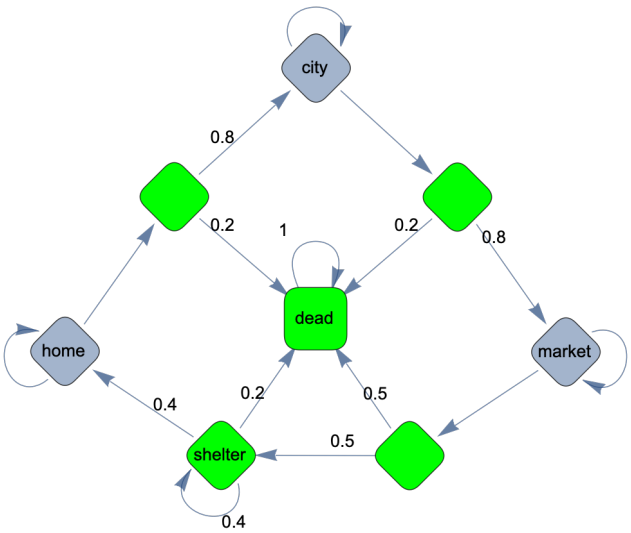
Lab - Week 6

Q-Learning

The purpose of this exercise is to ensure that you fully understand how Q-Learning is performed. You will perform the updates manually. Subsequently, you will extend the code of your grid world example with q-learning.

1. Warm-up: Q-Learning Step-by-step

We are reusing the "Seven Lives of Cats" example that you used in the last lab for value iteration.



Tasks

1. We will perform q-table updates for the first two steps in some episode manually. To start, initialise your q-table. You can initialise your q-table in any form you like except for with all-zero values. You also need to think about the final states; how should you initialize these? Why? As your agent executes its actions it experiences state updates (which are under the control of the environment). Your tutor will play the role of the environment and tell you what the consequences of your actions (state updates) are.
2. Let's have a look at how these state updates actually happened. Review your transition matrix from last week for this example. Recall you need to "pull back" the green nodes at which the agent makes no decision into the grey nodes. You should have ended up with a matrix that looks as follows (if you did not change the probabilities). T[1] are the transitions for the "stay" action and T[2] are the ones for the "run" action. The order of states is home, shelter, city, market, dead.

This time we are assuming that the task is episodic. The dead state is a terminal state (rather than an absorbing state as which it was treated previously). Based on this transition matrix, construct a full episode, i.e. write down the state and reward sequence for a single episode until it terminates. You may choose the actions as you like.

```
T=[
  [[1, 0, 0, 0, 0],
   [0.4, 0.4, 0, 0, 0.2],
   [0, 0, 1, 0, 0],
   [0, 0, 0, 1, 0],
   [0, 0, 0, 0, 1]],
  [[0, 0, 0.8, 0, 0.2],
   [0.4, 0.4, 0, 0, 0.2],
   [0, 0, 0, 0.8, 0.2],
   [0, 0.5, 0, 0, 0.5],
   [0, 0, 0, 0, 1]]]
```

3. Perform the q-table update for the last step in your episode manually (assuming that the table you obtained from the previous sub-task is the state of the q-table just before the last transition happens)

2. Project Phase 1

The remainder of this lab will be used to working on the grid world code that you started in the last lab. At this point you should have a code “shell” that can execute the agents actions but does not yet have the capability to learn the correct actions. The objective now is to implement q-learning to let the agents learn to solve the task effectively.

1. Create a function to return the state description of the agent.
2. Create a function to implement the policy.
3. Create a function to that performs a chosen action (step) for an agent and returns the appropriate reward.
4. Create a function to perform a single q-Table update.
5. Integrate the above functions to perform q-learning with your agent.
6. Explore and evaluate the learning and fine-tune your algorithm.