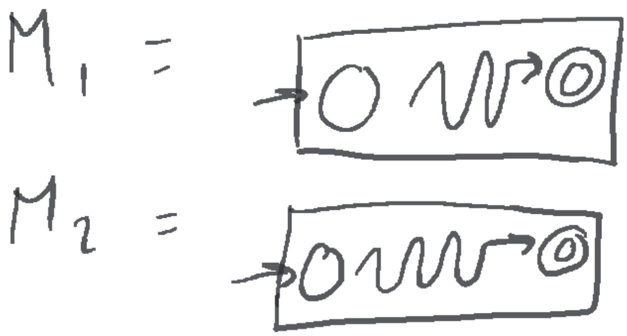
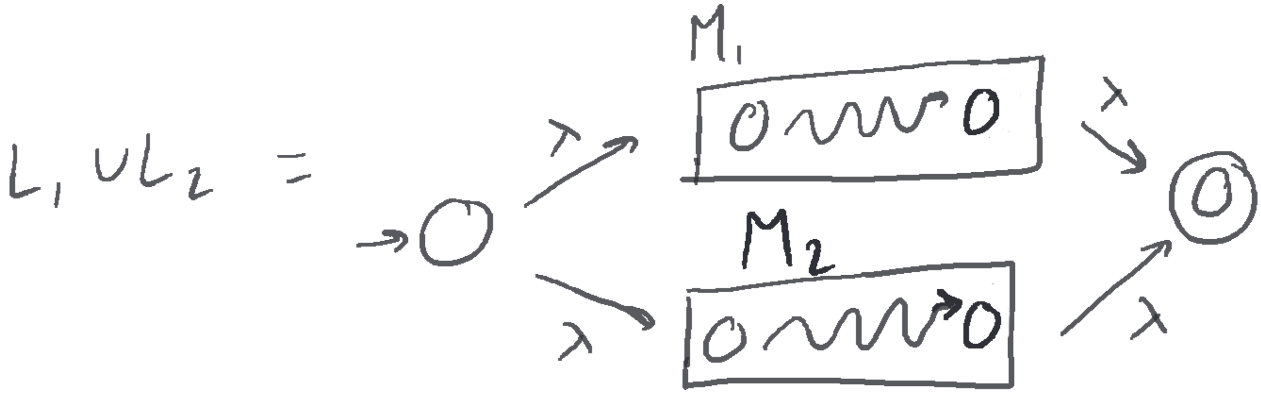
CS 3186 --- Assignment #8

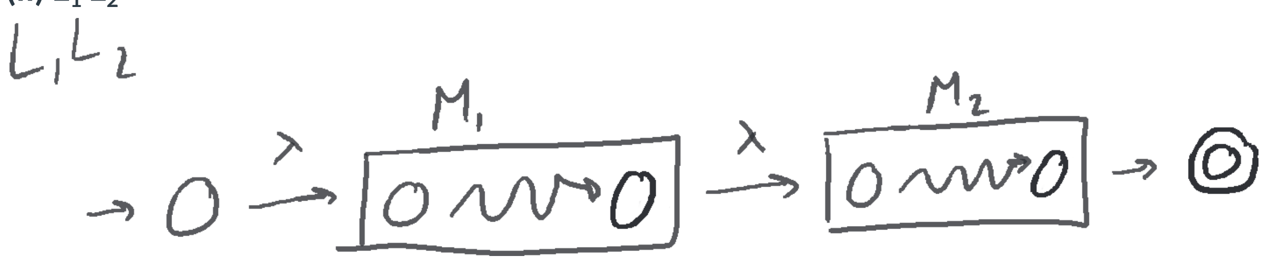
(I) Given the description of L_1 and L_2 as regular in the form of acceptors M_1 and M_2. Show that the following languages are regular by constructing an automaton using generic descriptions of M below:



(i) $L_1 \cup L_2$

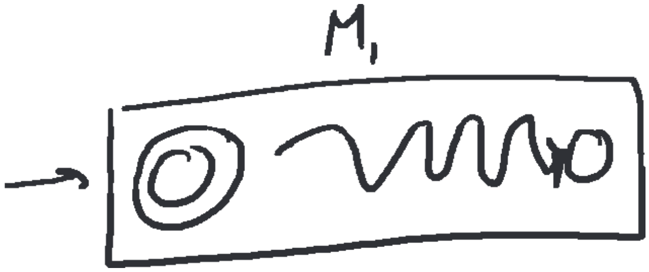


(ii) $L_1 L_2$



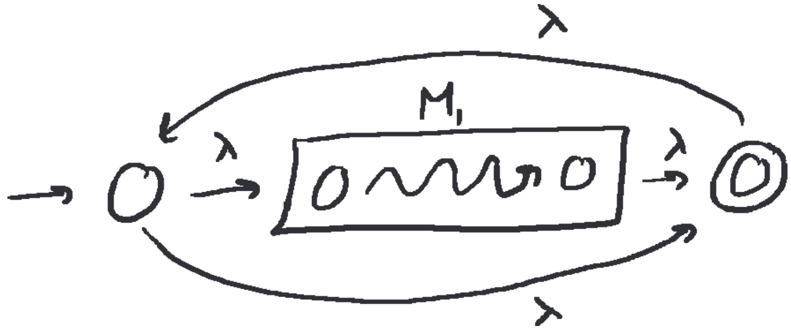
(iii) L_1 complement

$\overline{L_1}$



(iv) L_1^*

L_1^*



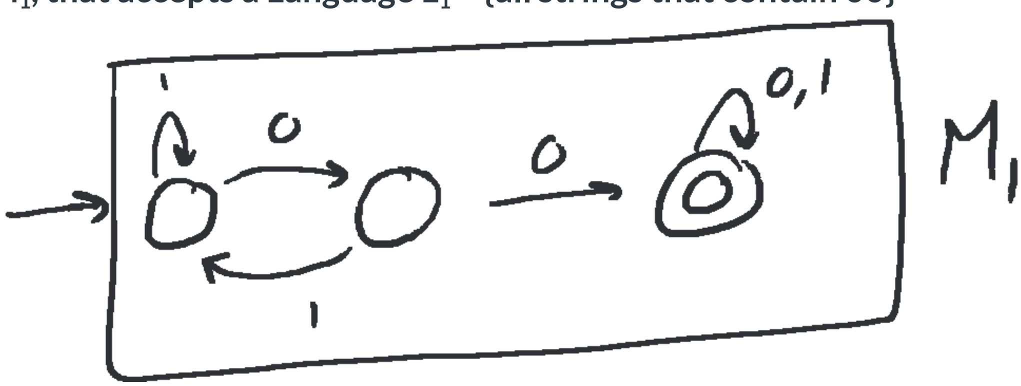
(v) L_1^R

L_1^R

(II) Σ = {0, 1}

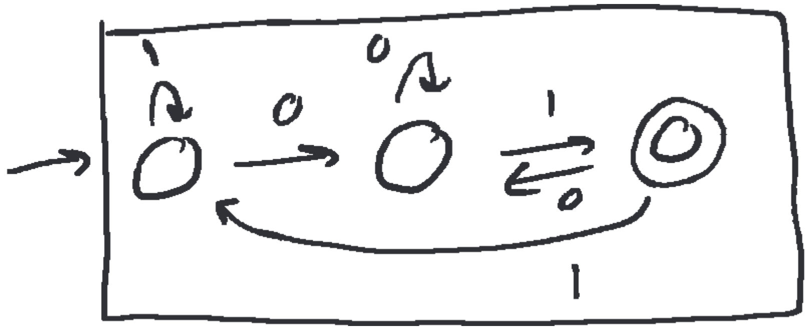
(i) Give a DFA, M_1, that accepts a Language L_1 = {all strings that contain 00}

L_1 =



(ii) Give a DFA, M_2, that accepts a Language L_2 = {all strings that end with 01}

$L_1 =$

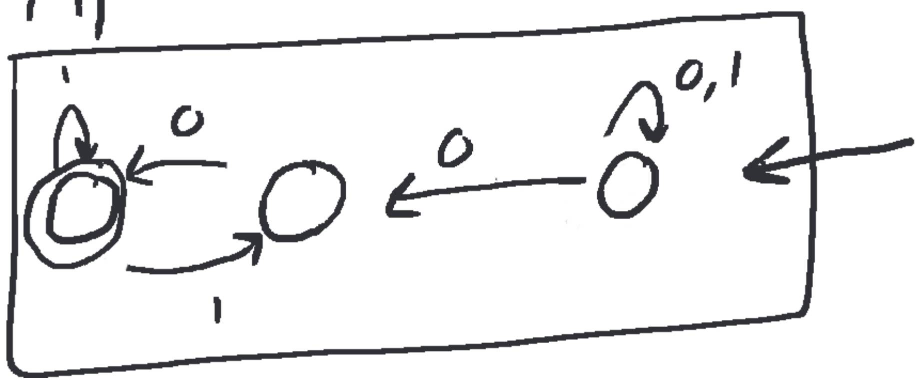


M_2

(iii) Give acceptor for Reverse of L_1

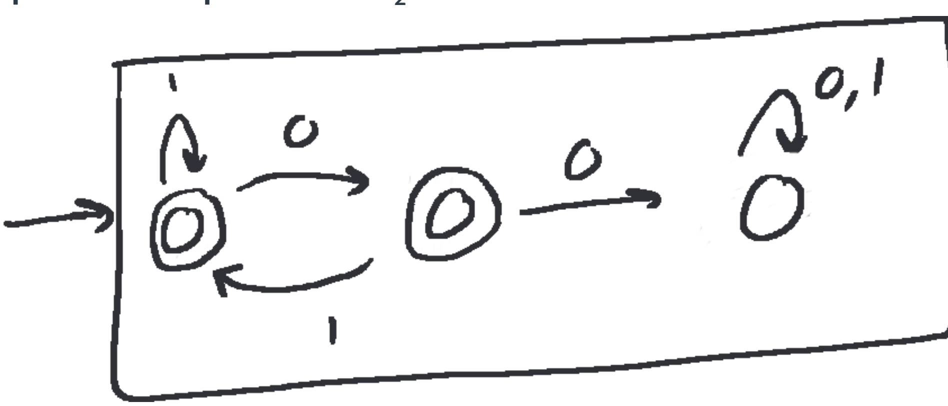
M_1

L_1^R



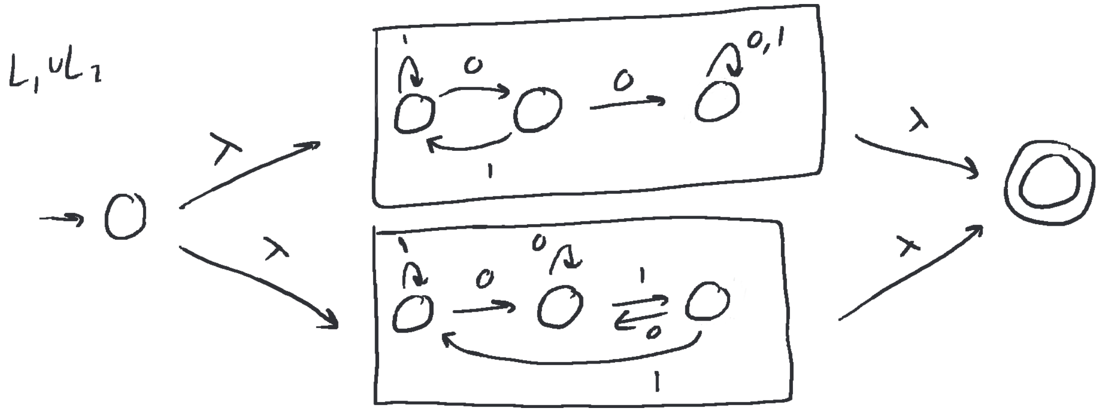
(iv) Give acceptor for complement of L_2

$\overline{L_1} =$

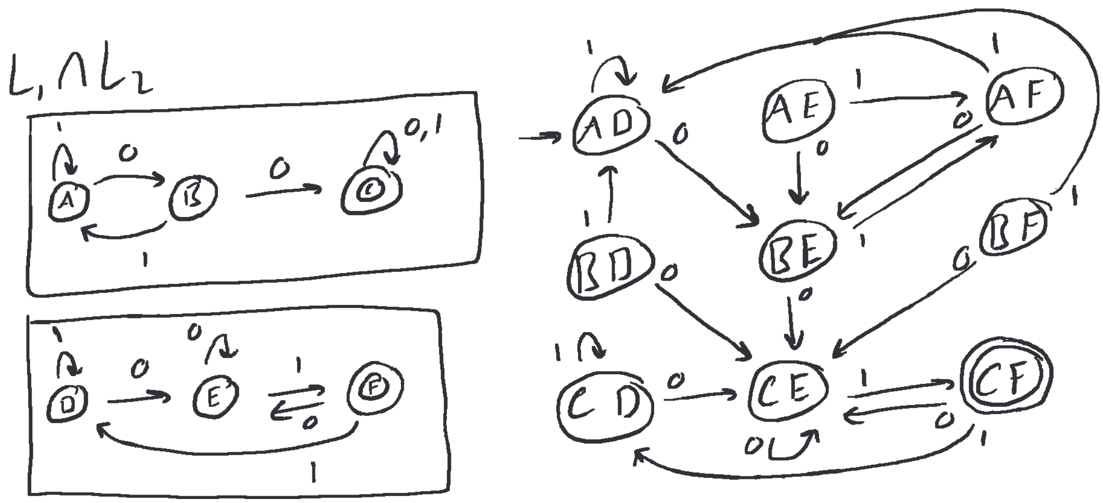


M_1

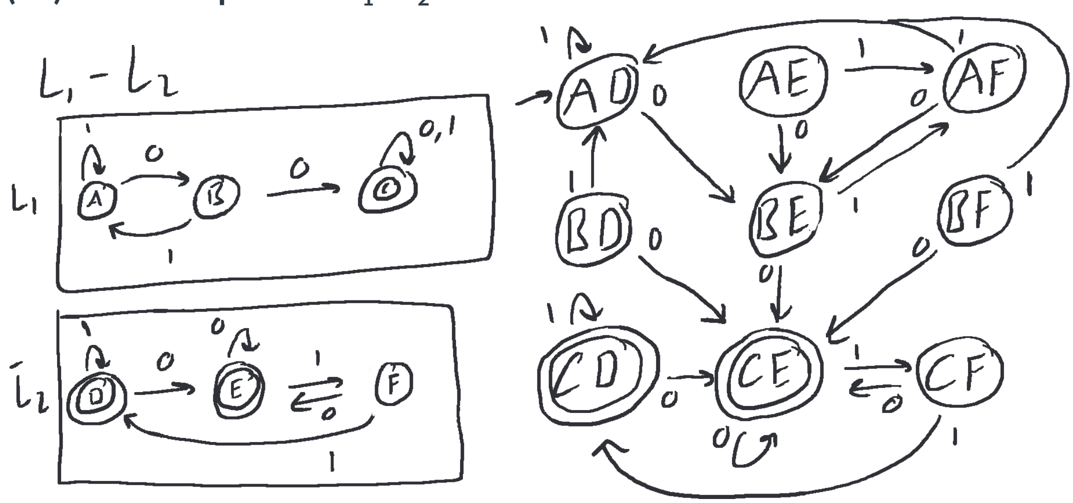
(v) Give acceptor for L_1 union L_2



(vi) Give acceptor for L_1 intersection L_2

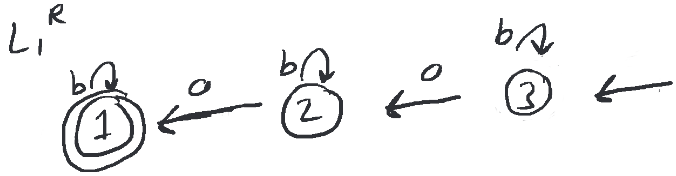


(vii) Give acceptor for L_1 - L_2

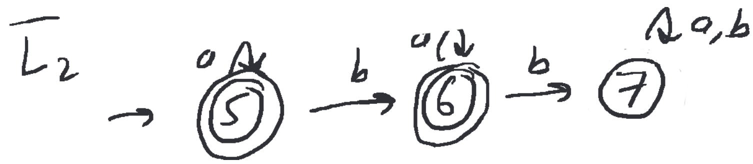


(III) Give the DFAs for the two languages {w| w has exactly two a's} and {w| w has at least two b's}. Redo exercises (iii) through (vii)

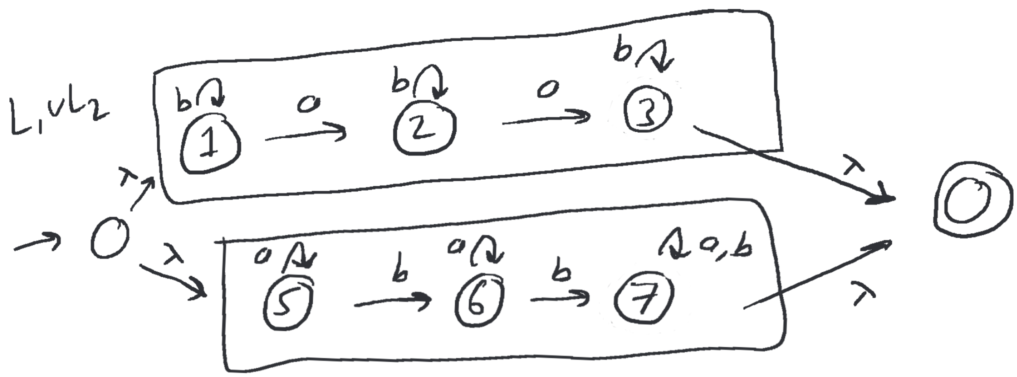
iii)



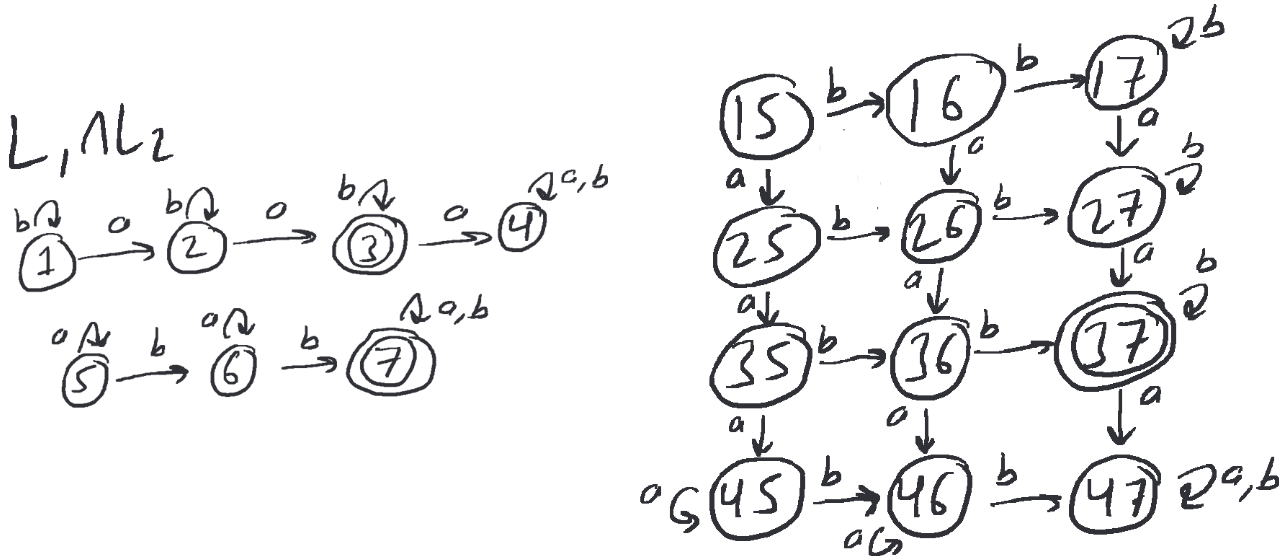
iv)



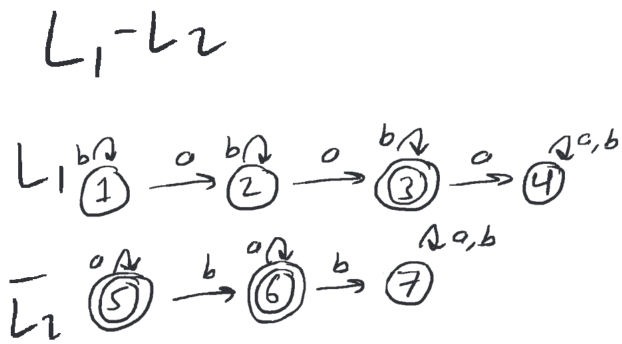
v)

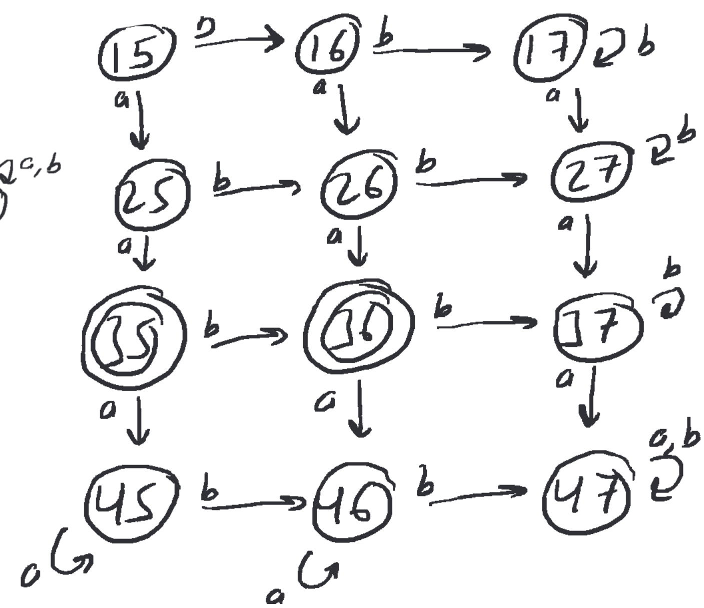


vi)

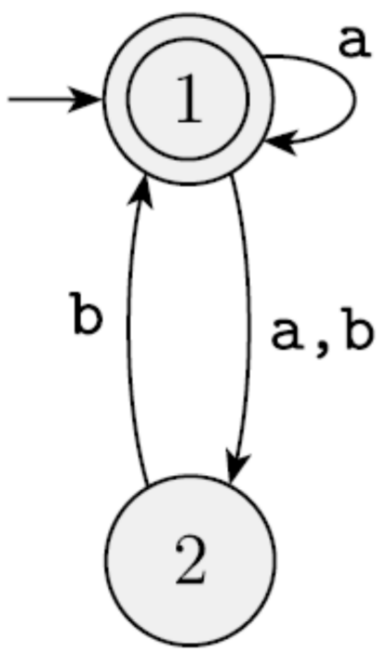


vii)

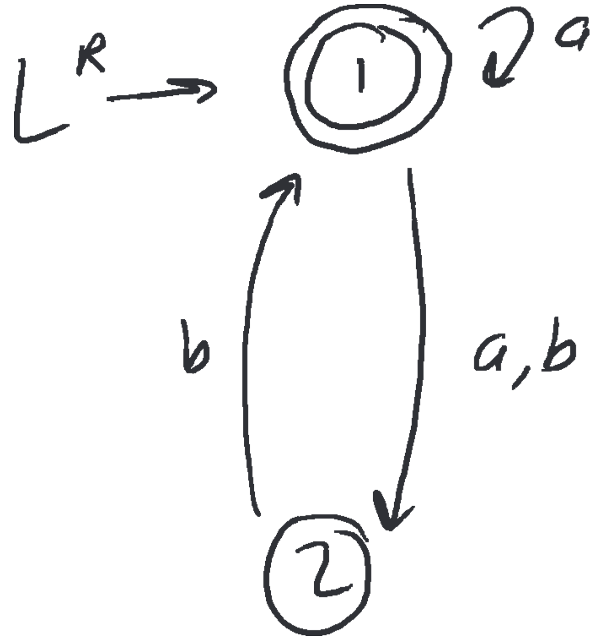




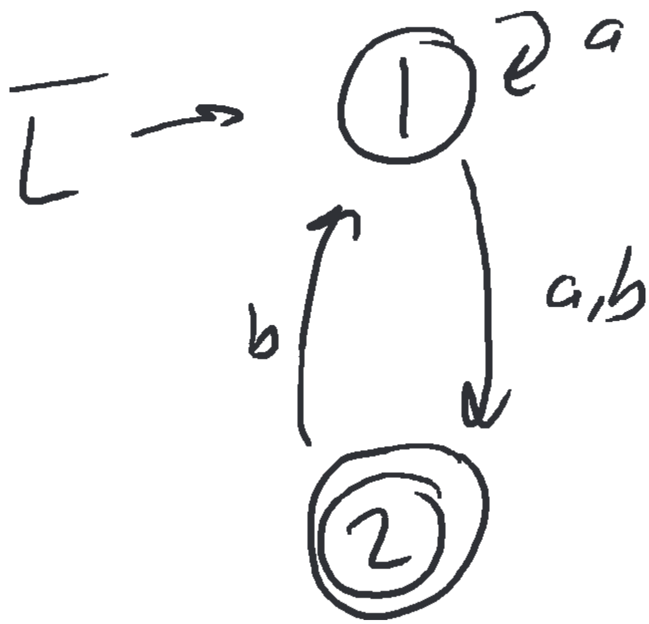
(IV) Given the automaton below for a language L Construct an automaton for



(i) Reverse of L



(ii) Complement of L



CS 3186

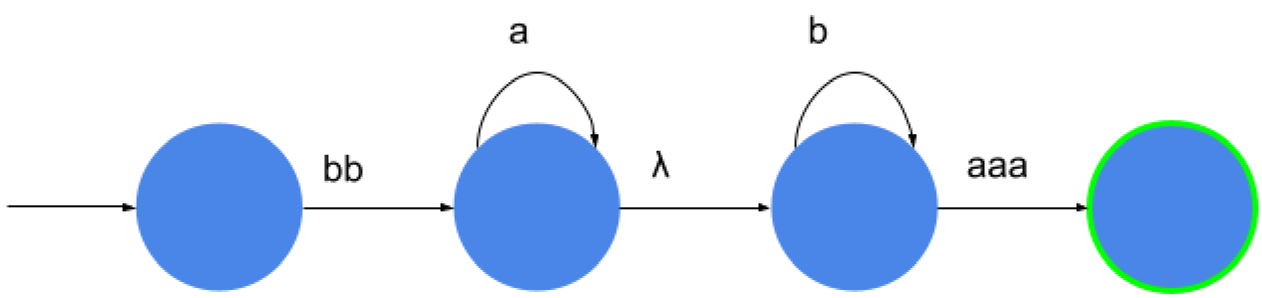
Assignment #9

(I) State the pumping lemma with all its conditions and indicate how you go about proving that a language is not regular.

First you need to take a valid regular Language that has an infinite number of strings. There exists m where m is the number of states. Then you take a string (w) that is greater than or equal to the number of states m. We then write string w as xyz such that |xy| ≤ m and |y| ≥ 1. For all i ≥ 0: xy^iz belongs to L, if string y is "pumped" or inserted any number of times then the string will still remain in L. After that you find a contradiction. For example, find any one string for some value of "i" which is not in L. Lastly we can conclude that our assumption that L is regular is not true. Hence, L is not regular.

(II) If the languages are regular, give an automaton. Otherwise, show it is not regular by using pumping lemma.

L = $\{b^2a^nb^ma^3 \mid m, n \geq 0\}$.
Can give a automaton here



L = $\{b^2a^nb^ma^3 \mid m = n\}$.
1,2. Given Language L, L is an infinite language as it defines an infinite number of strings. There exists a DFA with S amount of states.

3. We choose w = $b^2a^sb^sa^3$ |w|=2s+5 >= s

4. W = $b^2a^sb^sa^3$ = xyz (j+k+l) = s

$x = b^2a^j$, $y = a^k$, $z = a^lb^sa^3$

Where k >= 1

5. For all i ≥ 0; $xy^iz \in L$
i=0; $xz \in L$
i=2; $xyyz \in L$

6. Say i = 2; xyyz = $b^2a^ja^ka^ka^lb^sa^3 = b^2a^{s+k}b^sa^3$
We have s+k which is not equal to s. Meaning this is a contradiction.
7. Our assumption that L is regular is not true. Hence L is not regular.

L = {ww^R| w is any string over {a.b}}
1,2. Given Language L, L is an infinite language as it defines an infinite number of strings. There exists a DFA with M amount of states.

3. Choose w = a^mb and $w^R = ba^m$ |w| = 2m+2 >= m

4. $x = a^j$, $y = a^k$, $z = a^lbba^m$
Where k >= 1 and (J+k+l)=m

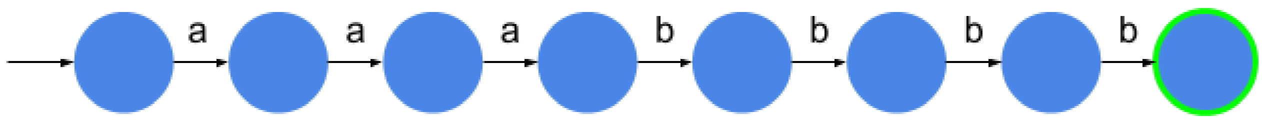
5. For all i ≥ 0; $xy^iz \in L$

i=0; $xz \in L$

i=2; $xyyz \in L$

6. Say i =2; xyyz = $a^ja^ka^ka^lbba^m$ = $a^{m+k}bba^m$
We have m+k which is not equal to m. Meaning this is a contradiction.
7. Our assumption that L is regular is not true. Hence L is not regular.

L = {a^{3}b^4}
Can give automata



L = {a^{n}b^{n+1}}
1,2. Given Language L, L is an infinite language as it defines an infinite number of strings. There exists a DFA with M amount of states.

3. Choose w = $a^m b^{m+1}$ |w| = 2m +1 >= m

4. $x = a^j$, $y = a^k$, $z = a^l b^{m+1}$

Where k >=1 and (j+k+l)=m

5. For all i ≥ 0; $xy^i z \in L$

i=0; $xz \in L$

i=2; $xyyz \in L$

6. Say i =2; xyyz = $a^j a^k a^k a^l b^{m+1} = a^{m+k} b^{m+1}$

We have m+k which is not equal to m. Meaning this is a contradiction.

7. Our assumption that L is regular is not true. Hence L is not regular.

L = {$a^m b^n a^{m+n}$ | m,n ≥ 0 }

1,2. Given Language L, L is an infinite language as it defines an infinite number of strings. There exists a DFA with S amount of states.

3. Choose w = $a^s b^s a^{s+s}$ |w| = 4s >= s

4. $x = a^j$, $y = a^k$, $z = a^l b\ a^{s+s}$

Where k >=1 and (j+k+l)=s

5. For all i ≥ 0; $xy^i z \in L$

i=0; $xz \in L$

i=2; $xyyz \in L$

6. Say i =2; xyyz = $a^j a^k a^k a^l b\ a^{s+s} = a^{s+k} b a^{s+s}$

We have s+k which is not equal to s. Meaning this is a contradiction.

7. Our assumption that L is regular is not true. Hence L is not regular.

CS 3186 --- Assignment #10

(I) Define/Describe a context free grammar G and the language L(G)

Context free grammar G = G(V,T,S,P)
V = Variables, S = Starting variable, T = Terminals, P = Production rules
L(G) = {w: S =>* w, w ∈ T*}

(II) Define/Describe a sentential form in a derivation.

The sentential form of a sentence follows:
$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n \Rightarrow w$
Where w_x are sentential forms of the derivation of w. Apply each derivation by replacing a variable with another variable or terminal based on the production rules.

(III) Differentiate between a leftmost and a rightmost derivation sequence.

In a leftmost derivation sequence, the variables are replaced one at a time from left to right.
In a rightmost derivation sequence, the variables are replaced one at a time from right to left.

(IV) Define an ambiguous grammar.

An ambiguous grammar is when two or more derivatives can end up with the same string.

(V) Sometimes only the production rules of a grammar are defined with the starting nonterminal given by the first rule.
R-> XRX | S
S-> aTb | bTa
T-> XTX | X | λ
X->a | b

a. What are the variables of G?
R, S, T, X

b. What are the terminals of G?
a, b, λ

c. Which is the start variable of G?
R

d. Give 3 strings of varying lengths in L(G).
ab, aaba, bbabaa, babaabb

e. Give 3 strings *not* in L(G).
aa, bbb, λ, aaaaa…...

f. True or False: T => aba
False, T => XTX | X | λ, none directly give aba

g. True or False: T =>* aba.
True, T => XTX => XXX => aba

h. True or False: T => T
False, T => XTX | X | λ, T can't lead back to T

i. True or False: T =>* T.
False, T => XTX | X | λ ,T can't lead back to a single T

j. True or False: XXX =>* aba.
True, X => a | b so XXX => aba

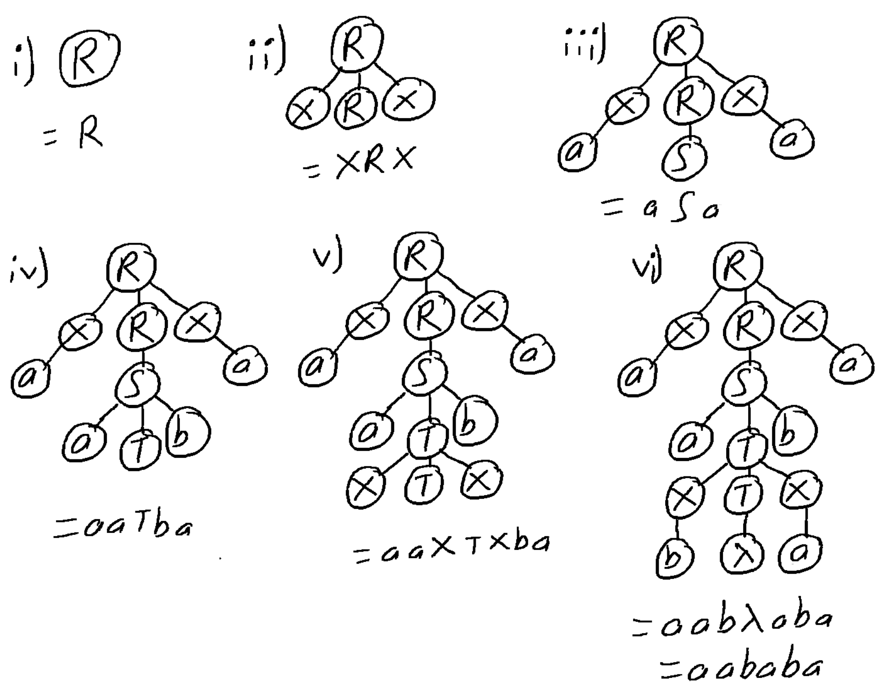
k. True or False: X =>* aba.
False, X => a | b there are only terminals X can lead to

l. True or False: T =>* XX.
True, T => XTX | X | λ, so T => XTX => X λX = XX

m. True or False: T =>* XXX
True, T => XTX | X | λ, so T => XTX => XXX

n. True or False: S =>* λ
False, S => aTb | bTa which have terminals, so S can't end up with only λ

(v) Using the rule of the above grammar, using leftmost derivation (or using a rightmost derivation) show step by step the partial derivation trees, yield for each of the sentential forms in deriving aababa (as described in the notes)



(VI) Show G is ambiguous, give two leftmost, two rightmost & two derivation trees

G = ({S,A,B,,D}, {a,b,c},S,P}
Where P, the production rules are:

$$S \rightarrow BC \mid AD$$
$$B \rightarrow aBb \mid \lambda$$
$$C \rightarrow cC \mid \lambda$$
$$A \rightarrow aA \mid \lambda$$
$$D \rightarrow bDc \mid \lambda$$

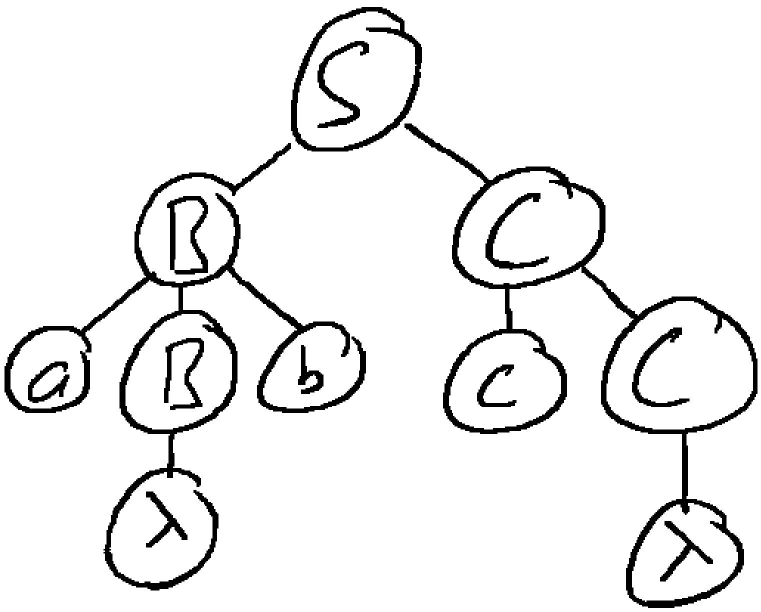
Leftmost:

S => BC => aBbC => aλbC => aλbcC => aλbcλ = abc
S => AD => aAD => aλD => aλbDc => aλbλc = abc

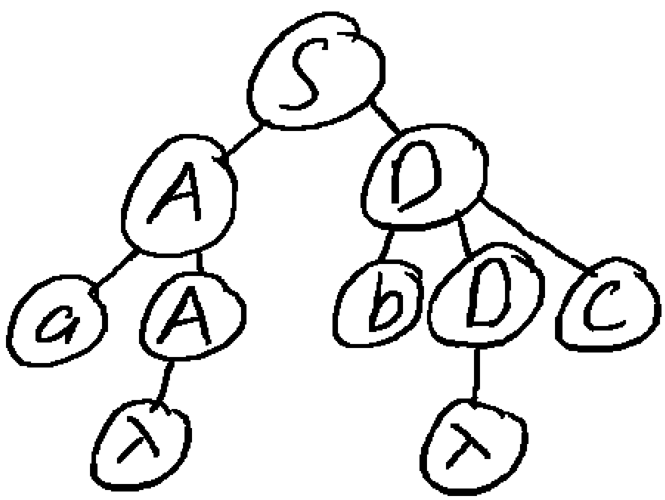
Rightmost:

S => BC => BcC => Bcλ => aBbcλ => aλbcλ = abc
S => AD => AbDc => Abλc => aAbλc => aλbλc = abc

Derivation Trees



= a b c



= a b c

CS 3186 --- Assignment #11

(1) Define a PDA as a 7-tuple and describe each of the components.

PDA = ($Q, \Sigma, \Gamma, \delta, q_0, z, F$)
Q = set of states, Σ = input alphabet, Γ = stack alphabet, δ = transition function, q_0 = initial state, z = stack start symbol, F = set of final states.

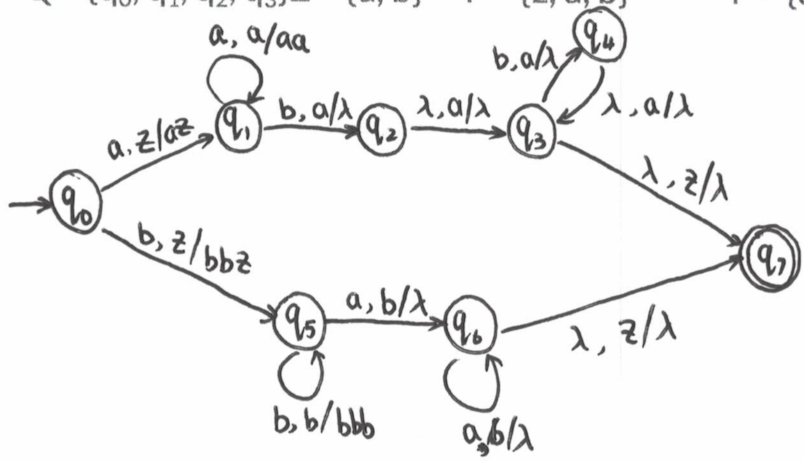
(2) Create a PDA that recognizes the following context free language with terminals {a,b}
L = {w | number of a's = twice the number of b's; String w can only have a's followed by b's or b's followed by a's}
i.e., it should accept aab, aaaabb, baa, bbaaaa,.. and so on.
(i) Describe your algorithm

If the first letter is a, then:
  when a, if top is z or a then push a
  when b, if top is a then pop two a's
If the first letter is b, then:
  when b. if top is z or a then push two b's
  when a, if top is b then pop b

(ii) Give the description as a complete 7-tuple with a transition diagram

NPDA$(Q, \Sigma, \Gamma, \delta, q_0, z, F)$

$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{a, b\}$ $\Gamma = \{z, a, b\}$ $F = \{q_3\}$



PDA $\{Q, \Sigma, \Gamma, \delta, q_0, z, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, $\Sigma = \{a, b\}$, $\Gamma = \{z, a, b\}$, $F = \{q_7\}$

(iii) Show configuration sequences on aabbbb leading to rejection. (Nejectionote that this is an easier problem than simply saying that L = {w |number of a's = twice the number of b's} Then we need to account for strings like aba, abbaaa,.. which complicates the logic.

$\delta(q_0, aabbbb, z) \vdash \delta(q_1, abbbb, az) \vdash \delta(q_1, bbbb, aaz)$

$\vdash \delta(q_2, bbb, az) \vdash \delta(q_3, bbb, z) \vdash \delta(q_7, bbb, z)$

no action defined. reject.

(3) Create a PDA that recognizes the following context free language with terminals {a,b,c}

L = {wck | w ∈ {a, b}* and k = |w|}

(Hint: It is only asking for the # of c's = total number of a's + b's)

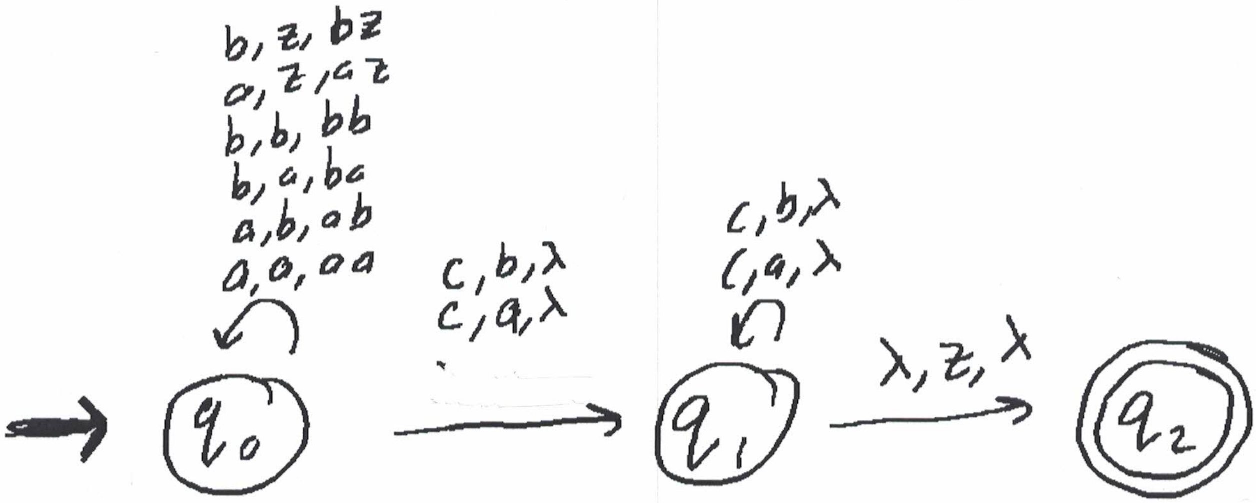
(i) Describe your algorithm

When a,	if top is z,	then push a
	if top is a,	then push a
	if top is b,	then push a
When b,	if top is z,	then push b
	if top is a,	then push b
	if top is b,	then push b
When c,	if top is a,	then pop a
	if top is b,	then pop b

(ii) Give the description as a complete 7-tuple with a transition diagram

NPDA(Q, Σ, Γ, δ, q_0, z, F)

$Q = \{q_0, q_1, q_2\}$ $\Sigma = \{a, b, c\}$ $\Gamma = \{z, a, b, c\}$ $F = \{q_2\}$



(iii) Show configuration sequences on babbcccc leading to acceptance.

$\delta(q_0, babbcccc, z)$

$\delta(q_0, abbcccc, bz)$

$\delta(q_0, bbcccc, abz)$

$\delta(q_0, bcccc, babz)$

$\delta(q_0, cccc, bbabz)$

$\delta(q_1, ccc, babz)$

$\delta(q_1, cc, abz)$

$\delta(q_1, c, bz)$

$\delta(q_1, \lambda, z)$

$\delta(q_2, \lambda, \lambda)$

(4) Example 7.5 is considered in the notes. Give all possible configuration sequences to account for all the choices on string babbba (similar to slide #45)