

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

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

$$L_B = \underbrace{(aba \cup ab \cup ca)}_X L_B \cup \underbrace{\epsilon}_M$$

$$\text{Arden's rule} \quad X = \frac{L \times U_M}{U_X} = L^*$$

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

$$L_B \rightarrow L_A$$

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

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

Q1 Summer 2010

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

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

$$\text{Step 1)} \quad L_A = aL_B \cup bL_C$$

$$L_B = aL_A \cup \epsilon$$

$$L_C = bL_A \cup bL_B$$

$$L_A = aL_B \cup bL_C$$

$$L_A = a(aL_A \cup \epsilon) \cup bL_B$$

$$L_A = aaL_A \cup \epsilon \cup bL_B$$

$$L_A = aaL_A \cup \epsilon \cup b(aL_A \cup \epsilon)$$

$$L_A = aaL_A \cup \epsilon \cup baL_A \cup \epsilon$$

$$L_A = aaL_A \cup baL_A \cup \epsilon$$

$$L_A = (aa \cup ba)L_A \cup \epsilon$$

$$L_A = \underbrace{(aa \cup ba)}_x L_A \cup \underbrace{\epsilon}_m$$

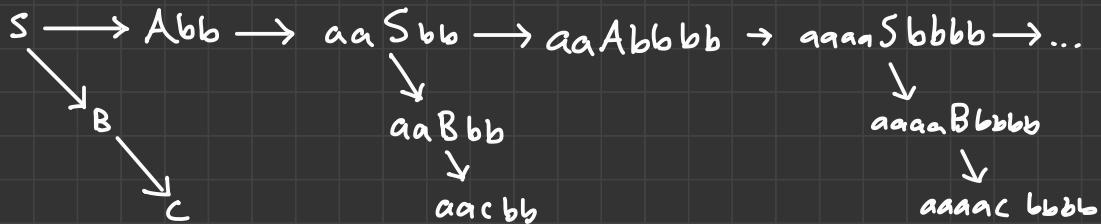
$$\text{Kleene's theorem} = X = L^* \cup \epsilon = L^*$$

$$X = (aa \cup ba)^* \epsilon$$

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

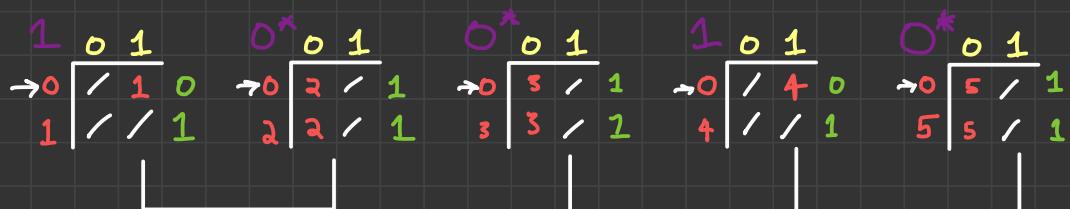
Q3 Summer 2022

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	0
3	3	0
4	5	/
5	5	/

NOT

$$(\overline{10^*})^*$$

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

$$0^* 1 0^*$$

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

union

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

	0	1
→ 0	3,6 1,4 0	
1	2 1 0	
2	2 1 0	
3	3 4 0	→ DFA
4	5 6 1	
5	5 6 1	
6	6 6 1	

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

	0	1
→ 0	3,6 1,4 0	
3,6	3,6 4,6 1	
2,5	2,5 1,6 1	
4,6	5,6 6 1	
2,5	2,5 1,6 1	
2,6	2,6 1,6 1	
5,6	5,6 6 1	
6	6 6 1	
2,6	2,6 1,4 1	

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

	0	1
→ A	B	C 0
B	B	D 1
E	E	F 1
D	G	H 1
F	E	F 1
I	I	F 1
G	G	H 1
H	H	H 1
I	I	F 1

Rename

Reduce

$$\rightarrow A$$

	0	1
BDEFGHI	BDEFGHI	0
BDEFGHI	BDEFGHI	1

$$\rightarrow 1$$

	0	1
2	2 2	0
2	2 2	1

"NOT"

$$\rightarrow 1$$

	0	1
2	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 $B \rightarrow BdSe$

$$\begin{aligned} S &\rightarrow bSb \mid Ba \mid a & \checkmark \\ B &\rightarrow BdSc \mid dSc \mid S & \checkmark \end{aligned}$$

Remove Unit productions

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

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

$B \rightarrow S$ is Unit production

Remove $B \rightarrow S$ - replace S with A

$$\begin{aligned} S &\rightarrow bSb \mid Ba \mid a \\ B &\rightarrow BdSc \mid dSc \mid bSb \mid Ba \mid a \end{aligned}$$

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

\nearrow Also be an A

- convert to chomsky normal form

Replace S with A

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

$$\text{Let } X_a \rightarrow a \quad X_b \rightarrow b$$

$$S \rightarrow AS|AA \quad A \rightarrow SA | X_a X_b | AA$$

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

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

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

$$S = 1 \quad A = 2 \quad X_a = 3 \quad X_b = 4 \quad Z = 5$$

Given $S_2 \rightarrow A_2 S_1 A_2$
 $A_i \rightarrow X_3 \dots | X_j \dots$

Rule is \exists

i=1 : $1 < 2$ ✓ Good, Move on

$i=2$ 2 & 1 Bad, change production $A_2 \rightarrow S_A | X_a X_b | AA$

Substitute S into A where S causes LR

$$2 \in J \quad A \rightarrow \underline{ASA} | AAA | X_a X_b | AA$$

$\alpha_1 \quad \alpha_2 \quad \beta_1 \quad \alpha_3$

$Z \rightarrow S A | S A Z | A A | A A Z | A I A Z$

$$A \rightarrow \underline{X_a X_b} \mid \underline{X_a X_b Z}$$

$$Z \rightarrow \alpha_n | \alpha_n Z$$

$$A \rightarrow \beta_n | \beta_n Z$$

$i = 3$

$i \in S$

$i=5$ $5 \neq 1$ we have $Z \rightarrow SA | SAZ | AA | AZ | A | AZ$
 $5 \neq 2$ $s \quad 1 \quad 1 \quad 2 \quad 2 \quad 1 \quad 2$

In Z Replace S with what S+ originally produces

$$\sum_s \rightarrow ASA|AAA|ASAZ|AAAZ|AA|AAZ|A|AZ$$

542

In Z replace first A with what new A produces

$Z \rightarrow X_a X_b SA | X_a X_b Z SA | X_a X_b AA | X_a X_b Z AA |$

X_a X_b SAZ | X_a X_b Z SAZ | X_a X_b AAZ | X_a X_b Z AAZ |

$X_a X_b A$ | $X_a X_b Z A$ | $X_a X_b AZ$ | $X_a X_b Z AZ$

$$X_a X_b | X_a X_b Z | \cancel{X_a X_b} \sum (X_a X_b Z Z)$$

Duplicates

$$5 \cancel{+} 3$$

We can replace X_a with a

Now we have :

$$S \rightarrow AS \mid AA$$

$$A \rightarrow aX_b \mid aX_bZ \quad X_a \rightarrow a \quad X_b \rightarrow b$$

$$\begin{aligned} Z \rightarrow & aX_bSA \mid aX_bZA \mid aX_bAA \mid aX_bZA \mid \\ & aX_bSAZ \mid aX_bZSAZ \mid aX_bAAZ \mid aX_bZA \mid \\ & aX_bA \mid aX_bZA \mid aX_bAZ \mid aX_bZAZ \mid \\ & aX_b \mid aX_bZ \mid aX_bZZ \end{aligned}$$

Back substitution. Replace left S in S production rules with all A productions

$$\begin{aligned} S \rightarrow & aX_bS \mid aX_bZA \mid aX_bA \mid aX_bZA \mid \\ & aX_b \mid aX_bZ \mid - \\ A \rightarrow & aX_b \mid aX_bZ \\ Z \rightarrow & aX_bSA \mid aX_bZA \mid aX_bAA \mid aX_bZA \mid \\ & aX_bSAZ \mid aX_bZSAZ \mid aX_bAAZ \mid aX_bZA \mid \\ & aX_bA \mid aX_bZA \mid aX_bAZ \mid aX_bZAZ \mid \\ & aX_b \mid aX_bZ \mid aX_bZZ \end{aligned}$$

$$X_a \rightarrow a$$

$$X_b \rightarrow b$$

The result of A' is indicated above as Z .

We have no S' because S caused no left recursion.

Q6 Summer 2022

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 $i \geq 0$
(that GPT-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 $\notin L \therefore \text{Contradiction}$

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 $\text{ODDCF}_A \cap \text{NOTCF}_A$ 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 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.

Let λ represent blank, and b 's on the right as ' c '. We know that there must be two more a 's than b 's, and there must be one more b than c .

To formulate a turing machine, start from the left of the tape and prime first two left a 's as A , then continue right. Once the first b is reached, mark it as B . This ensures there are $n+2$ a 's and $n+1$ b 's. Now we must check for equivalence of a 's b 's and c 's. Go back to the beginning of the tape. Now move right to next a . Mark it as A . Move right until next b . Mark it as B . Now move right unto the next c and mark it as a C . Now move to the beginning of the tape. Move right to find the next a . If we find the a , repeat the equivalence cycle. If no a is found, skip only A 's B 's and C 's and if we reach λ then move to accept state, otherwise reject.

Turing Machine on next page.

	a	b	c	A	B	C	λ
q_0	(q_1, A, R)	/	/	/	/	/	/
q_1	(q_2, A, R)	/	/	/	/	/	/
q_2	(q_2, a, R)	(q_3, B, L)	/	/	/	/	/
q_3	(q_3, a, L)	(q_3, b, L)	/	(q_4, A, R)	(q_3, B, L)	/	/
q_4	(q_5, A, R)	/	/	/	(q_4, B, R)	/	(q_f, λ, R)
q_5	(q_5, a, R)	(q_6, B, R)	/	/	(q_5, B, R)	/	/
q_6	/	(q_7, B, L)	/	/	/	/	/
q_7	(q_7, a, L)	/	/	(q_4, A, R)	(q_7, B, L)	/	/
q_f	/	/	/	/	/	/	/

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) $L_2 - L_1 = L_2 \cap L_1'$ 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 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 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 not recursive but r.e.

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

(ii)(Q1) Since L_1 is recursive, a TM 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. Therefore this problem is not recursive but r.e.

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

(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 $L_1 \cap L_2$ 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. Therefore this problem is not recursive but r.e.

(iii) (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 $w_1 \dots w_n$ move like this



This movement is guaranteed to traverse every word despite it halting or not. Therefore it will terminate if a word is encountered in ENUM. not recursive but r.e.