

1. Write regular expressions for the following informally described languages:
  - a. All strings of a's and b's with the subsequence abb.

$\Sigma = \{a, b\};$

$(a|b)^*abb(a|b)^*$

- b. All strings of a's and b's with an even number of a's and an odd number of b's.

**The expression**

**$(00|11)^*((01|10)(00|11)^*(01|10)(00|11)^*)^*$  given in the notes is for going from a state back to itself. Let R denote this regular expression, replacing 0 by a and 1 by b.**

**There are two paths from the starting state (let's call it S) to the state representing even number of a's and odd number of b's (let's call it T): S b T or S aba T.**

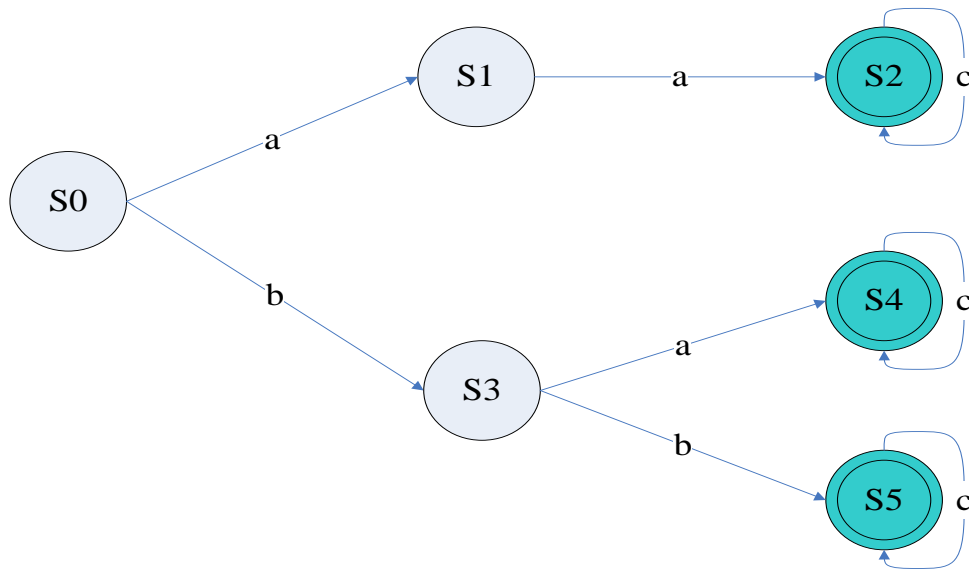
**From each state (starting, intermediate, and ending), one can go arbitrarily and back to that state. So, you can replace each state by R.**

**So the answer is  $RbR | RaRbRaR$**

**Note that regular expressions do not have a minimized form. So, there is no need to further minimize anything.**

2. Consider the regular expression  $aac^* | b(a|b)c^*$  defined on  $\Sigma = \{a, b, c\}$ .

a. Construct the NFA for the regular expression. You can directly draw the NFA without going through the RE-to-NFA steps.



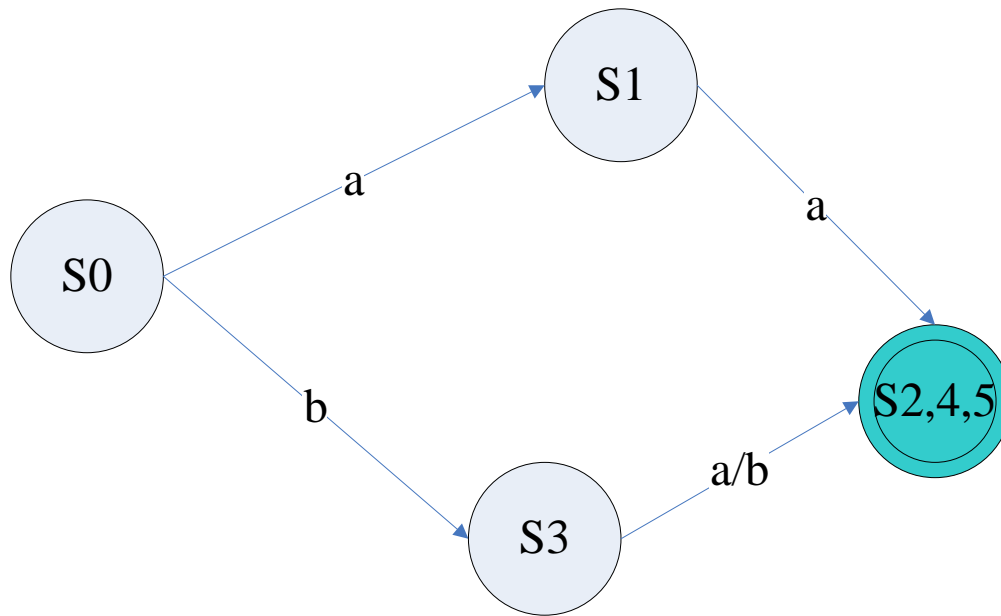
b. Convert the NFA to DFA. Show the conversion steps.

	a	b	c
S0	S1	S3	X
S1	S2	X	X
S2	X	X	S2
S3	S4	S5	X
S4	X	X	S4
S5	X	X	S5

c. Minimize the DFA. Show the minimization steps.

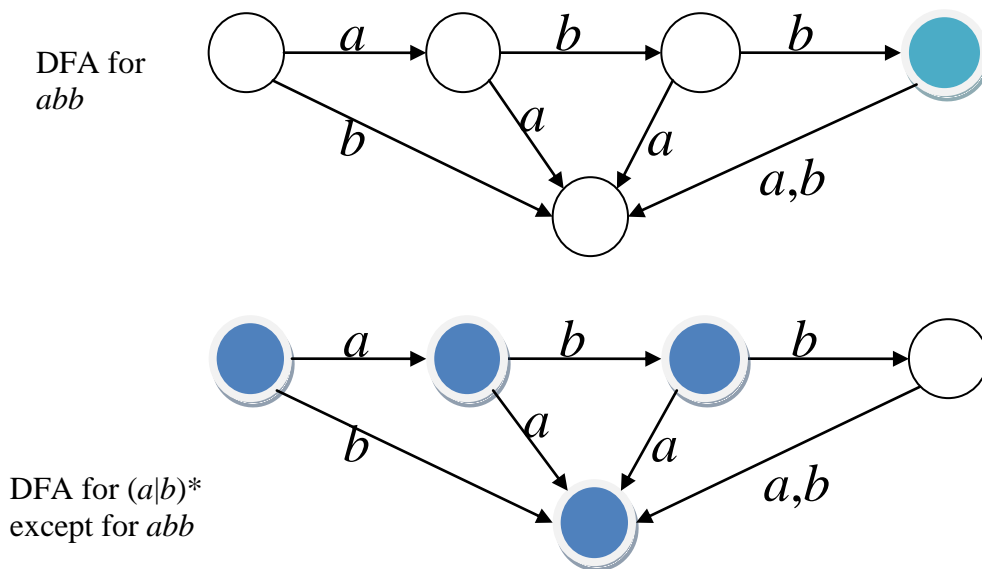
**S0 S1 S3 ε | S2 S4 S5**

**S0 ε | S1 | S3 | S2 S4 S5**



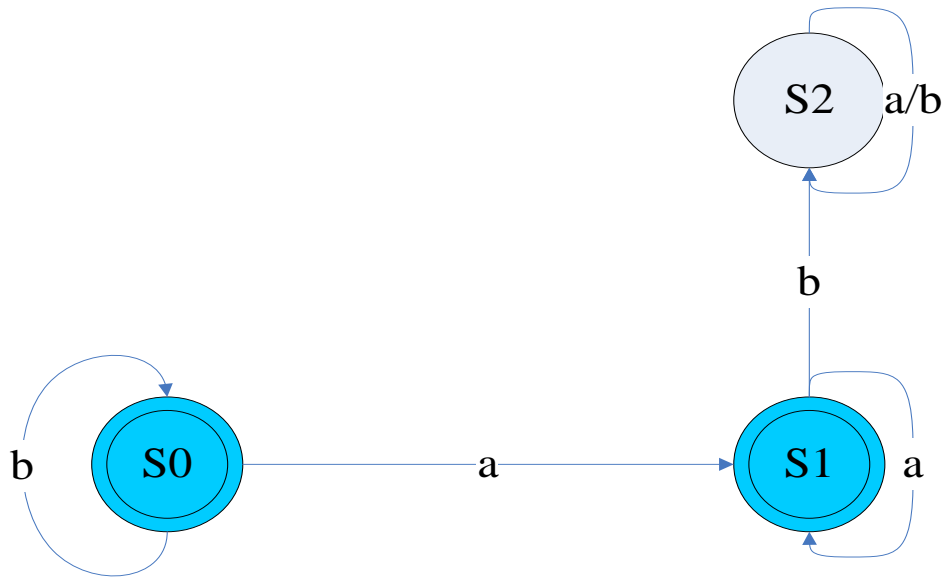
3. Consider  $\Sigma = \{a, b\}$ . Answer the following DFA related questions. When constructing DFA, there is no need to show your construction steps, but you need to informally state how you get the DFAs.

a. Construct a DFA that accepts  $(a|b)^*$  except for  $abb$ .

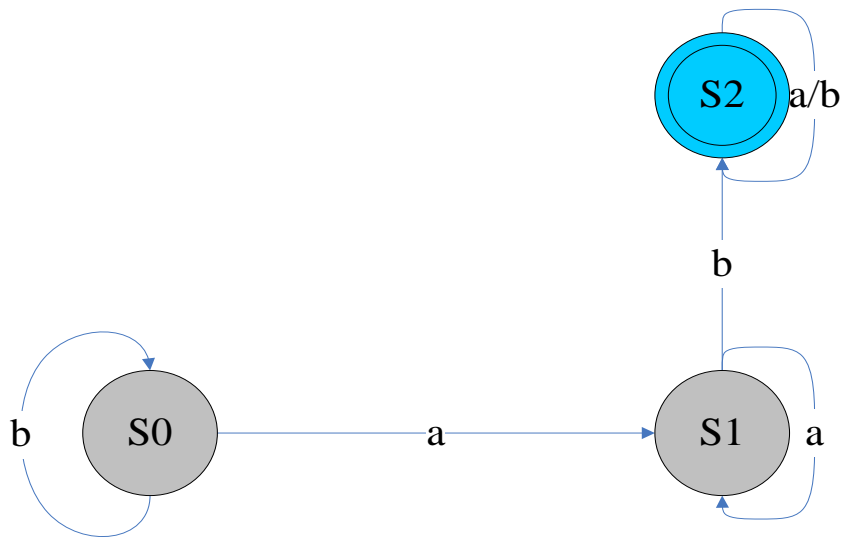


b. Construct a DFA that accepts  $(a|b)^*$  except for  $b^*a^*$ .

DFA for  $b^*a^*$



DFA for  $(a|b)^*$  except for  $(b^*a^*)$



c. Based on the techniques you use in (a) and (b), can you come up with a DFA construction algorithm for the “except for” type of languages? (Just focus on the main idea.)

**1) Draw the DFA of the RE in the exception statement. The DFA should be a complete DFA, i.e., every state should have a transition for each input symbol.**

**2) Each accepting state becomes non-accepting state and each non-accepting state becomes accepting state.**

4. A token recognizer is designed to handle the following tokens, where  $\Sigma = \{a, b, c, d\}$ .

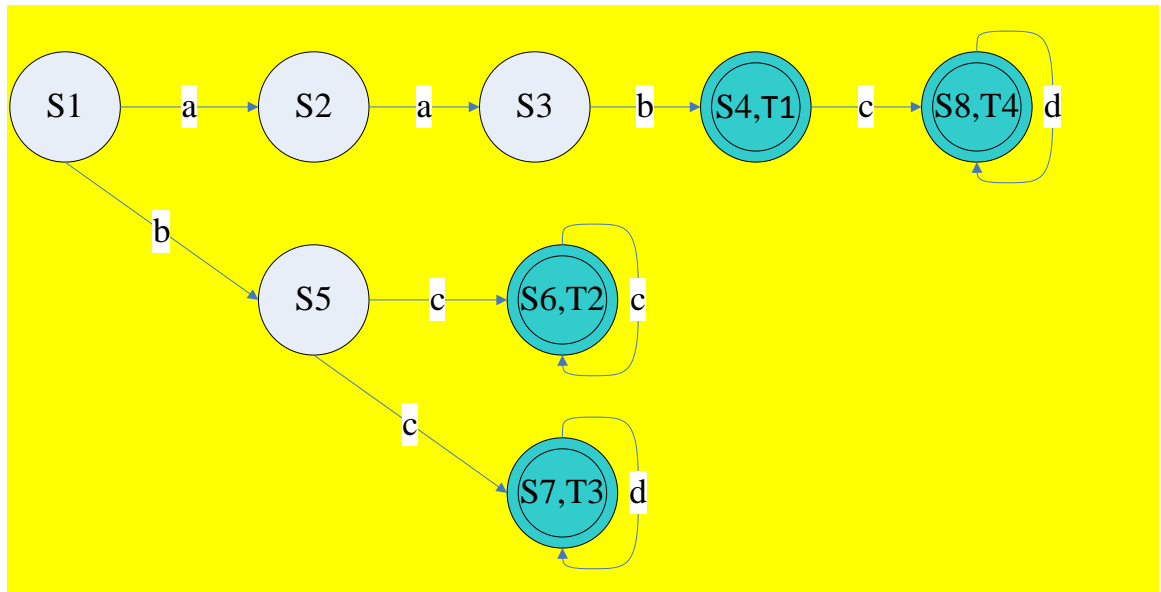
T1 = aab

T2 = bcc\*

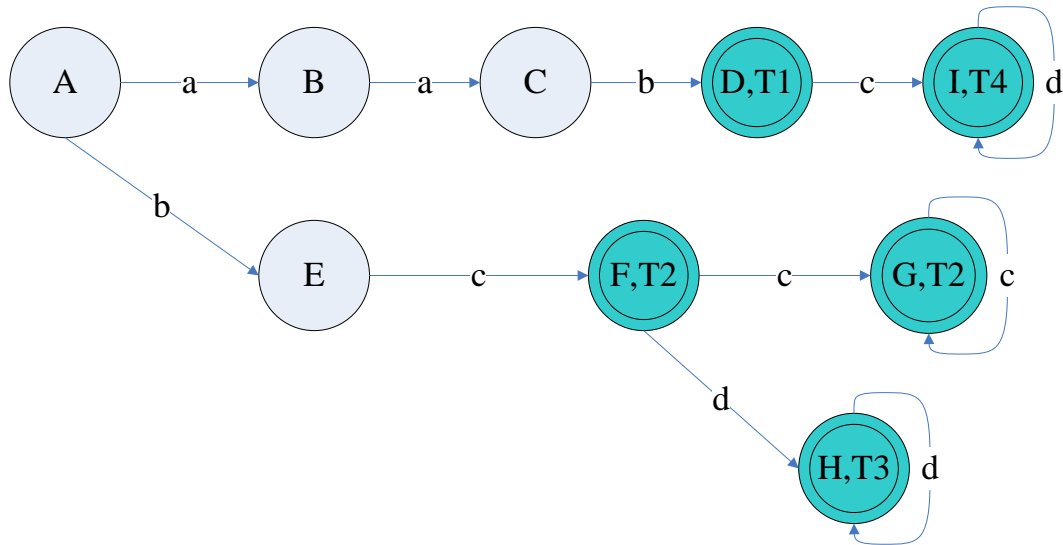
T3 = bcd\*

T4 = aabcd\*

a. Construct a minimized DFA for token recognition and the tokens are defined as follows. The DFA should specify the specific token names (T1, T2, ...) it accepts at the corresponding final states. You do not need to show the steps for the construction if you can draw the DFA directly. Note that the longest matching and first matching rules for ambiguity resolution should be used.



	A	b	c	d	comments
1	2	5			A,1
2	3				B,2
3		4			C,3
4			8		D,4, T1
8				8	I,8,T4
5			6,7		E,5
6,7			6	7	F,6,T2
6			6		G,6,T2
7				7	H,7,T3



b. Execute your DFA to process the following string to identify tokens. List every token string and its token name. Also, describe all the backtracking actions taken during the process.

Aabaabccccdbcbccccaabbccddd

- aab | aabccccdbcbccccaabbccddd -- got token aab(T1)
- aab | aabccddd | bcbccccaabbccddd -- got token aabccddd (T4)
- aab | aabccddd | bc | bccccaabbccddd -- got token bc(T2)
- aab | aabccddd | bc | bcccc | aabbccddd -- got token bcccc(T2)
- aab | aabccddd | bc | bcccc | aab | bcddd -- got token aab(T1)
- aab | aabccddd | bc | bcccc | aab | bcddd -- got token bcddd (T3)

5. Consider the following grammar defined over

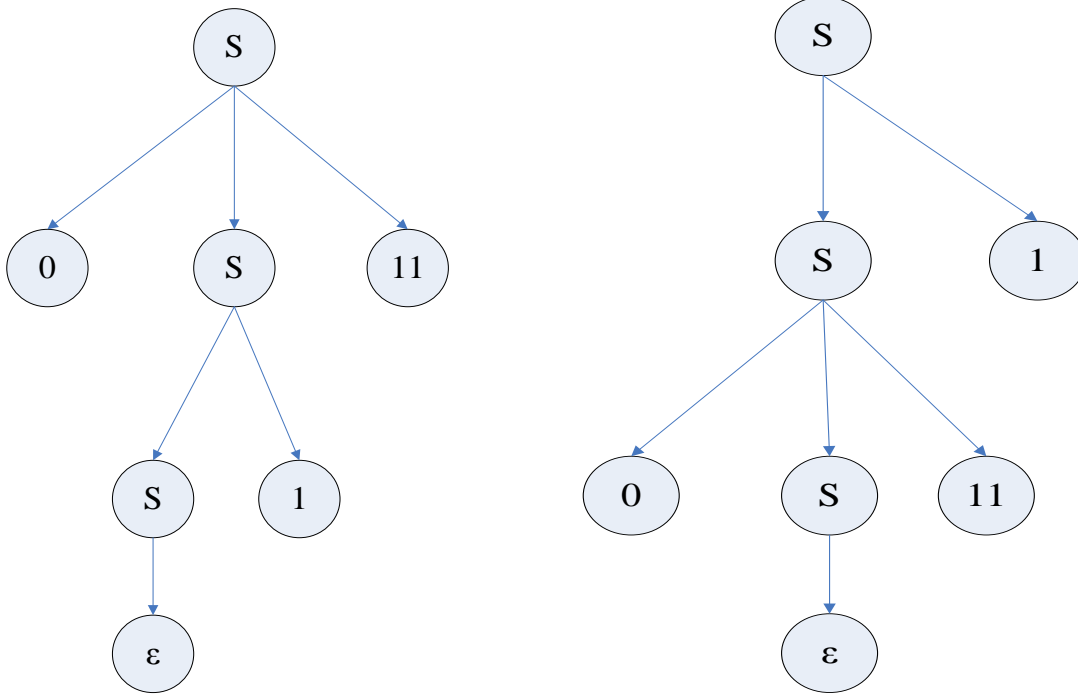
(a) Briefly describe the language generated by this grammar.

$0^n 1^{2n+m}$

(b) Show that this grammar is ambiguous by giving a string that can be parsed in two different ways and showing the two corresponding parse trees.

**Ways 1:  $S \rightarrow 0S11 \rightarrow 0S111 \rightarrow 0111$**

**Ways 2:  $S \rightarrow S1 \rightarrow 0S111 \rightarrow 0111$**



(c) Rewrite the grammar to eliminate the ambiguity.

**$S \rightarrow 0S11 \mid T$**

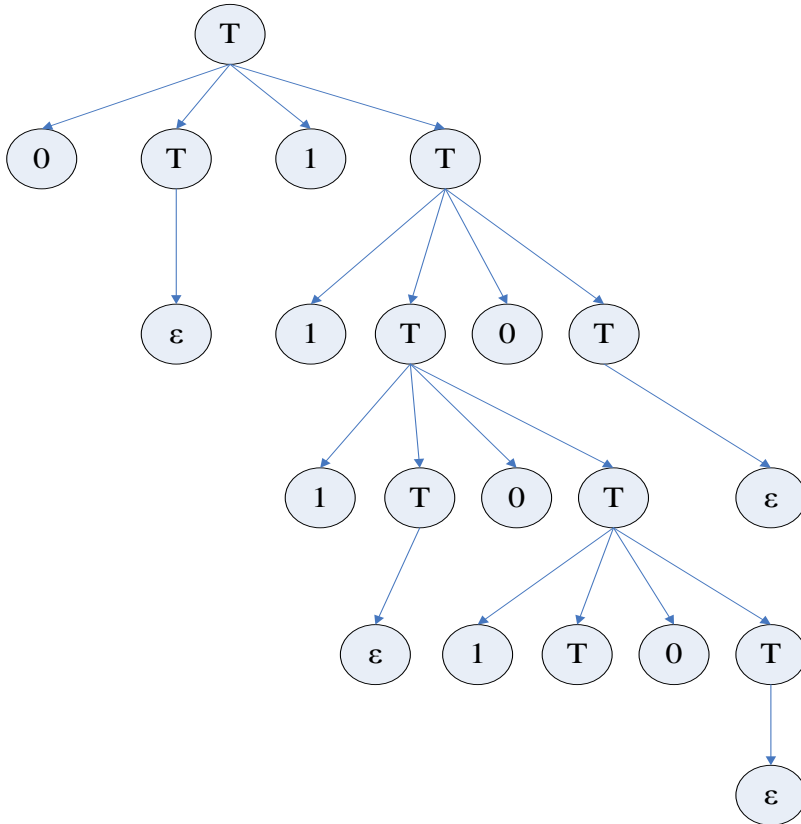
**$T \rightarrow T1 \mid \epsilon$**

6. Let L be a language defined over  $\Sigma = \{0, 1\}$  and L consists of all strings with the same number of 0's and 1's.

a. Give a context free grammar for L.

**$T \rightarrow 0T1T \mid 1T0T \mid \epsilon$**

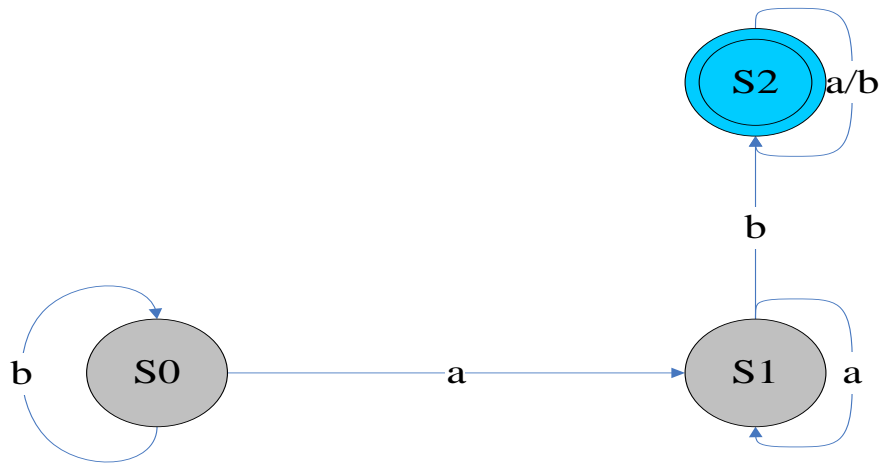
b. Show a parse tree for the string 01110100.



c. Give the leftmost derivation for (b).

<b>0T1T</b>	<b>T → 0T1T</b>
<b>→ 01T</b>	<b>T → ε</b>
<b>→ 011T0T</b>	<b>T → 1T0T</b>
<b>→ 0111T0T0T</b>	<b>T → 1T0T</b>
<b>→ 01110T1T00</b>	<b>T → 0T1T</b>
<b>→ 01110 1 00</b>	<b>T → ε</b>

7. Construct a regular grammar for the language L, where L accepts  $(a|b)^*$  except for  $b^*a^*$  (start from your answer for 3).



**$S \rightarrow bS \mid aT$**

**$T \rightarrow aT \mid bU$**

**$U \rightarrow aU \mid bU \mid \epsilon$**