## 

To define the Successor function, we first define the Current location of the agent 1, and agent 2, as following:

Since the state variable doesn’t contain exact information on the agent 1 and agent 2 locations, we have to loop through and find a tile with values 1 and 2.

We have to check that the following conditions hold:

* Player can move to a tile which isn’t gray, or other player
* Player can move to a tile which isn’t a wall

Defining a group of Moves which can be done in a game:

For example, a move of first player upwards is: , where:

Defining a condition which will be checked:

Now, we can define the successor function:

## 

Defining the condition for

## 

The branching factor can be 4 in the initial turn of a any player, if surrounded by unvisited or white tiles. From second turn on, it will be 0-3, since we can never visit the tile which we have already visited.

## 

Simple player iterated on all the directions (4 directions – UP, DOWN, LEFT, RIGHT), checking if they are feasible. If a direction is feasible, the ‘state\_score’ is being calculated. The move with the best score is performed. If state\_scores are equal, FIRST direction which was calculated wins.

According to the ‘state\_score’, it will prefer the state with the LEAST number of further states available for that player, as long as this will not lead to 0.

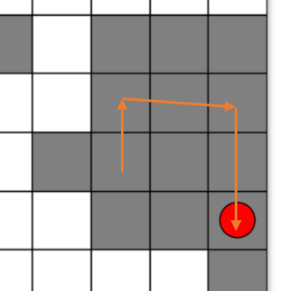
PROS:

* Given a certain area with no obstacles (a large space), the agent will for sure fill all the tiles

CONS:

* Sees only 1 step into the future
* Doesn’t take enemy location into account
* Can run into the dead end, while being able to easily avoid it (connects to the first CON)

Example of a dead-end (map number 3)



## 

As stated in PRO, the agent will fill the whole given area in an optimal way, for example here, I have marked it’s path:

