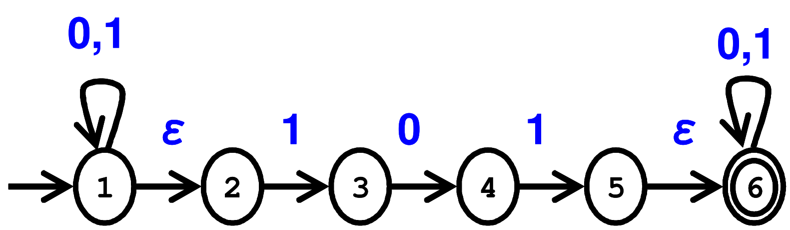
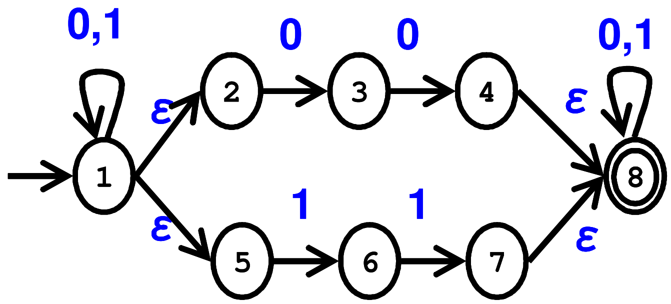
CMSC 330, Practice Problems 2 (SOLUTIONS)

1. Regular expressions and languages
   a. From the perspective of formal language theory, what is a language?
      Set of strings
   b. Given the language A = {"aa", "c"} and B = {"b"}, what is the language AB?
      {"aab", "cb"}
   c. Given the language A = {"aa", "c"}, what is the language A^0?
      { ε }
   d. Given the language A = {"aa", "c"}, what is the language A^2?
      { "aaaa", "cc", "aac", "caa" }
   e. Given the language A = {"aa", "c"}, what is the language A*?
      { ε, "aa", "c", "aaaa", "cc", "aac", "caa", "aaaaaa" … }
   f. Give a regular expression for all binary numbers including the substring "101".
      (0|1)*101(0|1)*
   g. Give a regular expression for all binary numbers with an even number of 1's.
      (0*10*1)*0* or 0*(10*10*)*
   h. Give a regular expression for all binary numbers that don't include "000".
      (01 | 001 | 1)*(0 | 00 | ε)

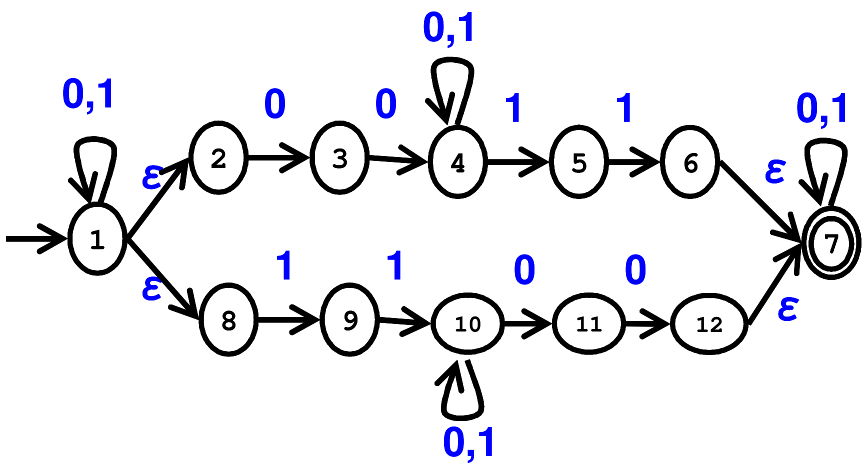
2. Finite automata
   a. When does a NFA accept a string?
      If there any path for the string that ends at a final state for the NFA
   b. How long could it take to reduce a NFA with n states and t transitions to a DFA?
      $\mathbf{2^n}$
   c. Give a NFA that only accepts binary numbers including the substring "101".



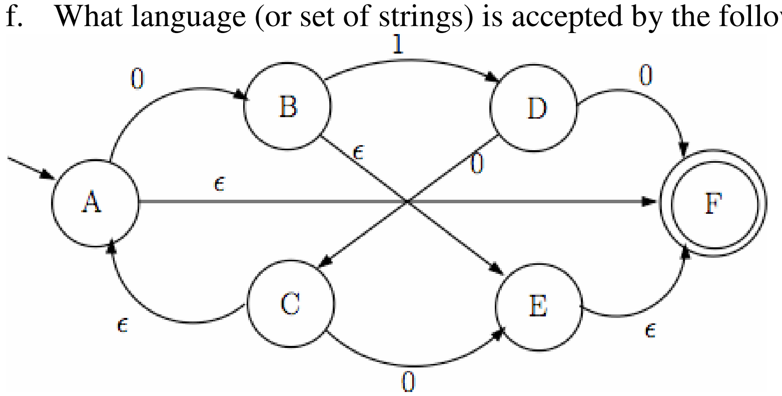
d. Give a NFA that only accepts binary numbers that include either “00” or “11”.



e. Give a NFA that only accepts binary numbers that include both “00” and “11”.



f. What language (or set of strings) is accepted by the following NFA?

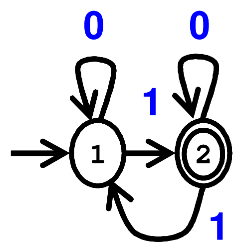


(010)*(0|ε)

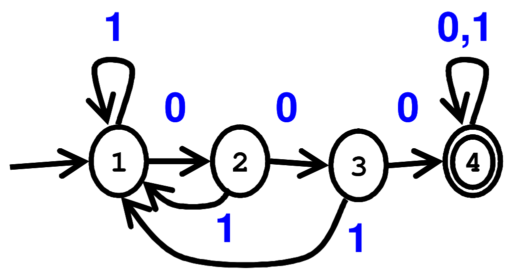
g. Compute the ε-closure of the start state for each of the NFA above.

- **For NFA in (c) ε–closure(1) = {1,2}**
- **For NFA in (d) ε–closure(1) = {1,2,5}**
- **For NFA in (e) ε–closure(1) = {1,2,8}**
- **For NFA in (f) ε–closure(A) = {A,F}**

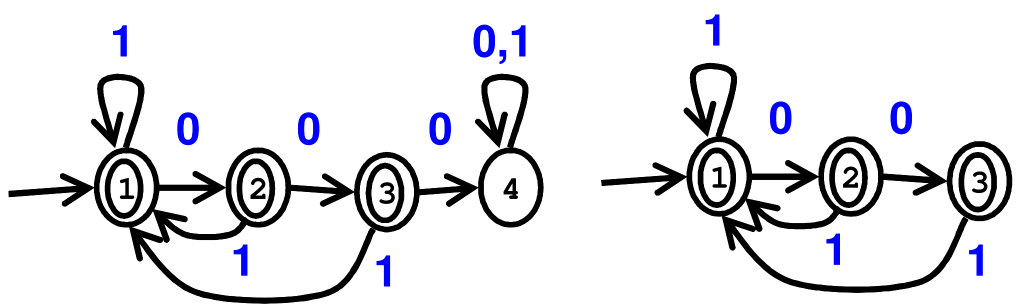
h. Give a DFA that only accepts binary numbers with an odd number of 1's.



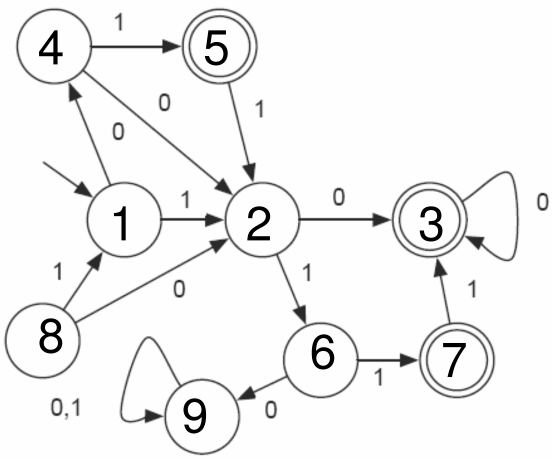
i. Give a DFA that only accepts binary numbers that include "000".



j. Give a DFA that only accepts binary numbers that don't include "000".



k. What language (or set of strings) is accepted by the following DFA?



Described as a list of strings:

{ "01", "111", "0011", "01111", "10", "000", "0110", "1111", "00111", "011111"…}
where all underlined strings may have any number of 0s appended

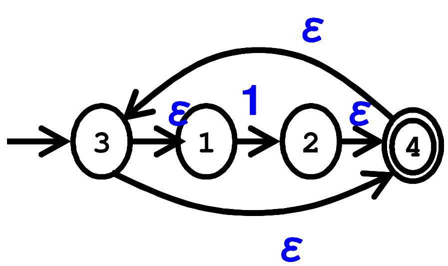
Described as a regular expression: 01 | (1 | 00 | 011)(11 | (0 | 111)0*)

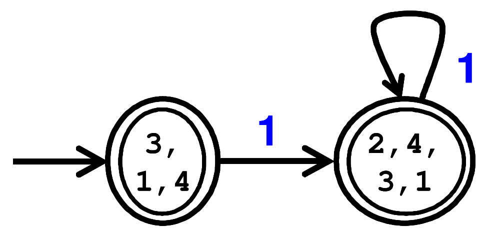
Explanation (for each underlined portion of RE)

- **01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 1 to 5 and accepts**
- **01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 1 to 2, then…**
- **01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 2 to 7 and accepts**
- **01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 2 to 3, then…**
- **01 | (1 | 00 | 011)(11 | (0 | 111)0*) accepts w/ 0 or more 0's**

1. For each regular expression: 1*, (0|01)*0
   a) Reduce the RE to an NFA using the algorithm described in class.
   b) Reduce the resulting NFA to a DFA using the subset algorithm.
   c) Show whether the DFA accepts / rejects the strings "1", "11", "101"
   d) Minimize the resulting DFA using Hopcroft reduction
   e) Are any 2 of the minimized DFA identical?

1* → NFA → DFA





Accept / reject

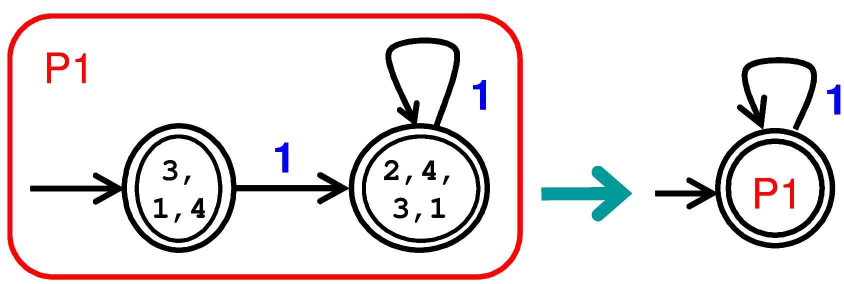
- "1"   {3,1,4} → {2,4,3,1} accept
- "11"   {3,1,4} → {2,4,3,1} → {2,4,3,1} accept
- "101"   {3,1,4} → {2,4,3,1} → reject

Minimized DFA

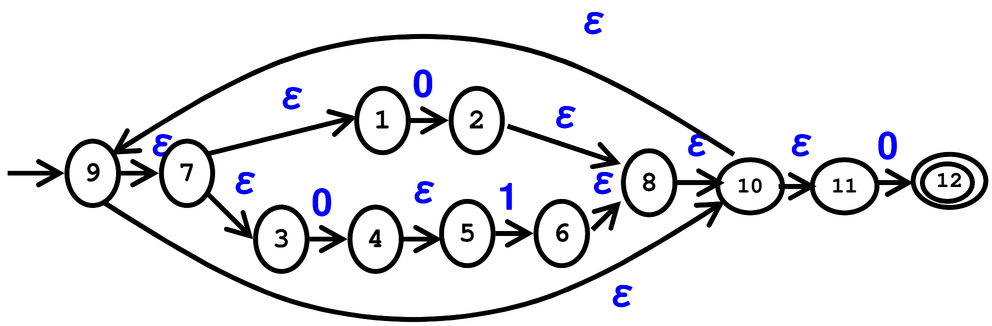
Initial partitions: accept ={ {3,1,4}, {2,4,3,1} } = P1,
nonfinal = Ø

- move({3,1,4}, 1) → P1
- move({2,4,3,1}, 1) → P1

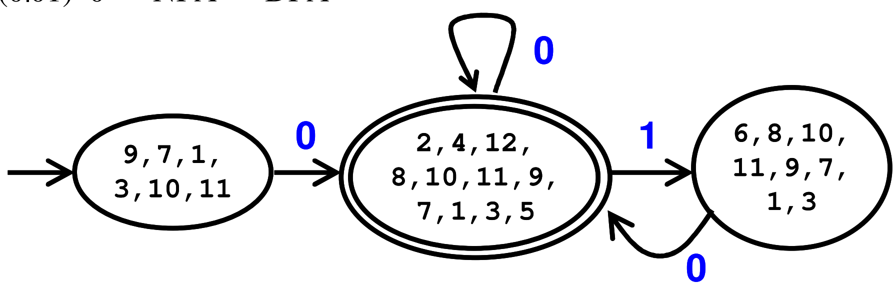
No need to split P1, minimization done. After cleanup, minimal DFA is



(0|01)*0 → NFA



(0|01)*0 → NFA → DFA



Accept / reject

- "1" {9,7,1,3,10,11} → reject
- "11" {9,7,1,3,10,11} → reject
- "101" {9,7,1,3,10,11} → reject

Minimized DFA

Initial partitions: accept ={ {2,4…}} = P1,
nonfinal ={ {9,7…}, {6,8…}} = P2

- move({9,7…}, 0) → P1
- move({6,8…}, 0) → P1
- move({9,7…}, 1) → reject
- move({6,8…}, 1) → reject

No need to split P2, minimization done. After cleanup, minimal DFA (different from previous minimal DFA) is