CSE 105: Homework Set 2

Joshua Wheeler

April 16, 2014

1. Complemenation

Part 1A
Let

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

$$B = B(M)$$

is the language recognized by the machine.
Since

$$B = B(M) = \{b|\text{b is recognized by M}\}$$

We may define its complement

$$B' = \{b \mid \text{b is not recognized by M}\} =$$

$$\{b' \mid \text{b' is recognized by M}\} =$$

$$\{b' \mid b'\text{recognizes the complement of b}\}$$

Where b' $\in$ B', b' $\in$ B, b $\in$ B

The Machine to recognize B' would be

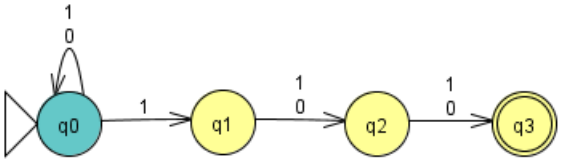
$$M' = \{Q, \Sigma, \delta, q_0, F'\}$$

where F' = Q - F

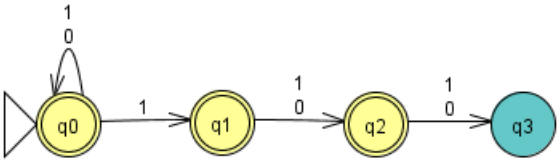
Part 1B
Let M be the NFA from the book in example 1.30 which recognizes

$$C = \{c \mid \text{ c contains a 1 in the third position from the end } \}$$

The NFA is

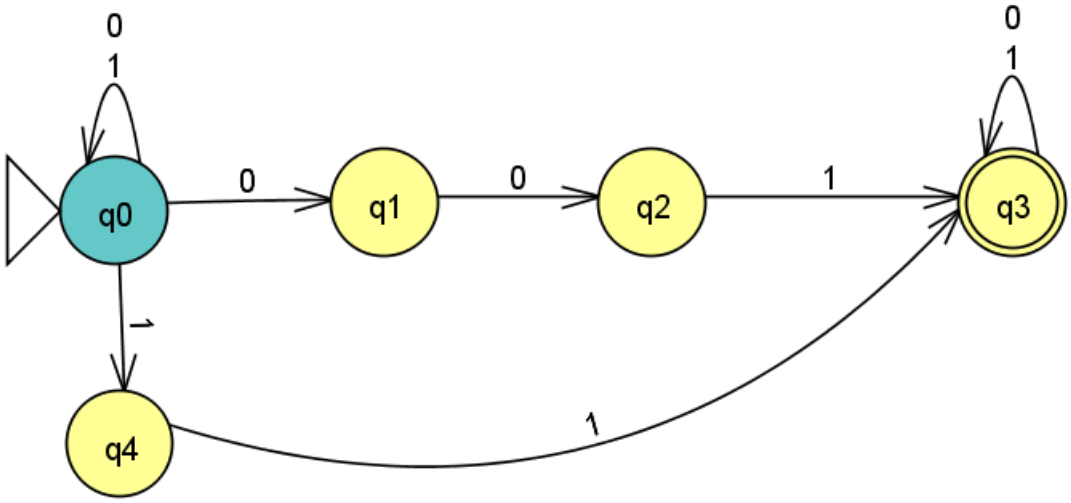


The complement of this NFA should be
C' = $\{C \mid$ c does not contain a 1 in the third position from the end$\}$
but if we swap the accept and nonaccept states we have the NFA



This NFA will accept every string $\{x \mid x \in \Sigma^*\}$
which clearly is not the complement of C. This implies that the set of languages recognized by the NFA does not work if we swap the accept and non-accept states but it is closed by complement. As in, it is possible to find another NFA with different transitions and states from the original NFA which accepts the complement of the input string. This is because we have already proved that every NFA can be transformed into a DFA. IF we take any NFA, transform it to a DFA first and then take the complement of that DFA then convert it back to an NFA, then we have successfully found that NFA's are closed under complementation.

2. NFA Design



This NFA uses the power of parallel processing to determine whether or not the string 001 or 11 is a substring of the input by using 5 states. The initial state, q0, always points back to itself. This is in order to catch invalid strings. q0 will always run in parallel with every other state until the end of the string where an accept state may or may not have been reached. If the string 001 and 11 go undetected then all other states eventually become stuck and the only state reading the string is q0. If the string starts with a 1, q0 transitions to q4 and listens for a second 1 and if that second 1 is detected the string will indefenitely stay in the accept state, q3. Any other characters in the string will be valid after the 11 substring is detected so q3 always points back to itself for any input. Similarly, once the first 0 is detected, q0 will transition to q1. On the second 0 q1 transitions to q2 and then on a 1 leading two 0's, it will go to the accept state q3 or become stuck otherwise. The below NFA and input table demonstrate this behavior.

Below are the test inputs for this DFA:

Input	Result
001	Accept
11	Accept
1001	Accept
011	Accept
10101001	Accept
101010001	Accept
010101001	Accept
01010001	Accept
101011	Accept
1010111001	Accept
0101101	Accept
00000001111111111	Accept
111100000	Accept
01010101010101	Reject
10101010101010	Reject

3. Epsilon Moves

Part 3A:
This NFA counts the number of 1's over the unary alphabet $\Sigma = \{1\}$.

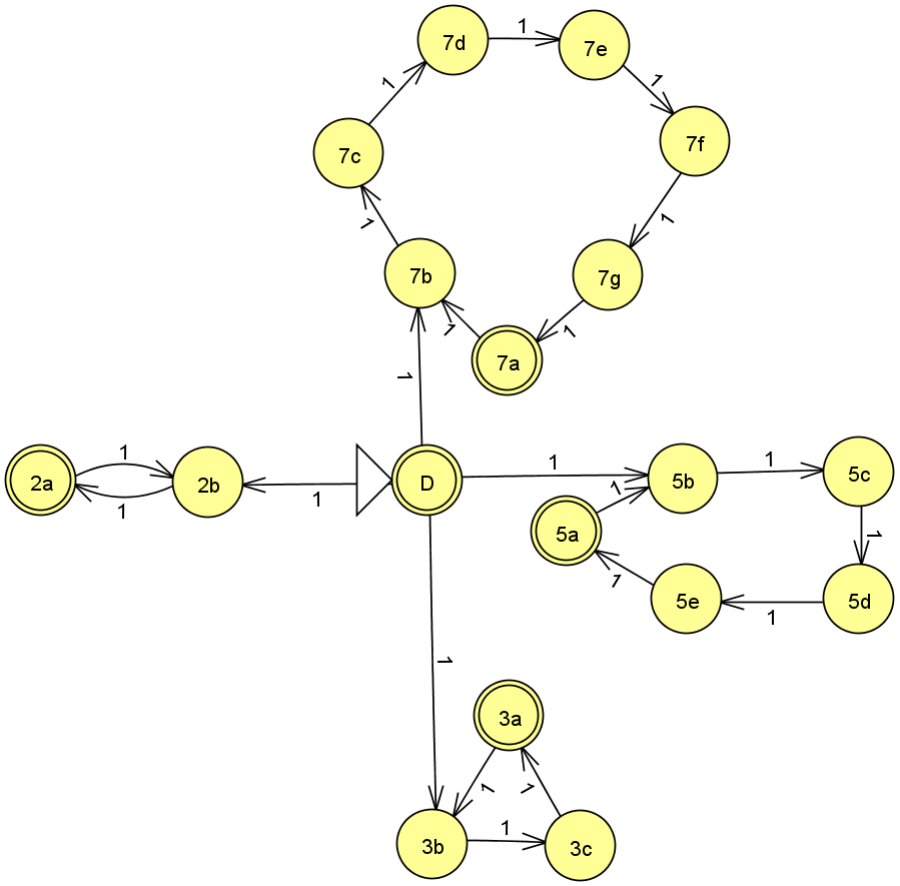
A string will be accepted if the number of 1's is a multiple of the prime numbers 2, 3, 5 or 7. The ϵ runs these checks of multiplicity in parallel. In mathmematical terms, this machine recognizes a language

$$L = \{\{1\}^n \mid 2 \text{ divides } |\{1\}^n| \text{ or } 3 \text{ divides } |\{1\}^n| \text{ or } 5 \text{ divides } |\{1\}^n| \text{ or } 7 \text{ divides } |\{1\}^n|\}$$

where n is an integer

Part 3B:
This NFA is very similar to that of the original. In order to make the NFA mimic the same behavior without the ϵ the initial state was changed to an accept state. This is because in the original NFA, the first ϵ move went to an accept state before any inputs were accepted. Next, in order to maintain

divisibility, the initial state of the new NFA maps to a chain of states detecting the divisibility of 2, 3, 5 and 7 respectively. It has the same behavior as the original NFA, since after a multiple of 2, 3, 5 or 7 1's are detected, the NFA will accept the string. This is accomplished by forcing the initial state to map to the the state in the respective divisibility chain that has detected the first one. These divisibility chains then loop around to detect the subsequent multiples. The NFA and a list of corresponding inputs are shown below.



Input	Result
1	Reject
11	Accept
111	Accept
1111	Accept
11111	Accept
111111	Accept
1111111	Accept
11111111	Accept
111111111	Accept
1111111111	Accept
11111111111	Reject
111111111111	Accept
1111111111111	Reject
11111111111111	Accept
111111111111111	Accept
1111111111111111	Accept
11111111111111111	Reject
111111111111111111	Accept
1111111111111111111	Reject
11111111111111111111	Accept
111111111111111111111	Accept
1111111111111111111111	Accept
11111111111111111111111	Reject

4. State Complexity
   Part 4A:
   In order to show that $\equiv_L$ is a recurrence relation we must prove that it is reflexive, symmetric and transitive.

   Reflexivity Proof:
   We can see that X$\equiv_L$X is trivially true. It is saying that x is indistinguishable from itself. In other words:

$$x \equiv_L x \implies \{x, x \mid \forall z \in L, xz \in L, xz \in L\} = \{x \mid \forall z \in L, xz \in L\}$$

   Therefore X$\equiv_L$X

   Symmetric Proof:
   In this case, we want to prove $x \equiv_L y \Leftrightarrow y \equiv_L x$
   By definition we can see that

$$x \equiv_L y \implies \{x, y \mid \forall z \in L, xz \in L, yz \in L\}$$

   Also,

$$y \equiv_L x \implies \{y, x \mid \forall z \in L, yz \in L, xz \in L\}$$

   These are equivalent statements. This relation must be symmetric. If x is indistinguishable from y then y must be indistinguishable from x.

Transitivity Proof:
To prove transitivity we must show that

$$(x \equiv_L w) \wedge (w \equiv_L y) \implies x \equiv_L y$$

We can reason through this. The left side of the implication says that $xz \in L$ whenever $wz \in L$ and also that $wz \in L$ whenever $yz \in L$. Since this is true, then $wz \in L$ will only be true whenever both $xz \in L$ and $yz \in L$. Hence, the right side of the implication must be true. Therefore, the relation is transitive.

Because we have proved that the relation is reflexive, symmetric and transitive, it must be an equivalence relation.

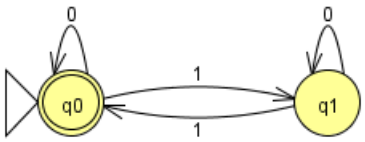
Part 4B
Let X=010 and y=0110
For any z, x and y are distinguishable since the DFA will only accept a string if it has zero or more 0's and an even number of 1's. If we choose a string z such that z has one or more 1's then the evennness or oddness of x and y will always toggle. For example, lef z=10. Then xz=01010 and yz=011010 so xz has an even number of 1's and will be accepted whereas yz has an odd number of 1's and will be rejected. This proves that there are at least two elements which are pairwise distinguishable so the index of L must be at least 2.

Part 4B Extra Credit:
Since the DFA recognizes
$\{w \mid$ w has zero or more 0's and an even number of 1's $\}$
it is somewhat intuitive that the index of L is exactly 2. If we pick a string $x \in X$ such that
X = $\{x \mid$ x has an odd number of 1's $\}$
and a $y \in Y$ such that
Y = $\{y \mid$ y has an even number of 1's $\}$
then $x \notin L$ and $y \in L$. Now if we tried to choose a third string from some other set W where
$W = \{w \mid w \in \{0,1\}^*\}$
the element $w \in W$ will either have an odd or even number of 1's. So any w we choose has the property that $w \in X$ or $w \in Y$ and since this is the case, any third string we pick will be indistinguishable with either x or y.

Part 4C:
This DFA can be described by the relation

$W = \{w \mid w$ has zero or more 0's and an even number of 1's $\}$

The minimized DFA is:



We can see this is equivalent to the original DFA because in JFlap they both produced the following equivalent outputs:

Input	Result
0	Accept
00	Accept
01	Reject
10	Reject
11	Accept
001	Reject
010	Reject
011	Accept
100	Reject
101	Accept
110	Accept
111	Reject
1000	Reject
1001	Accept
1010	Accept
1011	Reject
1100	Accept
1101	Reject
1110	Reject
1111	Accept