CSCE-433 Formal Languages & Automata
CSCE-627 Theory of Computability

Spring 2022

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: MWF 10:30–11:30am

Senior Grader: Avdhi Shah
Office: N/A
Phone: tba
Email: `avdhi.shah@tamu.edu`
Office Hours: tba

Solutions to Assignment #3

1. Use the pumping lemma for regular languages to show that the following languages are not regular (you might find it useful to study the solutions in the textbook to Exercise 1.29, parts (a) and (c)):

a) $\{www | w \in \{a,b\}^*\}$
b) $\{a^i(ab)^j(ca)^{2i} | i > 0, j > 0\}$
c) the set of properly nested parentheses (e.g., includes "()(())()" but not ")(")
d) (CSCE 433 students only) $\{a^n b^m | n < m\}$
e) (CSCE 627 students only) $\{a^i b^j c^{2j} | i \geq 0, j \geq 0\}$

Proof. In the proofs for all cases, we will assume that p is the pumping length given in Pumping Lemma, for which you can use either the version given in the textbook, or the version we presented in class.

a) Name the language by L_1. Consider the string $x = a^p ba^p ba^p b$ in L_1 (thus, $w = a^p b$). Assum the contrary that L_1 is regular. Then by Pumping Lemma, we can pump on the prefix a^p of x. Thus, this a^p can be written as $a^p = a^i a^k a^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = a^i (a^k)^r a^j ba^p ba^p b = a^i a^{rk} a^j ba^p ba^p b$$

is in the language L_1 for all $r \geq 0$. However, if we let $r = 2p + 2$ (note $k > 0$), then the first half of the string $x_{2p+2} = a^i a^{2pk+2k} a^j ba^p ba^p b$ is a sequence of all a's, while its second half contains b's. Thus, the string $x_{2p+2} = a^i a^{2pk+2k} a^j ba^p ba^p b$ cannot be of the pattern www for any $w \in \{a,b\}^*$, so cannot be in the language L_1. This contradiction shows that the language L_1 is not regular.

b) Name the language by L_2. Consider the string $x = a^p(ab)(ca)^{2p}$ in L_2. Assum the contrary that L_2 is regular. Then by Pumping Lemma, we can pump on the prefix a^p of x. Thus, this a^p can be written as $a^p = a^i a^k a^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = a^i (a^k)^r a^j (ab)(ca)^{2p} = a^i a^{rk} a^j (ab)(ca)^{2p}$$

is in the language L_2 for all $r \geq 0$. However, if we let $r = 2p$ (note $k > 0$), then there are at least $2pk + i + j \geq 2p$ copies of a's that appear before the first appearance of (ab), while there are only $2p$, which is strictly less than $2(2p) = 4p$, copies of (ca)'s that appear at the end of the string. Thus, the string $x_{2p} = a^i a^{2pk} a^j (ab)(ca)^{2p}$ cannot be of the pattern $a^i(ab)^j(ca)^{2i}$ for any $i > 0$ and $j > 0$, so cannot be in the language L_2. This contradiction shows that the language L_2 is not regular.

c) Name the language by L_3. The proof for this language is simpler. Consider the string $x = (^p)^p$ in L_3 (i.e., x starts with p copies of the left parenthesis "(", which are followed by p copies of the right parenthesis ")"). Assum the contrary that L_3 is regular. Then by Pumping Lemma, we can pump on the prefix $(^p$ of x. Thus, this $(^p$ can be written as $(^p = (^i(^k(^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = (^i(^{rk}(^j)^p$$

is in the language L_3 for all $r \geq 0$. However, if we let $r = 0$ (note $k > 0$), then the string is $x_0 = (^{p-k})^p$. Since $k > 0$, the parentheses are not balanced in x_0, so the string x_0 is not in the language L_3. This contradiction shows that the language L_3 is not regular.

d) Name the language by L_4. Consider the string $x = a^p b^{p+1}$ in L_4. Assum the contrary that L_4 is regular. Then by Pumping Lemma, we can pump on the prefix a^p of x. Thus, this a^p can be written as $a^p = a^i a^k a^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = a^i a^{rk} a^j b^{p+1}$$

is in the language L_4 for all $r \geq 0$. However, if we let $r = 2$, since $k > 0$, we have $i + 2k + j = i + k + j + k = p + k \geq p + 1$. Thus, the string is $x_2 = a^i a^{2k} a^j b^{p+1}$ is not in the language L_4. This contradiction shows that the language L_4 is not regular.

e) Name the language by L_5. Consider the string $x = b^p c^{2p}$ in L_5 (thus, we let $i = 0$ and $j = p$). Assum the contrary that L_5 is regular. Then by Pumping Lemma, we can pump on the prefix b^p of x. Thus, this b^p can be written as $b^p = b^i b^k b^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = b^i b^{rk} b^j c^{2p}$$

is in the language L_5 for all $r \geq 0$. However, if we let $r = 0$, since $k > 0$, we have $i + j = p - k < p$. Thus, the string $x_0 = b^i b^j c^{2p}$ contains (strictly) fewer than p copies of b but $2p$ copies of c. Thus, x_0 is not in the language L_5. This contradiction shows that the language L_5 is not regular. $\square$

2. [Textbook, page 90, Exercise 1.46 (a) and (c)] Prove that the following languages are not regular. You may use the pumping lemma and the closure of the class of regular languages under union, intersection, and complement.

(a) $\{0^n 1^m 0^n \mid m, n \geq 0\}$

(c) $\{w \mid w \in \{0,1\}^* \text{ is not a palindrome}\}$

Proof. Again in the proofs below, we will assume that p is the pumping length given in Pumping Lemma, for which you can use either the version given in the textbook, or the version we presented in class.

(a) Name the language by L_a. Consider the string $x = 0^p 1 0^p$ in L_a (thus, we have $n = p$ and $m = 1$). Assum the contrary that L_a is regular. Then by Pumping Lemma, we can pump on the prefix 0^p of x. Thus, this 0^p can be written as $0^p = 0^i 0^k 0^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = 0^i 0^{rk} 0^j 1 0^p$$

is in the language L_a for all $r \geq 0$. However, if we let $r = 0$, since $k > 0$, we have $i + j = p - k < p$. Thus, the string $x_0 = 0^{i+j} 1 0^p$ satisfies $i + j < p$, and is not in the language L_a. This contradiction shows that the language L_a is not regular.

(c) Name the language by L_c. Assum the contrary that L_c is regular. Since the class of regular languages is closed under complement, the complement $\overline{L_c}$ of L_c should be also regular.

The complement $\overline{L_c}$ of L_c is defined as:

$$\overline{L_c} = \{w \mid w \in \{0,1\}^* \text{ is a palindrome}\}.$$

We apply Pumping Lemma on the language $\overline{L_c}$. Recall that a string is a palindrome if it reads the same forward and backward. Since $\overline{L_c}$ is regular, we can assume that p is the pumping length for the regular language $\overline{L_c}$.

Consider the string $x = 0^p10^p$, which is obviously a palindrome, thus is in the language $\overline{L_c}$. By Pumping Lemma, we can pump on the prefix 0^p of x. Thus, this 0^p can be written as $0^p = 0^i0^k0^j$, where $i + k + j = p$ and $k > 0$ such that the string

$$x_r = 0^i0^{rk}0^j10^p$$

is in the language $\overline{L_c}$ for all $r \geq 0$. However, if we let $r = 0$, since $k > 0$, we have $i + j = p - k < p$. Thus, the string $x_0 = 0^{i+j}10^p$ is not a palindrome, thus, is not in the language $\overline{L_c}$. This contradiction shows that the language $\overline{L_c}$ is not regular. As a consequence, the language L_c is not a regular language as well. □

3. Write a context-free grammar for each of these languages; include a brief English intuition for how it works.

(a) $\{a^mb^ia^n | i = m + n, m \geq 0, n \geq 0\}$

(b) set of all strings over $\{a, b\}$ that have the same number of a's as b's; includes ϵ

(c) the complement of $\{a^nb^n | n \geq 0\}$, where $\Sigma = \{a, b\}$

Solution.

(a) Rewrite the language as $\{a^mb^mb^na^n \mid m \geq 0, n \geq 0\}$. Each string is the contatenation of a^mb^m and b^na^n for some values of $m \geq 0$ and $n \geq 0$. We can use concatenation in the production rules expanding the start variable S, and then separately use A and B to generate the two parts of the string:

$$\begin{array}{rcl} S & \rightarrow & AB \\ A & \rightarrow & aAb \mid \epsilon \\ B & \rightarrow & bBa \mid \epsilon \end{array}$$

(b) We consider three cases: a as the beginning of a substring and b is at the end of the substring with any (legal) string in between), or vice versa, or we have two legal strings side by side.

$$S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$$

(c) Categorize strings in this language into three groups: (1) those in which there is a b that precedes an a; (2) those in which all a's precede all b's but the number of a's is larger than the number of b's; and (3) those in which all a's precede all b's but the number of a's is smaller than the number of b's. We will have three "subgrammars", one for each group. The first group is created by variables S_1 and U, where U generates any string over $\{a, b\}$. Variables S_2 and S_3 are used to create strings in the second and third groups, respectively. Both of them use variable E, which creates strings of the form a^nb^n. For the second group, the $a^n b^n$ format strings are preceded by additional a's, created using variable A. For the third group, the $a^n b^n$ format strings are followed by additional b's, created using variable B.

$$
\begin{aligned}
S &\rightarrow S_1 \mid S_2 \mid S_3 \\
S_1 &\rightarrow UbUaU \\
U &\rightarrow aU \mid bU \mid \epsilon \\
S_2 &\rightarrow AE \\
A &\rightarrow a \mid aA \\
E &\rightarrow aEb \mid \epsilon \\
S_3 &\rightarrow EB \\
B &\rightarrow b \mid bB
\end{aligned}
$$

□

4. [Textbook, page 155, Exercise 2.9] Give a context-free grammar that generates the language

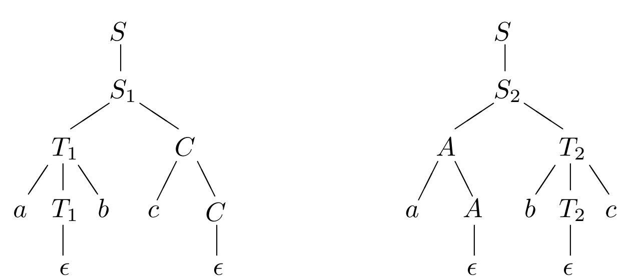
$$A = \{a^i b^j c^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}.$$

Is your grammar ambiguous? Why or why not?

Solution. The idea is to use nondeterminism in the first grammar rule from the start variable S to decide whether to have the number of a's equal the number of b's, or to have the number of b's equal the number of c's. Then we have separate "subgrammars" for those two cases. For generating strings with equal numbers of a's and b's, we use the variables S_1, T_1 and C, and use concatenation to append any number of c's at the end. An analogous idea is used for the other case.

$$
\begin{aligned}
S &\rightarrow S_1 \mid S_2 \\
S_1 &\rightarrow T_1 C \\
T_1 &\rightarrow aT_1 b \mid \epsilon \\
C &\rightarrow cC \mid \epsilon \\
S_2 &\rightarrow AT_2 \\
T_2 &\rightarrow bT_2 c \mid \epsilon \\
A &\rightarrow aA \mid \epsilon
\end{aligned}
$$

The grammar is ambiguous. For example, the string abc can be derived by the grammar in two different ways (i.e., has two different parse trees):



5. [Textbook, page 156, Exercise 2.14] Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9. Be sure to show all your steps.

$$\begin{aligned} A &\rightarrow BAB \mid B \mid \epsilon \\ B &\rightarrow 00 \mid \epsilon \end{aligned}$$

Solution. The procedure proceeds as follows.

Step 1: Add a new start variable$S$, and its associated rule.

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow BAB \mid B \mid \epsilon \\ B &\rightarrow 00 \mid \epsilon \end{aligned}$$

Step 2: Remove ϵ rules. First, remove $A \rightarrow \epsilon$ and get:

$$\begin{aligned} S &\rightarrow A \mid \epsilon \\ A &\rightarrow BAB \mid B \mid BB \\ B &\rightarrow 00 \mid \epsilon \end{aligned}$$

Next, remove $B \rightarrow \epsilon$ and get:

$$\begin{aligned} S &\rightarrow A \mid \epsilon \\ A &\rightarrow BAB \mid B \mid BB \mid AB \mid BA \mid A \\ B &\rightarrow 00 \end{aligned}$$

Step 3: Remove unit rules. First, remove $A \rightarrow A$ and get:

$$\begin{aligned} S &\rightarrow A \mid \epsilon \\ A &\rightarrow BAB \mid B \mid BB \mid AB \mid BA \\ B &\rightarrow 00 \end{aligned}$$

Then remove $S \rightarrow A$ and get:

$$\begin{aligned} S &\rightarrow BAB \mid B \mid BB \mid AB \mid BA \mid \epsilon \\ A &\rightarrow BAB \mid B \mid BB \mid AB \mid BA \\ B &\rightarrow 00 \end{aligned}$$

Then remove $S \rightarrow B$ and get:

$$\begin{aligned} S &\rightarrow BAB \mid 00 \mid BB \mid AB \mid BA \mid \epsilon \\ A &\rightarrow BAB \mid B \mid BB \mid AB \mid BA \\ B &\rightarrow 00 \end{aligned}$$

Then remove $A \rightarrow B$ and get:

$$\begin{aligned} S &\rightarrow BAB \mid 00 \mid BB \mid AB \mid BA \mid \epsilon \\ A &\rightarrow BAB \mid 00 \mid BB \mid AB \mid BA \\ B &\rightarrow 00 \end{aligned}$$

Step 4: Reduce all right-hand sides to two symbols. There is only one problematic such right-hand side: BAB, and we introduce a new variable C to take care of that:

$$\begin{array}{rcl} S & \rightarrow & BC \mid 00 \mid BB \mid AB \mid BA \mid \epsilon \\ C & \rightarrow & AB \\ A & \rightarrow & BC \mid 00 \mid BB \mid AB \mid BA \\ B & \rightarrow & 00 \end{array}$$

Step 5: Replace terminals on right-hand sides that are not singletons with variables. There is only one to worry about: 0, and we introduce a new variable U to take care of that:

$$\begin{array}{rcl} S & \rightarrow & BC \mid UU \mid BB \mid AB \mid BA \mid \epsilon \\ U & \rightarrow & 0 \\ C & \rightarrow & AB \\ A & \rightarrow & BC \mid UU \mid BB \mid AB \mid BA \\ B & \rightarrow & UU \end{array}$$

Now the grammar is in Chomsky normal form. $\square$