Q1 Summer 2022

Construct a regular expression over $\{a,b,c\}$ for the language accepted by this nfa:

	a	b	c	
→A	B	/	/	0
B	C	/	A	1
C	/	A,B	/	0

Step 1) $L_A = aL_B$

$L_B = aL_C \cup cL_A \cup \epsilon$

$L_C = b(L_A \cup L_B) = bL_A \cup bL_B$

$L_A \rightarrow L_C$ substitute in L_A to $L_C \rightarrow \ldots$

$L_C = b(aL_B) \cup bL_B = baL_B \cup bL_B$

$L_C = (ba \cup b)L_B$

$L_C \rightarrow L_B$ sub in L_C to $L_B \rightarrow \ldots$

$L_B = a[(ba \cup b)L_B] \cup cL_A \cup \epsilon$

$L_B = (aba \cup ab)L_B \cup cL_A \cup \epsilon$

$L_A \rightarrow L_B$ sub in L_A to L_B

$L_B = (aba \cup ab)L_B \cup caL_B \cup \epsilon$

$L_B = (aba \cup ab \cup ca)L_B \cup \epsilon$

Kleene's theorem = $X = LXUM = L^*M$

$L_B = (aba \cup ab \cup ca)^* \epsilon$

$L_B \rightarrow L_A$

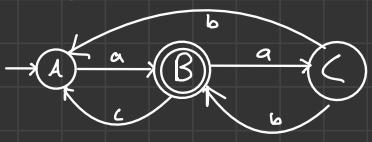
$L_A = a[(aba \cup ab \cup ca)^* \epsilon]$

$X = a(aba \cup ab \cup (a)^*$

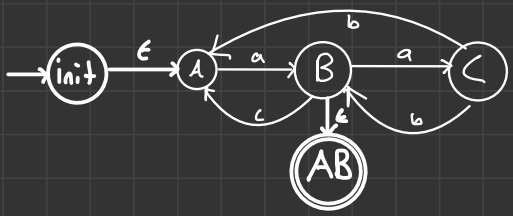
Method to double check Previous Page

	a	b	c	
→A	B	/	/	0
B	C	/	A	1
C	/	A,B	/	0

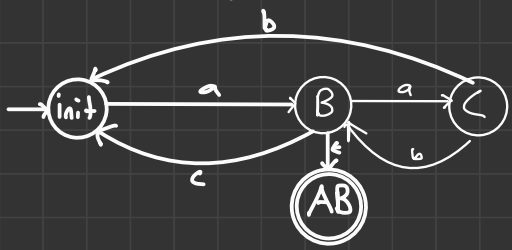
1. original



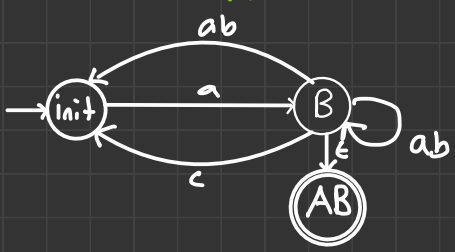
2. normalize



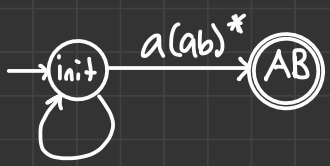
3. Remove state A



4. Remove state C

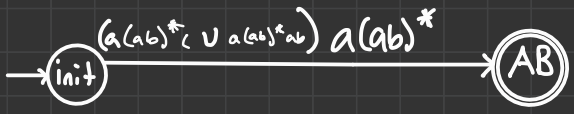


5. Remove state B



$a(ab)^*\epsilon \cup a(ab)^*ab$

6. Remove Cycle

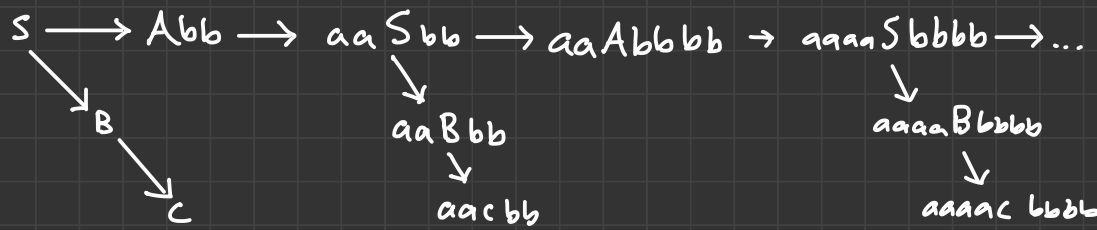


$$X = (a(ab)^*\epsilon \cup a(ab)^*ab)\,a(ab)^*$$
$$X = a(ab)^*(\epsilon \cup ab)\,a(ab)^*$$
$$X = a(ab)^*(\epsilon a \cup aba(ab)^*\cup ab)$$
$$X = a(aba \cup ab \cup \epsilon a)^*$$

Q2 Summer 2022

Prove that a language $L(G)$ is not regular where G is the following cfg. $G = (\{S, A, B\}, \{a, b, c\}, \{S \to Abb \mid B, A \to aaS, B \to c\}, S)$

Note: You must first determine $L(G)$



Therefor $L(G) = \{a^{2n} c\, b^{2n} \mid n \geq 0\}$

To prove L is a regular language, assume L is regular $\exists$ dfa $\mathcal{C}$ accepting L. N is the number of states in $\mathcal{C}$

Consider the case $x = b^{2c} d\, e^{2c}$ with $|x| = N$

$x = w \cdot v$ $\quad w = b^{2N}$ $\quad v = de^{2N}$ $\quad |w| = N$

Pumping Lemma

$w = w_1 \cdot w_2 \cdot w_3$ such that $|w_2| > 0$ and $T(q_0, w)$ equals wherever $w \cdot v \in L$ $\quad w_1 (w_2)^s w_3 \in L$ $\forall s \geq 0$ $\quad T(q_0, w_1 (w_2)^s w_3)$ $\forall s \geq 0$

If $s = 0$ $\quad w_1 (w_2)^0 w_3 = w_1 w_3$ $\quad |w_1 w_3| = N - |w_2| < N, |w_2| > 0$

$\Rightarrow$ $b^{2N - |w_2|} d\, e^{2N} \in L$ and

$$\begin{aligned} T(q_0, v) &= T(q_0, wv) \\ &= T((q_0, w), v) \\ &= T((q_0, w_1 w_3), v) \\ &\in L(\mathcal{C}) \end{aligned}$$

Which is accepting the automata.

but $b^{2N - |w_2|} d\, e^{2N} \in L$

This is a contradiction, therefor $L(G)$ is NOT regular

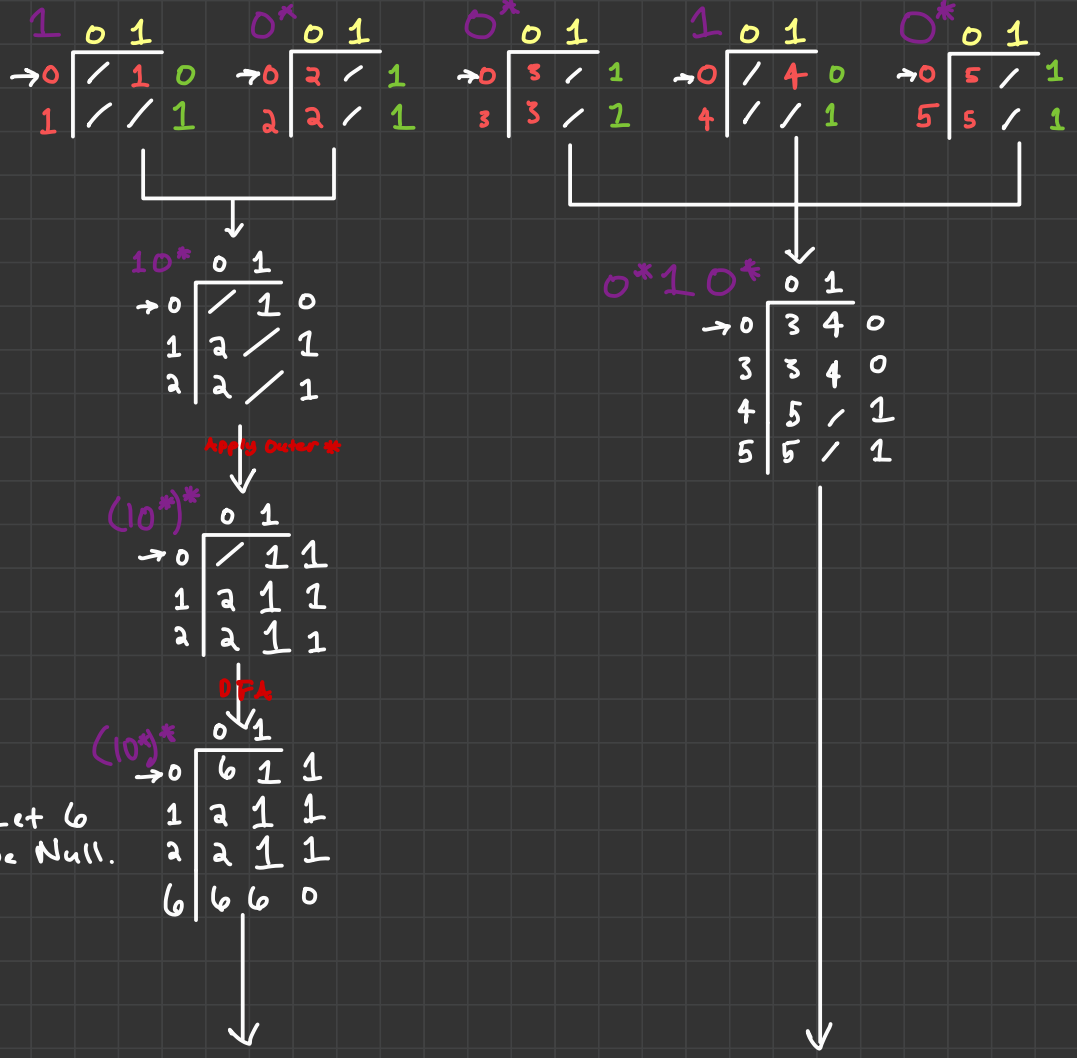
Q3 Summer 2022

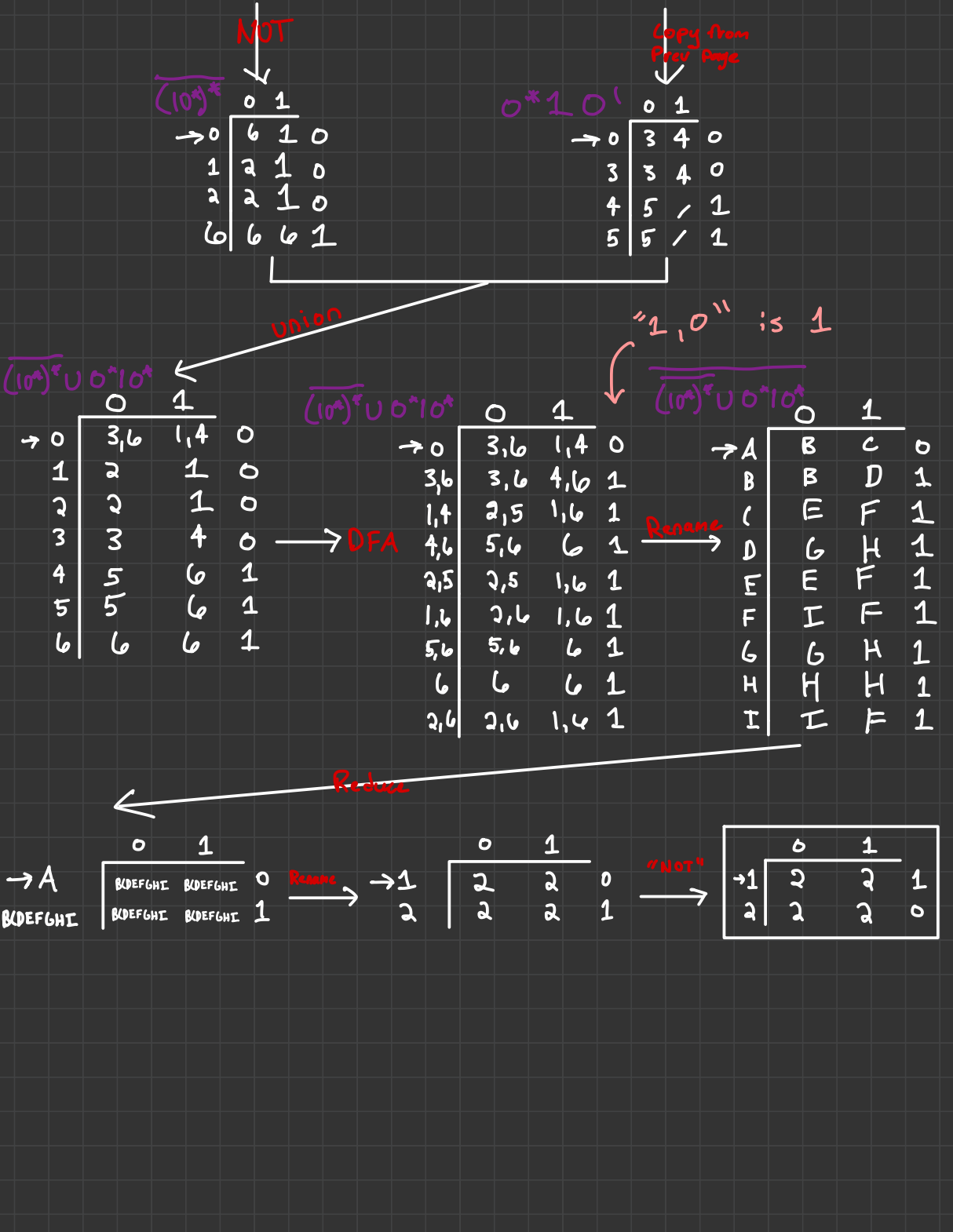
Construct a reduced dfa for the following regular expression over {0, 1} $[(10^*)^* \cap \overline{0^*10^*}]$ Note: You must first determine the nfas for $(10^*)^*$ and 0^*10^*, then do the intersection

If there is an intersection, convert it to union by de Morgan's law

$$\overline{[\overline{(10^*)^*} \cup \overline{0^*10^*}]}$$

1 = accepted 0 = not accepted




NOT
$\overline{(10^*)^*}$
Copy from Prev Page
0^*10^*
union
$\overline{(10^*)^*} \cup 0^*10^*$
"1,0" is 1
DFA
Rename
Reduce
"NOT"
A
BCDEFGHI

Q4 Summer 2022

Construct a Chomsky normal form grammar for L(G) for the following cfg G:

$G = (\{S, B\}, \{a, b, c, d\}, \{S \to bSb \mid Ba,\ B \to BdSc \mid S \mid \epsilon\}, S)$

Note: You must first remove all ϵ- productions and unit productions.

$S \to bSb \mid Ba \qquad B \to BdSc \mid S \mid \epsilon$

Remove ϵ-productions $\qquad X_1 \to \ldots \to \epsilon$

- To remove $B \to \epsilon$, look for all productions whose right side contain B
- Replace each occurence of 'B' in each of these productions with ϵ
- Add the resultant productions to the grammar

Remove $B \to \epsilon$. We find $S \to Ba$ and $B \to Bd S\epsilon$

$S \to bSb \mid Ba \mid a$
$B \to BdSc \mid dSc \mid S$

Remove unit productions

- Unit productions are when a symbol leads to an alone uppercase

Productions: $S \to bSb \mid Ba \mid a \qquad B \to BdSc \mid dSc \mid S$

$B \to S$ is unit production

Remove $B \to S$ replace every instance of B with

$S \to bSb \mid Ba \mid a$
$B \to BdSc \mid dSc \mid bSb \mid Ba \mid a$

Replace all RHS lowercase that are not alone with $X_{lowercase}$

Given $S \to bSb \mid Ba \mid a \qquad B \to BdSc \mid dSc \mid bSb \mid Ba \mid a$

Let $X_a \to a \quad X_b \to b \quad X_c \to c \quad X_d \to d$

$S \to X_bSX_b \mid BX_a \mid a \qquad B \to BX_dSX_c \mid X_dSX_c \mid X_bSX_b \mid BX_a \mid a$

Replace each production $A \to B_1 \dots B_n$ where $n > 2$ with $A \to B_1C$ where $C \to B_2 \dots B_n$, Repeat this step for all productions having two or more RHS symbols.

Given $S \to X_bSX_b \mid BX_a \mid a \qquad B \to BX_dSX_c \mid X_dSX_c \mid X_bSX_b \mid BX_a \mid a$

$S \to X_bH_0 \mid BX_a \mid a \qquad B \to BH_1 \mid X_dH_3 \mid X_bH_4 \mid BX_a \mid a$

$H_0 \to SX_b \qquad H_1 \to X_dH_2 \qquad H_2 \to SX_c \qquad H_3 \to SX_c \qquad H_4 \to SX_b$

$S \to X_bH_0 \mid BX_a \mid a$
$B \to BH_1 \mid X_dH_3 \mid X_bH_4 \mid BX_a \mid a$

$H_0 \to SX_b$
$H_1 \to X_dH_2$
$H_2 \to SX_c$
$H_3 \to SX_c$
$H_4 \to SX_b$

$X_a \to a$
$X_b \to b$
$X_c \to c$
$X_d \to d$

Website:

$S \to H_1H_0 \mid BH_2 \mid a$
$B \to H_4H_3 \mid H_5H_3 \mid H_2H_0 \mid BH_2 \mid a$

Q5 Summer 2022

based on 2018 solution

Not finished

Construct a Greibach normal form grammar for L(G) for the following cfg G:

$G = (\{S, B\}, \{a, b\}, \{S \to AS|A, A \to SA|ab\}, S)$

Note: You must first remove all unit productions. You must derive all the productions for S and A. Indicate how the results look for S' and A'

$S \to AS|A \quad A \to SA|ab$

- remove ϵ productions
- remove unit productions

Eliminate $S \to A$ consider the possibility that every instance of S could also be an A

$S \to AS|AA \quad A \to SA|ab|AA$

Replace S with A

- convert to chomsky normal form

Replace all RHS lowercase that are not alone with $X_{lowercase}$

Let $X_a \to a \quad X_b \to b$

$S \to AS|AA \quad A \to SA|X_aX_b|AA$

Replace each production $A \to B_1 \dots B_n$ where $n > 2$ with $A \to B_1C$ where $C \to B_2 \dots B_n$, Rep at this step for all productions having two or more RHS symbols. None exist here.

We are now in chomsky normal form

$S \to AS \mid AA \quad A \to SA \mid X_a X_b \mid AA \quad X_a \to a \quad X_b \to b$

Change the names of the non terminal symbols into some A_i in order of ascending i

$S = 1 \quad A = 2 \quad X_a = 3 \quad X_b = 4$

Given $S \to AS \mid AA$

$A_i \to X_j \ldots \mid X_j \ldots \mid$

$i = 1 \quad 1 < 2 \quad 1 < 2$

$i = 2 \quad 2 \not< 1$

Substitute S into A where S causes LR

$j = 1 \quad A \to ASA \mid AAA \mid X_a X_b \mid AA$

$1 = 1 \qquad\qquad \alpha_1 \quad \alpha_2 \quad \beta_1 \quad \alpha_3$

$B \to X_a X_b \mid X_a X_b B'$

$B' \to SA \mid SAB' \mid AA \mid AAB' \mid A \mid AB'$

$X \to \beta_n \mid \beta_n X'$

$X' \to \alpha_n \mid \alpha_n X'$

$C \to X_a X_b \mid X_a X_b C'$

$C' \to SA \mid SAC' \mid AA \mid AAC' \mid A \mid AC'$

$D \to X_a X_b \mid X_a X_b D'$

$D' \to SA \mid SAD' \mid AA \mid AAD' \mid A \mid AD'$

Not Done ↑

Q6 Summer 2022

Prove that the following language is not context free.
$L = \{a^{n+2} b^{n+1} b^n \mid n \geq 0\}$

Assume L is a context free language then $\exists\ G(N,T,P,S)$ in CNF s.t. $L = L(G)$

Let t = no. of variables.

Assume a word $z = a^{2t+2} b^{2t+1} b^{2t}$ $\quad |z| > 2^t$

By pumping lemma, $z = uvwyx$ where $|vx| \geq 1$ and $uv^i w x^i y \in L(G)$ for all $i \geq 0$

ChatGPT-4 seems to generate cases well. They all make sense

Case 1: V and X consists of only of left a's, if $i = 0$
We will have too few a's.

Case 2: V and X consists of only middle b's if $i = 0$
We will have too few middle b's

Case 3: V and X consists of only right b's if $i = 0$
We will have too few right b's

Case 4: V and X consists of only of left a's and some middle b's
for $i = 2$, left a's will increase but middle b's won't.
String not in L

Case 5: V and X consists of only of middle and right b's
for $i = 2$, b's will increase but a's won't. too few a's

Case 6: V and X consists of all three sections. the ratio of a's to b's will change resulting in a string not in L.

$z \in L(G)$ but $\notin L$ $\quad \therefore$ Contradiction

Each case above results in contradiction $\therefore$ we have proved that L is not context free.

Q7 Summer 2022

Consider the class of $EVENCF_A$ of all CFL whose words are all of even length, over the fixed alphabet A.

(a) Is $EVENCF_A$ countable?
(b) Is the class $NOTCF_A$ countable where $NOTCF_A$ consists of all languages over A that are not context free?
(c) Is the class $ODDCF_A \cap NOTCF_A$ countable?

For each question you must give a precise argument substantiating your answer.

(a) The set of all words of even length over the fixed alphabet A is not infinite, therefor $EVENCF_A$ is countable.

(b) $EVENCF_A$ is a CFL and $NOTCF_A$ consists of all languages that are not context free. Therefor, we don't know if $NOTCF_A$ is context free or not. It could be complex or it could be infinite.

Let A^* represent all plausable conditions of the fixed alphabet.

$\therefore A^*$ is countable infinite
$\therefore 2^{A^*}$ is also countable infinite
$\therefore CF_A$ is countable
$\therefore 2^{A^*} - CF_A = NOTCF_A$ = infinite

$\therefore$ We can not account for all languages represented by $NOTCF_A$. Therefor, $NOTCF_A$ is not countable.

(c) Since $NOTCF_A$ consists of all languages that are not in CF_A, and every element of $ODDCF_A$ is in CF_A, no element can be both $ODDCF_A$ and $NOTCF_A$. Therefor $ODDCF_A \cap NOTCF_A$ will yield to an empty set. $ODDCF_A \cap NOTCF_A = \{\emptyset\}$

The cardinality of $\{\emptyset\}$ is 0 which is countable.

$\therefore ODDCF_A \cap NOTCF_A$ is countable.

Q8 Summer 2022 Not Complete

Construct a Turing Machine for the language in question 6, $L = \{a^{n+2} b^{n+2} b^n \mid n \geq 0\}$

Note: Describe first the process in english; then translate into moves of the turing machine.

Notations used:

a → left a's b_M → middle b's b_R → right b's ß → blanks

For the turing machine, I started from the left of the tape. First, I transform 'a' to 'A' when I read 'a' and continue right. Once I reach 'b_M' transform it into 'B_M' and continue right. once 'b_R' is reached, transform it into 'B_R' and change directions. Ignore all a, b_M, B_M, B_R until you reach 'A' and then change direction to the right again. If the next input is 'a' transform it to A and loop back to q_1 state. Else ignore all B_M until you find b_M and change it to B_M. Finally, move right ignoring all the B_R's until you find anymore b_R's and change it to B_R. Finally if the next input is ß, move to the final accepting state.

Imagine you have a word in L say n = 1. $W = aaabb_M b_R$ Use this to create your turing machine by above process.

A A a B_M b_M B_R

	a	b_M	b_R	A	B_M	B_R	ß
q_0	1 (q_1, A, R)						
q_1	2 (q_1, a, R)	3 (q_2, B_M, R)					
q_2		4 (q_2, b_M, R)	5 (q_3, B_R, L)				
q_3				6 (q_4, A, R)			
q_4	7 (q_1, A, R)				8 (q_5, B_M, R)		

Q9 Summer 2022

- Recursive always halts
- r.e. halts only on acceptance
- non-r.e. never halts

Let L_1 and L_2 be arbitrary languages, subject to specification in either (i) or (ii). Consider the following four questions:

(Q1) Does $L_2 - L_1$ contain a given fixed word w?
(Q2) Is $L_2 - L_1$ not empty?
(Q3) Does $L_2 \cap L_1$ contain a given fixed word w?
(Q4) Is $L_2 \cap L_1$ not empty?

For each of these four questions explain with reasons whether the problem is recursive, not recursive but r.e., or non-r.e. Provided
(i) Both L_1 and L_2 are recursive.
(ii) L_1 is recursive and L_2 is r.e. but not recursive.

(i) Given L_1 and L_2 are recursive

(i)(Q1) $L_2 - L_1 = L_2 \cap L_1^c$ Recursive languages are closed under complementation and intersection.
$\Rightarrow L_2 \cap L_1^c$ is recursive
$\Rightarrow L_2 - L_1$ is recursive

If the language is recursive then we can construct a TM that always halts by acceptance or rejectance.

Therefor the problem is recursive

(i)(Q2) $L_2 - L_1 \neq \emptyset$ if $L_2 \neq L_1$. To prove the languages are different a word must be found that is in one but not the other. The TM has to run forever to enumerate all possibilities with no guaranteed halt, therefor this problem is not recursive but r.e.

(i)(Q3) As we know, recursive languages are closed under intersection.

$L_2 \cap L_1$ is recursive

If the language is recursive then we can construct a TM that always halts by acceptance or rejectance.

Therefor the problem is <u>recursive</u>

(i)(Q4) $L_2 - L_1 \neq \emptyset$ if $L_2 \neq L_1$. To prove the languages are different a word must be found that is in one but not the other. The TM has to run forever to enumerate all possibilities with no guaranteed halt, therefor this problem is <u>not recursive but r.e.</u>

(ii) Given L_1 is recursive and L_2 is r.e. but not recursive.

(ii)(Q1) Since L_1 is recursive, a T.M is able to determine whether or not it contains x in finite time.

Since L_2 is r.e. but not recursive, a TM is able to halt upon acceptance, but if not accepted it may run infinitely.

To decide if $L_2 - L_1$ contains x, we run both above TM in parallel. However, if w is not in L_2 the process will never halt. Therefor this problem is <u>not recursive but r.e.</u>

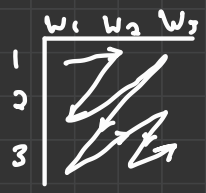
(ii)(Q2) We can enumerate the elements of L_2 using a TM and check if they are not in L_1 using TM_{L1}. If we find an element, we can conclude it is not empty. But if it is not empty, the process will not halt. Therefor, the question is <u>not recursive but r.e.</u>

(ii)(Q3) Since L_1 is recursive, a T.M is able to determine whether or not it contains x in finite time.

Since L_2 is r.e. but not recursive, a TM is able to halt upon acceptance, but if not accepted it may run infinitely.

To decide if $L_2 \cap L_1$ contains x, we run both TM_{L_1} and TM_{L_2} However, if w is not in L_2 the process will never halt. Therefor this problem is <u>not recursive but r.e.</u>

(ii)(Q4) Find a word in the language, brute force. May never halt because $L_1 \cap L_2$ is not recursive. We will use another trick. Given $w_1 \ldots w_n$ move like this



This movement is guaranteed to traverse every word despite it halting or not Therefor it will terminate if a word is enumerated in ENUM. <u>not recursive but r.e.</u>