CS 337 **Test 2** **4/5/06**

Open book and notes.

Max points = 50 Time = 50 min Do all questions.

1. (Finite State Machine; 20 points) The finite state machine in Figure 1 compares the magnitudes of two binary strings of equal length. The machine is fed the bits alternately from both strings, from high to low order. So, given that the first string is 010 and the second one is 011, the machine receives 001101. The machine accepts such a string if and only iff the first string is smaller in magnitude than the second string (for 001101, the machine will accept). Note that the machine is in a reject state after receiving an odd number of bits (i.e., when it has received more bits from the first string than from the second).

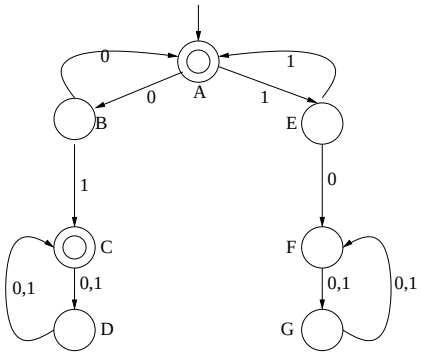

Figure 1: Finite State Machine to compare binary strings

   (a) (16 points) We would like to prove that the given machine meets the specification. But you will only do a part of the proving. Annotate the states with appropriate predicates; see P.72 of the book. Specify what needs to be proved for the transitions incoming and outgoing from each state and in the initial state. You don't have to prove the propositions.

   Hint: Let f be the bits of the first string and s of the second string at any state of the machine. Thus, if the machine has received 001, $f = 01$ and $s = 0$. Write $pre.f$, when f is nonempty, for the prefix of f which includes all but the last bit of f.

   First, write a predicate over f, $pre.f$ and s in each state. Next, generate one proposition for each edge and the initial state.

   (b) (4 points) Modify the machine so that it accepts iff the two input strings are different.

2. (Haskell; 30 points)

   (a) (8 points) Define a function that creates a list of unique elements from a sorted list. So, a possible input is `[2,2,3,3,4]` and the corresponding output is `[2,3,4]`. Use only the basic comparison operator (`==`) and recursion.

   (b) (9 points) Given is a non-empty list of elements L. Write a function that creates a list of the same length as L, and its i^{th} element is the list of elements up to and including the i^{th} element of L. Thus, given L to be `[2,3,4]`, the output should be `[[2],[2,3],[2,3,4]]`.

   Hint: Try to use `map` function.

   (c) (3 points) what is the type of the function being defined in (2b)?

   (d) (10 points) Prove that `map f (rev xs) = rev (map f xs)`, for any function `f` and list `xs`. Use the following definition of `rev`:

```
rev []       = []
rev (x: xs) = (rev xs) ++ [x]
```

   Use: `map f (xs ++ ys)= (map f xs) ++ (map f ys)`, and `map f (x:xs) = [f x]:(map f xs)`.