ASP Challenge Problem:
Automated Warehouse Scenario

Nourhan ElNaggar - nelnagga@asu.edu
Arizona State University

Abstract-This portfolio report represents warehouse project for CSE579. Most of the content was taking from report project milestone 4 that was made earlier for documentation[6]

I. Introduction

With the growth of technology we are living nowadays, robots are becoming an essential part in the field of automation in order to fulfill human needs in the optimum possible way. As Kaveh Azadeh (2017) said on his research paper "Robotized handling systems are increasingly applied in distribution centers. They require little space, provide flexibility in managing varying demand requirements, and are able to work 24/7. This makes them particularly fit for e-commerce operations"[2].

For this problem it is about automating the process of delivering an order using robots. As Carolina Monory (2020) mentioned "That doesn't mean robots are taking over or stealing jobs from human workers. Rather, companies are taking advantage of the accuracy of machines, as well as their ability to work continuously for long hours without fatigue, to make warehouse tasks safer and more efficient. Some warehouse automation setups cover everything from unloading trailers to fulfilling orders, but humans are still part of the scene".[1] The problem can be described as follows:

- Simulates the real world with a grid that can be with any number of rows and columns.
- There exists number of robots that can do certain actions on the grid like moving forward and backward between the nodes, picking up the requested product from a shelf, putting down an order on a node and delivering an order to a pickup station.
- The problem contains constraints such as that two robots cannot cross each other on the grid and the robot cannot put the order down on certain nodes that is being called as "highway".

The objective is to make a robot deliver an order to the pickup station with the minimum number of actions on the grid so that it reaches the optimum solution.

II. Explanation of solution

The solution of this problem is divided into three parts, each part contains sections that describes the main rules or constrain needed to solve the problem, as well as the Clingo code below each description for demonstration.

The first part shows the main conditions and constraints that prevents the robot and shelf objects from being placed on a certain location with respect to the time T that the action was made on. Note that these objects need to be encoded before using these constraints on them.

A - Robot object constraints:

A.1- A constrain that prevents a robot from being at two locations at the same time:

```
:- robotLocation(RI ,object(node,L1),T),robotLocation(RI ,object(node,L2),T),
    L1!=L2, node(L1),node(L2),robot(RID), T=0..n.
```

A.2- A constrain that prevents two robots from being at the same node at the same time:

```
:- robotLocation(RI1 ,object(node,L1),T),
 robotLocation(RI2 ,object(node,L1),T),
 RI1!=RI2,  node(L1),robot(RI1),robot(RI2), T=0..n.
```

A.3- A constrain that prevents two robots from swapping their locations at a certain time:

```
:- robotLocation(RID1, object(node,L1),T),
 robotLocation(RID1, object(node,L2),T+1),
 robotLocation(RID2, object(node,L2),T),
 robotLocation(RID2, object(node,L1),T+1),
 robot(RID1),robot(RID2),
 node(L1),node(L2),  L1!=L2, RID1!=RID2.
```

B - Shelf object constraints:

B.1- A constraint that prevents a shelf to be on two different locations at the same time:

```
:-shelfLocation(SI ,object(node,L),T),
```

```
shelfLocation(SI , object(node,L1),T),
node(L),node(L1),shelf(SI),L!=L1,T=0..n.
```

B.2- A constraint that prevents a shelf to be with two robots at the same time:

```
:-shelfLocation(SI, object(robot,RI1),T),
 shelfLocation(SI, object(robot,RI2),T),
 shelf(SI),robot(RI1),robot(RI2),RI1!=RI2, T=0..n.
```

B.3- A constraint that prevents a robot from carrying two shelves at the same time:

```
:-shelfLocation(SI1, object(robot,RI),T),
shelfLocation(SI2, object(robot,RI),T),
 shelf(SI1),shelf(SI2),robot(RI), SI1!=SI2,T=0..n.
```

Second part shows the effects for each action a robot is able to do on the grid- moving, picking up shelf, putting down a shelf and delivering- with respect to a certain time.

A- Robot move effect:

If a robot is moving at time T form location L, he will be at location L1 at time T+1, where we use the pair of X and Y nodes and add the step of moving for each row and column.

```
robotLocation(RID,object(node,L1),T+1):-
 robotLocation(RID,object(node,L),T),
 location(L, pair(X,Y)),
 location(L1, pair(X+DX,Y+DY)),
 movingRobot(RID, move(DX,DY),T).
```

B- Robot Picks up a shelf effect:

The effect of that action is that the shelf location will be on the robot location at time T+1.

```
shelfLocation(SID,object(robot,RID),T+1):-
shelfLocation(SID,object(node,Loc_Robot),T),
robotLocation(RID,object(node,Loc_Shelf),T),
pickingUpShelf(RID,SID,T).
```

C- Robot puts down a shelf effect:

Effect for the shelf location will be at L which is the same location a robot is standing at in a certain time.The below rule can demonstrate the action constraint:

```
 shelfLocation(SID,object(node,L),T+1):-
puttingDownShelf(RID,SID,T),
shelfLocation(SID,object(robot,RID),T),
robotLocation(RID,object(node,L),T).
```

D- Robot delivers an order Effect:

A robot RID delivering an order with a delivered description of quantity at time T with the amount that has not been delivered at time T+1:

```
request(ORDER, object(node,L),
orderdesc(PRID,Left_Order-Delivered_Quantity),T+1):-
deliver(RID,ORDER, deliverdesc(SID,PRID,Delivered_Quantity),T),
request(ORDER, object(node,L),
orderdesc(PRID,Left_Order),T).
```

For the third part, it shows the general constraints that prevents the occurrence of action -that its effect was described on the second part- if it happened at a certain time:

A- Robot Picks up a shelf constraints:

A.1- a shelf can not be picked up by two robots at a certain time:

```
:- pickingUpShelf(RID1,SID,T),
pickingUpShelf(RID2,SID,T),RID1!=RID2,robot(R1),robot(R2),shelf(SID).
```

A.2- If a robot is carrying a shelf, he can not pick another:

```
:- pickingUpShelf(RID,SID1,T),
shelfLocation(SID2, object(robot,RID),T),SID1!=SID2.
```

A.3- Can not pick up a shelf if the same shelf is being carried with another robot:

```
:- pickingUpShelf(RID1,SID,T),
shelfLocation(SID, object(robot,RID2),T),RID1!=RID2.
```

B - Robot puts down a shelf constraints:

B.1- A shelf can only be put down by one robot at a certain time:

```
:- puttingDownShelf(RID1,SID,T),
puttingDownShelf(RID2,SID,T), RID1!=RID2,
robot(R1),robot(R2),shelf(SID).
```

B.2- A robot can only put down a shelf if he has a one:

```
:- puttingDownShelf(RID,SID,T),
 not shelfLocation(SID, object(robot,RID),T).
```

B.3- A highway H is a special node that a robot RID can not put down a node on it:

```
:- puttingDownShelf(RID,SID,T),
    robotLocation(RID, object(node,H),T), highway(H).
```

C- Robot delivers an order amount constraint:

- robot can not deliver an order unless its quantity is less than the product quantity:

```
:- deliver(RID,ORDER, deliverdesc(SID,PRID,Delivered_Quantity),T),
productLocation(PRID, object(shelf,SID),
productdesc(quantity,Product_Quantity),T),  Delivered_Quantity>Product_Quantity.
```

The main goal is to deliver all the order requests at the last maximum time n with the minimum number of actions:

```
:- not request(ORDER, object(node,_),orderdesc(PRID,0),T), T=n.
#minimize{1,ROBOT,ACTION,T:occurs(ROBOT,ACTION,T)}.
```

III. Results

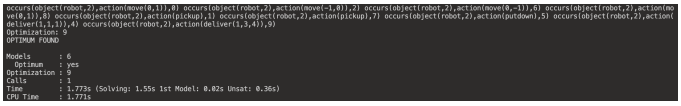
Below are each instance test case visualized with Aspirlo tool[3][4][5] to represent a grid of rows and columns, shelves as the circles with different colors for each product, robots are the squares, highways as the nodes with the light blue color, picking stations as the yellow squares. Red circles happen when a robot picks up a shelf at certain node.The below screenshots demonstrates the occurrence of each action that is showing on the terminal of running program.

The command line that was used to reach the below output:

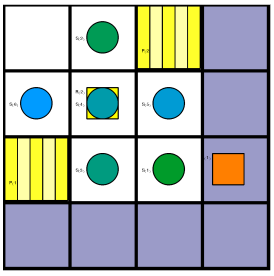
- clingo Automated-Warehouse.lp -c n=10 0

Where Automated-Warehouse.lp is the name of the code file and n is the maximum number of actions a robot can do to reach a solution.

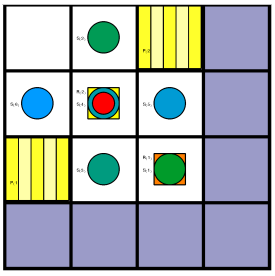
- Sample test case 1: Minimum of 10 Actions



Results for sample test 1

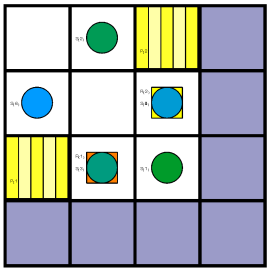


Initial test case



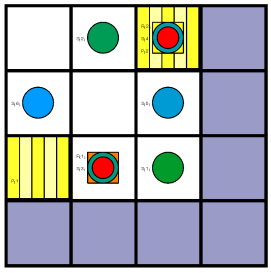
occurs(object(robot,1),action(move,(-1,0)),0).

occurs(object(robot,2),action(pickup,()),0).



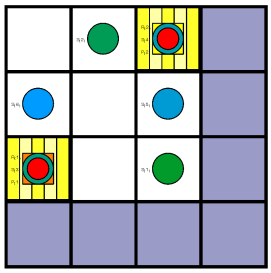
occurs(object(robot,1),action(move,(-1,0)),1).

occurs(object(robot,2),action(move,(1,0)),1).



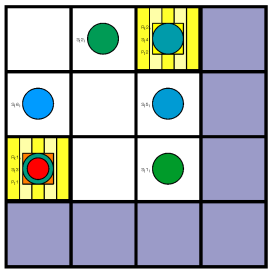
occurs(object(robot,2),action(move,(0,-1)),2).

occurs(object(robot,1),action(pickup,()),2).



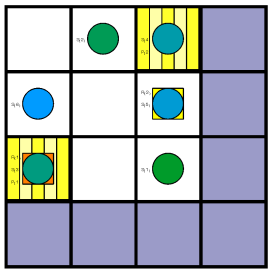
occurs(object(robot,1),action(move,(-1,0)),3).

occurs(object(robot,2),action(deliver,(2,2,1)),3).



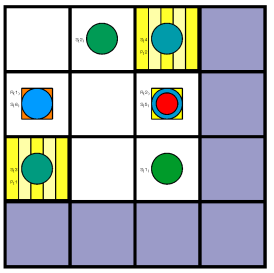
occurs(object(robot,2),action(putdown,()),4).

occurs(object(robot,1),action(deliver,(1,1,1)),4).



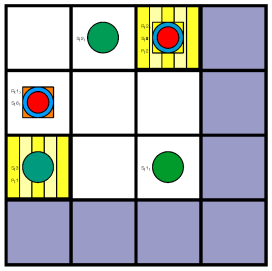
occurs(object(robot,2),action(move,(0,1)),5).

occurs(object(robot,1),action(putdown,()),5).



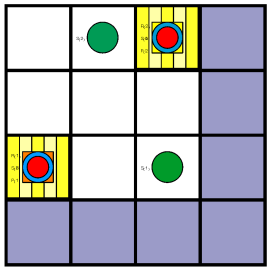
occurs(object(robot,1),action(move,(0,-1)),6).

occurs(object(robot,2),action(pickup,()),6).



occurs(object(robot,2),action(move,(0,-1)),7).

occurs(object(robot,1),action(pickup,()),7).

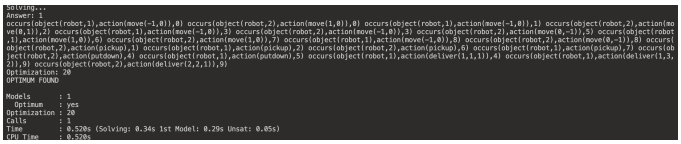


occurs(object(robot,1),action(move,(0,1)),8).

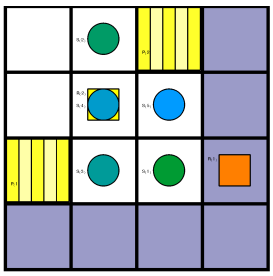
occurs(object(robot,1),action(deliver,(1,3,4)),9).

occurs(object(robot,2),action(deliver,(3,4,1)),9).

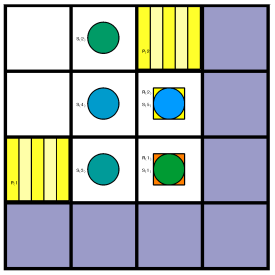
- Sample test case 2: Minimum of 10 Actions



Results for sample test 2

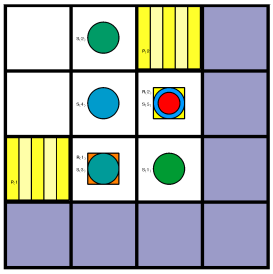


Initial test case



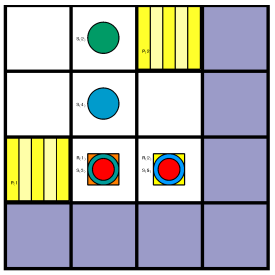
occurs(object(robot,1),action(move,(-1,0)),0).

occurs(object(robot,2),action(move,(1,0)),0).



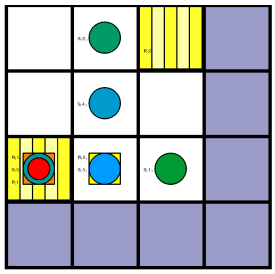
occurs(object(robot,1),action(move,(-1,0)),1).

occurs(object(robot,2),action(pickup,()),1).



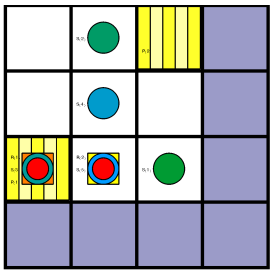
occurs(object(robot,2),action(move,(0,1)),2).

occurs(object(robot,1),action(pickup,()),2).



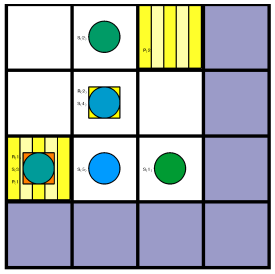
occurs(object(robot,1),action(move,(-1,0)),3).

occurs(object(robot,2),action(move,(-1,0)),3).



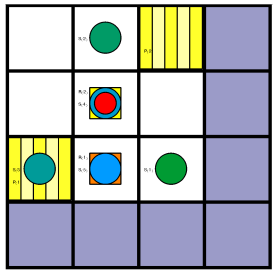
occurs(object(robot,2),action(putdown,()),4).

occurs(object(robot,1),action(deliver,(1,1,1)),4).



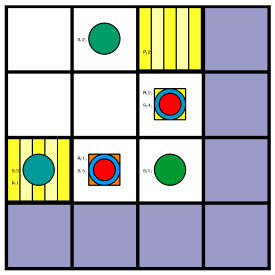
occurs(object(robot,2),action(move,(0,-1)),5).

occurs(object(robot,1),action(putdown,()),5).



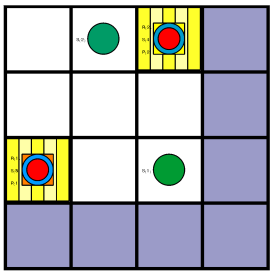
occurs(object(robot,1),action(move,(1,0)),6).

occurs(object(robot,2),action(pickup,()),6).



occurs(object(robot,2),action(move,(1,0)),7).

occurs(object(robot,1),action(pickup,()),7).



occurs(object(robot,1),action(move,(-1,0)),8).

occurs(object(robot,2),action(move,(0,-1)),8).

occurs(object(robot,1),action(deliver,(1,3,2)),9).

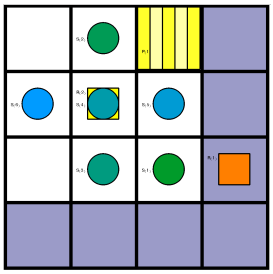
occurs(object(robot,2),action(deliver,(2,2,1)),9).

-Sample test case 3: Minimum of 7 Actions

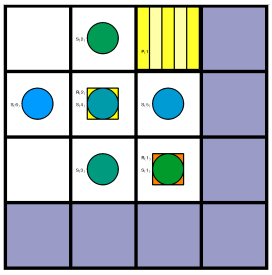
```
Answer: 4
occurs(object(robot,1),action(move(0,-1)),0) occurs(object(robot,1),action(move(-1,0)),1) occurs(object(robot,1),action(move(0,-1)),3) occurs(object(robot,2),action(move(0,-1)),3) occurs(object(robot,1),action(move(0,1)),5) occurs(object(robot,2),action(move(1,0)),5) occurs(object(robot,2),action(pickup),0) occurs(object(robot,1),action(pickup),2) occurs(object(robot,1),action(deliver(2,4,1)),4) occurs(object(robot,2),action(deliver(1,2,1)),6)
Optimization: 10
OPTIMUM FOUND

Models       : 4
  Optimum    : yes
Optimization : 10
Calls        : 1
Time         : 0.367s (Solving: 0.22s 1st Model: 0.04s Unsat: 0.17s)
CPU Time     : 0.366s
```

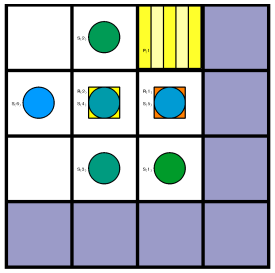
Results for sample test 3



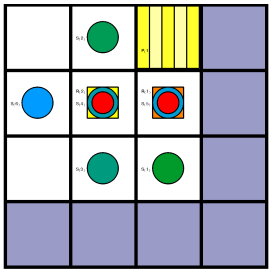
Initial test case



occurs(object(robot,1),action(move,(-1,0)),0).

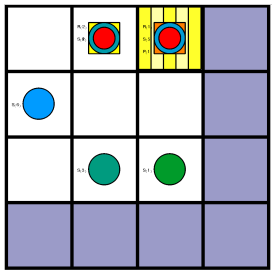


occurs(object(robot,1),action(move,(0,-1)),1).



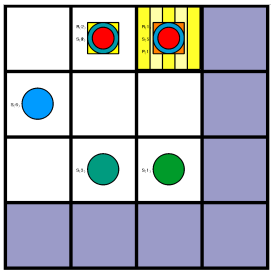
occurs(object(robot,1),action(pickup,()),2).

occurs(object(robot,2),action(pickup,()),2).

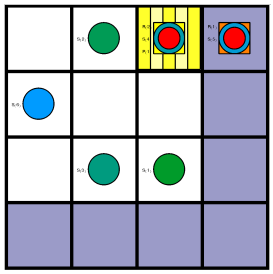


occurs(object(robot,1),action(move,(0,-1)),3).

occurs(object(robot,2),action(move,(0,-1)),3).



occurs(object(robot,1),action(deliver,(2,4,1)),4).



occurs(object(robot,1),action(move,(1,0)),5).

occurs(object(robot,2),action(move,(1,0)),5).

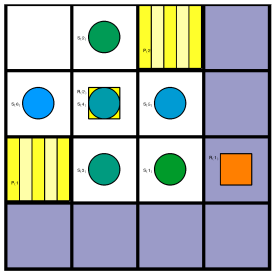
occurs(object(robot,2),action(deliver,(1,2,1)),6).

-Sample test case 4: Minimum of 5 Actions

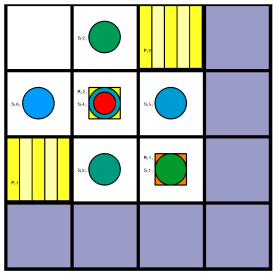
```
occurs(object(robot,1),action(move(-1,0)),0) occurs(object(robot,1),action(move(-1,0)),1) occurs(object(robot,2),action(move(1,0)),1) occurs(object(robot,2),action(move(0,-1)),2) occurs(object(robot,1),action(move(-1,0)),3) occurs(object(robot,2),action(pickup),0) occurs(object(robot,1),action(pickup),2) occurs(object(robot,2),action(deliver(2,2,1)),3) occurs(object(robot,1),action(deliver(1,1,1)),4) occurs(object(robot,2),action(deliver(3,2,2)),4)
Optimization: 10
OPTIMUM FOUND

Models       : 4
  Optimum    : yes
Optimization : 10
Calls        : 1
Time         : 0.374s (Solving: 0.23s 1st Model: 0.02s Unsat: 0.21s)
CPU Time     : 0.373s
```

Results for sample test 4

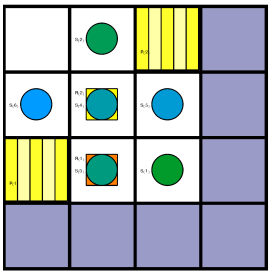


Initial test case

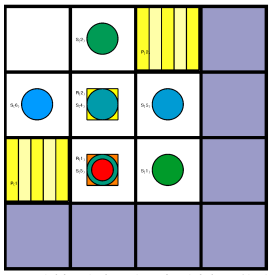


occurs(object(robot,1),action(move,(-1,0)),0).

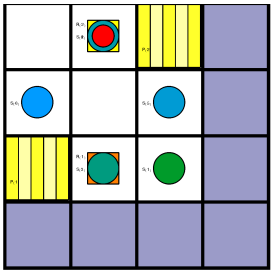
occurs(object(robot,2),action(pickup,()),0).



occurs(object(robot,1),action(move,(-1,0)),1).

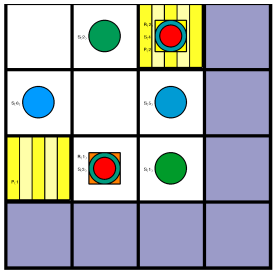


occurs(object(robot,1),action(pickup,()),2).



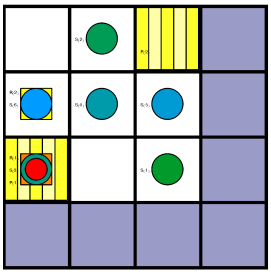
occurs(object(robot,1),action(move,(-1,0)),1).

occurs(object(robot,2),action(move,(0,-1)),1).



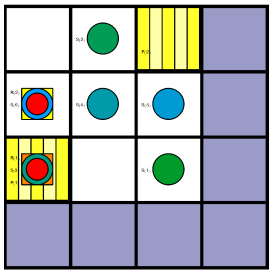
occurs(object(robot,2),action(move,(1,0)),2).

occurs(object(robot,1),action(pickup,()),2).



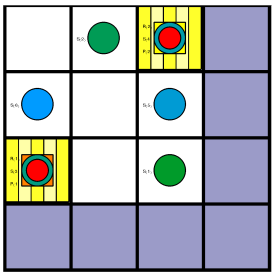
occurs(object(robot,1),action(move,(-1,0)),3).

occurs(object(robot,2),action(move,(-1,0)),3).



occurs(object(robot,2),action(pickup,()),4).

occurs(object(robot,1),action(deliver,(1,1,1)),4).

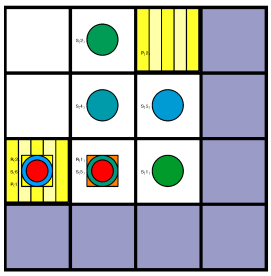


occurs(object(robot,1),action(move,(-1,0)),3).

occurs(object(robot,2),action(deliver,(2,2,1)),3).

occurs(object(robot,1),action(deliver,(1,1,1)),4).

occurs(object(robot,2),action(deliver,(3,2,2)),4).



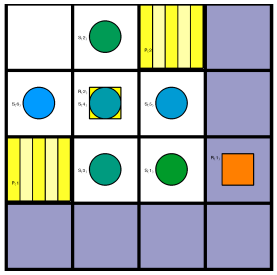
occurs(object(robot,1),action(move,(1,0)),5).

occurs(object(robot,2),action(move,(0,1)),5).

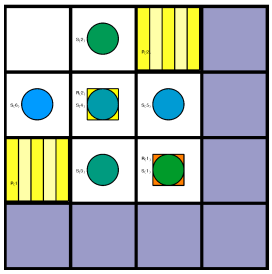
occurs(object(robot,2),action(deliver,(1,3,4)),6).

- Sample test case 5: Minimum of 7 Actions

Results for sample test 5



Initial test case



occurs(object(robot,1),action(move,(-1,0)),0).

IV. Lessons learned

Solving a problem with logic programming made me learn how to look at the abstract view of solving the problem by writing the needed rules and constraints without telling the program how to solve the problem. Moreover, this course taught me that reasoning about the needed action is very powerful as we cannot gather all the data in the world and some data need to be anticipated by the rules

and constraints in order to reach a solution for a problem in a domain. Finally, machines are different than humans, humans are able to understand the tacit knowledge very clearly even without telling it explicitly, while machines need a representation of a knowledge in order to reach a solution.

References

[1]Carolina Monory, October 19th, 2020. What is warehouse automation - https://6river.com/what-is-warehouse-automation/

[2]Kaveh Azadeh, René de Koster. 2017 Robotized Warehouse Systems: Developments and Research Opportunities https://core.ac.uk/download/pdf/154412962.pdf

[3]Gebser, M., Obermeier, P., Otto, T., Schaub, T., Sabuncu, O., Nguyen, V., & Son, T. C. (2018). Experimenting with robotic intra-logistics domains.

[4]https://potassco.org/doc/start/

[5]https://github.com/potassco/asprilo

[6] Nourhan ElNaggar Mar 3rd, 2021. Project milestone 4