Planning and Approximate Reasoning

PART 1: Planning

November, 13th, 2019

Time: 2 h. You cannot use any printed or electronic materials.
Please give clear and detailed explanations (with examples, if appropriate).

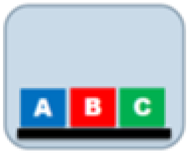
---

1. **Representation for High Level Planning**
   a. (1 point) Many applications use discrete or continuous planning systems to operate and act in the real world. Many other applications deal with a mixture of discrete and continuous variables, which are known as hybrid planning systems. Give an example for each of these 3 planning systems.
   b. (1 point) One example of hybrid planning systems is the autonomous mobile manipulation. The goal of it is the execution of complex manipulation tasks, in dynamic environments, in which cooperation with humans may be required. To achieve this goal, several scientific and engineering challenges, currently beyond the state of the art in robotics, must be addressed. Explain briefly the main steps should be performed for mobile manipulation of known objects?

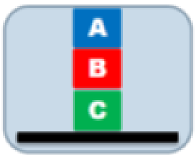
2. **Classical Planning**
   a. (1 point) Explain the three main Properties of Planning Algorithms?
   b. (1 point) The Standford Research Institute Problem Solver (STRIPS) is an automated planning technique that works by defining the domain and the planning problem in order to achieve a certain goal. STRIPS works with stacks and describes the world providing objects, actions, preconditions and effects. Explain in details why STRIPS planning cannot provide a solution for non-deterministic problems?

3. **PDDL Implementation**

(2 point) The blocks world is one of the most famous planning domains in AI. Given a set of wooden blocks with similar shapes and different colours sitting on a table. The goal is to build one or more vertical stacks of blocks. The catch is that only one block may be moved at a time: it may either be placed on the table or placed atop another block using a robot arm. Because of this, any blocks that are, at a given time, under another block cannot be moved. For the problem shown in the figure below, write the domain PDDL file for solving this problem. Note: S0 is the initial state and g is the goal.



s_0



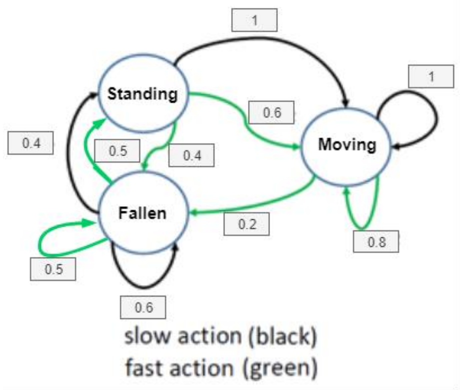
g

4. **Markov decision process (MDP)**

(2 point) MDP models a sequential decision problem, in which a system evolves over time and is controlled by an agent. The system dynamics are governed by a probabilistic transition function **P** that maps states ***s*** and actions **a** to new states ***s´***. At each time, an agent receives a reward R(***s***,***a***) that depends on the current state ***s*** and the applied action **a.** Given a certain policy ***π***, the expected accumulated

reward for a certain state, ***s***, is known as the value for that state according to the policy, ***π***: $V^{\pi}(s) = R(s,a) + \gamma(\sum P(s,s',a)R(s,s',a))$, where γ is a parameter known as the discount factor, 0 ≤ γ < 1. In the figure below, you can find a system with 3 states (standing, fallen and moving) with two actions (slow and fast). The probabilities transition values (*p*) of each state shown on the figure. Compute the expected accumulated reward with a policy, ***π* =** (fast), assuming the discount factor *γ* is 0.8 and the immediate rewords of the three states are:

Standing	Moving	Fallen
0	+2	-2

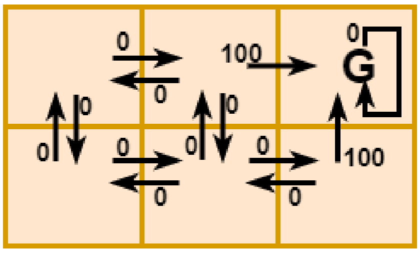


5. Reinforcement Learning (RL)

(2 point) Q Learning is the most used RL. By the usage of this algorithm, the agent learns the quality (Q value) of each action (i.e. policy) based on how much reward the environment returns with. Q Learning uses the table to store the value of each environment's state along with the Q value. To use the matrix Q, the agent simply traces the sequence of states, from the initial state to goal state. The algorithm finds the actions with the highest reward values recorded in matrix Q for current state using the following training rule:

$$\boxed{\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')}$$

where s is the current state, *s´* is the next state, *a* is the current action, *a´* is the next action, and *γ* is the discount factor. For the example shown below, a simple deterministic case represented as a grid, each cell represents a distinct state, and each arrow a distinct action. The immediate reward function *R(s,a)* gives *100* for actions entering the goal state *G*, and zero otherwise. Find the optimal policy corresponding to actions with maximal Q value. Note: start at top left corner with fixed policy – clockwise, and the initial *Q(s,a) = 0; γ = 0.9.*



R(s,a) (immediate reward) values