Automated Warehouse Scenario: ASP Solver

Karan Dabas,
Arizona State University
Tempe, Arizona, USA
kdabas1@asu.edu

Abstract

Given a warehouse with robots and various shelves with a variety of products on them, our object is to achieve complete automation such that robots could deliver the ordered products to the picking stations in the warehouse without any collision and in optimal possible time.

Introduction

I will be using Clingo answer set programmming language to develop the entire warehouse world and find the best possible plan of action to complete the orders in a given scenario. This project was one of the problem published in ASP Challenge 2019 submitted by Martin Gebser and Philipp Obermeier.

Problem Statement

The problem states that, we are given a set of robots in a warehouse. This warehouse is like one of the amazon fulfillment centers where robots automatically locate and bring the ordered products to the picking station to complete it. We are given an automated warehouse scenario where the warehouse is divided into a rectangular grid with multiple cells. A few robots that can move on the grid, such that they can only move horizontally or vertically from their current position on the grid. The robots cannot move diagonally , not allowed to swap positions with the adjacent robot and also cannot move out of the defined warehouse grid. Apart from the robots, the warehouse grid also have some shelves, each shelf contains a variety of products with some amount of units of such products. Some cells in the grid are marked as picking station, this is where the robot needs to bring the shelf with ordered products to complete a given order. Each order is mapped to one of the picking station. We are told that a robot can move under a shelf and pick it up and move the shelf to other cells such that no other shelf or robot is on that cell. No two robots can be on the same cell, and a robot carrying a shelf cannot move to a cell with other shelves on it. If such actions occur it will be treated as a collision. We are to avoid collision at all cost. The problem also states that some of the cells in the grid are defined as highway, this means that a robot with or without self can move freely on these cells with cannot drop off shelf on it at any given time. For an example setup of the warehouse see Figure-1. The objective is to complete the given orders with the minimum makespan. A makespan is defined as the greatest time step ‘t ’occurring within action of a plan. This means the solution should be such that it completes all orders in a scenario in the least possible time.

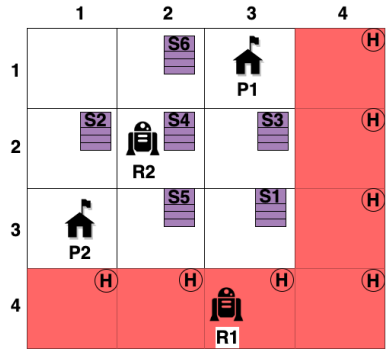

Figure 1: Warehouse setup from the inst1.lp. Here H represent the highway nodes, R represents robots, S represent shelves and P represent the picking stations.

Background Work

As seen from the problem statement, we have a variety of objects in a warehouse and a set action that can be performed. In order to identify these actions and the a way to represent such objects and respective actions of the warehouse world I first watched and understood the concepts taught to us in Module-3, Module-4 and Module-5 in class. The topics include basic arithmetic in clingo, clingo directives(#show, #include), pooling, negation, choice rules, how to use local and global variables, how to add constraints in clingo, cardinality bounds, concept of Generate-(Define)-Test to help define a world with a large number of actions and later based on the conditions prune out to find the most suitable solutions, representation of functions like one-to-one, onto, one-to-one-correspondence, how to do aggrega-

tion,summation in clingo(#count, #sum) along with how to do ASP optimization like #minimize, #maximize, how to identify and define states in a given system/world, define fluents, commonsense law of inertia, various state constraints, effect and precondition of actions, action constraints, and domain independent axioms like actions are exogenous, fluents are initially exogenous, uniqueness and existence of fluents.

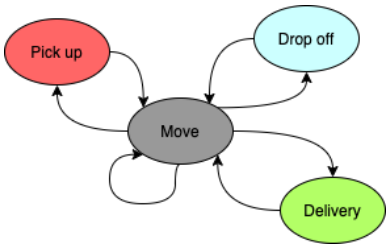

Figure 2: States the robot can go to in the defined automated warehouse world.

Based on the taught concepts, I was able identify and define the states of the robot in the warehouse problem. From Figure-2, the robot has 4 possible states that it can go to. A robot can **move** in the grid, it can **pickup** shelves, **drop off** shelves, **deliver** the shelf with ordered product to the picking station.

Methodology

I have successfully identified all the required tasks need to complete the project. In subsections below I show what all I have completed, challenges encountered and plan to resolve them along with the list of all the task and expected date to complete them. I have also attached my complete list of rules for warehouse problem, all my code till date and rough work in Appendix section at the end.

Progress

See Table 1 where I have listed breakup of all the tasks to complete the project along with my current status on completing these tasks. Refer to the Appendix section below for complete overview of my work.

Challenges Encountered

Below is the list of challenges I faced till date:

1. Identify all the object in the problem.
2. Understand the complex input description given in problem.
3. How to represent the warehouse initial state in clingo.
4. Figure out all the constraints mentioned across the problem.
5. Understand the complex output description given in problem.
6. List down all the actions allowed in the problem.
7. How to represent all actions in clingo.
8. Figure out how to define the effects and preconditions of actions.
9. How to define common sense law of inertia in clingo for this problem.
10. Identify all possible collisions among robots, robot with shelf on it and other shelves.
11. Define the action over time such that it could later be used to minimize the overall time for optimal solution(minimum makespan).

EDOC	Tasks	Status
2/28	Identify all objects in Warehouse	Done
3/01	Complete code to setup warehouse with given initial states	Done
3/01	Define robot movement	Done
3/05	Define grid control	Done
3/06	Identify all actions in a warehouse	Done
3/10	Code for actions are exogenous	Done
3/12	Define common sense law of inertia for warehouse world	Done
3/15	Code for common sense law of inertia	Done
3/20	Identify all required constraints (Robot move, pickup, dropoff, shelf, picking station, delivery, highway	Done
3/22	Code for robot movement constraints	Done
3/24	Code for shelf constraints	Done
3/28	Code for robot pickup action constraint on shelf	
3/30	Code for robot dropoff constraint on shelf	
3/31	Code for highway constraint	
4/02	Code for picking station constraint	
4/05	Code for delivery constraint	
4/05	Add optimization to find the minimum makespan	
4/05	Deliver with final results	

Table 1: List of tasks and current progress of the project. EDOC represent Expected date of completion.

Plan of Action

I have already overcome few of the challenges listed above, I did a thorough read of the given problem description file and listed all the mentioned object, actions, various constraints which can also be seen in the Appendix section below. I did face some issues with how to represent it in clingo, but with the help of the background work mentioned above I was successfully able to overcome them. Currently I am left with adding constraints to the my codebase.

Plan for Future Tasks

Table-1 shows, that I still have a lot of work left to complete. I have set various targets for completing the unfinished tasks(see EDOC from table). I am on track to complete the project by first week of April. Currently working on adding various constraints and getting final results.

Appendix-A

Below is the complete rule book that I developed based on my understating of the given automated warehouse problem.

1. Each Robot may but not necessarily have to do any action(move,pickup,dropoff) in a given time instance.
2. A robot can only perform at most one action per time instance.
3. A robot cannot swap with other robots on it.
4. A robot cannot move outside the initialized grid in Warehouse.
5. A same robot cannot be on more than one cell/location.
6. At most one robot on a given cell/location.
7. A robot cannot go to a cell with shelf if it has a shelf on it before.
8. A shelf can be at only one cell/location. (only on one node/robot in a grid and can only be either on a node or on a robot)
9. A Shelf cannot be at two cells at a given time instance T.
10. A Shelf cannot be at two robots at a given time instance T.
11. A Shelf cannot be at at a cell and on robot at a given time instance T.
12. Two different shelves cannot be on the same robot at a given time instance T.
13. Two or more shelves cannot be on one node.
14. A robot can only pickup one shelf which is on its current cell/location.
15. Only one robot can pickup the shelf.
16. Only the shelf picked up by robot should be on it.
17. A robot cannot pick other shelf if already has one on it.
18. A robot cannot dropoff shelf if it does not have it.
19. A robot cannot put down shelf on highway
20. Shelf cannot be on the Highway.
21. picking station cannot be on the Highway.
22. A robot need to be on the order picking station.
23. Delivery is only possible if robot has the product on its shelf that was ordered.
24. Delivery is only possible if robot has the shelf with the ordered product.
25. Delivery not allowed if the asked amount of units of a product is more than available units of the product on the shelf.
26. A robot does not change location if no move action is done on it.
27. A shelf remains at the same cell/location if no action(like pickup) is done on it.
28. A shelf remain on a robot if no dropoff action is done on it.
29. An order is not complete if it is not yet delivered.
30. Units of product on a shel remains the same if not delivered in any order.

Appendix-B

My current code for robot movement and warehouse setup that takes all the input from the given initial state and defines all necessary functions, my code for sort and object declaration, actions are exogenous in the warehouse, common sense law of inertia, and state constraints.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% setupwarehouse.lp %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pair(X,Y):- init(object(node,C),value(at
,pair(X,Y))).
nodeAt(C,pair(X,Y)):- init(object(node,
C),value(at,pair(X,Y))).
node(C):- nodeAt(C,pair(X,Y)).

highwayAt(C,pair(X,Y)):- init(object(
highway,C),value(at,pair(X,Y))).
highway(C):- highwayAt(C,pair(X,Y)).

robotAt(R,object(node,C),0):- init(obje
ct(robot,R),value(at,pair(X,Y))), nodeAt
(C,pair(X,Y)).
robot(R):- robotAt(R,object(node,C),0).

shelfAt(S,object(node,C),0):- init(obje
ct(shelf,S),value(at,pair(X,Y))), nodeAt
(C,pair(X,Y)), not highwayAt(C,pair(X,Y
)).
shelf(S):- shelfAt(S,object(node,C),0).

productOn(I,object(shelf,S),with(quant,
U),0):- init(object(product,I),value(on,
pair(S,U))).

product(I):- productOn(I,object(shelf,S)
,with(quant,U),0).

pickingStationAt(P,C):- init(object(
pickingStation,P),value(at,pair(X,Y))),
not highwayAt(C,pair(X,Y)), nodeAt(C,
pair(X,Y)).
pickingStation(P):- pickingStationAt(
P,C).

order(O):- init(object(order,O),value(
pickingStation,P)).
deliverAt(O,object(node,C),contains
(I,U),0):- init(object(order,O),value
(line,pair(I,U))), pickingStationAt(
P,C), init(object(order,O),value(
pickingStation,P)).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% gridcontrol.lp %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Length and width of the Warehouse grid.
numGridWidth(N):- N=#count{X
:nodeAt(C,pair(X,Y))}.
numGridLength(N):- N=#count{Y
:nodeAt(C,pair(X,Y))}.
% Total number of cells/nodes initialized
during the init state.
numTotalCells(N):- N=#count{C
:node(C)}.

% Number of robots in the Warehouse.
numRobot(N):- N=#count{R:robot(R)}.

% Number of Shelves in the Warehouse.
numShelves(N):- N=#count{S:shelf(S)}.

% number of Orders
numOrders(N):- N=#count{O:order(O)}.

% Number of Picking Stations in
Warehouse.
numPickingStation(N):- N=#count{P
:pickingStation(P)}.

% Number of Highway cells declared
in a Warehouse.
numHighway(N):- N=#count{C:highway(C)}.

% Number of products initially
declared in a Warehouse.
numProducts(N):- N=#count{I:product(I)}.

%Check Grid for holes, if found the
initial states are not valid
:- numTotalCells(N), numGridWidth(W),
numGridLength(L), not N = W*L.


#show numGridWidth/1.
#show numGridLength/1.
#show numTotalCells/1.
#show numRobot/1.
#show numShelves/1.
#show numOrders/1.
#show numPickingStation/1.
#show numHighway/1.
#show numProducts/1.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% robotmovement.lp %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Only horizontal and vertical movement
possible for a robot.
move(1,0;0,1;-1,0;0,-1).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% compute.lp %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Action Effect %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Moving a robot.
robotAt(R,object(node,C),T+1):- robotAt
(R,object(node,C1),T), robotmove(R,DX,
DY,T), nodeAt(C1, pair(X,Y)), nodeAt(C,
pair(X+DX,Y+DY)).

% Robot picking a shelf.
shelfAt(S,object(robot,R),T+1):- robot(R
), robotAt(R,object(node,C),T), shelfAt(
S,object(node,C),T), pickup(R,S,T).

% Robot dropoff shelf.
shelfAt(S,object(node,C),T+1):- robotAt
(R,object(node,C),T), shelfAt(S,object(
robot,R),T), dropoff(R,S,T).

% Robot delivering the order, so the
amount of item required get reduced by
the amount delivered and some for
product inventory in Warehouse.
deliverAt(O,object(node,C),contains(
I,U-U1),T+1):- deliverAt(O,object(node
,C),contains(I,U),T), delivery(R,O,S,
I,U1,T).

productOn(I,object(shelf,S),with(quant
,U-U1),T+1):- productOn(I,object(shelf
,S),with(quant,U),T), delivery(R,O,S,
I,U1,T).

productAvail(I,U-U1):- delivery(R,_,_,
I,U1,T).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Law of Inertia %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A robot does not change location if
no move action is done on it.
robotAt(R,object(node,C),T+1):- robotAt
(R,object(node,C),T), not robotmove(R,_,
_,T), T<t.

% A shelf remains at the same cell/
location if no action(like pickup)
is done on it.
shelfAt(S,object(node,C),T+1):- shelfAt
(S,object(node,C),T), not pickup(_,S,T)
, T<t.

% A shelf remain on a robot if no
dropoff action is done on it.
shelfAt(S,object(robot,R),T+1):- shelfAt
```

```
(S,object(robot,R),T), not dropoff(R,S,T
), T<t.

% An order is not complete if it is not
yet delivered.
deliverAt(O,object(node,C),contains(I,U)
,T+1):- deliverAt(O,object(node,C),
contains(I,U),T), productOn(I,object(
shelf,S),with(quant,U1),T), not
delivery(_,O,S,I,U1,T), T<t.

% Units of product on a shel remains
the same if not delivered in any order.
productOn(I,object(shelf,S),with(quant,
U),T+1):- productOn(I,object(shelf,S),
with(quant,U),T), not delivery(_,_,S,
I,_,T), T<t.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State Constraints %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% Robot Constraints %%%%%%
% A robot can only perform atmost one
action per time instance.
:- occurs(object(robot,R),Ac1,T),
occurs(object(robot,R),Ac2,T),
Ac1!=Ac2, T=0..t-1.

% A robot cannot swap with other
robots.
:- occurs(object(robot,R1),
move(DX1,DY1),T), robotAt(R1,
object(node,C1),T), occurs(object
(robot,R2),move(DX2,DY2),T),
robotAt(R2,object(node,C2),T),
R1!=R2, C1!=C2, nodeAt(C1,pair
(X1,Y1)), nodeAt(C2,pair(X2,Y2))
, X1+DX1 = X2, X2+DX2 = X1,
Y1+DY1=Y2, Y2+DY2=Y1.

% A robot cannot move outside the
initialized grid in Warehouse.
:- robotmove(R,DX,DY,T),
robotAt(R,object(node,C),T),
nodeAt(C,pair(X,Y)), X+DX > L,
numGridLength(L).

:- robotmove(R,DX,DY,T),
robotAt(R,object(node,C),T),
nodeAt(C,pair(X,Y)), X+DX < 1.

:- robotmove(R,DX,DY,T),
robotAt(R,object(node,C),T),
nodeAt(C,pair(X,Y)), Y+DY > W,
numGridWidth(W).

:- robotmove(R,DX,DY,T),
robotAt(R,object(node,C),T),
nodeAt(C,pair(X,Y)), Y+DY < 1.

% A same robot cannot be on
more than one cell/location.
:- robotAt(R,object(node,C1),T),
robotAt(R,object(node,C2),T), C1!=C2.

% Atmost one robot on a given cell/
location.
:- robotAt(R1,object(node,C1),T),
robotAt(R2,object(node,C1),T), R1!=R2.

% A robot cannot go to a cell with
shelf if it has a shelf on it before.
:- shelfAt(S,object(robot,R),T),
robotAt(R,object(node,C),T),
shelfAt(S1,object(node,C),T), S!=S1.
```

Appendix-C

My rough work on sketching out the tasks from the entire problem description. See Figure- 3,4,5,6,7

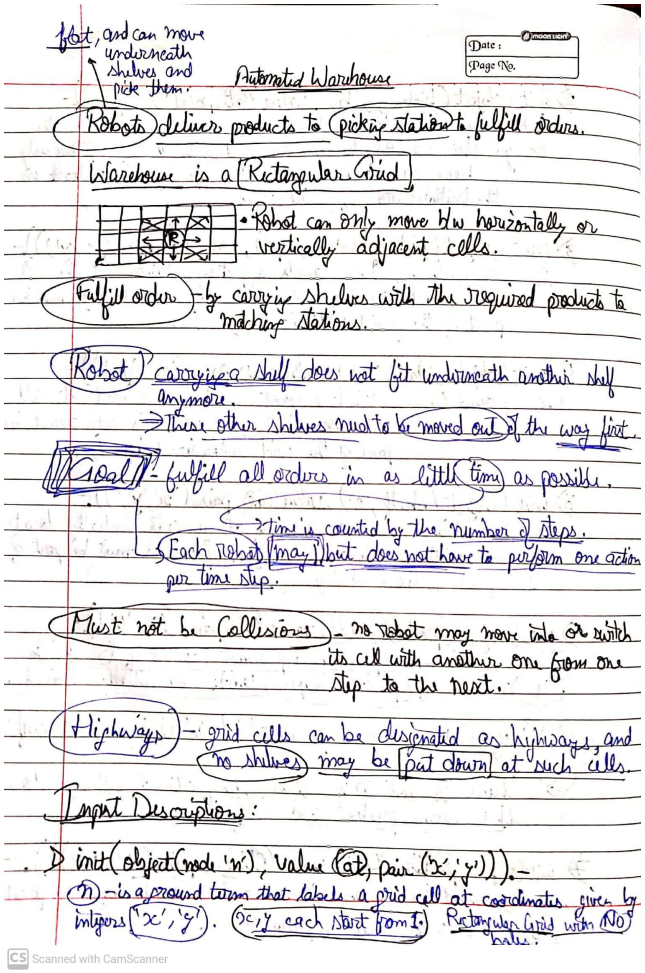

Figure 3: Rough work 1.

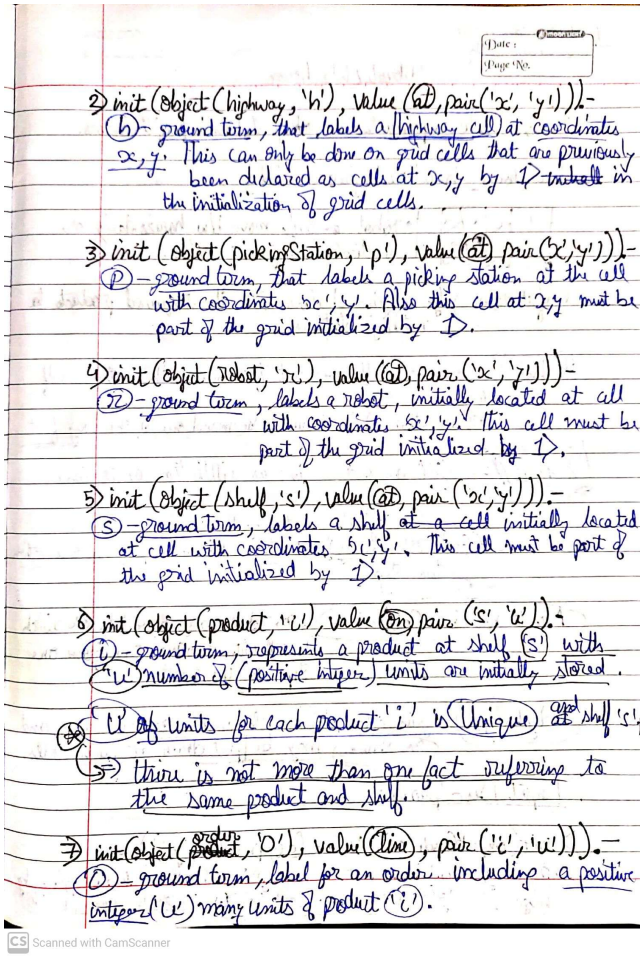

Figure 4: Rough work 2.

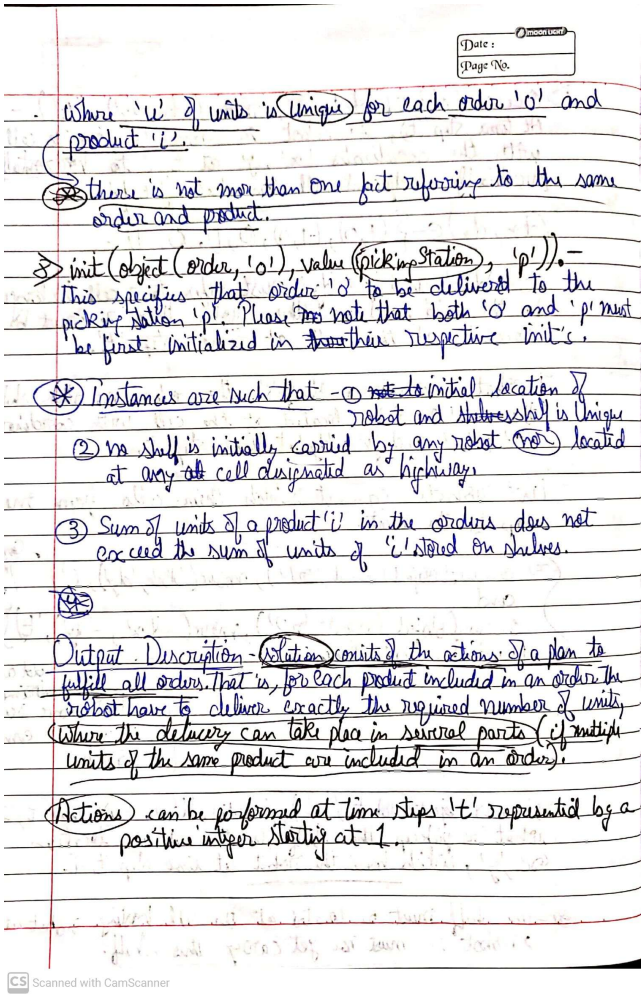

Figure 5: Rough work 3.

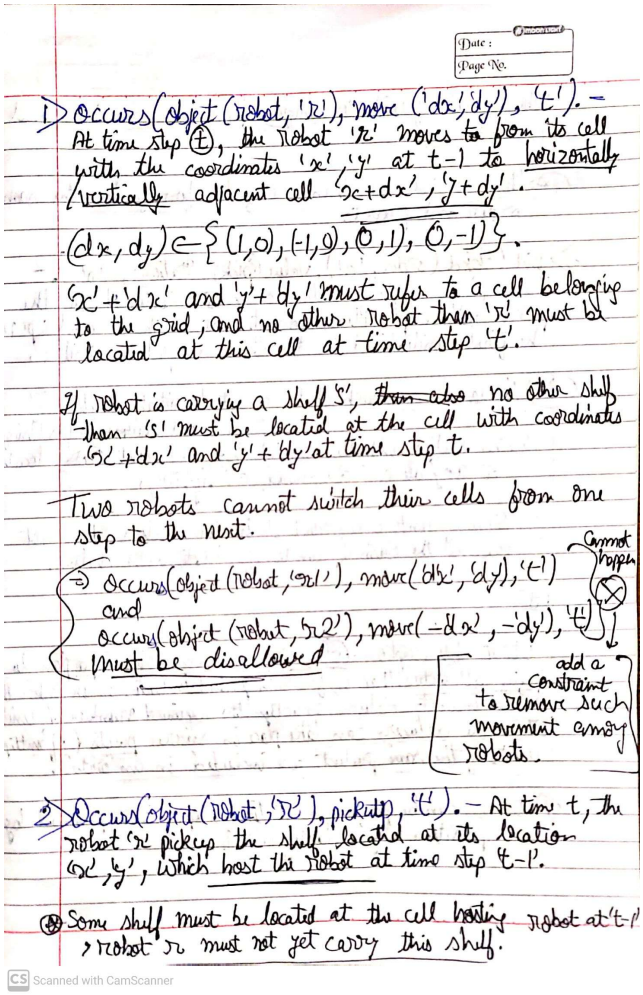

Figure 6: Rough work 4.

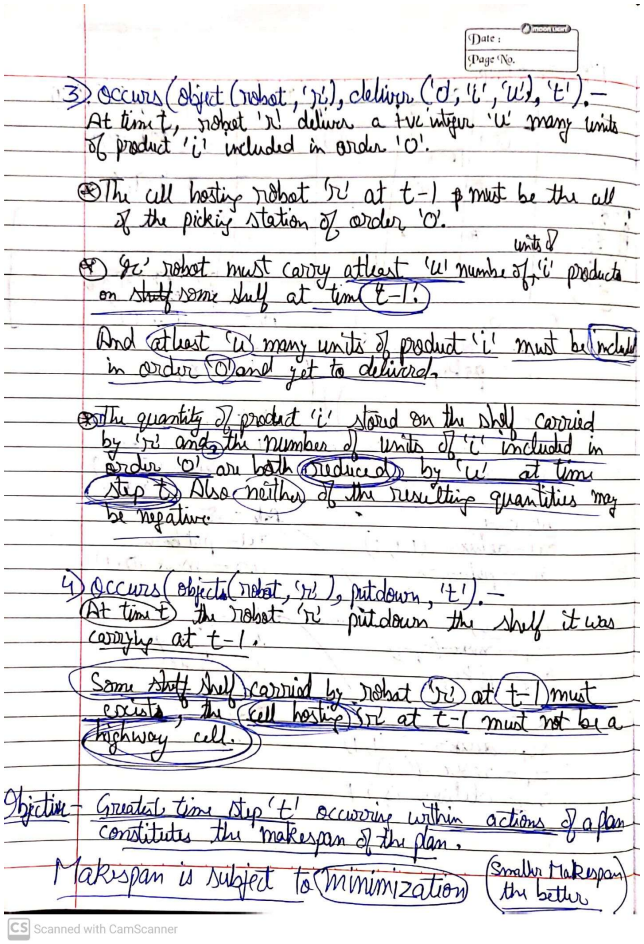

Figure 7: Rough work 5.