COSC 3340/6309 **Final Examination**

Thursday, July 5, 2018, 11 am – 2 pm

Open Book and Notes **Final grades only through PeopleSoft**

<u>YOU MUST USE THE CONSTRUCTIONS GIVEN IN CLASS</u>

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

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

2. Prove that the language L(G) is not regular where G is the following cfg:

$G = (\{S,A,B\}, \{a,b,c\}, \{S \to Ab|B, A \to aS, B \to c\}, S)$.

Note: You <u>must</u> first determine L(G).

3. Construct a reduced dfa for the following extended regular expression over {0,1}:

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

Note: You <u>must</u> first determine nfas for (10*)* and 0*10*, then do the intersection. The answer must then be reduced.

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

$G = (\{S,B\}, \{a,b,c,d\}, \{S \to Sb|Ba, B \to BdBc|S|\varepsilon\}, S)$.

Note: You <u>must</u> first remove all ε- and all unit productions.

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

$G = (\{S,A\}, \{a,b\}, \{S \to ASS|A, A \to SSS|baba\}, S)$.

Note: You <u>must</u> first remove all unit productions. You <u>must</u> derive all the productions for S and A; indicate how the result looks for S' and A'.

6. Prove that the following language L is not contextfree: $L = \{ 0^n1^{n+1}0^{n+2} \mid n \geq 1 \}$.

7. Consider the class CF_A of all context free languages over the fixed alphabet A.

(a) Is CF_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 $CF_A \cap NOTCF_A$ countable?

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

8. Construct a Turing machine for the language in Question 6, $L = \{ 0^n1^{n+1}0^{n+2} \mid n \geq 1 \}$.

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

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

(Q1) Does $L_1 - L_2$ contain a given fixed word w? (Q2) Is $L_1 - L_2$ empty?

(Q3) Does $L_1 \cap L_2$ contain a given fixed word w? (Q4) Is $L_1 \cap L_2$ empty?

For each of these four questions **<u>explain with reasons</u>** whether the general problem is <u>recursive</u>, <u>not recursive but r. e.</u>, or <u>non-r. e.</u>, provided

(i) Both L_1 and L_2 are <u>recursive</u>.

(ii) L_1 is <u>r. e., but not recursive</u> and L_2 is <u>recursive.</u>

Note that there are **eight** different questions to be answered.

Points: 1: 6 2: 8 3: 15 4: 11 5: 12 6: 12 7: 13 8: 8 9:15

Pio Estepa

0883903

Lemma

1. $X_A = bX_B \cup bX_C$ $\qquad X = LX \cup m$

$X_B = aX_A \cup cX_C \cup \varepsilon$ $\qquad X = L^* m$

$X_C = bX_A \cup bX_B \cup \varepsilon$

Plug X_C to X_B

$$X_B = \underbrace{aX_A \cup cbX_A}_{m} \cup \underbrace{cb}_{L}\underbrace{X_B}_{X} \cup \underbrace{c \cup \varepsilon}_{m}$$

$$X_B = (cb)^*(aX_A \cup cbX_A \cup c \cup \varepsilon) = (cb)^* aX_A \cup (cb)^* cbX_A \cup (cb)^* c \cup (cb)^*$$

Plug X_C to X_A

$$X_A = bX_B \cup bbX_A \cup bbX_B \cup b$$

Plug X_B to X_A

$$X_A = bbX_A \cup (b \cup bb)X_B \cup b$$

$$X_A = bbX_A \cup (b\cup bb)(cb)^* aX_A \cup (b\cup bb)(cb)^* cbX_A \cup (b\cup bb)(cb)^* c \cup (b\cup bb)(cb)^* \cup b$$

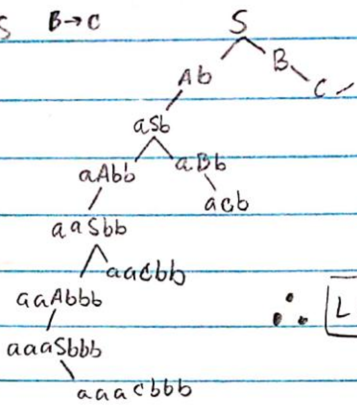
Factor X_A

$$X_A = \underbrace{(bb \cup (b\cup bb)(cb)^* a \cup (b\cup bb)(cb)^* cb)}_{L}\underbrace{X_A}_{X} \cup \underbrace{(b\cup bb)((cb)^* c \cup (cb)^*) \cup b}_{m}$$

$$X_A = (bb \cup (b\cup bb)(cb)^* a \cup (b\cup bb)(cb)^* cb)^* [(b\cup bb)((cb)^* c \cup (cb)^*) \cup b]$$

$$= \boxed{(bb \cup (b\cup bb)(cb)^*(a \cup cb))^* ((b\cup bb)((cb)^* c \cup (cb)^*) \cup b)}$$

$S \to Ab | B \quad A \to aS \quad B \to c$



$\therefore \boxed{L(G) = \{a^n c b^n \mid n \geq 0\}}$

First, assume $L(G)$ is regular, then there exists a DFA where $L(G)$ has a pumping length "n" such that any string "S" may be divided to 3 parts $S = xyz$ where all these conditions must be true:

Pumping Lemma: (1) $xy^iz \in L(G)$ for every $i \geq 0$

(2) $|y| > 0$

(3) $|xy| \leq n$

Let pumping length $n = 3$

let $S = \underbrace{aa}_{x}\underbrace{a}_{y}\underbrace{cbbb}_{z}$

(1) $xy^iz \in L(G)$, let $i = 2$ condition 1 fail

$S = aaaacbbb \to xy^2z \notin L(G)$ b/c you have $\{a^{n+1} c b^n\}$ $\therefore$ Fail

(2) $|y| > 0$

$|y| = 1$ ✓ condition 2 pass

(3) $|xy| \leq n$ where $n = 3$

$|x| = 2 \quad |y| = 1$

$\therefore |xy| = 3 \leq 3$ ✓ condition 3 pass

assumed that $L(G)$ is regular so it has to pass all 3 conditions

wever, it didn't pass condition (1) $\therefore$ it's a contradiction

$\therefore$ $L(G)$ is not regular.

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

$q_1 q_2$ $\quad$ $q_3 q_4 q_5$

	0	1	
q_0	/	q_1	0
q_1	/	/	1

	0	1	
q_0	q_2	/	0
q_2	/	/	1

(10^*)

q_0	/	q_1	0
q_1	q_2	/	1
q_2	q_2	/	1

$\downarrow *$

	0	1	
q_0	q_2	/	1
q_2	q_2	/	1

$\downarrow *$ $(10^*)^*$

q_0	/	q_1	1
q_1	q_2	q_1	1
q_2	q_2	q_1	1

q_0	q_3	/	1
q_3	q_3	/	1

q_0	/	q_4	0
q_4	/	/	1

q_0	q_5	/	1
q_5	q_5	/	1

q_0	q_3	q_4	0
q_3	q_3	q_4	0
q_4	/	/	1

0^*10^*

q_0	q_3	q_4	0
q_3	q_3	q_4	0
q_4	q_5	/	1
q_5	q_5	/	1

DFA $\quad$ complement

				complement
q_0	q_3	q_4	0	1
q_3	q_3	q_4	0	1
q_4	q_5	/	1	0
q_5	q_5	/	1	0
/	/	/	0	1

q_0	q_3	q_1, q_4	1
q_1	q_2	q_1	1
q_2	q_2	q_1	1
q_3	q_3	q_4	1
q_4	q_5	/	0
q_5	q_5	/	0
/	/	/	1

DFA

		0	1		complement
A	q_0	q_3	$q_1 q_4$	1	0
$\rightarrow$ B	q_3	q_3	q_4	1	0
C	q_4	q_5	/	0	1
D	q_5	q_5	/	0	1
E	$q_1 q_4$	$q_2 q_5$	q_1	1	0
F	$q_2 q_5$	$q_2 q_5$	q_1	1	0
G	q_1	q_2	q_1	1	0
H	q_2	q_2	q_1	1	0
I	/	/	/	1	0

Renamed

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

Reduction:

0	1
{A B E F G H I}	{C D}
{A E F G H I} {B}	{C D}
{A} {E F G H I} {B}	{C D}
{A} {B} {E F G H I}	{C D}
(1) (2) (3)	(4)

$\rightarrow$

	0	1	
1	2	3	0
2	2	4	0
3	3	3	0
4	4	3	1

4. $S \to Sb \mid Ba \quad B \to BdBc \mid S \mid \varepsilon$

Chomsky

$A \to a \quad A \to BC$

Eliminate $B \to \varepsilon$

$S \to Sb \mid Ba \mid a$

$B \to BdBc \mid dBc \mid Bdc \mid dc \mid S$

Eliminate $B \to S$

$S \to Sb \mid Ba \mid Sa \mid a$

$B \to BdBc \mid SdBc \mid BdSc \mid SdSc \mid dBc \mid dSc \mid Bdc \mid Sdc \mid dc$

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

$\boxed{S \to SX_b \mid BX_a \mid SX_a \mid a}$ ✓ CNF

$B \to BX_dBX_c \mid SX_dBX_c \mid BX_dSX_c \mid SX_dSX_c \mid X_dBX_c \mid X_dSX_c \mid BX_dX_c \mid SX_dX_c \mid X_dX_c$

$\boxed{B \to BB_1 \mid SB_1 \mid BB_3 \mid SB_3 \mid X_dB_2 \mid X_dB_4 \mid BB_5 \mid SB_5 \mid X_dX_c}$ ✓ CNF

$B_1 \to X_dB_2$ ✓

$B_2 \to BX_c$ ✓

$B_3 \to X_dB_4$ ✓

$B_4 \to SX_c$ ✓

$B_5 \to X_dX_c$

$S \to ASS \mid A \quad A \to SSS \mid baba$

Greibach

$X \to X\alpha \mid B_1 \mid B_2$

$X \to B_1 \mid B_2 \mid B_1 X' \mid B_2 X'$

$X' \to \alpha \mid \alpha X'$

Because we have S production on S, we have to make a new starting state S_0. ∴

$S_0 \to S \quad S \to ASS \mid A \quad A \to SSS \mid baba$

Eliminate unit production $S_0 \to S$

$S_0 \to ASS \mid A$

Eliminate $S \to A$

$S_0 \to ASS \mid AAS \mid ASA \mid AAA$

$S \to ASS \mid AAS \mid ASA \mid AAA$

$A \to SSS \mid ASS \mid SAS \mid SSA \mid AAS \mid ASA \mid SAA \mid AAA \mid baba$

$S_0 \to AS_1 \mid AA_1 \mid AS_2 \mid AA_2$

$S_1 \to SS \quad S_2 \to SA \quad A_1 \to AS \quad A_2 \to AA \quad X_a \to a \quad X_b \to b \quad X_1 \to X_b X_a$

$X_2 \to aX_1$

$S \to AS_1 \mid AA_1 \mid AS_2 \mid AA_2$

$A \to SS_1 \mid AS_1 \mid SA_1 \mid SS_2 \mid AA_1 \mid AS_2 \mid SA_2 \mid AA_2 \mid bX_2$

to A

$A \to AS_1S_1 \mid AA_1S_1 \mid AS_2S_1 \mid AA_2S_1 \mid AS_1 \mid AS_1A_1 \mid AA_1A_1 \mid AS_2A_1 \mid AA_2A_1 \mid$
$AS_1S_2 \mid AA_1S_2 \mid AS_2S_2 \mid AA_2S_2 \mid AA_1 \mid AS_2 \mid AS_1A_2 \mid AA_1A_2 \mid AS_2A_2 \mid AA_2A_2 \mid$
$AA_2 \mid \underbrace{bX_2}_{B_1}$

Eliminate left recursion

$\boxed{A = bX_2 \mid bX_2A'}$ ✓ GNF

Plug A to S

$$\boxed{\begin{aligned} S \to\ & bX_2S_1 \mid bX_2A'S_1 \mid bX_2A_1 \mid bX_2A'A_1 \mid \\ & bX_2S_2 \mid bX_2A'S_2 \mid bX_2A_2 \mid bX_2A'A_2 \end{aligned}}$$ ✓ GNF

(Next Page)

$A' \rightarrow S_1S_1 \mid S_1S_1A' \mid A_1S_1 \mid A_1S_1A' \mid S_2S_1 \mid S_2S_1A' \mid A_2S_1 \mid A_2S_1A' \mid SS \mid SSA' \mid$

$S_1A_1 \mid S_1A_1' \mid A_1A_1 \mid A_1A_1A' \mid S_2A_1 \mid S_2A_1A' \mid A_2A_1 \mid A_2A_1A' \mid S_1S_2 \mid S_1S_2A' \mid$

$A_1S_2 \mid A_1S_2A' \mid S_2S_2 \mid S_2S_2A' \mid A_2S_2 \mid A_2S_2A' \mid AS \mid ASA' \mid SA \mid SAA' \mid$

$S_1A_2 \mid S_1A_2A' \mid A_1A_2 \mid A_1A_2A' \mid S_2A_2 \mid S_2A_2A' \mid A_2A_2 \mid A_2A_2A' \mid$

$AA \mid AAA'$ not in GNF

To get A' to GNF, we have to plug in S & A into A'.

That will result in:

20 S prods from A' • 8 prods from S = 160 productions

20 A prods from A' • 2 prods from A = 40 productions

A' in GNF form will then have $160 + 40 = 200$ productions!

No S' productions b/c S had no left recursions.

6. $L = \{0^n 1^{n+1} 0^{n+2} \mid n \geq 1\}$

Assume L is CFL, then $\exists G = (N, T, P, S)$ in CNF so that $L = L(G)$.

By Pumping Lemma, we let $z \in L(G)$ where z can be expressed as

$z = uvwxy$ & with $|vx| \geq 1$, then we have:

$uv^iwx^iy \in L(G) \ \forall i \geq 0$

Let's consider: $z = 0^{2^n} 1^{2^{n+1}} 0^{2^{n+2}} \in L(G)$

Proof by Contradiction:

Case 1: V and X are the only 0's on the left side. if $i = 2$, then there are too many 0's on the left side.

Case 2: V and X are the only 0's on the right side if $i = 2$ then there are too many 0's on the right side

Case 3: V and X are only 1's. if $i = 0$ then there are not enough 1's in the middle

Case 4: V and X are left 0's and 1's. If $i = 2$, then there are too many 0's on the left side

Case 5: V and X are right 0's and 1's. If $i = 2$, then there are too many 0's on the right side.

Case 6: V and X are the left and right 0's respectively. If $i = 2$, then there are too many 0's on both sides and not enough 1's in the middle.

$z \in L(G)$ but $\notin L$ $\therefore$ contradiction

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

7. a. CF_A is a context free language if it's accepted by a pda. We know that pda is a finite automata. Therefore, CF_A is countable.

b. CF_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 or a problem.

Let A^* indicate all plausible conditions of the fixed alphabet:

$\therefore A^*$ is countable infinite

$\therefore 2^{A^*}$ is also countable infinite

$\therefore CFA$ is countable

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

$\therefore$ We cant 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, no element can be in both A and B. Therefore, $CF_A \cap NOTCF_A$ will yield to an empty set. $CF_A \cap NOTCF_A = \{\emptyset\}$

An empty set is a finite set with a cardinality of 0.

$|\{\emptyset\}| = 0$ which is countable.

$\therefore CF_A \cap NOTCF_A$ is countable

$L = \{0^n 1^{n+1} 0^{n+2} \mid n \geq 1\}$

Notations used:

$a \rightarrow$ 0's on the left of 1's

$b \rightarrow$ 1's in the middle

$a' \rightarrow$ 0's on the right of 1's

$\not{b} \rightarrow$ blank

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 a b is reached, transform it to B and continue to the right. Once an a' is reached, transform it into A' and change direction. Ignore all a, b, B, A' until you reach A. and 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 the B's until you find b and change it into B. Finally, move to the right ignoring all the A's until you find two a's and change them into A'. Finally, if the next input is blank $\not{b}$, move to the final accepting state.

	a	b	a'	A	B	A'	$\not{b}$
q_0	(q_1, A, R)	–	–	–	–	–	–
q_1	(q_1, a, R)	(q_2, B, R)	–	–	(q_1, B, R)	–	–
q_2		(q_2, b, R)	(q_3, A', L)	–		(q_2, A', R)	–
q_3	(q_3, a, L)	(q_3, b, L)	–	(q_4, A, R)	(q_3, B, L)	(q_3, A', L)	–
q_4	(q_1, A, R)	(q_5, B, R)	–	–	(q_4, B, R)	–	–
q_5	–	–	(q_6, A', R)	–	–	(q_5, A', R)	–
q_6	–	–	(q_7, A', R)	–	–	–	–
q_7	–	–	–	–	–	–	$(q_F, \not{b}, R)$
q_F	–	–	–	–	–	–	– accepting state

Test n=1

$\cancel{a}\cancel{b}\cancel{b}\cancel{a}\cancel{a}\cancel{a'}$ $\not{b}$ ✓

Test aabbáa' ✓

A a B b A' a $\not{b}$

i) L1 and L2 are recursive languages

Q1) Does L1-L2 contain a fixed word w?

L1 - L2 = L1 intersection $L2^c$

Recursive languages are closed under complementation and intersection

=> L1 intersection $L2^c$ is Recursive

=> L1 - L2 is recursive.

If the language is recursive then we can construct a turing machine that always halts. If w belongs to L1 - L2, then turing machine halts and shout accepted, otherwise halts and shout rejected.

=> problem is recursive.

Q2) Is L1 - L2 empty?

L1 - L2 = L1 intersection $L2^c$

Recursive languages are closed under complementation and intersection

=> L1 intersection $L2^c$ is Recursive

=> L1 - L2 is recursive.

If the language is recursive then we can construct a turing machine that always halts. The turing machine accepting the language will halt and reject on all input.

=> problem is recursive.

Q3) Does L1 intersection L2 contain a fixed word w?

As we know recursive languages are closed under intersection,

L1 intersection L2 is recursive.

If the language is recursive then we can construct a turing machine that always halts. If w belongs to L1 intersection L2, then turing machine halts and shout accepted, otherwise halts and shout rejected.

=> problem is recursive.

Q4) Is L1 intersection L2 empty?

As we know recursive languages are closed under intersection,

L1 intersection L2 is recursive.

If the language is recursive then we can construct a turing machine that always halts. The turing machine accepting the language will halt and reject on all input.

=> problem is recursive.

ii) L1 is r.e but not recursive, L2 is recursive.

Q1) Does L1 - L2 contain a fixed word w?

L1 - L2 = L1 intersection $L2^c$

L1 is recursive enumerable but not recursive.

Recursive languages are closed under complementation => $L2^c$ is recursive

=> $L2^c$ is recursive as well as recursive enumerable.

=> L1 intersection $L2^c$ is recursive enumerable but not recursive.

=> L1 - L2 is recursive enumerable but not recursive.

If the language is recursive enumerable but not recursive then we can construct a turing machine that always halts If w belongs to L1 - L2, otherwise for sme input which does not belong to L1 - L2 it might enter a loop.

=> problem is recursive enumerable but not recursive.

Q2) Is L1 - L2 empty?

L1 - L2 is recursive enumerable but not recursive.

=> problem is recursive enumerable but not recursive..

Q3) Does L1 intersection L2 contain a fixed word w?

L1 is recursive enumerable but not recursive.

L2 is recursive as well as recursive enumerable.

L1 intersection L2 is null

=> L1 intersection L2 is regular

=> L1 intersection L2 is recursive and recursively enumerable

=> problem is recursive enumerable.

Q4) Is L1 intersection L2 empty?

L1 is recursive enumerable but not recursive.

L2 is recursive as well as recursive enumerable.

L1 intersection L2 is null

=> L1 intersection L2 is regular

=> L1 intersection L2 is recursive and recursively enumerable

=> problem is recursive enumerable.