Automated Warehouse Scenario – Project Report

Shashank Davalgi Omprakash
1219510734 - sdavalgi@asu.edu

Abstract

The Automated warehouse scenario for product pickup, dropping at any specific locations at huge warehouses like Amazon, is important for optimal product deliveries. In this project, I have implemented an automated warehouse scenario in which robots pick up orders from the picking stations and deliver to fulfill orders. I have used Answer Set Programming with CLINGO to implement this project by inducing the knowledge gained by the course CSE 579 – Knowledge Representation and Reasoning taught by Dr. Joohyung Lee. I've learned many concepts and programming at some point of the task implementation with useful insights into future applications.

Problem Statement

The use case is to deliver the packages by the robots located in the warehouse. Robots need to pick up shelves, present over the warehouse, and drop them at the picking stations. It reflects the real-world automated warehouse scenario. A warehouse is denoted as a rectangular grid, and the robots can move horizontally or vertically but not diagonally. The robots are flat and can move underneath shelves and pick them up. While the robots move around, pick up and put down shelves, deliver shelves, deliver products, or remain idle, but there must be no crashes, i.e., no robot may move into or switch its cell with another one from one step to the next. There is something called highways, where no shelves may be put down at such cells. The overall goal is to fulfill all the orders in as little time as possible, where time is counted in steps and each robot may (but does not have to) perform one action per time step.

Project Background

The Automated Warehouse Scenario is a typical and important backbone of various E-commerce platforms and other business corporations serving a myriad of customers with a multitude range of products from various domains. The automated warehouse is a necessity for faster, optimal, and profitable delivery of products and completion of order transactions in the least amount of time.

The simplified version of the automated warehouse scenario serves as a standard example for the application of the Knowledge representation, Answer set programming, and Clingo. I have written this project in answer set programming language with **CLINGO**. Clingo tackles NP-Hard class problems using Answer set solvers. Answer set solvers are building blocks of ASP. It uses these Answer set solvers to create stable models to solve similar problems. The Answer set solvers reduce the search problems into computing stable models. The **Davis–Putnam–Logemann–Loveland(DPLL) algorithm** is a complete, backtracking-based search algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form, i.e., for solving the CNF-SAT problem. An enhanced version of the DPLL algorithm is used to design the answer set solvers.

The **ASP Challenge Problem: Automated Warehouse Scenario** is used as a base description for this project. All the requirements and instructions from this are utilized in the implementation of the project. Subsequently, I have referenced and applied the topics especially - exogeneity and commonsense law of inertia into this project. The project is comparable to the Blocks world problem illustrated in the course. Thus, the blocks world problem lecture from the course is referred to as the foundation and starting point for the project implementation. The class lectures and notes were very helpful and handy on the course of the project implementation and learning.

Approach to Solve the Problem

Project is divided into various subtasks as mentioned below:

- At the beginning, Grid and Highway rules were written.
- In the X-axis and the Y-axis, the robot movement is implemented.
- Shelf pickup, putdown is implemented.

- Delivery to the picking station and constraints are completed.

The below images show the grid and product constraints written in as an example. node(C, D) defines a node X present at the location (X, Y). highway(C, D) defines a highway H to be present at a location (C, D). pickingStation(P, C, D) defines a pickingStation to be present at the location (C, D).

```
%%%%%%%%%%%%%%%%%%%%%%%
% Grid Constraints
%%%%%%%%%%%%%%%%%%%%%%%
node(C, D):- init(object (node, X) , value (at,
pair(C, D))).

pickingStation(P, C, D):- init(object(
pickingStation, P), value(at, pair(C ,D))).

highway(C, D):- init(object(highway, H), value(at,
pair(C ,D))).
```

Figure 1. Grid Constraints

The above-mentioned tasks were on the given sample instances. Test cases include - moving the robots along the positive X-axis, positive Y-axis, negative X-axis, negative Y-axis, picking up the shelf, putting down the shelf, delivering shelf to specific pickup stations is performed. I also wrote rules for not letting two robots pass through each other. Next, I implemented Commonsense Law of Inertia to carry out the four aspects given in the problem such as the item's location, the robot's location, the carry action state, and the order state. The below image shows the logic for the same. The rule at() denotes an object X of item I at the location (C, D) at time T and will be in the same place at time T+1. The rule on() denotes an object X at location (S, Y) and shelf K at the time T will be in same place at time T+1. The carry() rule denotes a robot R carrying shelf S at time T will still be carrying it at the time T+1. And finally, the order() rule donates an order) with E amount of items of type I at time T will have the same amount as well as time T+1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constraints for the Commonsense law of inertia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
{at(object (X ,I), C, D, T + 1)} :- at(object (X,
I), C, D, T), T = 0..m-1.

{on(X, K, S, Y, T+ 1)} :- on(X, K, S, Y, T), T =
0..m-1.

{carry(R, S, T + 1)} :- carry(R, S, T), T = 0..m-1.

{order(O, I, E, T + 1)}:- order(O, I, E, T), T =
0..m-1.
```

Figure 2. Constraint for Commonsense law of inertia

Main Results and Analysis

All the sample instances given for the project were executed. Also, all the test cases were covered and executed, respectively. Clingo was used for project execution and implemented of the warehouse scenario with all the actions, instances, and constraints strictly as per the project description.

The stable model solutions from clingo execution allowed us to view all the robotic actions and all the actions associated with the completion of the order(s). These actions are tracked as an atomic operation in terms of time steps. The atom changes with every time step change. So, all the actions executed in the program are considered as atoms associated with time steps.

The Table 1 tabulates the minimum number of steps required for all the given five simple instances.

Simple Instances	Timesteps Required
1	**13**
2	**11**
3	**7**
4	**10**
5	**6**

Table 1: Minimum number of time steps required for solving given simple instances for the project.

Using two threads and Intel i5 10^{th} Gen quad-core CPU, the test cases were run. From the above table, I analyzed that the time taken by the solver was more when the time steps was more. And the third and the fifth instance took less than a second to complete. Below are the screenshots for couple of the instances.

```
clingo version 5.3.0
Reading from warehouseScenario.lp ...
Solving...
Answer: 1
occurs(object(robot,1),move(-1,0),1) occurs(object(robot,2),move(-1,0),1)
occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),move(0,1),3)
occurs(object(robot,1),move(-1,0),5) occurs(object(robot,2),move(0,-1),5)
occurs(object(robot,2),move(1,0),7) occurs(object(robot,1),move(1,0),8)
occurs(object(robot,2),move(1,0),8) occurs(object(robot,1),move(0,-1),9)
occurs(object(robot,2),move(0,-1),10) occurs(object(robot,1),move(1,0),11)
occurs(object(robot,1),move(0,-1),12) occurs(object(robot,2),move(1,0),12)
occurs(object(robot,1),pdown,7) occurs(object(robot,2),pdown,6) occurs(
object(robot,2),pkup,2) occurs(object(robot,1),pkup,4) occurs(object(
robot,2),pkup,9) occurs(object(robot,1),pkup,10) occurs(object(
robot,2),deliver(1,3,4),4) occurs(object(robot,1),deliver(1,1,1),6) occurs(
object(robot,2),deliver(3,4,1),11) occurs(object(robot,1),deliver(2,2,1),13)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 217.092s (Solving: 162.30s 1st Model: 162.29s Unsat: 0.00s)
CPU Time     : 376.448s
Threads      : 2        (Winner: 2)
```

Figure 3. Instance 1 output

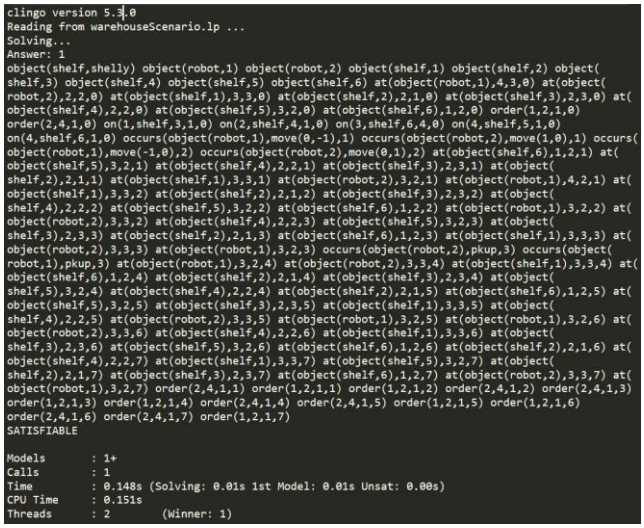

Figure 4. Instance 3 output

Figure 5. Instance 5 output

During the development of this project, I encountered few issues as mentioned below:

- Figuring out the edge scenarios in ASP to avoid the errors.
- Listing down all the constraints to avoid unexpected and improper results was time consuming.
- Figuring out the movements of robots in the grids with shelves and constraints.

Conclusion

For this course project, a simplified version of the Automated Warehouse scenario was assigned. This involved designing a solution to automate pick up, put down, and delivery of the products(shelves) at pickup stations in the warehouses with optimal and inexpensive steps. We approached this problem with all the knowledge and learnings from the course CSE 579 - Knowledge Representation and Reasoning taught by Dr. Joohyung Lee. Concepts of clingo was applied which devised an answer using the Answer set programming implemented in Clingo to achieve this goal.

Information gained from this course helped in developing the solution in conventional development, testing, integration, and standardization served in overall learning of the course. The answer to the problem consists of definitions and constraints for objects involved in the project – nodes, picking stations, robots, shelves, products, highways. And actions were robot actions – move, pick-up, put down, deliver. Fulfilling all the orders was the desired output for the scenario. Clingo answered the warehouse scenarios and produced minimal time solutions. For further development, we aim at improvising on the simplified version of the automated warehouse scenario which will involve more constraints and features from real world implementation.

Future Work

This project is a simplified version of the real-world scenario of Automated warehouses such as Amazon and others. In the future, this problem can be modified and extended further to cover more constraints and features which will bring it a step or two closers to the real world working of automated warehouses, such as:

- Additional structural accuracy and constraints need to be inculcated in the project.
- We can extend the project to account for warehouse locations.
- A case of multiple warehouses can be included.
- We can include warehouses employees also into consideration.
- An automated warehouse with manual interventions can be executed.
- Static and dynamic Restocking options for the shelves can be arranged.
- Restocking options concerning supply as per shipment truck arrives at the warehouse can be considered.

Overall, we can add various objects such as Restocking stations, multiple warehouses, separate robots for orders, and restocking. In summary, we can extend and upgrade this present project with more features that will enable it to inch towards the real world.

References

Fabricius, Felicitas & Bortoli, Marco & Selmair, Maximilian & Group, Bmw & Reip, Michael & Steinbauer, Gerald & Gebser, Martin. (2020). Towards ASP-based Scheduling for Industrial Transport Vehicles.

Martin Gebser and Philipp Obermeier. Automated Warehouse Scenario. http://www.mat. unical.it/~dodaro/aspchallenge2019/automated-w arehouse-scenario.package.zip

Gebser, Martin & OBERMEIER, PHILIPP & OTTO, THOMAS & Schaub, Torsten & SABUNCU, ORKUNT & Nguyen, Van & Son, Tran. (2018). Experimenting with robotic intra-logistics domains. Theory and Practice of Logic Programming. 18. 502-519. 10.1017/S1471068418000200.

Wikipedia contributors. (2020, May 20). Answer set programming. In Wikipedia, The Free Encyclopedia, from https://en.wikipedia.org/w/index.php?title=Answer_set_program ming&oldid=957788946

Wikipedia contributors. (2020, June 14). DPLL algorithm. In *Wikipedia, The Free Encyclopedia*, from https://en.wikipedia.org/w/index.php?title=DPLL_algorithm&old id=962444653

https://asprilo.github.io/

Shashank Davalgi Omprakash, Automated Warehouse Scenario, Progress Report.