1. One potential reinforcement learning task would be a poker playing agent. The state would include information regarding the size of the pot, the number of players remaining in the hand, the agent’s cards, the cards on the board, the other players’ actions, etc. The actions would be very limited: they would only include folding, calling, and raising (checking and betting are only specific cases of calling and raising when the current bet is zero). The reward would be easy to calculate: the change in the size of the agent’s stack after each action, which would include increases when the agent wins the pot.

Another potential task would be a helicopter flying agent. The state would include information regarding the absolute GPS coordinates, the relative position such as pitch and yaw, the RPM’s of the engine and rotors, environmental conditions such as wind speed, etc. The actions would include control over the four flight control inputs of a helicopter: the cyclic (moving forward or sideways), the collective (moving up or down in altitude), the anti-torque pedals (yaw), and the throttle (engine power). A positive reward signal could be given for arriving at an instructed destination, and a negative reward signal could be given every time step that the helicopter takes to arrive at the destination, with a large negative reward signal given for crashing.

A final potential task would be controlling a non-player controlled character (NPC) in a video game, such as a Massively Multiplayer Online Role Playing Game (MMORPG). The state would include information regarding the location and strength of human controlled characters and other NPC’s, and possibly information regarding previous encounters with these characters. The actions would include movement in the 3-D world, fleeing from or attacking other characters, buying and selling from other characters, etc. The reward could vary depending on the type of NPC, but could include a negative reward for dying, a positive reward for killing another character, a positive reward for increasing in wealth, etc.

1. I would argue that reinforcement learning is not appropriate for all goal-directed learning tasks. For example, in an unsupervised learning problem such as clustering, the task does involve learning, and there is a goal involved, but there is no clear way to represent a “reward” or “action” taken by the unsupervised learning agent. In fact, the lack of error or reward information for the algorithm to work with is the reason the task is called “unsupervised”.
2. The “right” place to draw the line between the agent and the environment can vary depending on the particular task and agent. However, a general rule is that the boundary should not be so “far in” that the agent can change aspects of the environment arbitrarily, since the goal is to learn actions that the environment will respond to, not to have complete control of the environment. Also, the boundary should not be so “far out” that there is a significant difference between the action the agent chooses, and what action is actually carried out after all of the intervening activity occurs. For example, in the driving scenario there is a large “gap” between what the driving agent can actually control (such as the steering wheel and pedals), and the tire torques, so that the reward signal the agent receives when it believes that the tire torques are the values it has chosen, could actually be based on dramatically different tire torques due to external factors that the agent does not have any information about, such as the state of the transmission. However, as already mentioned, this can vary depending on the agent. An agent learning to drive might begin by having its actions control simple electrical signals to its joints and motors, and have the reward signal be based on the position of the pedals and steering wheel. Then once it has learned to solve that problem, its actions could become the position of the pedals and steering wheel, and have the reward signal be based on the position and velocity of the car. Once it has learned to solve that problem, the actions could become the position and velocity of the car, and the reward signal could be based on achieving some higher level goal such as reaching some destination. In other words, actions can become more “high level” and complex as the agent learns the policies of lower level actions and state information.
3. The return at each time would be: –Gamma^k, where k is the number of time steps until failure. This is nearly the same as for the continuing task, except that whereas the episodic task returns to time step 0 after each failure, the continuing task simply repositions the pole without resetting the time step. Therefore, all of the future failures would contribute to the negative reward signal at each time, but due to discounting they would become progressively smaller in magnitude as they become farther in the future.
4. The problem is that the agent is receiving a total reward of 1 regardless of how long it takes to exit the maze. In order to convey to the agent that the time taken to complete the task should be made as short as possible, either there should be a negative reward signal at every time step that does not result in the agent exiting the maze, or the positive reward for exiting the maze should be discounted.
5. When the vision system has access to only one frame, this does not provide the Markov state. For example, if a ball was thrown across the visual field of the vision system, and the single frame captured the ball somewhere in front of the visual system, then the previous frames would provide critical information regarding where the ball is likely to be in the next frame, because they would provide the means to estimate the velocity of the ball. If the camera is broken and receives no images, then it is not provided with any information regarding the current state at all. Since the Markov state contains information regarding the current state, the broken camera cannot have access to the Markov state.
6. See attached.
7. See attached.
8. See attached.
9. The question is somewhat ambiguous because of conflicting statements. On page 60 of the textbook it states, “episode termination [is] the entering of a special absorbing state that transitions only to itself and that generates only rewards of zero.” However, the question states, “Now consider adding a constant C to *all* the rewards in an episodic task” [italics mine]. I will consider each case separately. If the constant C is not added to the absorbing state reward, then adding a constant to every other reward does affect the relative values of states. This is because the equation for calculating K (the difference between V(s) when adding or not adding the constant C), as given by Solution 3.10, is the same as for the continuing task except that the maximum time step of k is not infinite, but is rather the time step T when the terminal state is reached. This number will vary depending on the current state, and so K is not a constant. As an example, consider the simple case of a linear, deterministic sequence of states, with s1 defined as the state one time step before the terminal state, and s2 defined as the state three time steps before the terminal state. Then s1’s value will only have C added to it, since the reward of the absorbing state has been defined as 0, while s2 will have the following extra value added to it:

C + Gamma(C) + Gamma^2(C)

For the case where C is added to the reward of the absorbing state, then adding the constant C makes no difference for the same reason as in the continuous task. That is, the same constant K defined by the equation in Solution 3.10 will be added to the value of every state, since the subsequent state to the absorbing state is defined to be the absorbing state for an infinite number of time steps.

17. See attached.