ECE 153a/253
Homework 4

Due: Wed Nov. 3, 2010

Problems:

1. Design a UML state machine for the Coke Machine 1.1 that doesn't take your money if the machine jams.

2. Consider a random deterministic FSM with n states and m transitions implemented using nested switch and state table patterns. In the following problem, we are only interested in the order of growth as the variables are increased. O(n,m) -- thus ignore any constant factors. Choose a unit (i.e. function calls or program statments) and determine how that number changes as you increase the stated quantity. Similarly, estimations for time ignore the units of the problem but count the number of function calls, object dereferences etc. What is desired is only the order of change -- so in terms of n, does the measured quantity change like n (linear growth), like log(n) logarithmic growth, like n^2 ...

a. Estimate the total program size in terms of n and m for both kinds of implementation.

b. Estimate the execution time in terms of n and m for both kinds of implementation.

3. Download and build the code for the qhsmtst example. The code has a Makefile which should compile correctly on any linux machine and should be easy to convert to the terminal windows environment or on the MicroBlaze platform.

a. Instrument the code in main.c to allow you to estimate the relative probability of each of the repeatable states of the machine, given a random selection of input signal at each dispatch. Run the experiment for 10 sets of 10,000 dispatch calls and estimate your confidence in the probabilities.

4. Consider a finite state machine controlling a elevator for a 5 story building such as HFH. The elevator is a simple one with 1 call button on each floor and 5 in the cab. On arrival at each selected floor, the elevator pauses for 10 seconds before proceeding to its next destination. It moves from floor to floor at 5 seconds per floor unless it is stopped on a floor. Buttons are active until the elevator arrives on a given floor to clear it. Any combination of buttons is possible...

a. Design a HFSM to control the elevator and implement your design using the qepn event code.

b. For your design, determine the average time to arrive on each floor after the elevator is called, assuming that a random call button is pressed every 25 seconds. Once the elevator arrives, assume that the user gets in and punches his desired floor and that only one ueser enters at a time. (You have executable code -- just run it for long enough that you have a sound statistical sample for the average call time on each floor). For a sanity check, re-run with a delay of 120 sec (instead of 25) between calls.

If you have the time and interest-- consider the following: How would you implement a policy to minimize the amount of upward traversal the elevator needs to do (to lower the power cost)? Can you think of a way to measure the average number of asends per call? If you implement your policy, what do you expect to happen to the call time measured in part b?