CSE 311: Foundations of Computing I

Homework 8 (due Friday, Dec 2nd at 6:00 PM)

1. DFAs [Online] (30 points)

For each of the following, create a *DFA* that recognizes exactly the language given.

(a) [10 Points] The set of all binary strings that contain at least two 1's *or* at most two 0's.

(b) [10 Points] Binary strings where if we treat them as a binary number, that number is congruent to 2 modulo 5. For example, 00111 is in the language (because $7 \equiv 2 \bmod 5$) but 011 is not.

(c) [10 Points] Consider a binary string w of even length. Let odd(w) be characters at odd position in w and even(w) be characters at even position in w. For example, if $w = 100110$ then odd$(w) = 101$ and even$(w) = 010$. Consider even(w) and odd(w) as binary numbers. Let L be the language

$$L = \{w \in \{0,1\}^* : w \text{ has even length and odd}(w) > \text{even}(w)\}.$$

For example, $00101101 \in L$ because $0110 > 0011$ but $001101 \notin L$ because $010 \not> 011$. Also, $110011 \notin L$.

> You must submit and check your answers to this question using https://grinch.cs.washington.edu/cse311/fsm.
> You also need to submit documentation in Canvas. For each state s of your machine write which strings will take DFA from the start state to s.

2. NFAs [Online] (15 points)

For each of the following, create an *NFA* that recognizes exactly the language given.

(a) [5 Points]The set of binary strings that contain 11 or do not contain 00

(b) [5 Points] The set of binary strings that contain 11 and do not contain 00

(c) [5 Points] The set of binary strings such that if they contain 11 then they do not contain 00.

> You must submit and check your answers to this question using https://grinch.cs.washington.edu/cse311/fsm.

3. FSM design [Online] (15 points)

In the old days of video gaming, you had to go through a very specific set of actions in order to deal damage to an enemy boss. Your FSM should have inputs **S**(sword), **A**(arrow), **D**(dodge), **P**(pause). The outputs are **OUCH**(boss damage), **OOF**(player damage), **FAIL**(player death), **WIN**(player wins).

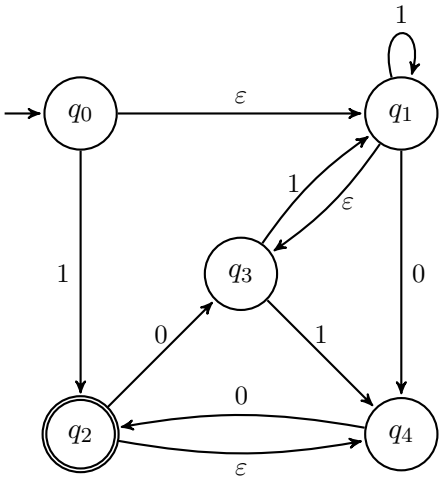
- If the actions are **D**, **A**, **D**, **S** in order, you should output **OUCH**.
- The preceding sequence can have pauses interspersed, but when the game is paused, no inputs are registered. (So after one pause, no other inputs matter until pause is pressed again.)
- Any other sequence of 4 (non-pause) actions should result in output **OOF**.
- After an **OUCH** or **OOF** output, the sequence of actions is reset.
- If two **OOF** outputs occur in a row without an **OUCH** in between, you should output **FAIL**.
- If **OUCH** output occurs, you should output **WIN**.

From the **FAIL** or **WIN** states, no actions matter.

You must submit and check your answers to this question using
https://grinch.cs.washington.edu/cse311/fsm.

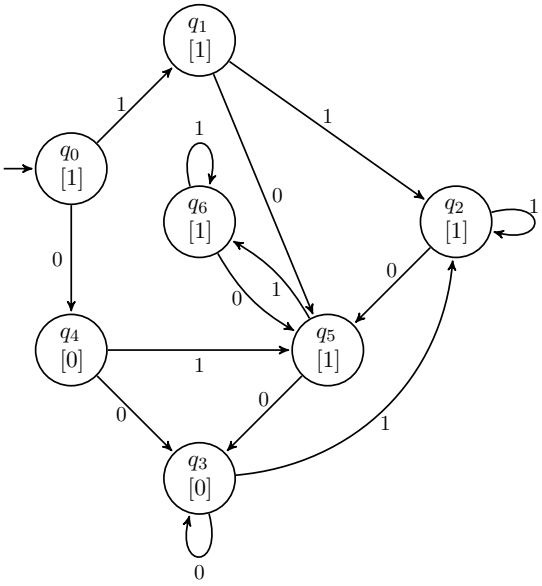
4. NFA to DFA (15 points)

Use the construction from lecture to convert the following NFA to a DFA. Label each state of the DFA using appropriate states of the original NFA.



5. Minimize This (15 points)

Use the algorithm for minimization that we discussed in class to minimize the following automaton. For each step of the algorithm write down the groups (of states), which group was split in the step the reason for splitting that group. At the end, write down the minimized DFA.



6. Regular Expression → NFAs [Online] (10 points)

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

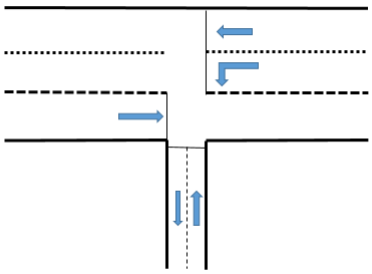
(a) [5 Points] $0110^*(110^* \cup 001^*)^*$.

(b) [5 Points] $(00^*1)^* \cup ((11^*0)^*(0011^*)^*)^*$.

You must submit and check your answers to this question using
https://grinch.cs.washington.edu/cse311/fsm.

7. EXTRA CREDIT: Traffic Control (-NoValue- points)

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, WG, WY, WR, LG, LY, LR, NG, NY, NR.) 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.)