

Q1 Summer 2022

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

	a	b	c	
$\rightarrow A$	B	/	/	0
B	C	/	A	1
C	/	A, B	/	0

$$\text{Step 1) } L_A = aL_B$$

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

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

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

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

$$L_C = (ba \cup b)L_B$$

$L_C \rightarrow L_B$ subst in L_C to $L_B \rightarrow \dots$

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

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

$L_A \rightarrow L_B$ subst in L_A to L_B

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

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

Kleene's theorem $= X = L^* \cup M = L^*$

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

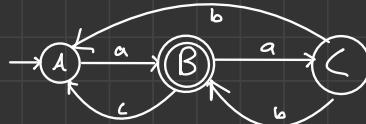
$$L_A \rightarrow L_B$$

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

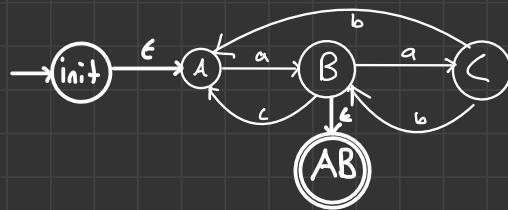
$$X = a(aba \cup ab \cup ca)^*$$

$\rightarrow A$	a	b	c	
B	/	/		0
C	/	/	A	1
		A,B	/	0

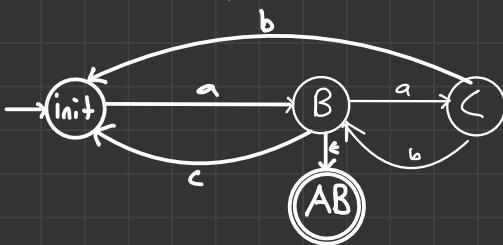
1. original



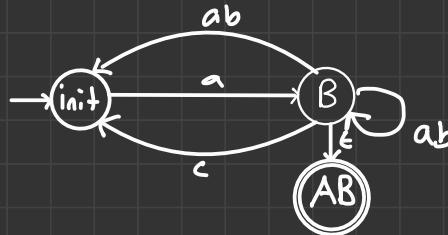
2. normalize



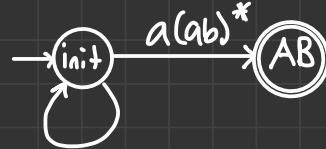
3. Remove state A



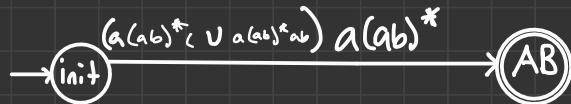
4. Remove state C



5. Remove state B



6. Remove cycle



$$X = (a(ab)^*_c \cup a(ab)^*_ab) a(ab)^*$$

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

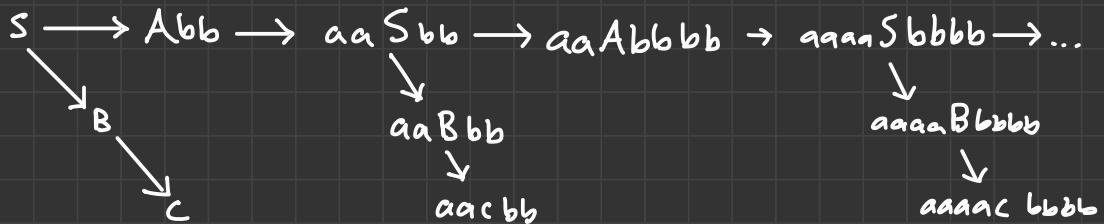
$$X = a(ab)^* ((a \cup aba(aab)^* \cup ab)$$

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

Q2 Summer 2022

Prove that a language $L(G)$ is not regular where
 G is the following CFG. $G = \{S, A, B, a, b, c\}, \{S \rightarrow AabbB, A \rightarrow aaS, B \rightarrow \epsilon\}, S$

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



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

To prove L is a regular language, assume L is regular \exists DFA T accepting L . N is the number of states in T

Consider the case $x = b^a d e^a$ with $|x| = N$
 $x = w \cdot v \quad w = b^{2N} \quad v = de^{2N} \quad |w| = N$

Pumping Lemma

$w = w_0 \cdot w_1 \cdot w_2$, such that $|w_2| > 0$ and $T(q_0, w)$ equals wherever $w \cdot v \in L$ $w_1(w_2)^s w_2 \in L \quad \forall s \geq 0 \quad T(q_0, w_1(w_2)^s w_2) \quad \forall s \geq 0$

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

$$\Rightarrow b^{2N - |w_2|} d e^{2N} \in L \quad \text{and} \quad T(q_0, v) = T(q_0, wv) \\ = T((q_0, w), v) \\ = T((q_0, w_1 w_2), v) \\ \in L(\epsilon)$$

Which is accepting the automata.

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

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

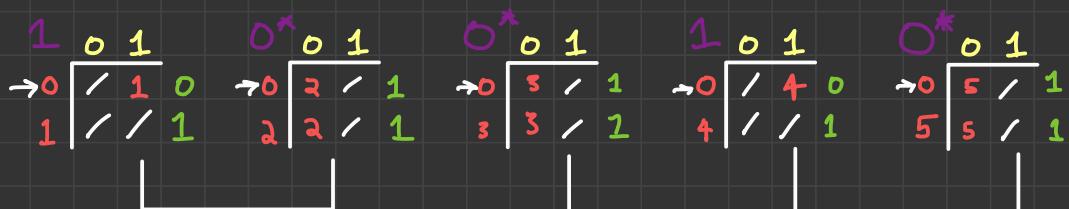
Q3 Summer 2022

Construct a reduced dfa for the following regular expression over $\{0, 1\}$ $[(0^*)^* \cap \overline{0^*10^*}]$. 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

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

1 = accepted 0 = not accepted



$$10^*$$

0	0	1
1	2	/
2	2	/
1	2	/

Applying outer *

$$(10^*)^*$$

0	0	1
1	2	1
2	2	1
1	2	1

DFA

$$(10^*)^*$$

0	0	1
1	2	1
2	2	1
6	6	0

Let 6
be Null.

$$0^*10^*$$

0	0	1
3	3	/
4	/	1
5	5	/

NOT

$(\overline{10^*})^*$		
	0	1
$\rightarrow 0$	6 1 0	
1	2 1 0	
2	2 1 0	
6	6 6 1	

Copy from
Prev Page

$0^* 1 0^*$		
	0	1
$\rightarrow 0$	3 4 0	
3	3 / 0	
4	/ / 1	
5	5 / 1	

union

$(\overline{10^*})^* \cup 0^* 1 0^*$

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow 0$	3,6 1,4 0	
1	2 1 0	
2	2 1 0	
3	3 6 0	
4	6 6 1	
5	5 6 1	
6	6 6 1	

$(\overline{10^*})^* \cup 0^* 1 0^*$

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow 0$	3,6 1,4 0	
3,6	3,6 6 1	
2,6	2,6 1,6 1	
6	6 6 1	
2,6	2,6 1,6 1	
1,6	1,6 2,6 1	

"1,0" is 1

$(\overline{10^*})^* \cup 0^* 1 0^*$

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow A$	B C 0	
B	B D 1	
B	E F 1	
C	D D 1	
D	E F 1	
E	F E 1	
F	E F 1	

Reduce

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow A$	B C D E F B C D E F 0	
B C D E F	B C D E F B C D E F 1	

Rename

$\rightarrow 1$

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow 1$	2 2 0	
2	2 2 1	

"NOT"

$\rightarrow 1$

$(\overline{10^*})^* \cup 0^* 1 0^*$		
	0	1
$\rightarrow 1$	2 2 1	
2	2 2 0	

Q4 Summer 2022

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

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

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

$$S \rightarrow bSb \mid Ba \quad B \rightarrow BdSc \mid S \mid \epsilon$$

Remove ϵ -Productions $X_1 \rightarrow \dots \rightarrow \epsilon$

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

Remove $B \rightarrow \epsilon$. We find $S \rightarrow Ba$ and $B \rightarrow BdSe$

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

Remove Unit productions

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

Productions: $S \rightarrow bSb \mid Ba \mid a \quad B \rightarrow BdSc \mid dSc \mid S$

$B \rightarrow S$ is Unit production

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

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

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

Given $S \rightarrow bSb|Ba|a$ $B \rightarrow BdSc|dSc|bSb|Ba|a$

Let $X_a \rightarrow a$ $X_b \rightarrow b$ $X_c \rightarrow c$ $X_d \rightarrow d$

$S \rightarrow X_b S X_b | BX_a | a$ $B \rightarrow BX_d S X_c | X_d S X_c | X_b S X_b | BX_a | a$

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

Given $S \rightarrow X_b S X_b | BX_a | a$ $B \rightarrow BX_d S X_c | X_d S X_c | X_b S X_b | BX_a | a$

$S \rightarrow X_b H_0 | BX_a | a$ $B \rightarrow BH_1 | X_d H_3 | X_b H_4 | BX_a | a$

$H_0 \rightarrow SX_b$ $H_1 \rightarrow X_a H_2$ $H_3 \rightarrow SX_c$ $H_4 \rightarrow SX_b$
 $H_2 \rightarrow SX_c$

$S \rightarrow X_b H_0 | BX_a | a$
 $B \rightarrow BH_1 | X_d H_3 | X_b H_4 | BX_a | a$

$H_0 \rightarrow SX_b$	$X_a \rightarrow a$
$H_1 \rightarrow X_a H_2$	$X_b \rightarrow b$
$H_2 \rightarrow SX_c$	$X_c \rightarrow c$
$H_3 \rightarrow SX_c$	$X_d \rightarrow d$
$H_4 \rightarrow SX_b$	

Q5 Summer 2022

based on 2018 solution
Not finished

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

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

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'

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

- remove ϵ productions
- remove Unit productions

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

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

- convert to chomsky normal form
- Replace all RHS lowercase that are not alone with $X_{\text{lowercase}}$

$$\begin{aligned} & \text{Let } X_a \rightarrow a \quad X_b \rightarrow b \\ & S \rightarrow AS|AA \quad A \rightarrow SA|X_a X_b|AA \end{aligned}$$

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

We are now in Chomsky normal form

$$S \rightarrow AS_1 A_1 \quad A \rightarrow SA_1 | X_a X_b | A_1 A$$

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

$$S = 1 \quad A = 2$$

Change the names of the Non-Terminal Symbols into

some A_i in order of descending i

Alter the rules so that non-terminals are in ascending order, such that, if the production is of the form $A_i \rightarrow A_j x_j$, then $i < j$ and should never be $\geq j$

Remove Left Recursion

Q6 Summer 2022 Not finished

Prove that the following language is not context free.

$$L = \{a^{n+2} b^{n+2} b^n \mid n \geq 0\}$$

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

Let $t = \text{no. of variables.}$

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

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

Once Exam 3 is released, determine if we need to disprove every case or not

How do we determine all cases?

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 $\text{ODDCF}_A \cap \text{NOTCF}_A$ 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, therefore EVENCF_A is countable.

(b) EVENCF_A is a CFL and NOTCF_A consists of all languages that are not context free. Therefore, 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 plausible conditions of the fixed alphabet.

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

\therefore We can not account for all languages represented by NOTCF_A . Therefore, 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 . Therefore $\text{ODDCF}_A \cap \text{NOTCF}_A$ will yield to an empty set. $\text{ODDCF}_A \cap \text{NOTCF}_A = \emptyset$

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

$\therefore \text{ODDCF}_A \cap \text{NOTCF}_A$ is countable.

Q8 Summer 2022

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

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

Notations used:

$a \rightarrow$ left a 's $b_M \rightarrow$ middle b 's $\lambda \rightarrow$ blank
 $b_R \rightarrow$ right b 's

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.

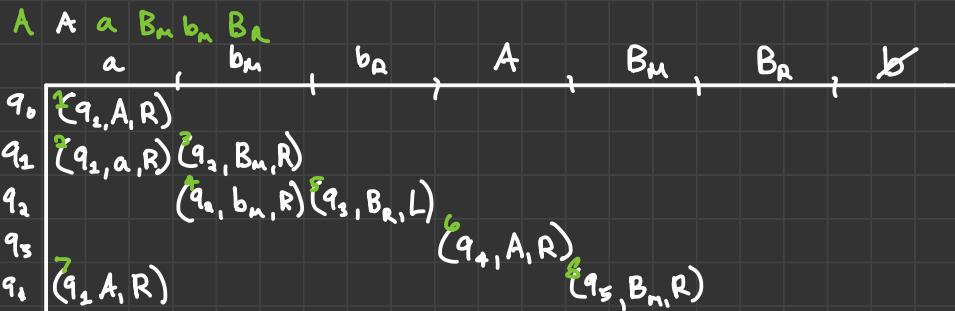
Once I reach ' b_M ' transform it into ' B_M ' and continue right.

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_2 state. Else ignore all B_M until you find b_N 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 λ , move to the final accepting state.

Imagine you have a word in L say $n=1$. $w = aaaa bbb bbb$

Use this to create your turing machine by above process.



Q9 Summer 2022

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) $L_2 - L_1 = L_2 \cap L_1'$ Recursive languages are closed under complementation and intersection.

$\Rightarrow L_2 \cap L_1'$ 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 rejection.

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 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 rejection.

Therefore the problem is recursive.

(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, therefore this problem is R.R.

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

(ii)(Q1) Should I ask less about this?