Homework 1: Chapters 1 - 5

The following exercises are due at the beginning of class on **Thursday, October 1**. Some of these problems may take a while to solve, so I recommend that you work on this assignment over the course of multiple days.

1. ***[10 pts.]*** Prove the following:
   a) For every purely reactive agent, there is a behaviorally equivalent standard agent.
   b) There exist standard agents that have no behaviorally equivalent purely reactive agent.
2. ***[10 pts.]*** There are two ways of specifying tasks by utility functions: by associating utilities with either states ($u : E \rightarrow \Re$) or with runs ($u : R \rightarrow \Re$). The second type of utility function is strictly more expressive than the first. Give an example of a utility function over runs that cannot be defined simply by associating utilities with states.
3. ***[45 pts. total]*** Consider the vacuum-world example from Chapter 3 (pp. 51-53). Note: This is different from the environment we used for our programming assignment. In particular, there are no obstacles and the agent does not need to return home after it is done cleaning the room.
   a) ***[15 pts.]*** Imagine that your agent must be able to perform the task in environments of arbitrary size. Assume that the agent's internal state includes a ground atom of the form worldSize(*width*, *height*) that is used to indicate the maximum x coordinate (*width* – 1) and maximum y coordinate (*height* – 1). Write a set of general rules that will allow the agent to function properly regardless of the world size. Try to keep this set of rules as small as possible. Hint: You may find it convenient to use predicates to represent some mathematical comparisons and to treat arithmetic expressions as terms (in fact, this could be considered shorthand for pre-defined functions that return the result of applying arithmetic operations to their terms). You may also assume the presence of other helpful mathematical predicates, as long as you are clear about how they should be defined.
   b) ***[15 pts.]*** Now consider a solution to the problem that uses situation calculus.
      i) Give precondition and effect axioms for the *forward* action. Note, you may need to modify some of the domain predicates so that they become fluents. As above, you may use arithmetic expressions as terms.
      ii) Give a successor state axiom for the fluent Facing(*d*,*s*), where *d* is the direction the agent is facing in situation *s*.
   c) ***[15 pts.]*** Finally, use the book's version of the subsumption architecture approach to design a purely reactive agent for this environment. You can assume a "wall" percept which tells the agent that it is facing the boundary of the room. You do not need to implement this agent, only give a formal description of it. How does it compare with the logic-based example? Which do you think will perform better? Which is simpler?
4. ***[10 pts.]*** Consider the architecture presented by Bratman, Israel, and Pollack in "Plans and Resource-Bounded Practical Reasoning." Compare Fig. 1 of this paper to the pseudo-code for a practical reasoning agent presented in Fig. 4.3 of our textbook (p. 76). Among other things, discuss which modules correspond to which functions, how the inputs and outputs are similar or different, and how the architectures differ in capabilities.

5. ***[10 pts.]*** Consider the Concurrent MetateM program in Figure 3.7 (p. 63). Explain the behavior of the agents in the system.
6. ***[15 pts.]*** Consider the single-agent Mars explorer example from Chapter 5 (pp. 92-94). Assume that the world is specified as a grid with the mothership in the center at location(0,0). The agent has actions: **GoNorth, GoEast, GoSouth, GoWest, PickupSample,** and **DropSample** and the percepts: **Obstacle(***direction***),** **Gradient(***direction***)**, **CarryingSample** and **AtSample.** Describe the design of a BDI agent to solve the problem. What types of knowledge will be needed to form its Beliefs? Summarize what desires should be generated by the options() function, and under what conditions each is possible. Finally, describe how the filter() function should select between competing options. You don't have to give pseudo-code, but you should give enough details so that I could see how to flesh out your design.