CSE 579 Course Project: Milestone 4 - Individual Project Report
Automated Warehouse Scenario

Hannah O. Ajoge
Masters of Computer Science, Arizona State University.
hajoge@asu.edu or ohuajo@gmail.com

Problem Statement

This report applies answer set programming (ASP) to solve a practical real-world problem of robots being used to fulfill orders in a warehouse.

This problem involves a grid based warehouse that contain shelves and robots. The shelves carries specific products and the function of the robots is to retrieve specific shelve(s) of desired products and make them available at the right location, at the quickest possible time, based on given orders. This problem can be broken down into the following components/descriptions:

- Warehouse is rectangular grid of cells arranged in rows and columns just like a chess or sudoku board.
- Each shelve of product is situated on a cell, except cells that are designated as highways.
- Each robot can go to a specific cell, where a specific shelve of product is and pick the shelve; provided the robot is not already loaded.
- Each robot can go through any cell, except the robot is loaded and there is a shelve on its path. Then the shelve has to first be moved out of the way first.
- Robots can only perform one active at each time point and time is counted in steps.
- All robots do not have to be active at each time point.
- Robot activities include move around, pick up of shelves, put down of shelves and deliver products.
- Robots should not collide, regardless of their state of activity.

Project Background

This projects build on knowledge and concepts that was taught in the lecture videos on ASP. ASP is a declarative programming language which is useful for solving difficult knowledge-intensive problems especially NP-hard search problems. ASP is an offshoot of nonmonotonic nonmonotonic reasoning subfield of knowledge representation (Lifschitz 2008). Stable models of logic programming is the focal point of ASP semantics. Stable model semantics in ASP are based on application of the knowledge of autoepistemic logic and that of default logic for the analysis of negation as failure (Brewka, Eiter, and Truszczyński 2011).

The term ASP was first used just twenty years ago. At that time, the meaning of ASP was also formally elaborated. Just ten years after the inception of ASP, ASP became well recognized in the knowledge representation and logic programming communities (Marek, Niemela, and Truszczynski 2011). The origin of ASP can be traced back to the famous 1969 paper by McCarthy and Hayes. In this paper, McCarthy and Hayes introduced and established knowledge representation and reasoning as a prominent subfield of artificial intelligence. McCarthy and Hayes then referred to what is now known as knowledge representation as "epistemological part" and suggested first-order language as a formalization of knowledge representation (Marek, Niemela, and Truszczynski 2011). The limitations of first-oder language (such as monotonicity, lack of defaults and lack of a way to specify inductive definitions) led many researchers to seek an alternative logic. This eventually led to the birth of ASP as coined by Vladimir Lifschitz in 1999 (Marek, Niemela, and Truszczynski 2011).

The building blocks of answer set programs are atoms literals and rules. Atoms are factual statements that could be true or false. Literals are atoms and their negations. Where "a" (and other letters or phrases starting will small letters) represent atoms and "not a" represent the negation, rules are expressions of the following form:

$a \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n.$ (1)

The rule (1) infers that "a" the head is true if all the literals to the right of ← (known as the body) are true.

For example the rule below:

light_on ← power_on, not broken. (2)

The rule (2) means an assertion that the light is on, if it is established that the power is on and the lamp is not broken. Rules may not contain a body and in that case the ← can be omitted. For example the following two rules which are the same:
power_on ← . (3)
power_on . (4)
The rules (3) and (4) are called facts, as heads are regarded to be unconditionally true. Likewise a rule may have no head and be unconditionally regarded as false. For example the rule below:
← boy, girl. (5)
The rule (5) can be inferred to mean, there is no possibility of being a boy and also a girl.
The take-home here is that because of the strong declarative nature of ASP, "rapid prototyping
and development of software for solving search and optimization problems, and facilitates modifications and refinements leading to better performance" (Marek, Niemela, and Truszczynski 2011) is easily supported through simple and relatively few rules.

ASP is executed in tools known as answer set solvers. The first answer set solver is known as smodels, and subsequently others suchs as dlv, cmodels, pbmodels, Clasp/clingo, Wasp and dlv-hex and oClingo followed. For this project and the course as a whole, I utilized clingo. Clingo is as a solver, grounds and solve logic programs (https://potassco.org/).

Approach to Solving the Problem

This projects build on knowledge and concepts that was taught in the lecture videos on ASP. I found the Methodology of ASP: Generate-(Define)-Test Method video of week 4 very important to my generating clingo codes. I also frequently went back to check the videos from weeks 4 and 5, to grasp concepts and help in trouble shooting my codes.

I developed my clingo program by generating a search space of possible solutions and then add some constraints/actions to simplify my encoding. While my code is below in Appendix, just below are a summary of my encoding. In the generating phase of my code, I generated twenty seven (27) atoms which are named (bold and italicized) as following with their meanings in brackets:
rowSize (row size), ***colSize*** (column size), ***tRobots*** (total robots), ***tNodes*** (total nodes), ***tShelves*** (total shelves), ***tProducts*** (total products), ***tPStation*** (total picking stations), ***tOrders*** (total orders), ***node*** (specific grid), ***nodeAt*** (specific grid and it's coordinates), ***pair*** (specific grid coordinate), ***highway*** (specific highway), ***pickingStationAt*** (specific pickingstaion and the grid label), ***pickingStation*** (specific pickingstaions), ***robotAt*** (specific robot at specific node at specific time), ***robot*** (specific robot), ***shelfOn*** (specific shelve on specific node at specific time), ***shelf*** (specific shelf), ***productOn*** (specific product on specific shelf with quantity at specific time), ***product*** (specific product) ***orderAt*** (specific order at specific node which contains specific unit of items at specifi time), ***order*** (specific order), ***move*** (allowed robot movement, one step on x or y axis), ***robotMove*** (specific robot and its one step movement at specific time), ***pickUpShelf*** (specific robot picks up specific shelf at specific time), ***putDownShelf*** (specific robot put down specific shelf at specific time) and ***deliver*** (specific robot deliver specific order consisting of specific number of specific item on specific shelf at specific time)
In the testing stage of my code, I declared actions and constrains which can be summarized below using the descriptive comments from the code; "%%%" precedes subsections and "%" precedes intended instruct achieved by rule(s) detailed in appendix.

```
%%% Robots
% A robot can't execute 2 actions at same time
% No robot collision allowed
% A robot can only be on one node at a time
% Robot cannot move outside of the grid
% Effect and law of inertia of moving a robot
%% %Shelves
% No shelf on 2 nodes
% No 2 shelves on the same node
% No shelf on 2 locations (robot, node)
% Shelf cannot be put down on highway
% Disallow shelves on the highway
%%% Robot interaction with shelves and others
% Only one robot can pick a shelve
% Robots don't move through a node with shelf
% A robot handles one shelf at a time.
% A robot only pick shelves in the same node as the robot
% Only one robot can putdown shelve
% Robot has to have shelf to putdown
% No shelf on 2 robots
% No 2 shelves on the same robot
% Effect and law of inertia of picking up a shelf
%Effect  and law of inertia of putting down a shelf
%%% Orders and picking station
% Order must be fulfilled
% Delivery of the right item quantity and shelve at picking station
%Effect  and law of inertia of delivering a product
%%%Minimize time, show and constant
```

I executed my clingo program, which I named projectWarehouse.lp in clingo using the following commands:

- clingo projectWarehouse.lp simpleInstances/inst1.asp -t4
- clingo projectWarehouse.lp simpleInstances/inst2.asp -t4
- clingo projectWarehouse.lp simpleInstances/inst3.asp -t4
- clingo projectWarehouse.lp simpleInstances/inst4.asp -t4
- clingo projectWarehouse.lp simpleInstances/inst5.asp -t4

To visualize the robot's movements I had to manually use paper and pencil because I was unable to get Asprilo visualizer to work on my computer. Visualizing the movements, assisted me in testing and debugging my code.

Main Results and Analysis

The output of my clingo commands are as following:

For inst1.asp, 27 actions within 15 steps(time):

```
---------- truncated because too long ----------
Answer: 11
occurs(object(robot,1),move(0,-1),1)
occurs(object(robot,1),move(-1,0),2)
occurs(object(robot,1),move(1,0),4)
occurs(object(robot,2),move(1,0),4)
occurs(object(robot,1),move(0,-1),5)
occurs(object(robot,2),move(0,-1),5)
occurs(object(robot,1),move(-1,0),7)
occurs(object(robot,2),move(0,1),7)
occurs(object(robot,2),move(-1,0),9)
occurs(object(robot,1),move(0,1),10)
occurs(object(robot,2),move(0,1),10)
occurs(object(robot,1),move(-1,0),11)
occurs(object(robot,1),move(-1,0),12)
occurs(object(robot,2),move(-1,0),12)
occurs(object(robot,1),move(0,1),14)
occurs(object(robot,2),move(1,0),14)
occurs(object(robot,2),move(0,-1),15)
occurs(object(robot,2),pickup,1)
occurs(object(robot,1),pickup,3)
occurs(object(robot,2),pickup,11)
occurs(object(robot,1),pickup,13)
occurs(object(robot,2),putdown,8)
occurs(object(robot,1),putdown,9)
occurs(object(robot,2),deliver(2,2,1),6)
occurs(object(robot,1),deliver(3,4,1),8)
occurs(object(robot,2),deliver(1,1,1),13)
occurs(object(robot,1),deliver(1,3,4),15)
Optimization: 120
---------- truncated because too long ----------
OPTIMUM FOUND
Models       : 11
  Optimum    : yes
Optimization : 120
Calls        : 1
Time         : 41.591s (Solving: 41.18s 1st Model: 0.08s Unsat: 34.23s)
CPU Time     : 151.734s
Threads      : 4       (Winner: 4)
```

For inst2.asp, 22 actions within 11 steps(time):

```
---------- truncated because too long ----------
Answer: 32
occurs(object(robot,1),move(-1,0),1)
occurs(object(robot,2),move(0,1),1)
occurs(object(robot,1),move(0,-1),2)
occurs(object(robot,2),move(-1,0),3)
occurs(object(robot,1),move(1,0),5)
occurs(object(robot,2),move(1,0),5)
occurs(object(robot,1),move(0,1),6)
occurs(object(robot,1),move(0,1),7)
occurs(object(robot,2),move(0,-1),7)
occurs(object(robot,1),move(-1,0),8)
occurs(object(robot,1),move(-1,0),9)
occurs(object(robot,1),move(-1,0),10)
occurs(object(robot,2),move(1,0),10)
occurs(object(robot,1),move(0,-1),11)
occurs(object(robot,2),move(0,-1),11)
occurs(object(robot,2),pickup,2)
occurs(object(robot,1),pickup,3)
occurs(object(robot,2),pickup,8)
occurs(object(robot,2),putdown,6)
occurs(object(robot,2),deliver(1,1,1),4)
occurs(object(robot,1),deliver(1,3,2),12)
occurs(object(robot,2),deliver(2,2,1),12)
Optimization: 78
---------- truncated because too long ----------
OPTIMUM FOUND
Models       : 32
  Optimum    : yes
Optimization : 78
Calls        : 1
Time         : 7.746s (Solving: 7.51s 1st Model: 0.10s Unsat: 1.90s)
CPU Time     : 29.770s
Threads      : 4       (Winner: 4)
```

For inst3.asp, 13 actions within 8 steps(time):

```
---------- truncated because too long ----------
Answer: 21
occurs(object(robot,1),move(-1,0),1)
occurs(object(robot,1),move(0,-1),2)
occurs(object(robot,1),move(1,0),4)
occurs(object(robot,2),move(1,0),4)
occurs(object(robot,1),move(0,-1),5)
occurs(object(robot,2),move(0,-1),5)
occurs(object(robot,1),move(-1,0),7)
occurs(object(robot,2),move(0,1),7)
occurs(object(robot,2),move(-1,0),8)
occurs(object(robot,1),pickup,3)
occurs(object(robot,2),pickup,3)
occurs(object(robot,2),deliver(1,2,1),6)
occurs(object(robot,1),deliver(2,4,1),8)
Optimization: 36
---------- truncated because too long ----------
OPTIMUM FOUND
Models       : 21
  Optimum    : yes
```

Optimization : 36
Calls : 1
Time : 0.792s (Solving: 0.64s 1st Model: 0.02s Unsat: 0.24s)
CPU Time : 2.582s
Threads : 4 (Winner: 4)

For inst4.asp, 19 actions within 10 steps(time):

---------- truncated because too long ----------
Answer: 24
occurs(object(robot,1),move(0,-1),1)
occurs(object(robot,2),move(0,-1),1)
occurs(object(robot,1),move(-1,0),2)
occurs(object(robot,1),move(-1,0),3)
occurs(object(robot,2),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),6)
occurs(object(robot,1),move(1,0),7)
occurs(object(robot,2),move(1,0),7)
occurs(object(robot,2),move(-1,0),9)
occurs(object(robot,1),move(1,0),10)
occurs(object(robot,2),pickup,2)
occurs(object(robot,1),pickup,4)
occurs(object(robot,2),pickup,8)
occurs(object(robot,2),putdown,4)
occurs(object(robot,1),deliver(2,2,1),8)
occurs(object(robot,1),deliver(3,2,2),9)
occurs(object(robot,2),deliver(1,1,1),10)
Optimization: 55
---------- truncated because too long ----------
OPTIMUM FOUND
Models : 24
Optimum : yes
Optimization : 55
Calls : 1
Time : 5.872s (Solving: 5.52s 1st Model: 0.02s Unsat: 2.24s)
CPU Time : 21.860s
Threads : 4 (Winner: 4)

For inst5.asp:

---------- truncated because too long ----------
Answer: 36
occurs(object(robot,1),move(-1,0),1)
occurs(object(robot,2),move(0,1),1)
occurs(object(robot,1),move(0,-1),2)
occurs(object(robot,1),move(-1,0),3)
occurs(object(robot,1),move(-1,0),4)
occurs(object(robot,2),move(-1,0),4)
occurs(object(robot,1),move(0,1),6)
occurs(object(robot,2),move(1,0),6)
occurs(object(robot,2),move(0,1),7)
occurs(object(robot,2),pickup,2) occurs(object(robot,1),pickup,5)
occurs(object(robot,2),deliver(1,1,1),5)
occurs(object(robot,1),deliver(1,3,4),7)
Optimization: 28
---------- truncated because too long ----------
OPTIMUM FOUND
Models : 36
Optimum : yes
Optimization : 28
Calls : 1
Time : 0.805s (Solving: 0.46s 1st Model: 0.01s Unsat: 0.17s)
CPU Time : 2.106s
Threads : 4 (Winner: 4)

My result shows the actions taken by the robots to execute the orders in the five instances and the time (steps) taken. To execute the actions the robots carried out movements according to the given constraints, such as robots should not collide, robots should not go through shelves when carrying a shelf, robots should stay within the rectangular grid, etc.

Conclusion

This project utilizes simple intuitive modelling language known as ASP to solve hard computation problem of automating warehouse scenario involving multiple autonomous robots. In addition, this report provide opportunity for reader(s) to be informed about the origin and basics of ASP.

As I previously indicated in my progress report, the time available to execute the project is limited, especially for MCS online students with just seven (7) weeks to complete the course. For example, I may have more functional code if I was able to visualize my output with Asprilo instead of manually drawing each stage; which can easily be prone to error. My suggestion is that Asprilo should be incorporated earlier (weeks before), may be as part of milestone 2; and assistance (for example videos) for installation in different operating systems made available.

Overall this project is one of the most practical, real world utilization of ASP and I have executed to the best of my ability.

Opportunities for Future Work.

ASP which was initially mainly of an academic niche, now has much industrial use. As a person from biomedical background, I envisage that ASP will great for tackling hard system biology problems, as biochemical pathways could easily be represented in ASP. Specifically for this project, having nodes in the grid, were the robots cannot be will be interesting addition to the project.

.

References

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. ASP at a glance. *Communications of the ACM, 54(12): 92–103.* https://doi.org/10.1145/2043174.2043195

Lifschitz, V. 2008. "What Is ASP?", Proc. AAAI Conf. Artificial Intelligence, 1594-1597.

Marek, V.; Niemela, I.; and Truszczynski, M. 2011. Origins of Answer-Set Programming - Some Background And Two Personal Accounts. https://arxiv.org/pdf/1108.3281.pdf