Inteligência Artificial
Trabalho 1

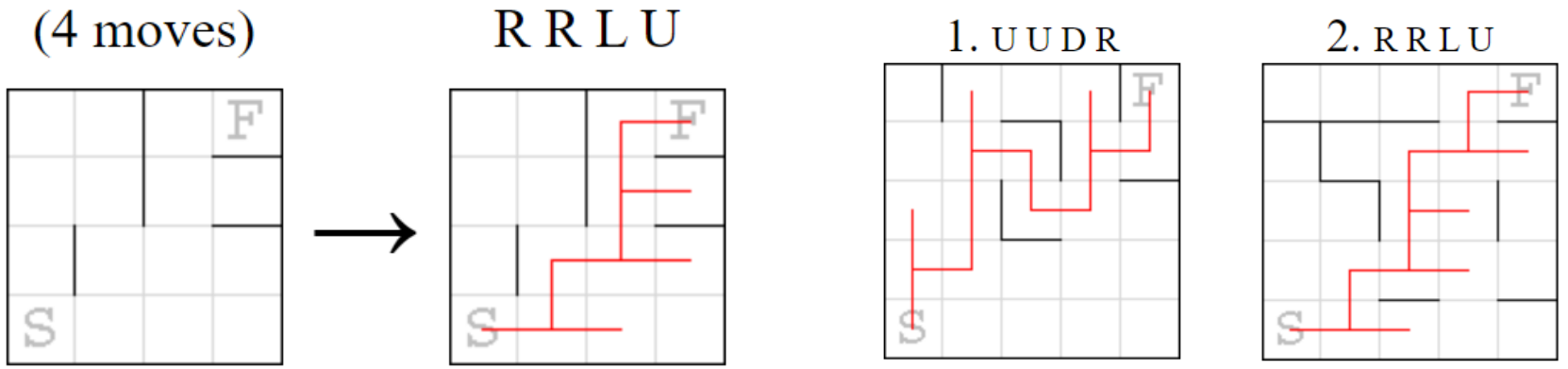
Juan Bellon, up201908142

Luísa Araújo, up201904996

Nuno Melo, up202003324

Robot Mazes

- Single player game.
- **Main goal**: Find the shortest sequence of necessary moves to reach the exit.
- From the departure point until the exit the robot is going to follow every move of the sequence in a cycle.
- If the robot bumps into a wall it does not move and passes to the next instruction (operator) of the sequence.
- Example:



References

- https://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf
- http://bryukh.com/labyrinth-algorithms/

```
1  Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)
2  while the OPEN list is not empty {
3    Take from the open list the node node_current with the lowest
4         f(node_current) = g(node_current) + h(node_current)
5    if node_current is node_goal we have found the solution; break
6    Generate each state node_successor that come after node_current
7    for each node_successor of node_current {
8      Set successor_current_cost = g(node_current) + w(node_current, node_successor)
9      if node_successor is in the OPEN list {
10       if g(node_successor) ≤ successor_current_cost continue (to line 20)
11     } else if node_successor is in the CLOSED list {
12       if g(node_successor) ≤ successor_current_cost continue (to line 20)
13       Move node_successor from the CLOSED list to the OPEN list
14     } else {
15       Add node_successor to the OPEN list
16       Set h(node_successor) to be the heuristic distance to node_goal
17    }
18     Set g(node_successor) = successor_current_cost
19     Set the parent of node_successor to node_current
20   }
21   Add node_current to the CLOSED list
22 }
23 if(node_current != node_goal) exit with error (the OPEN list is empty)
```

```python
from heapq import heappop, heappush


def heuristic(cell, goal):
    return abs(cell[0] - goal[0]) + abs(cell[1] - goal[1])


def find_path_astar(maze):
    start, goal = (1, 1), (len(maze) - 2, len(maze[0]) - 2)
    pr_queue = []
    heappush(pr_queue, (0 + heuristic(start, goal), 0, "", start))
    visited = set()
    graph = maze2graph(maze)
    while pr_queue:
        _, cost, path, current = heappop(pr_queue)
        if current == goal:
            return path
        if current in visited:
            continue
        visited.add(current)
        for direction, neighbour in graph[current]:
            heappush(pr_queue, (cost + heuristic(neighbour, goal), cost + 1,
                                path + direction, neighbour))
    return "NO WAY!"
```

Formulation as a search problem (Part 1)

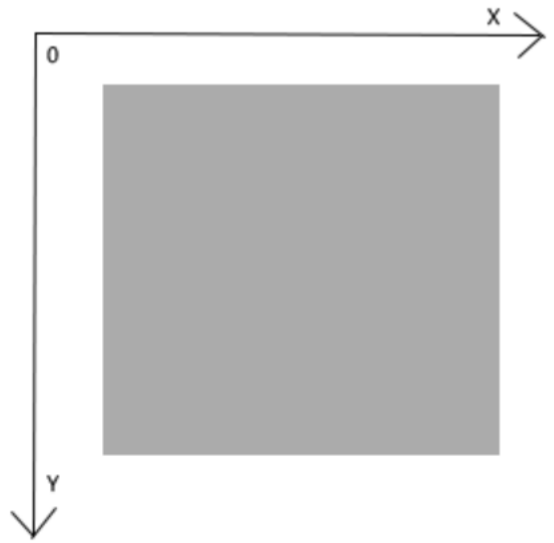
1. **State Representation**: place(x,y) represents what is located in the (x,y) coordinates. It can be a wall, the exit door or even nothing! Let's assume:

- (x,y) – Represents the robot's current position.
- if place(x,y) == 0 then (x,y) is free.
- if place(x,y) == 1 then a wall occupies (x,y).
- If place(x,y) == 2 then the exit is in (x,y).

2. **Initial State:** Can be any (x,y) where place(x,y) = 0. For example, (0,0) & place(0,0) = 0 can be our initial state.
3. **Objective Test:** Any (x,y) where place(x,y) = 2
4. **Heuristic Function:** h(n) = | n.x – f.x | + | n.y – f.y |, where f is the final coordinate.

   Cost: Length of the sequence of moves. A* = Heuristic + Cost

Formulation as a search problem (Part 2)



1. Operators:

- Let’s assume xSize and ySize are the labyrinth’s dimensions.
- Considering the xy axis has its origin in the superior left corner.

Name	Preconditions	Effect	Cost
UP	y>0 & place(x,y)!=1	y = y-1	1
DOWN	y<ySize & place(x,y)!=1	y = y+1	1
LEFT	x>0 & place(x,y)!=1	x = x-1	1
RIGHT	x<xSize & place(x,y)!=1	x = x+1	1

Implementation

- **Programming language**: Python.
- **Development Environment**: Visual Studio Code and/or Pycharm.
- **Data Structures**:

1. Maze - Two dimensions array or vector, for example:

   [[1,0,0,1],[1,0,1,1],[1,0,0,2],[1,1,0,1]]

2. Robot – List with length = 2 [x,y], for example:

   [1,5]

- **Code**