AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS (AASMA)

2018/2019

LAB 2 – IMPLEMENTING THE LOADING DOCKS IN NETLOGO AND JAVA

1. GOALS

1. Introduction to the *NetLogo* tool
2. *NetLogo* implementation of the *Loading Docks* scenario using reactive rules (Lab 1)
3. *Java* implementation of the *Loading Docks* scenario using reactive rules (Lab 1)

2. INTRODUCTION

This document contains a description of *Loading Docks* using *NetLogo* http://ccl.northwestern.edu/netlogo/

2.1 THE WAREHOUSE

Warehouse is represented in a 13x13 grid, as shown in Fig.1. The grid cells (*patches*) may have different colors and each one identifies a different element on the warehouse:

- **Black cells:** are the walls of the warehouse.
- **Dark grey cells:** form the ramp where the boxes initially are.
- **Blue, red, yellow and green cells:** are the blue, red, yellow and green shelves, respectively.
- The **remaining cells** represent the free floor of the warehouse.

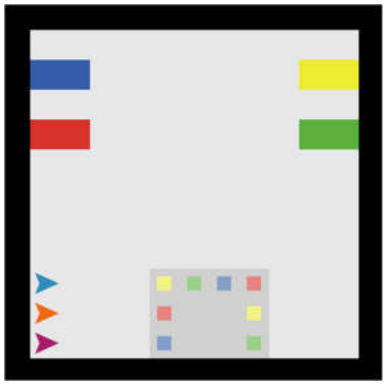

FIGURE 1: LOADING DOCKS SCENARIO IN NETLOGO

The cells have four relevant properties:

- **pxcor** and **pycor:** are the cell's coordinates in the warehouse grid.
- **kind:** defines the type of the cell (ramp, shelf, floor and wall). The values of this property might be defined with one of the following global variables: **ROOM_FLOOR**, **SHELF**, **RAMP** and **WALL**.
- **shelf-color:** defines the color of the cell, which only makes sense in case of being a shelf (**kind = SHELF**), otherwise its value may be ignored. The values of this property might be: **COLOR_BLUE**, **COLOR_GREEN**, **COLOR_YELLOW**, **COLOR_RED**.

2.2 THE AGENTS

The three agents/robots in L*oading Docks* are *turtles* in the *NetLogo* environment with properties:

- **xcor**, **ycor**: are the turtle's coordinates in the warehouse grid.
- **heading**: defines the turtle's current direction.

Fig.2 shows the coordinate system for our warehouse grid. The origin is in the center of the grid and one can assume the agents' coordinates (**xcor** and **ycor**) belong to the {-5,5} interval (excluding walls).

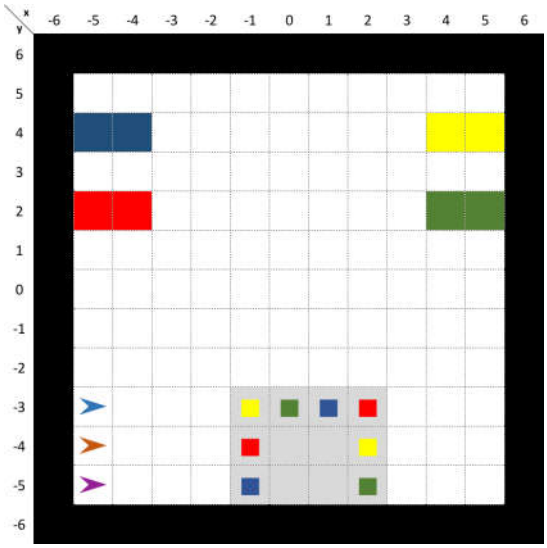

FIGURE 2: COORDINATE SYSTEM OF THE WAREHOUSE

The values of the **heading** property range between 0 e 360, denoting the clockwise rotation degree of the robot around its vertical axis. The 0 value means the robot is upturned.

Additionally, a robot also includes another property, called **cargo**, which contains a reference to the box being carried. When a robot doesn't carry any box, the value of **cargo** is **WITHOUT_CARGO**, another global variable in our environment.

2.3 THE BOXES

Boxes are also defined as *turtles* in this *NetLogo* scenario. Their main properties are:

- **xcor**, **ycor**: define the coordinates in the warehouse grid.
- **box-color:** defines the color of the box and might be one of the following global variables: **COLOR_BLUE**, **COLOR_GREEN**, **COLOR_YELLOW** or **COLOR_RED**.

2.4 THE INTERFACE

Besides the graphical representation of the grid (Figure 1), the interface of the simulation environment for the Loading Docks scenario also contains 3 *Buttons* and 2 *Monitors* (Figure 3).

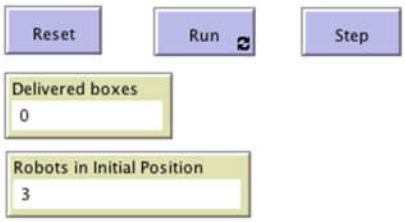

FIGURE 3: MONITORING AND CONTROLLING INTERFACE

The three buttons control the simulation:

- **Reset:** resets the initial state of the environment.
- **Run:** continually executes the simulation. By repressing this button, the simulations pauses.
- **Step:** executes the simulation only one time.

The Monitors show the current state of the goal.

- **Delivered boxes:** counts the number of boxes that are placed on the shelves.
- **Robots in Initial Position:** counts the number of robots in their initial positions.

By watching this monitoring boxes, one can check how far from achieving the goal the agents are, as well as their efficiency. The simulation stops as long as the final state occurs.

2.5 SUMMARY

The following table sums up the properties and constants in this scenario:

Cells/Patches	
kind:	**ROOM_FLOOR, SHELF, RAMP, WALL**
shelf-color:	**COLOR_BLUE, COLOR_GREEN, COLOR_YELLOW, COLOR_RED**
pxcor:	{-6, 6}
pycor:	{-6, 6}
Robots	
cargo:	**WITHOUT_ARGO, <ref-box>**
xcor:	{-5, 5}
ycor:	{-5, 5}
heading:	{0, 360}
Boxes	
xcor:	{-5, 5}
ycor:	{-5, 5}
Box-color:	**COLOR_BLUE, COLOR_GREEN, COLOR_YELLOW, COLOR_RED**

3. EXERCISES

3.1 AGENT BEHAVIOR IN NETLOGO

Implement a reactive agent to solve the *Loading Docks* problem, by defining the sensors, actuators and behavioral rules in the provided *NetLogo* file. Do not forget that actuators migh have pre-conditions.

Note: Behavioral rules should be implemented in the **robot-loop** procedure and should use the sensors and actuators in order to preserve the encapsulation. Any other implemented procedure to reset the robots initial state should be added to **init-robot** procedure.

3.2 AGENT BEHAVIOR IN JAVA

Study the *Java* implementation of the *Loading Docks* problem using the provided *Java* project.

- open the *Java* project in Eclipse and run the Main class
- explore the provided classes and design the object structure diagram

Implement an identical reactive agent to the agent designed in 3.1 in *Java*.

3.3 IMPROVEMENTS

Insert some of the improvements mentioned in the previous class to the reactive agents in 3.1 **or** 3.2.