Homework 7: Operational Semantics

1. (15 points) Give a terminal transition system for a deterministic Turing machine with a one-way tape. You should include a description of the set of configurations (Γ), the set of final configurations (T), the axioms and inference rules that define the transition relation ($\Rightarrow$), and both the input ($\mathcal{I}$) and output ($\mathcal{O}$) functions.

   A Turing machine consists of a finite state control, which can be thought of as a partial function δ that takes a state (from a set of states K) and a character read from the tape and returns a new state, a character to write on the tape, and a direction to move (right or left). The machine starts out with an initial tape (a string of characters from some alphabet Σ), its read head positioned at the leftmost character of the tape, and its finite state control in the state q_0. For each move, the machine reads a character from the tape at the current position, and then depending on the current state, writes a character at the current position, moves left or right one character on the tape, and goes to into a new state (as dictated by δ). If δ is not defined for the current state and read character, or if an attempt is made to move off the left end of the tape, then the machine halts.

2. (extra credit only) Formally define the language features you designed earlier in the term using a TTS. Does the formal definition clarify your English? Did you uncover any ambiguities?

3. (30 points) This exercise has 3 parts, choosing a language in which to implement the second part, implementing the second part in the language of your choice, and afterwards, evaluating the choice you made.

   (a) Look at the next part of this exercise, and pick either Prolog, Smalltalk, or SML as the language you would like to program that part in. State your choice, and give a paragraph describing your reasons for that choice. Give another paragraph describing why you do not think the other languages would be as suitable.

   (Look at the course specification for a description of how we will grade problems like this. Give technical reasons for your choice of language; you do not have to take run-time efficiency of the finished system into account.)

   (b) Implement a simple rewrite rule machine. To be specific you are to implement a relation/method/function `rrm` that takes as arguments a list of transition axioms and inference rules, and an initial configuration, and returns a final configuration, if there is one. A final configuration is a configuration to which no rules apply. You will have to decide on a precise syntax (form/encoding) for both the transition axioms and rules and for configurations.

   The simplification is that you do not have to deal with anything other than configurations, transitions between configurations, and inferences involving transitions. For example, you do *not* have to deal with assumptions or "environments", annotations, or side conditions (such as ensuring that there are no free variables in a given term).

   Note that transition systems need not be deterministic, nor must they always be terminating.

   (c) *After* you have finished the previous part, rethink your arguments in the first part (without changing them). Were they valid? How would you answer the first part if you had to do it all again?

4. (extra credit only) Add some provision for side-conditions to your rewrite rule machine.