Richard Zimmerman [U16632404] | Alec Braynen [U41628244]

One Robot Sensing Two Spaces Ahead

Background

Specifications

The Robot Does Not Hit Any Obstacles.

A[] grid[robotOne.myPosX][robotOne.myPosY] != 1

The array represented by the variable grid is a 2-d boolean array with the dimension of p and q where p represents the number of rows in the grid and q represents the number of columns in the grid. A status of false indicates the grid space represent by *s* and *t* in grid[*s*][*t*] indicates the grid square (*s,t*) is not blocked by a static obstacle. When grid[*s*][*t*] is true, this indicates the grid square (*s,t*) is blocked.

The statement above confirms that there is no such trace where the current position of the robot (in effect, any position the robot has traveled) is blocked by a static obstacle.

The Robot Does Not Cross any Boundaries.

A[] robotOne.myPosX >= 0 and robotOne.myPosX < dimX

A[] robotOne.myPosY >= 0 and robotOne.myPosY < dimY

In the above statement, robotOne (the only robot) has local variables myPosX and myPosY that represent the current X and Y coordinates of the robot on the grid, respectively. The statement evaluates to true only when all possible positions of the robot are with the bounds represented 0 (start of the grid), dimX (the number of rows in the grid), and dimY (the number of columns in the grid).

The Robot Will Eventually Reach the Destination.

A<> (robotOne.Complete)

The above statement confirms that for every trace, robotOne will reach the Complete state which is only entered if the robot current position (myPosX and myPosY) matches the coordinates of the destination point.

Model Summary

Grid Controller

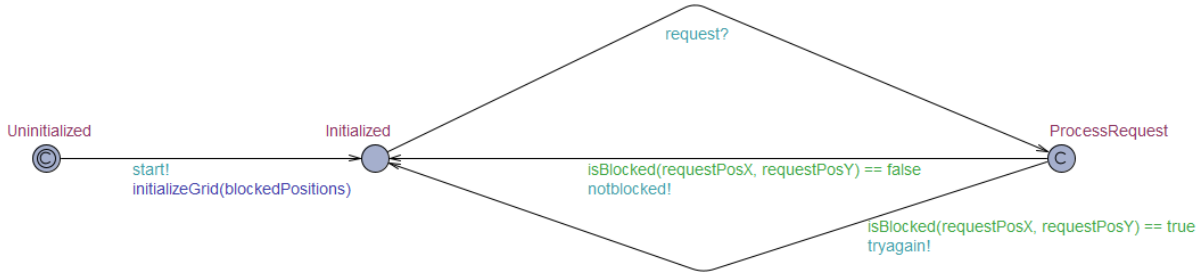

Figure 1 Model of Grid Controller in UPPAAL

The grid controller starts in a committed state named Uninitialized which, upon system start, is immediately exited and the transition from state Uninitialized to state Initialized is taken. This transition raises a signal on a broadcast channel named *start* and then calls a function that initializes the grid array (the 2d boolean array named *grid* mentioned in *Specifications -> The robot does not hit any obstacles*) which makes sure that the grid's static obstacles are enabled/instantiated.

The grid controller then waits in the Initialized state until it receives a signal on the *request* channel. Upon receiving the request signal, the grid controller enters the committed state ProcessReqeust and in the next "step" of the system, raises a signal on either the *notblocked* or *tryagain* channel which represents whether the grid targeted by the requester is not blocked or blocked, respectively.

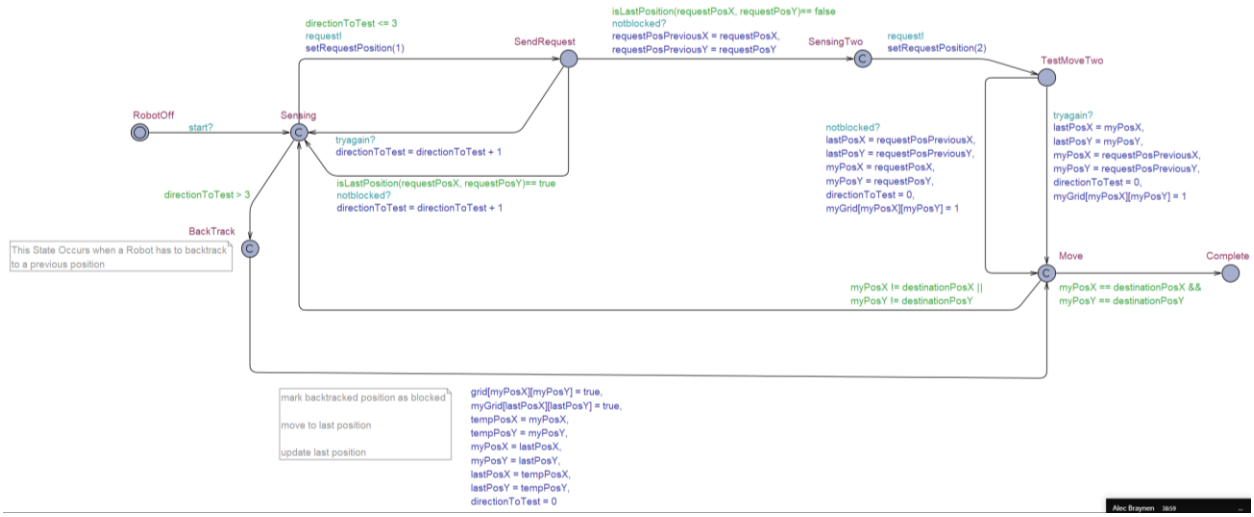

Figure 2 Model of RobotOne

Please the document for Part 1 of Project 1 for a description of how the robot works when sensing one-space ahead. The only differences between the 1-sensor robot and 2-sensor robot are the additional states SensingTwo and TestMoveTwo and the corresponding edges.

When the robot receives the first *notblocked* signal from the controller, the robot immediately builds another request to see if it can move two spaces in the same direction as the first successful

(notblocked) request. If the target space that is now two spaces away from the robot is free, the robot moves two-spaces to the new target, else the robot only moves one space at a time.