

ECE 537 - Foundations of Computing  
Prof. Sen  
**Homework #2**  
**Solutions**

1. Draw the diagram of a DFA that can recognize these languages over the alphabet  $\Sigma = \{a, b\}$ :

- (a)  $L = \{a^*b^*a^*\}$
- (b)  $L = \{w \mid w \text{ has at least three } a\text{'s and at least two } b\text{'s}\}$
- (c)  $L = \{w \mid w \text{ has an even length and an odd number of } a\text{'s}\}$
- (d)  $L = \Sigma^* - a^*b^*$ , which are the strings not in  $a^*b^*$
- (e)  $L = \{w \mid \text{every odd position is an } a\}$

**Answer:** See attached DFA diagrams.

2. Let  $B_n = \{a^k \mid k \text{ is a multiple of } n\}$ . Show that for each  $n \geq 1$ , the language  $B_n$  is regular.

**Answer:** For each  $n \geq 1$ , we can build a DFA with the  $n$  states  $q_0, q_1, \dots, q_{n-1}$  to count the number of consecutive a's modulo  $n$  so far. For each "a" that is input the counter increments by 1 and jumps to the next state in M. If we get another symbol, then we go to a fail state from which we never return. The machine accepts the string iff it stops at  $q_0$ . That means that the length of the string consists of all a's and its length is a multiple of  $n$ .

3. Suppose you have the "reverse" language  $L^R$  that is defined as follows  $L^R = \{w \mid w^R \in L\}$ , where  $w^R$  is defined as the symbols of the string in reverse order and  $L$  is a regular language. For example, if  $w = 01011$  then  $w^R = 11010$ . Is  $L^R$  a regular language? Show why or why not.

**Answer:** Yes,  $L^R$  is a regular language if  $L$  is regular. We cannot simply reverse the arrows in the DFA that recognizes  $L$ , because there might be several accept states and so the question naturally comes up as to where you actually begin. But you can turn it into an NFA that runs in reverse order, and we know that an NFA can be turned into a DFA. Hence  $L^R$  is a regular language.

4. Given language  $L$  over alphabet  $\Sigma = \{0, 1\}$  such that  $L = \{w \mid w \text{ contains equal number of "01" and "10"}\}$ . For example  $101 \in L$  because there is one "10" and one "01", but  $1010$  is not (there are two "10" and only one "01"). Is  $L$  a regular language? Show why or why not.

**Answer:**  $L$  is really the language of words that start and end with the same symbol, ie.,  $0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1$ . This is obviously regular.

5. Prove that NFA's can be exponentially smaller in size than DFA's by giving a family of languages  $L_k$  that can be recognized with an NFA of  $k$  states but need a minimal DFA of  $\Omega(2^k)$  states.

**Answer:** Let  $L_k$  to be the language of words over alphabet  $\Sigma = \{0, 1\}$  where the  $(k - 1)$ th-to-last symbol is a 1.  $L_k$  can be recognized with an NFA with  $k$  states as we showed in class. But the minimal DFA for  $L_k$  would have to have at least  $2^{k-1}$  states. To see this, suppose that  $u$  and  $v$  differ somewhere in their last  $k - 1$  symbols. In that case, there is some  $j < k$  such that the  $j$ th-to-last symbol of  $u$  is a 1 but a 0 in  $v$ . Then, if  $w$  is of length  $k - j - 1$ , then  $uw \in L$  but  $vw \notin L$ , so  $u \not\sim v$ . Therefore, we need at least  $2^{k-1}$  equivalence classes one for each final sequence of  $k - 1$  symbols.

6. Suppose we modify the NFA definition so that it accepts only if all the computation paths yield an accept answer. Show that this new NFA recognizes exactly the regular languages by demonstrating how to convert the new NFA into a DFA that recognizes the same language.

**Answer:** This is very simple and almost the same as the conversion from a standard NFA to a DFA. The only difference is that we change the definition of the accept state  $q'$  to make sure that all the states are accepting. Formally, we can write this as  $F' = \{q' \in 2^Q : q' \subseteq F\}$ .

7. We showed that a DFA can be modified to accept the complement of a language  $L$  by simply swapping its accepting and non-accepting states. Show by example that this is not true for NFA's.

**Answer:** This is very easy to show, because of the asymmetry that exists between the accept and reject condition of an NFA. An NFA accepts if *any* of its resulting states are accept, not all of them, but it will reject only if all of them reject. So take any NFA which ends up in some accept and some reject states, but accepts. Then reverse the accepting and rejecting states. The new NFA still accepts the same input.

8. Give two proofs (one using the pumping lemma and the other equivalence classes), that the language  $L = \{a^{2^n} \mid n \in \mathbb{N}\} = \{a, aa, aaaa, aaaaaaaaa\}$  is not regular.

**Answer:** Proof 1 (pumping lemma). Suppose that that pumping lemma holds for  $L$ . Then there exists a  $p$  such that for all  $w \in L$  with  $|w| \geq p$ , there would be some way to write  $w = xyz$  with  $|y| \leq p$  such that  $xy^iz \in L$  for all  $i \geq 0$ . This is a “unary” language in which the alphabet consists of only one symbol. This means  $w = a^l$  and  $y = a^k$  for some  $k < p$ , so  $xy^iz = w^{l+(i-1)k}$ . Then the pumping lemma becomes the claim that for all  $l \geq p$ , if  $a^l \in L$ , then  $a^{l+(i-1)k} \in L$  for all  $i \geq 0$ , for some  $k \leq p$ . In particular, considering  $xy^2z$ , the pumping lemma claims that  $a^{l+k} \in L$  for some  $k \leq p$ .

Now suppose that  $l \geq p$  and  $a^l \in L$ , let  $a^m$  be the next shortest word in  $L$ . For the pumping lemma to be true, we need the gap in their lengths  $m - l \leq p$ , otherwise  $xy^2z = a^{l+k}$  cannot be in the language since  $l + k \leq l + p < m$ . Therefore if the gaps in a unary language don't have a constant upper bound (it has arbitrarily large gaps) it cannot be regular.

For  $\{a^{2^n}$  in particular, if the pumping lemma is true with some value of  $p$ , let  $w$  be  $a^l$  where  $l$  is the smallest power of 2 greater than  $p$ . Then the length of the gap between  $w$  and the next shortest word  $a^{2l}$  is  $l$  which is greater than  $p$ .

**Proof 1 (Equivalence classes).** It turns out that none of the words in  $L$  are equivalent to each other. To see this, let  $u = a^{2^m}$  and  $v = a^{2^n}$  for any  $m \neq n$  and let  $w = u$ . Now  $uw = a^{2^{2^m}} = a^{2^{m+1}} \in L$ , but  $vw = a^{2^m} + a^{2^n} \notin L$  since the sum of two distinct powers of 2 is never a power of 2. Thus  $u \not\sim v$  for any distinct  $u, v \in L$ , so there are an infinite number of equivalence classes. Hence  $L$  is not regular.

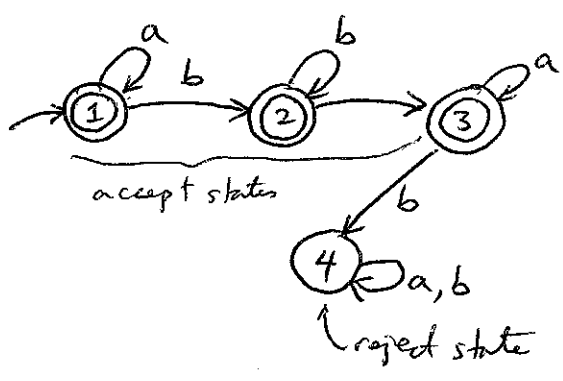
9. For a regular language  $L$  define  $L_{\frac{1}{2}} = \{x \mid xy \in L, \text{ such that } |y| = |x|\}$ . In other words,  $L_{\frac{1}{2}}$  is the first half of the words in  $L$ , so for every  $x \in L_{\frac{1}{2}}$ , there exists a string  $y$  such that  $xy \in L$ . Prove that  $L_{\frac{1}{2}}$  regular.

**Answer:** The basic idea is to simulate two copies of the DFA in parallel, one that moves forward from the start of state by reading  $x$ , the other that moves non-deterministically backwards from an accepting state by guessing  $y$ . Then if both states are equal after reading the string  $x$ , then we have met in the middle and  $xy \in L$ . Formally, let  $M = (Q, \Sigma, \delta, q_0, F)$  be a DFA for  $L$ , and define a new NFA  $M' = (Q', \Sigma, \delta', q'_0, F')$  as follows:  $Q' = Q \times Q$  (corresponding to the two copies of  $M$  running in parallel), let its start state  $q'_0$  have  $\epsilon$ -transitions from  $q'_0$  to each of  $q' \in \{(q_0, q_f) \mid q_f \in F\}$  (these are the accepting states of  $M$ ), and define the transition functions as  $\delta'((q_1, q_2), a) = \{(\delta(q_1, a), q_3) \mid \exists b : \delta(q_3, b) = q_2\}$ . Here  $b$  is the next symbol of  $y$  (read in reverse) which we guess to move backwards from  $q_2$  to  $q_3$ . Finally,  $F' = \{(q, q)\}$ . Then  $M'$  recognizes  $L_{\frac{1}{2}}$ .

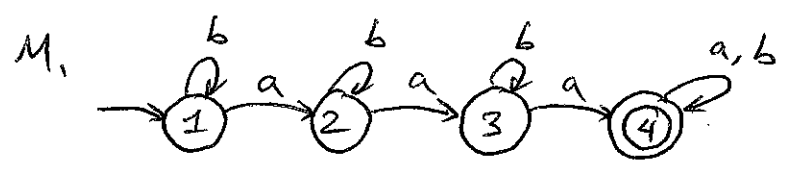
10. What are the equivalence classes of language  $L$  over alphabet  $\Sigma = \{a, b\}$ , where  $L = \{w \mid w \text{ contains the substring } aba\}$ ? Use these classes to draw the minimal DFA for  $L$ .

**Answer:** See attached solution and DFA diagram.

1. a)

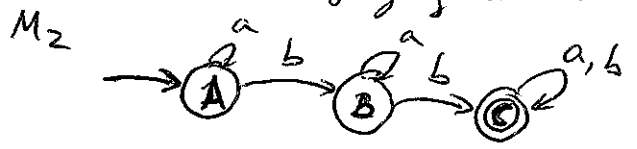


b) DFA for recognizing at least 3 a's



$Q_1 = \{1, 2, 3, 4\}$   
 $F_1 = \{4\}$

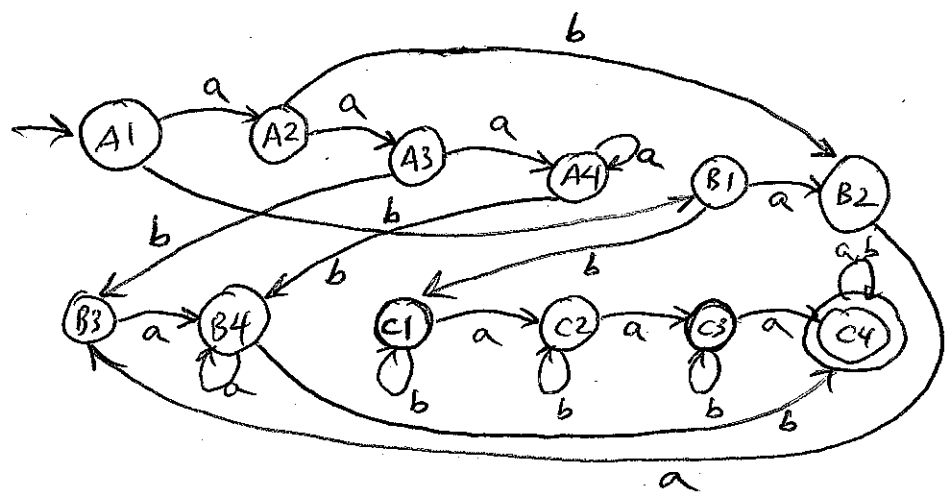
DFA for recognizing at least 2 b's



$Q_2 = \{A, B, C\}$   
 $F_2 = \{C\}$

Now take the intersection of the two:

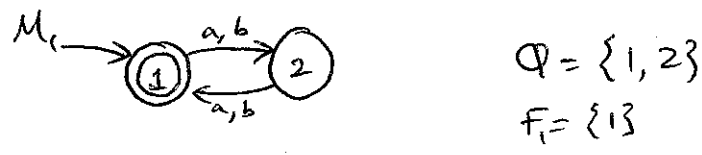
$Q = Q_1 \times Q_2 = \{A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4\}$   
 $F = \{C4\}$



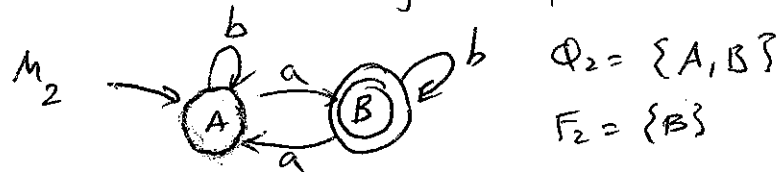
key: The (A1, A2, A3, A4)  
 (B1, B2, B3, B4)  
 and (C1, C2, C3, C4)  
 are all chained  
 together with a's  
 and then a 'b' moves  
 it from A to B to C.

c) Same idea as (B)

DFA for detecting even length



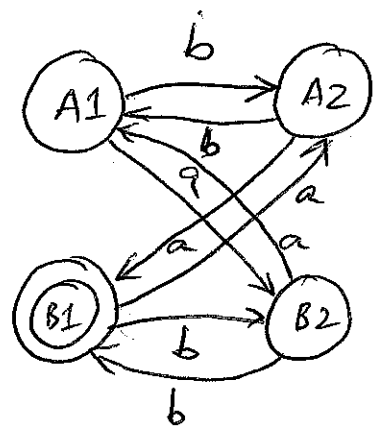
DFA for detecting odd # of a's



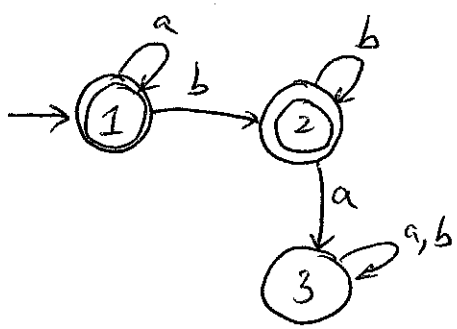
Now take the intersection of the two :

$Q = Q_1 \times Q_2 = \{A1, A2, B1, B2\}$

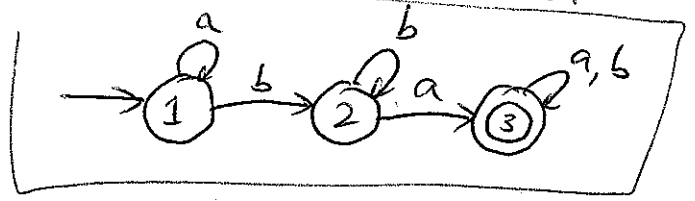
$F = \{B1\}$



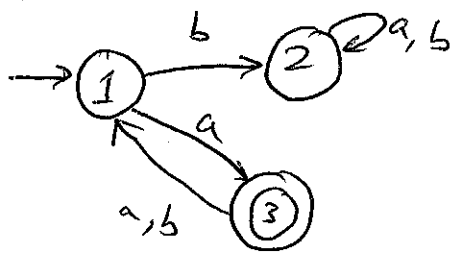
d) Recognize  $a^*b^*$



Now flip the accept & reject states :



e)



10. Equivalence classes:

1. Strings that do not have "aba" at end in "a"
2. Strings that do not have "aba" at end in "ab"
3. Strings that do not have "aba" at <sup>do not</sup> end in "ab"
4. Strings that have "aba"

