Homework 1: Chapters 1 - 5

The following exercises are due at the beginning of class on **Tuesday, October 2**. 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.]*** Show that state-based agents are equivalent in expressive power to standard agents, i.e. that for every state-based agent there is a behaviorally equivalent standard agent and vice versa.
2. ***[40 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) ***[10 pts.]*** Give the full definition (using pseudo-code if desired) of the *new* function, which defines the predicates to add to the agent's database. For convenience, you may assume that agent stores a global variable *action* which contains the last action it executed.
   b) ***[15 pts.]*** Now 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*). 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 functions to represent some arithmetic operations. You may also assume the presence of other helpful mathematical predicates, as long as you are clear about how they should be defined.
   c) ***[15 pts.]*** Finally, use Brook's subsumption architecture approach to design an agent for this environment. 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?
3. ***[25 pts. total]*** 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.**
   a) ***[10 pts.]*** Formulate the operations available to the agent using the STRIPS notation. Note, you must decide what set of predicates are sufficient. Hint: You may augment your STRIPS notation with the basic arithmetic operators.
   b) ***[15 pts.]*** Think about designing 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.

4. ***[10 pts.]*** Consider the following precondition and effect axioms. Assuming that these are the only axioms that are relevant to the fluent ***closed***, write a successor state axiom for it.

   $\text{Poss}(\text{Open}(d),s) \Leftrightarrow \text{closed}(d,s) \wedge \text{nextTo}(d,s) \wedge \neg\text{locked}(d,s)$
   $\text{Poss}(\text{Close}(d),s) \Leftrightarrow \neg\text{closed}(d,s) \wedge \text{nextTo}(d,s)$
   $\text{Poss}(\text{Explode}(e),s) \Leftrightarrow \text{combustible}(e,s)$
   $\text{Poss}(\text{Open}(d),s) \Rightarrow \neg\text{closed}(d, \text{Do}(\text{Open}(b),s))$
   $\text{Poss}(\text{Explode}(e),s) \wedge \text{near}(e,d,s) \Rightarrow \neg\text{closed}(d, \text{Do}(\text{Explode}(e),s))$
   $\text{Poss}(\text{Close}(d),s) \Rightarrow \text{closed}(d, \text{Do}(\text{Close}(b),s))$

5. ***[15 pts.]*** Consider the Jam system shown in Figure 4.6 (p. 85). Assume that PERFORM move $OBJ1 $OBJ2 has the following effects: add the fact that $OBJ1 is on $OBJ2, delete the fact that $OBJ2 is CLEAR (unless it is the table), and delete the fact that $OBJ1 is on whatever it used to be on. Trace the execution of this PRS agent until it has achieved its top-level goal. Show beliefs, intention stack, and executed action (if any) at each step. Be sure that the intention stack includes information that allows the agent to resume executing a plan that was suspended in order to accomplish a subgoal. If you wish, you may use a more compact representation for the agent's beliefs than that used by Jam (e.g., you might choose to use first-order logic).