University of Washington November 23, 2011
Department of Computer Science and Engineering
CSE 311, Autumn 2011

Homework 9, Due Friday, December 2, 2011

Problem 1:

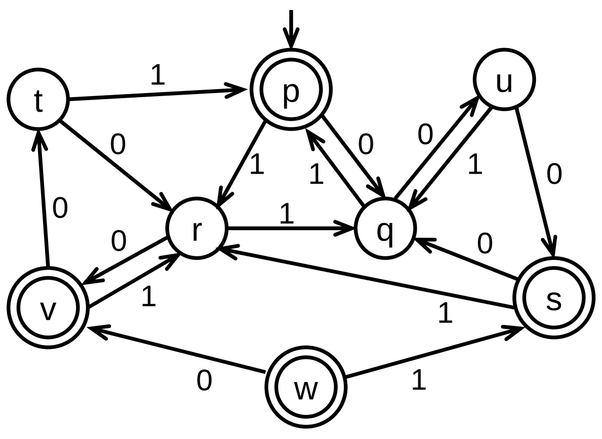
a.1) Give a three state DFA with state names a, b, and c to recognize the set of binary strings that have two consecutive 1's.

a.2) Give a three state DFA with state names 0, 1, and 2 to recognize the set of binary strings that have two consecutive 0's.

b) Use the above DFAs and the "product" construction for checking two properties at once to build a DFA with state names from $\{a, b, c\} \times \{0, 1, 2\}$ that recognizes strings that have two consecutive 1's and two consecutive 0's. (There are more efficient DFAs to recognize this set but just use the construction without optimizing it.)

Problem 2:
Design a finite state machine with outputs for a Candy Machine that dispenses a Gumball for 10 cents or M&M's for 15 cents. The machine takes nickels and dimes. It returns change if too much money is inserted or if the cost of the item selected is less than the amount of money deposited. A state is allowed to have multiple outputs.

Problem 3:
Apply the state minimization algorithm from the lectures to the DFA below. Write out the groups of states that you begin with as a sequence of sets of states. At each step, say which symbol and which group of states you are considering and how this splits the groups of states. Show how all the states are grouped after each step. When you have finished, draw the diagram for the resulting minimized DFA.

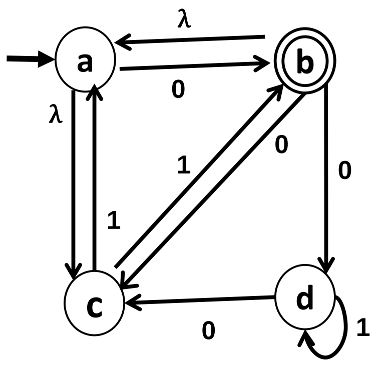


Problem 4:
Draw NFAs that recognize the languages described by each of the following regular expressions. Use the construction given in class or in the book or produce something simpler if you can.

a) $((01)^*1)^* \cup 11^*$

b) $(0 \cup 0^*1)^* \cup (1 \cup 1^*0)^*$

Problem 5:
Apply the construction given in lecture to convert the NFA below to a DFA that recognizes exactly the same language.



Problem 6:
Prove that the set of all binary strings with more 0's than 1's is not recognized by any DFA.

Extra Credit 7:
In this question you are to design a finite state controller for the traffic lights at an intersection of a busy two-way East-West street and a small two-way street that heads South of the E-W street but does not cross it: There is a left-turn lane for the westbound traffic on the E-W street to turn Southound and there is a sensor L that detects traffic waiting in this left-turn lane. There is also a sensor N to detect waiting Northbound traffic on the small street. We use input notation 0 to denote that neither sensor detects a car, B to denote that both sensors detect cars, and L and N to denote that only one of the sensors is activated. There are 4 traffic lights: Eastbound, Westbound. Northbound, and Left-turn that each cycles from Green to Yellow to Red back to Green in response to input signals. (We think of these outputs as EG,EY,ER, etc.) The length of time that a Yellow occurs is governed by a timer signal T that ends it. (We won't worry about how T is activated.)
Under normal circumstances, the Eastbound and Westbound lights are Green and the Northbound and Left-turn lights are Red. If there is waiting Northbound traffic, then all other lights must turn to Yellow and then to Red before the Northbound light can turn Green. Northbound traffic is rare enough that it takes precedence over all other traffic. If there is waiting Left-turn traffic, then the Eastbound and Northbound lights must turn to Yellow and then to Red before that traffic can turn left but Westbound traffic is unaffected. Left-turn traffic takes precedence over everything but Northbound traffic. So that the lights don't switch back instantly, after the Northbound light turns Green it must stay Green so long as there is still waiting traffic and for at least two timer signals before it turns Yellow. (There is no such requirement for the length of Green at any other traffic light.)