TDDC17 Artificial Intelligence

LAB 1

**Vacuum Agent**

The main decision-loop of the program depends on whether or not a queue called *action\_queue* contains any actions. If the *action\_queue* is empty, a path to an unknown node is calculated as well as the actions required to move to that point. If there exist actions in the queue, an action is polled from the queue and executed by the vacuum. The text below explains how the nearest unknown point is found and how the path to that point is calculated.

The vacuum agent has two fundamental functions, *find\_nearest\_unknown()* and *fill\_action\_queue(). Find\_nearest\_unknown()* uses a *Breadth First Search* to find the nearest unknown point and a hash-map that maps each point to a parent-point. Firstly, the function explores the points that are the closest neighbors of the current position. If none of these neighbor-points are unknown (and they are valid), we add them to the hash-map and continue checking the neighbors of the neighbor and so on. Each point explored that is not unknown, not a wall and has not been previously explored (i.e. a valid path) is saved in the hash-map and mapped to a parent point. When an unknown point is found, we iterate through the hash-map, finding the parents of the points and add them to a *path\_to\_target*-queue. The path is found by searching for the current point in the hash-map and saving the parent of that point in the *path\_to\_target-*queue. The previously found parent is then searched for in the hash-map and the parent of that parent is added to the *path\_to\_target*-queue. Once the parent of the “current parent” is null, we have reached our current position and the route has been determined (the parent of the current position is set to null at the start of the function). Shortly, the *path\_to\_target*-queue consists of points we have to route through in order to reach our goal point.

After the path that has to be taken is determined, we call *fill\_action\_queue()*. This function polls a point from the *path\_to\_target*-queue and determines what actions has to be taken in order to reach that point. The actions are determined by the current direction of the vacuum, the coordinates of the current position as well as the coordinates of the point that was polled from the queue. Each action taken is saved in *action\_queue*, which is then used when actually moving. *Fill\_action\_queue()* simulates the vacuum moving from the current position to the goal node. The vacuum actually moves when an action is polled from the *action\_queue* (in the main loop of the program).

Once the *action\_queue* is filled with the actions required to reach the goal point, we start the actual movement of the vacuum by polling actions from the queue. This is done until the queue is empty, meaning that the goal point has been reached. Then, *find\_nearest\_unknown()* is called again, finding another unknown point and the path to reach it…

When *find\_nearest\_unknown()* fails to find an unknown point, meaning that there are no valid paths to an unknown point (i.e. the vacuum has finished vacuuming), we call the function *go\_home()*. This function works just like *find\_nearest\_unknown()*, with the difference that it searches for the point that has the percept *HOME*. Once a route to that point is found, we fill the *action\_queue* with actions and execute them in the main loop. When the home point is reached the vacuum shuts down and we are finished.

The reason we chose to implement *Breadth First Search* is because we do not know the destination and want to find the nearest unknown point. *Breadth First Search* returns the shallowest point satisfying the conditions and is optimal when all the actions have the same cost, which is the case for this program.