Workshop 6 (Exploring the world)

Exercise 6: Bumble 1.4 – Explorer

Note: ***V-World workshops from here onwards are pretty much as Donald Nute designed them, i.e. although they are progressive and should lead to a well-developed understanding and appreciation for NPC agents and their behaviour in tile-based worlds, they should be treated in CI342 as reference. In my opinion they should be treated as a collection of (possibly) good ideas, or at least a collection of concepts to 'ponder.' This should not be seen as 'must do', but rather 'interesting to read' and potentially 'useful to consider', especially for your assignment.***

In Exercise 5, we modified Bumble so it creates a map of his world as he moves around it. Next, we need to modify Bumble so it uses its map to return to resources when they are needed. First, copy the code into Bumble 1.3 that you will need to find a path to a goal object using grid search. Then modify your code for Bumble 1.3 by changing

:- dynamic [agent/2,tried/0,last/1,pushed/0,hungry/0,hurt/0,mymap/3,here/2,last_seen/3].

to

:- dynamic
[agent/2,tried/0,last/1,pushed/0,hungry/0,hurt/0,mymap/3,here/2,last_seen/3,path/2].

Now you will need to add rules to your agent to tell it when to find a path and how to use a path. Let's consider the case where Bumble is hungry - a condition for each of a set of rules. Then I suggest you have a rule for each of these cases, and the rules should be in this order.

1. If Bumble is next to an apple, he should eat it.

2. If Bumble is near an apple, he should move toward it. (Bumble will only get to this rule if he is not next to an apple.)

3. If Bumble is next to a tree, he should push on it.

4. If Bumble is near a tree, he should move toward it.

5. If there is no tree on Bumble's map, then Bumble should continue to explore the world to try to find a tree.

6. If Bumble does not have a path to a tree stored, then he should try to find a Path to the nearest tree and store it as path(tree,Path), then follow it.

7. If Bumble has stored a path(tree,Path), then Bumble should try to follow the Path.

8. If Bumble has stored a path(tree,Path) and Bumble cannot move to the first location in Path because Bumble is not next to that location, then Bumble should retract(path(tree,_)) and try to find and follow a NewPath.

9. If Bumble has stored a path(tree,Path) and Bumble cannot move to the first location in Path (= [(X,Y)|Rest]) because some animate object is in that location, then Bumble should retract(path(tree,_)), retract(mymap(X,Y,_)), assert mymap(X,Y,obstacle), find a NewPath to the nearest tree, and follow it. In fact, you may want to mark temporary obstacles on your map at all locations next to Bumble that are occupied by some animate object. This will keep find_path/4 from generating a path through some immediate obstacle.

10. If all of the above fail and there is a tree on Bumble's map, then some obstacle is blocking any path to the tree. Bumble should remove (kill - with bug spray?) the obstacle or move directly away from it. This will give the obstacle a chance to move out of the way, and maybe Bumble can find a valid path to a tree on the next move.

11. If all the above fail, and there are unexplored regions in Bumble's world, then Bumble should continue to explore.

Another thing you might consider is the cost to follow a path to a tree. If the cost is greater than Bumble's strength, then Bumble should probably continue to explore and hope to find a closer tree. Chances may be slim, but if Bumble doesn't have the strength to reach a known tree, it's the only chance Bumble has.

You would create similar sets of rules for reaching a cross when Bumble is hurt, or for any other kind of goal Bumble might want to reach.

If you followed the suggestions above, then you put temporary 'obstacles' on your map to keep find_path/4 from again finding a path through the snail or hornet or other animate object that is in your way. You will want to remove those at the beginning of the next move. You could do this by changing

```
agent(Perceptions,Action) :-

set_bumble_states(Perceptions),

build_bumble_map(Perceptions),

bumble(Perceptions,Action).
```

to

```
agent(Perceptions,Action) :-

forall(retract(mymap(X,Y,obstacle)),assert(mymap(X,Y,o))),

set_bumble_states(Perceptions),

build_bumble_map(Perceptions),
```

bumble(Perceptions,Action).

You will also need to define a predicate

follow_path(Path,Dir)

that will return the direction you need to move to go to the first location in Path. Remember that you need to check first to make sure there is nothing in that location to prevent your move. You probably should do this before you call follow_path/2. If you do, then you should only be calling follow_path/2 in situations where it will succeed. Also remember that you will want to remove the first location from Path, creating a ShorterPath, and replace the stored Path with the ShorterPath for the next move.

This should give you enough ideas to write code that allows Bumble to return to any tree or cross on Bumble's map. Now, so long as Bumble stumbles across a tree and a cross before starving or being killed, Bumble should be able to survive pretty much forever. The other thing you need to do is implement a method for Bumble to explore.

One way to implement exploration is to put mymap(X,Y,frontier) on your map instead of mymap(X,Y,o) whenever (X,Y) is a location Bumble could occupy (empty, etc.) but which Bumble has not yet occupied. Then you can explore just by finding a path to the nearest frontier and following it. Remember, though, to change mymap(X,Y,frontier) to maymap(X,Y,o) once you have occupied a location. You could make this a step in your procedure for updating Bumble's map when you compute the new value for here/2. When you retract(here(_,_)) and assert(here(NewX,NewY)), just also retract(mymap(NewX,NewY,_)) and assert(mymap(NewX,NewY,o)).

End of workshops – here is what Nute suggests from here:

Possible exercise 7: Bumble 1.5 - Hunter

Starting with Bumble 1.4 - Explorer (see last workshop), build a version of Bumble that can catch the bird in the bird worlds.

Good luck with the assignment!
