CS5003 Foundations of C.S. Spring, 2017

Final Exam

PRINT NAME: _____

$\mathcal{SIGN}$: _____

Do any six problems. Write your answers clearly and neatly. Use the back if necessary.

1. Let $L \subseteq \{a, b, c\}^*$ be the language with definition

$$L = \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}, 1 \leq i_k \leq 2, 1 \leq j_k \leq 3\}.$$

a) Give a recursive definition of L.

♣ (there are many ways to solve this problem) BASIS: $\lambda \in L$ RECURSIVE STEP: $w \in L \Rightarrow \{wab, wab^2, wab^3, wa^2b, wa^2b^2, wa^2b^3\} \in L$ CLOSURES: All elements of W are obtained from the BASIS after a finite number of applications of the RECURSIVE STEP. ♠

b) Compute L_3 according to your definition.

♣

$$\begin{aligned} L_0 = \{\lambda\}, L_1 &= L_0 \cup \{w \in \{a, b\}^* \mid w = a^ib^j, 1 \leq i \leq 2, 1 \leq j \leq 3\}. \\ L_2 &= L_1 \cup \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1}a^{i_2}b^{j_2}, 1 \leq i_1, 1_2 \leq 2, 1 \leq j_1, j_2 \leq 3\}. \\ L_3 &= L_2 \cup \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1}a^{i_2}b^{j_2}a^{i_3}b^{j_3}, 1 \leq i_1, 1_2, 1_3 \leq 2, 1 \leq j_1, j_2, j_j \leq 3\}. \end{aligned}$$

c) Prove by induction that $n_b \leq 3n_a$.

♣

$$\begin{aligned} L_0 = \{\lambda\}, L_1 &= L_0 \cup \{w \in \{a, b\}^* \mid w = a^ib^j, 1 \leq i \leq 2, 1 \leq j \leq 3\}. \\ L_2 &= L_1 \cup \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1}a^{i_2}b^{j_2}, 1 \leq i_1, 1_2 \leq 2, 1 \leq j_1, j_2 \leq 3\}. \\ L_3 &= L_2 \cup \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1}a^{i_2}b^{j_2}a^{i_3}b^{j_3}, 1 \leq i_1, 1_2, 1_3 \leq 2, 1 \leq j_1, j_2, j_j \leq 3\}. \end{aligned}$$

♣ We show $n_b(w) \leq 3n_a(w)$ for all $w \in L_n$, by induction on n.

Base Case: If $w \in L_0$ then $w = \lambda$, and $0 = n_b(\lambda) \leq 3n_a(\lambda) = 0$ as required.

Inductive step. Let the statement be true for n. Let $w' \in L_{n+1}$, so by the recursive construction, $w' = wa^ib^j$, with $1 \leq i \leq 2$, $1 \leq j \leq 3$, and $w \in L_n$, so $n_b(w) \leq 3n_a(w)$ by the inductive hypothesis. We have $n_a(w) + 1 \leq n_a(w') \leq n_a(w) + 2$ and $n_b(w) + 1 \leq n_b(w') \leq n_b(w) + 3$. So $n_b(w') \leq n_b(w) + 3 \leq 3n_a(w) + 3$, by the inductive hypothesis, and $3n_a(w) + 3 = 3(n_a(w) + 1) \leq 3n_a(w')$. So $n_b(w') \leq 3n_a(w')$ as required.

Since the base case is true as well as the inductive step, the result is true for all $0 \leq n$ by induction.

♠

2. Let $L \subseteq \{a, b, c\}^*$ be the language with definition

$$L = \{w \in \{a, b\}^* \mid w = a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}, 1 \leq i_k \leq 2, 1 \leq j_k \leq 3\}.$$

a) Construct a context free grammar G for which $L = L(G)$.

♣ Since there is a proof in part b), we want the grammar to be as simple as possible:

$$G : S \to \lambda \mid abS \mid ab^2S \mid ab^3S \mid a^2bS \mid a^2b^2S \mid a^2b^3S. \qquad \spadesuit$$

b) Prove that $L = L(G)$.

♣ First show $L \subseteq L(G)$. Let $a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n} \in L$.

We show any string wS with $w \in L$ has a derivation $S \stackrel{*}{\Rightarrow} wS$.

Induction on n.

BASE CASE: If $n = 0$, the $w = \lambda$ and S is a sentential form, as required.

INDUCTIVE STEP: Suppose $a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n} \in L(G)$, and consider $w' = a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}a^{i_{n+1}}b^{j_{n+1}}$ L. $w' = wa^{i_{n+1}}b^{j_{n+1}}$. $w \in L(G)$ by the induction hypothesis, so there is a derivation sequence $S \stackrel{*}{\Rightarrow} wS$, and one of six rules of the grammar $S \Rightarrow a^{i_{n+1}}b^{j_{n+1}}$, so $S \stackrel{*}{\Rightarrow} wS \Rightarrow wa^{i_{n+1}}b^{j_{n+1}} = w'$ as required.

So there is a derivation $S \stackrel{*}{\Rightarrow} w'S$ induction, and using the rule $S \to \lambda$, we also have $S \stackrel{*}{\Rightarrow} w'S \Rightarrow w'$ and $w' \in L(G)$.

Next show $L(G) \subseteq L$.

We show that every sentential form of G is either w or wS, with $w \in L$. Induction on the number of rules applied.

BASE CASE: 0 rules applied, then the sentential form is $S = \lambda S$ and $\lambda \in L$.

INDUCTION STEP: Suppose the result is true whenever m rules have been applied. If we have a derivation $S \stackrel{m+1}{\Rightarrow} w'$, then by the induction hypothesis $S \stackrel{m}{\Rightarrow} a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}S \Rightarrow w'$ for some n. If the last rule applied is $S \to \lambda$ then $w' = a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n} \in L$, as required, or if one of the six other rules $w' = a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}a^{i_{n+1}}b^{j_{n+1}}S$, and $a^{i_1}b^{j_1} \cdots a^{i_n}b^{j_n}a^{i_{n+1}}b^{j_{n+1}} \in L$ so w' is also in the correct form.

So all sentential forms are either w or wS, with $w \in L$, by induction, so all elements of the language $L(G)$ are in L. $\spadesuit$

3. Which of the following sets are countable?

___ $\mathbb{Q} \times (\mathbb{Z} \times \mathbb{Z})$

♣ Cartesian product of countable sets is countable. ♠

___ $\mathcal{P}(\{1, 2\})$

♣ $\mathcal{P}(\{1, 2\})$ is finite. ♠

___ $\bigcup_{k=2}^{\infty} \{a_1, a_2, \ldots a_k\}^*$

♣ $\{a_1, a_2, \ldots a_k\}^*$ is countable, and a countable union of countable sets is countable. ♠

___ $\{a, b, c\}^*$

♣ Finite strings on a finite set is countable. ♠

___ The set of all languages which are not regular.

♣ The set of all languages on an alphabet is the set of subsets of a countable set, so uncountable. The set of regular expressions is countable. So the set of non-regular languages is uncountable. ♠

4. Give regular expressions OR a regular grammar for each of the following subsets of $\{a, b, c\}^*$.

♣ Note: In general, the regular expressions are closely related to the non-deterministic automata, while the regular grammars are closer to the the deterministic ones, so we should in general expect that to construct a regular expression will be easier than to construct a regular grammar, but we do now know how to do both. ♠

a) Strings with an even number of b's.

♣ $(a \cup b(a \cup c)^* b \cup c)^*$ ♠

b) Strings in which every a is either immediately preceded, or immediately followed by a b.

♣ $(ba \cup ab \cup b \cup c)^*$ is the standard wrong answer, since it forces a separate b to be matched to each a, which is not required, so missing aba. The a correct answer is $(ba \cup ab \cup aba \cup b \cup c)^*$. ♠

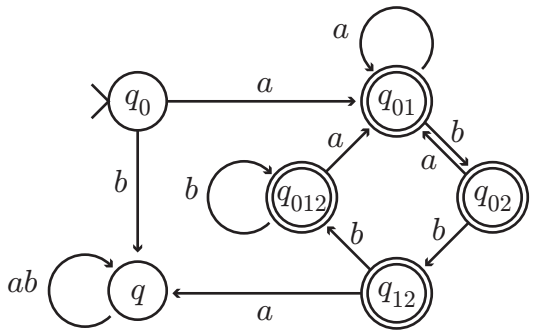
c) All strings not containing aba.

♣ In this case, the set is expressed as a complement, so we expect that a regular grammar might be easier:

$$\begin{aligned} G: (\text{no progress })S &\rightarrow \lambda \mid cS \mid bS \mid aA \mid c \mid b \mid a \\ (\text{ok ends in } a\ )A &\rightarrow aA \mid bB \mid cS \mid c \mid b \mid a \\ (\text{ok ends in } ab\ )B &\rightarrow bS \mid cS \mid c \mid b \end{aligned}$$

♠

5. Consider the following Deterministic Finite Automaton, M.



a) Find a regular expression for its language in $L(M) \subseteq \{a, b, \}^*$, or state why it doesn't have one.

♣ The language of every automaton has a regular expression, so you shouldn't waste time looking for a reason why not.

The start state is never returned to, and there is no escape from q. So no string starting with b is accepted. The cycle of accepting states tracks the number of b's with which the prefix ends: q_{01}, 0; q_{02}, 1; q_{12}, 2; and q_{012} 3 or more. The languages is all strings which start with a and do not contain the substring $abba$.

$$(a^+b \cup a^+b^3b^*)^*(a^* \cup a^+b^*)$$

It is easy to get the answer with expression graphs, but that is too tedious for me to type. ♠

b) Give a deterministic finite automaton whose language is the complement of $L(M)$ in $\{a, b\}^*$

♣ Just redraw the diagram interchanging the accepting and non-accepting states. ♠.

6. Let G be the grammar given by

$$\begin{aligned} G: S &\to S \mid AB \mid \lambda \\ A &\to aAaBC \mid ABC \mid a \\ B &\to BCA \mid b \\ C &\to bBcC \mid \lambda \end{aligned}$$

Convert this grammar to Chomsky Normal Form.

♣ First note that the rule $S \to S$ is useless and can be omitted.
Second, the fact that C is nullable must be rectified:

$$\begin{aligned} G': S &\to AB \mid \lambda \\ A &\to aAaBC \mid aAaB \mid ABC \mid AB \mid a \\ B &\to BCA \mid BA \mid b \\ C &\to bBcC \mid bBc \end{aligned}$$

Lastly we perform our splitting moves: note that the duplicated rules from the nullity moves don't need to be done completely separated.

$$\begin{aligned} G': S &\to AB \mid \lambda \\ A &\to U_aA_1 \mid AA_3 \mid AB \mid a \\ A_1 &\to AA_2 \\ A_2 &\to U_aA_3 \mid UaB \\ A_3 &\to BC \\ B &\to BB_1 \mid BA \mid b \\ B_1 &\to CA \\ C &\to U_bC_1 \\ C_1 &\to BC_2 \mid BU_c \\ C_2 &\to U_cC \\ U_a &\to a \\ U_b &\to b \\ U_c &\to c \end{aligned}$$

7. a) Let G be the grammar given by

$$\begin{aligned} G: S &\to A \mid B \\ A &\to aA \mid B \\ B &\to bB \mid C \mid D \\ C &\to cC \mid D \mid a \\ D &\to abc \end{aligned}$$

Construct an equivalent grammar which does not contain chain rules.

♣ It is easiest to compute bottom to top: $CHAIN(D) = D$, $CHAIN(C) = \{C, D\}$, $CHAIN(B) = \{B, C, D\}$, $CHAIN(A) = \{A, B, C, D\}$ and $CHAIN(S) = \{S, A, B, C, D\}$. So the grammar converts to:

$$\begin{aligned} S &\to aA \mid bB \mid cC \mid a \mid abc \\ A &\to aA \mid bB \mid cC \mid a \mid abc \\ B &\to bB \mid cC \mid a \mid abc \\ C &\to cC \mid a \mid abc \\ D &\to abc. \qquad ♠ \end{aligned}$$

b) For the grammar

$$\begin{aligned} G: S &\to aA \mid BD \\ A &\to aA \mid aAB \mid aD \\ B &\to aB \mid aC \mid BF \\ C &\to Bb \mid aAC \mid E \\ D &\to bD \mid bC \mid b \\ E &\to aB \mid bC \\ F &\to aF \mid aG \\ G &\to a \mid b \end{aligned}$$

Construct REACH and TERM in the correct order and remove the useless symbols

♣ We are not worried about the chain rules: $TERM_0 = \{D, G\}$, $TERM_1 = \{A, D, F, G\}$, $TERM = TERM_1 = \{S, A, D, F, G\}$ and conclude B, C, and E are not terminable. So the grammar converts to:

$$\begin{aligned} S &\to aA \\ A &\to aA \mid aD \\ D &\to bD \mid b \\ F &\to aF \mid aG \\ G &\to a \mid b \end{aligned}$$

Now $REACH_0 = \{S\}$, $REACH_1 = \{S, A\}$, $REACH = REACH_2 = \{S, A, D\}$, so F and G are unreachable. So the grammar converts to:

$$\begin{aligned} S &\to aA \\ A &\to aA \mid aD \\ D &\to bD \mid b \end{aligned}$$

$L(G) = a^2a^*b^+$. ♠.

8. For the grammar

$$\begin{aligned} G: S &\rightarrow AB \mid BA \mid \lambda \\ A &\rightarrow BB \mid AA \mid a \\ B &\rightarrow AA \mid b \end{aligned}$$

a) Trace the CYK algorithm to decide if *babbba* is in the language.

♣ Since S is not recursive, it is a waste of effort consider any S rules, or to record any S's until the very end.

$$\begin{array}{ccccccccccc} & & & & S,A & & & & & & \\ & & & \emptyset & & A,B & & & & & \\ & & A & & A & & A & & & & \\ & \emptyset & & A,B & & \emptyset & & A,B & & & \\ \emptyset & & \emptyset & & A & & A & & \emptyset & & \\ B & & A & & B & & B & & B & & A \end{array}$$

So the string is in the language. ♠

b) Convert to Greibach Normal Form.

♣ The grammar is already in Chomsky Normal Form and best to order $S < B < A$. Then the first order of business is to remove the initial B's in A rules:

$$\begin{aligned} S &\rightarrow AB \mid BA \mid \lambda \\ B &\rightarrow AA \mid b \\ A &\rightarrow AAB \mid bB \mid AA \mid a \end{aligned}$$

Then remove left recursion in A:

$$\begin{aligned} S &\rightarrow AB \mid BA \mid \lambda \\ B &\rightarrow AA \mid b \\ A &\rightarrow bB \mid a \mid bBC \mid aC \\ C &\rightarrow AB \mid A \mid ABC \mid AC \end{aligned}$$

Now A is compliant, and we can fix B and S in that order, and lastly, C

$$\begin{aligned} S &\rightarrow bBB \mid aB \mid bBCB \mid aCB \mid bBAA \mid aAA \mid bBCAA \mid aCAA \mid bA \mid \lambda \\ B &\rightarrow bBA \mid aA \mid bBCA \mid aCA \mid b \\ A &\rightarrow bB \mid a \mid bBC \mid aC \\ C &\rightarrow bBB \mid aB \mid bBCb \mid aCB \mid bB \mid a \mid bBC \mid aC \mid bBBC \mid aBC \mid bBCBC \mid \\ &\quad aCBC \mid bBC \mid aC \mid bBCC \mid aCC \end{aligned}$$