# Knowledge Representation and Reasoning
# MCS Portfolio Report

Maitri Ravikant Dodiya
Arizona State University
Tempe, US
mdodiya@asu.edu

## ABSTRACT
This document is a report outlining the challenges, solution approach and the associated complexities of the automated warehouse scenario problem. Moreover, multiple contingencies have been reported for future use. Details of the solution formulated, a snapshot of the result and application modules can be found below.

## Keywords
ASP, Clingo, Robots, Automation, Knowledge Representation and Reasoning (KRR).

## 1. INTRODUCTION
This is a portfolio report for the Knowledge Representation and Reasoning (CSE 579) course taken in Summer 2022. For this course one of the tasks we were required to complete was an individual project related to the concepts taught in class. We were supposed to propose a solution to either of the problems – Automated Warehouse Scenario or Insurance Referee Assignment. I chose Automated Warehouse Scenario.

In the last few decades, automation has changed the world significantly. Robots are increasingly being employed in warehouses[1] due to the recent surge in online shopping. In logistics, the management of warehouses is crucial for overseeing transportation and reducing manpower used. Optimization of movements made by robots in warehouses would lead to faster attainment of orders. The applications of this project are widespread ranging from storage and retrieval to supply chain management and many more. The use of robots in these areas leads to faster and more efficient delivery compared to manual labor. Programming robots to do these tasks will lead to flexible shipment of goods.

Entities and constraints to achieve the desired outcome for the aforementioned problem is a knowledge representation task[2]. One of the 2019 Answer Set Programming (ASP) Challenge problems was an automated warehouse scenario presented by Martin Gebser and Philipp Obermeier. Here, an automated warehouse scenario is considered where robots deliver products to picking stations to fulfill orders. This is a simplified version of Amazon's warehouse where robots can move horizontally or vertically between adjacent cells in a rectangular grid representing the warehouse. The robots are flat and move underneath shelves to pick up orders except when carrying orders. To fulfill orders, robots have to carry shelves with appropriate orders to the correct picking stations. The end goal is to fulfill orders in the least time possible, where time refers to the number of actions (steps) taken. Each robot can only take at most one step at a point in time.

## 2. DESCRIPTION OF SOLUTION
The problem describes warehouse management using Robots. It also asks to reach the most optimal solution from available ones, ie. a solution where the number of steps (time taken) to achieve the goal is minimized. Consequently, I decided to encode the problem details such as grid and entities like robots and items into the initial conditions [3] [4]. Apart from this, constraints were added to remove solutions which were invalid.

Throughout the process of reaching the solution, I referred to the official documentation of Clingo [5] as well. Initially, I created sketches of the instances provided and a logical solution imitating how the robots move. Along with that, I thought of a way to convert the input files to a simple format. I took an iterative approach to arrive at the best formulation of the problem in clingo. In terms of the scope of this project, I outlined the hard constraints and other requirements needed to ensure that the solution worked correctly.

In this scenario there is a g x g grid provided in the instance. There are m robots and n items. The task is to move each item from the initial position $(x_1, y_1)$ to $(x_2, y_2)$ utilizing less than or equal to m robots.

Consider Fig 1 below, representing an instance of this problem. A 4 x 4 gird has been provided. Here, R(i) denotes $i^{th}$ robot. Similarly, S(i) denotes the $i^{th}$ shelf.

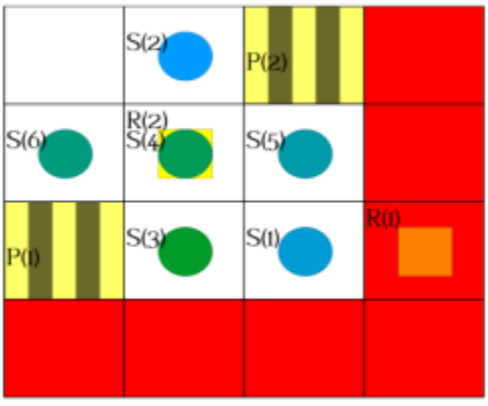


Fig 1. Visual representation of an instance

The various solution components used are as follows:

**1. Goal State:** Goal state determines the definition of being done with the instance ie. the state when a scenario is complete. In this case, the quantity of items in the order must be zero to ensure all items are delivered.

**2. Object Declarations:** These rules help define the domain, also known as the environment or world. Rules here assert things like robot location, picking stations, what an order is, etc.

**3. Fluents:** These are assisting rules to the object's domain.

**4. Law of Inertia:** These are inertial constraints placed to ensure that two robots can't occupy the same place at the same time. Apart from that, they can't switch places with each other at the same instance of time.

**5. Robot Actions:** These rules assert the different types of actions a robot can perform. This includes actions like picking an item up, putting it down, moving, or sitting idle. Moving encompasses other sub-actions such as moving up, down, left, or right.

All robot movements and state entities must follow the below criteria[6]:

● Valid movements for robots are vertical or horizontal.

● Robots cannot move diagonally or randomly from one cell to another cell.

● More than 1 robot cannot be simultaneously present in the same cell.

● If more than one robot is eligible for picking up an item, they must choose the owner of the task based on the distance of the robot from the item.

● If more than 1 robot needs to access the same cell, the order of accessing the order needs to be serialized.

● A robot cannot pick up the same order once it has been picked up by another robot.

● Target location must not contain an item already as it will lead to infinite wait for the robot.

● A cell in the highway must not be blocked by an item as it will lead to an infinite wait time for the robot.

● A robot cannot pick up a shelf that is already on it.

● A robot can only pick up a shelf if it is on the node containing the shelf.

● A robot cannot put down a shelf on the highway.

● A robot can only put down an item if it has picked it up.

● Two shelves cannot be on the same cell.

● The effect of fulfilling an order is that the quantity of pending orders reduces by one. This happens every time any robot fulfills its task successfully.

● Picking stations or shelves cannot be on a highway.

● Robots cannot move outside the grid.

● Robots cannot deliver items more than the number of orders or products.

● Once an item is picked up by a robot, it cannot pick another task until the delivery of the order is fulfilled.

Adding a constraint to a program eliminates answer sets that satisfy the body of the constraint. For this particular program, the constraints added ensured that invalid solutions such as those where two robots occupy the same position were eliminated.

## 3. DESCRIPTION OF RESULTS

Five instances were provided to test our solution on. These are Inst1.asp, Inst2.asp, Inst3.asp, Inst4.asp, and Inst5.asp.While testing the code on these instances, I learnt that if enough steps are not provided, the model will remain unsatisfiable. Hence, the minimum steps to find a satisfiable answer would be considered the optimal solution.

The optimal solution of Instance 1 is provided below in Fig 2. In the solution,

- object (robot, i) denotes that $i^{th}$ robot is an object in the grid.
- move (x, y) denotes moving an object to position (x, y).
- deliver (a, b, c) denotes delivery of order id a, item id b with number of units c.
- occurs (object (robot, i), action, t) denotes $i^{th}$ object performing the action at timestamp t.
- pickup denotes a robot picking up an order from a shelf.
- Putdown denotes a robot putting an order down.
- timeTaken(t) represents the total timeTaken to reach the optimal solution.
- numActions(n) represents the total number of actions performed to reach the optimal solution.

```
Solving...
Answer: 1
occurs(object(robot,1),move(-1,0),0) occurs(object(robot,1),
move(-1,0),1) occurs(object(robot,2),move(0,-1),1) occurs(object(robot,2),
move(1,0),2) occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),
move(0,1),5) occurs(object(robot,1),move(0,-1),6) occurs(object(robot,2),
move(0,-1),7) occurs(object(robot,1),
move(0,1),8) occurs(object(robot,2),pickup,0) occurs(object(robot,1),pickup,2)
occurs(object(robot,2),pickup,6) occurs(object(robot,1),pickup,7)
occurs(object(robot,2),putdown,4) occurs(object(robot,1),putdown,5)
occurs(object(robot,2),deliver(2,2,1),3) occurs(object(robot,1),deliver(1,1,1),4)
occurs(object(robot,2),deliver(3,4,1),8) occurs(object(robot,1),deliver(1,3,4),9)
timeTaken(9) numActions(19)
Optimization: 64
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 64
Calls        : 1
Time         : 0.580s (Solving: 0.45s 1st Model: 0.08s Unsat: 0.37s)
CPU Time     : 0.577s
```

Fig 2. Snapshot of the optimal solution for Instance 1.

I have minimized the number of actions needed to deliver the product by the robots. When we analyze the approach, we find that:

● All the instances ran successfully in optimal time and satisfiable models for each of them were found with the robots reaching the goal state.

● As evident in the solutions presented above, as the number of robots increases, the delivery time reduces. This happens because

more robots mean we can utilize as many empty cells as possible leading to few or no accidents.

● However, as the number of orders increases, the time taken for delivery also increases. This is expected because robots are only allowed to carry one shelf at a time.

● The use of #minimize allowed the models to reach an optimum solution in a minimum number of steps.

● Some rules required multiple revisits due to their complex nature and challenges faced in translating logical solutions to ASP based code in Clingo.

● The solution set increases exponentially with the increase in the size of the warehouse and the number of robots.

● However, an increase in the number of robots can also facilitate more tasks picked up in parallel.

● Increasing the number of tasks while keeping the number of robots constant will lead to an increase in the number of steps required for achieving the solution.

● As long as more than 1 item does not want to occupy the same final position, a solution is guaranteed to be reached.

| Instance File | Optimization | CPU Time |
|---|---|---|
| Inst1.asp | 64 | 0.577s |
| Inst2.asp | 72 | 1.142s |
| Inst3.asp | 31 | 0.290s |
| Inst4.asp | 20 | 0.480s |
| Inst5.asp | 31 | 0.520s |

Table 1. Optimal steps for each instance.

As shown in Table 1, the various CPU Time required to reach the goal and optimization for each instance is listed.

In summary, the CPU times vary depending on the scenario. The proposed solution was able to satisfy all the instances. Hence, this solution meets the project specifications.

# 4. CONTRIBUTION

This was an individual project. Hence, I was the only person responsible for creating a logical solution for each instance, converting the solution to descriptive logic and writing it in the language of Clingo. Consequently, I also created new test cases to determine the validity of my solution.

I downloaded and installed the windows version of clingo-5.4.0 using the official documentation for Clingo. I learnt keywords such as #minimize, #maximize, #optimize, and -t from the Answer Set Programming module. The -t4 (parallelization) and #optimize (optimization) features were used to reach the optimal solution quickly.

Performed multiple runs of the program on various inputs to identify patterns. I understood that as the number of robots increases, the time taken to find the optimal reduces unless it is an overcrowded grid. Hence, to optimize warehouse management the number of robots should also be optimally picked.

# 5. LESSONS LEARNED

In the course of finding a solution to the Automated Warehouse Scenario I learnt a lot. I am happy to have been presented with the opportunity to work on such an interesting problem. I can confidently say that I have implemented the concepts learnt in this course on a real-world scenario in a satisfactory manner.

The learnings I took away from this project are as follows.

● I learnt the value of applying knowledge representation techniques and the benefits of applying them on real world problems[7][8].

● I gained deep knowledge of the programming language - Clingo.

● I also learnt how to run threads in parallel using -t4(4 is the number of threads), optimize solutions and formulate constraints.

● I obtained experience in configuring an environment with Clingo.

● I learnt the importance of breaking a logical solution into smaller parts and then writing the code for it.

● An important takeaway is that one must test the code after each line is written and to think in a procedural manner. This helps in debugging scenarios where the code is missing a small detail resulting in “soft failures”.

● I learnt the value of commenting code. This helps in retracing the logic as well as a statement’s importance in the complete solution.

# 6. REFERENCES


[1] Schieweck, Steffen, Gabriele Kern-Isberner, and Michael ten Hompel. "Answer Set Programming for a Large, Knowledge-intense Domain: Practical Support of Warehouse Design." (2018).

[2] Van Harmelen, Frank, Vladimir Lifschitz, and Bruce Porter, eds. *Handbook of knowledge representation*. Elsevier, 2008.

[3] Tran Cao Son, Answer Set Programming. October 2005.

[4] Lifschitz, Vladimir. "More About the Language of Clingo." *Answer Set Programming*. Springer, Cham, 2019.

[5] Gebser, Martin, et al. "A user’s guide to gringo, clasp, clingo, and iclingo." (2008).

[6] Palù, Alessandro Dal, et al. "Answer set programming with constraints using lazy grounding." *International Conference on Logic Programming*. Springer, Berlin, Heidelberg, 2009.

[7] Azadeh, Kaveh, René De Koster, and Debjit Roy. "Robotized and automated warehouse systems: Review and recent developments." *Transportation Science* 53.4 (2019): 917-945.

[8] Erdem, Esra, Michael Gelfond, and Nicola Leone. "Applications of answer set programming." AI Magazine 37.3 (2016): 53-68.