction in the number of states generated. An interesting case is with 3 robots and 2 dishes, which generates nearly 6 million states compared to approximately 2 million states for 3 dishes, resulting in a threefold increase. This anomaly may be due to the various possible combinations of task distributions among robots, though further investigation is needed to better understand the underlying causes.

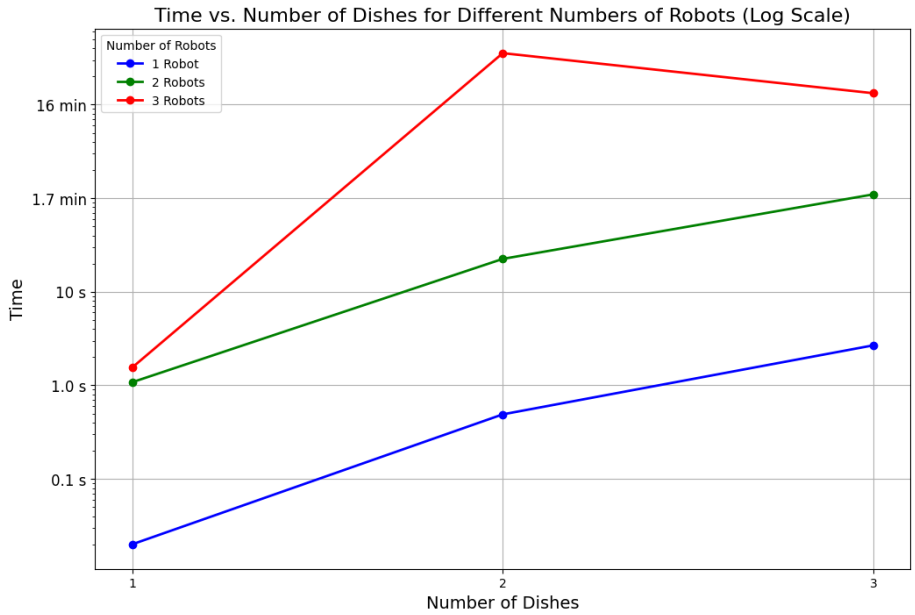

Figure 4: Time vs. Number of Dishes for Different Numbers of Robots

Figure 3 presents the computational time required to solve the problem as the number of dishes increases for varying numbers of robots. As in the previous graph, the scale is logarithmic. Interestingly, this graph closely resembles the previous one depicting the number of states, which makes sense because there is a strong positive correlation between the number of states and the computation time: the more states that are generated, the more time is required to process them. Using fewer robots results in reduced computation time, as fewer task combinations are generated. Similar to the pattern observed in the previous graph, the configuration with 3 robots requires significantly more time (59 minutes) to solve the problem with 2 dishes than with 3 dishes (22 minutes), highlighting an unusual increase in computational demand for this particular setup.

In summary, increasing the number of robots offers certain benefits, such as a reduction in the total number of actions required to manage multiple dishes. However, these benefits are overshadowed by a significant increase in the number of states generated and the computational time required for planning. The data indicate that while additional robots can improve task distribution, they introduce substantial coordination overhead, leading to exponential growth in solution complexity. These findings underscore the importance of optimizing multi-robot planning algorithms to effectively manage the trade-offs between task distribution and computational demands.

5.3 Impact of Number of Dishes on Solution Complexity

In this subsection, we analyze how increasing the number of dishes affects the complexity of the solutions. We used the same results from the previous study to assess the impact of the number of dishes on the same metrics. Table 3 summarizes the collected data, now emphasizing the number of dishes.

Table 3: Impact of Number of Dishes on Solution Complexity Metrics

Dishes	**Robots**	**Actions**	**States**	**Time (s)**
1	1	81	950	0,02
1	2	67	2,674	1,08
1	3	78	22,129	1,55
2	1	156	8,341	0,49
2	2	162	100,418	22,48
2	3	191	5,929,945	3556,07
3	1	247	68,155	2,68
3	2	204	562,221	110,11
3	3	220	1,935,327	1325,83

In Table 3 are presented the metrics for all combinations of robots and dishes. In this analysis, we focus on grouping the results based on the number of dishes, creating three distinct groups corresponding to configurations with 1, 2, and 3 dishes, as highlighted in the table. Following this, we will present and discuss the corresponding graphs for each complexity metric.

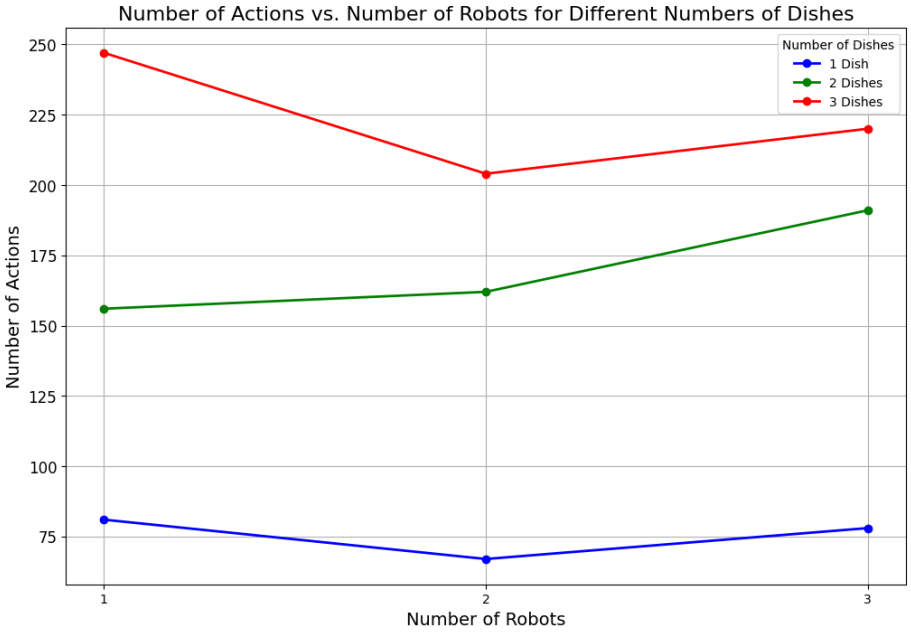

Figure 5: Number of Actions vs. Number of Robots for Different Numbers of Dishes

Figure 5 illustrates the number of actions required as the number of robots increases for varying numbers of dishes. This graph clearly demonstrates that altering the number of robots has minimal impact on the total number of actions. For each number of dishes (represented by the blue, green, and red lines), varying the number of robots does not yield significant changes, and no consistent pattern emerges indicating an increase or decrease in the number of actions as the number of robots is incremented.

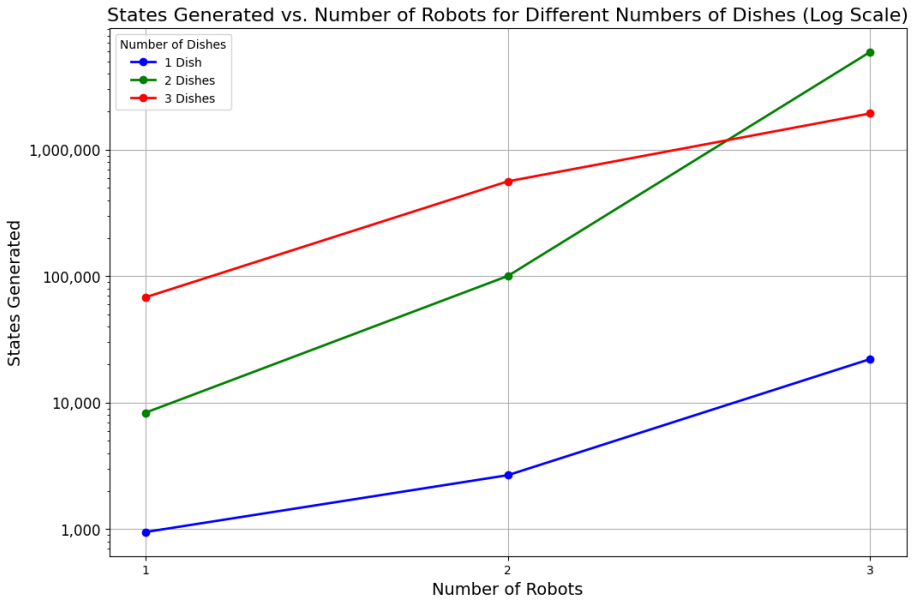

Figure 6: States Generated vs. Number of Robots for Different Numbers of Dishes

Figure 6 displays the relationship between the number of states generated as the number of robots increases for varying numbers of dishes. Firstly, it is important to note that the scale of the graph is logarithmic. The graph exhibits a clear positive correlation between the number of robots and the number of states generated, as evidenced by the approximately straight lines. In a logarithmic scale, such linear trends typically indicate an exponential relationship in the data. Therefore, this suggest that the number of states increases exponentially with each additional robot, which adversely affects the performance in this metric.

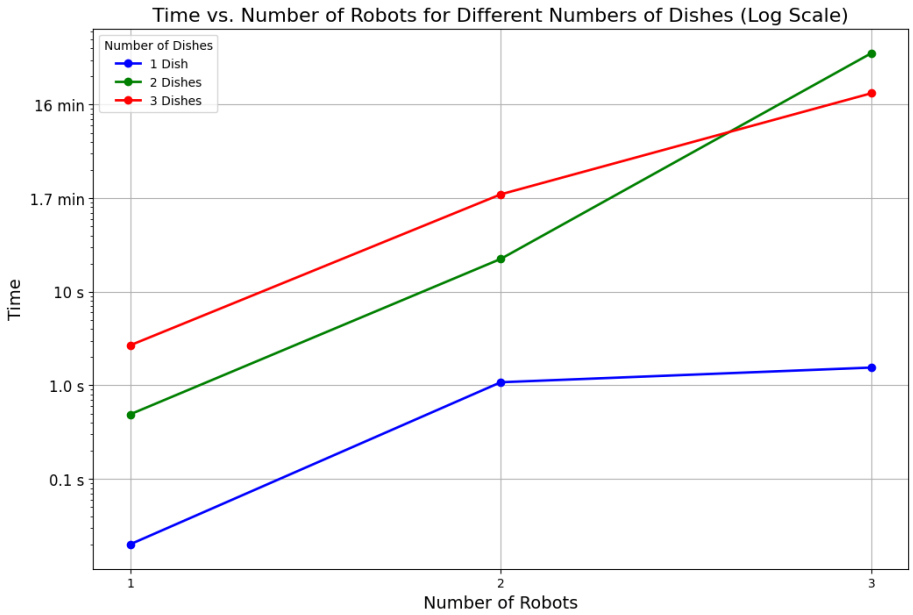

Figure 7: Time vs. Number of Robots for Different Numbers of Dishes

Figure 7 presents the computational time required to solve the problem as the number of robots increases for varying numbers of dishes. Similar to the previous graph, the scale is logarithmic, and the trends closely mirror those observed in the number of states generated, as in the previous study.

In summary, increasing the number of dishes makes the problem more complex in all cases, leading to an increase in the number of actions, generated states, and computation time.

6 Conclusions

In this project, we implemented and evaluated a robot sushi restaurant system using PDDL and the Metric-FF planner, incorporating numeric fluents and types to model complex scenarios. We conducted a series of test cases with varying number of dishes and robots, and assessed the planner's performance based on key metrics such as plan length and states generated. Additionally, we introduced unexpected situations for the robot to manage, such as handling broken tools and selecting their replacements.

Our analysis reveals that solution complexity is significantly influenced by the number of dishes. As illustrated in the graphs and tables, increasing the number of dishes the robots have to prepare leads to a marked rise in plan length, generated states, and computation time.

We have also observed that adding more robots leads to a poor trade-off between potential optimization gains and the increased complexity introduced for the planner. Since only one action can be executed at any given moment, and each action is assigned to a single robot, there is no real reduction in the number of steps in the plan. This limitation holds particularly when aiming for an optimal plan and when all robots are positioned within the same room; otherwise, this observation may not apply. Nonetheless, even in such cases, the trade-off remains unfavorable.

Overall, this project demonstrates that planning with Metric-FF can successfully perform this task; however, it is indeed approaching the limits of its capabilities. As the complexity increased, we encountered significantly longer planning times and were unable to request optimal solutions, as the planner could not manage the required computational load. This suggests that very complex scenarios may be beyond the feasible scope of this tool.