COMP 335 - Introduction to Theoretical Computer Science
Assignment 3

Ali Hanni - 40157164

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Fall 2021

Question 1

(15pts) Apply the Pumping Lemma to prove that the following languages are not regular.

(a) $L_1 = \{a^k b^n : n = 2^k\}$

(b) $L_2 = \{ww : w \in \{a^i b^i : i, j \geq 0\}\}$

(c) $L_3 = \{vw : v \in \{a, b\}^*, w \in \{c, d\}^*, |v| = |w|\}$

Answer

(a) We have to prove that there exist a string $w \in L_1$ where $|w| \geq m$ for $m > 0$, such that for all possible decomposition of w in xyz where $xyz = w$, $|xy| \leq m$, and $|y| \geq 1$ there exist at least one $i \geq 0$ such that $w = xy^i z \notin L_1$.

Let's assume that L_1 is a regular language.

We know that for all strings w element of L_1 and for any decomposition of w such that $w = xyz$, we have that $xy^i z$ must also be element of L_1 for $i \geq 0$. Now let $w = a^m b^{2^m} \in L_1$.

We know that $|a^m b^{2^m}| = m + 2^m \geq m$.
Now let $x = a^r$, $y = a^s$ and $z = a^{m-s-r} b^{2^m}$ for $r \geq 0$, $s \geq 1$ and $s + r \leq m$ such that:

$$w = xyz$$
$$|xy| = s + r \leq m$$
$$|y| = s \geq 1$$

For $i = 0$:

$$xy^0 z = xz$$
$$xz = a^r a^{m-s-r} b^m = a^{m-s} b^{2^m}$$

Since $|s| \geq 1$, we know that $m > m - s$. The string $a^{m-s} b^{2^m}$ is thus not a valid string of L_1 since $2^m \neq 2^{m-s}$ which contradicts the initial definition of L_1. Hence, the initial supposition is contradicted. L_1 does not respect the pumping lemma and is therefore not a regular language.

(b) We have to prove that there exist a string $w \in L_2$ where $|w| \geq m$ for $m > 0$, such that for all possible decomposition of w in xyz where $xyz = w$, $|xy| \leq m$, and $|y| \geq 1$ there exist at least one $i \geq 0$ such that $w = xy^iz \notin L_2$.

Let's assume that L_2 is a regular language.

We know that for all strings w element of L_2 and for any decomposition of w such that $w = xyz$, we have that xy^iz must also be element of L_2 for $i \geq 0$. Now let $w = w_1w_2 = a^mb^ma^mb^m \in L_2$ where $w_1, w_2 = a^mb^m$.

We know that $|a^mb^ma^mb^m| = 4m \geq m$.
Now let $x = a^r$, $y = a^s$ and $z = a^{m-s-r}b^ma^mb^m$ for $r \geq 0$, $s \geq 1$ and $s + r \leq m$ such that:

$$\begin{aligned} w &= xyz \\ |xy| &= s + r \leq m \\ |y| &= s \geq 1 \end{aligned}$$

For $i = 0$:

$$\begin{aligned} xy^0z &= xz \\ xz &= a^ra^{m-s-r}b^ma^mb^n = a^{m-s}b^ma^mb^m \end{aligned}$$

Since $|s| \geq 1$, we know that $m > m - s$. The string $a^{m-s}b^ma^mb^m$ is thus not a valid string of L_2 since $a^{m-s}b^m \neq a^mb^m$ which contradicts the initial definition of L_2. Hence, the initial supposition is contradicted. L_2 does not respect the pumping lemma and is therefore not a regular language.

(c) We have to prove that there exist a string $w \in L_3$ where $|w| \geq m$ for $m > 0$, such that for all possible decomposition of w in xyz where $xyz = w$, $|xy| \leq m$, and $|y| \geq 1$ there exist at least one $i \geq 0$ such that $w = xy^iz \notin L_3$.

Let's assume that L_3 is a regular language.

We know that for all strings w element of L_3 and for any decomposition of w such that $w = xyz$, we have that xy^iz must also be element of L_3 for $i \geq 0$. Now let $w = w = a^mb^ma^mb^m \in L_3$.

We know that $|a^mb^ma^mb^m| = 4m \geq m$.
Now let $x = a^r$, $y = a^s$ and $z = a^{m-s-r}b^ma^mb^m$ for $r \geq 0$, $s \geq 1$ and $s + r \leq m$ such that:

$$
\begin{aligned}
w &= xyz \\
|xy| &= s + r \leq m \\
|y| &= s \geq 1
\end{aligned}
$$

For $i = 0$:

$$
\begin{aligned}
xy^0z &= xz \\
xz &= a^r a^{m-s-r} b^m a^m b^n = a^{m-s} b^m a^m b^m
\end{aligned}
$$

Since $|s| \geq 1$, we know that $m > m - s$. The string $a^{m-s}b^m a^m b^m$ is thus not a valid string of L_3 since $|a^{m-s}b^m| \neq |a^m b^m|$ which contradicts the initial definition of L_3. Hence, the initial supposition is contradicted. L_3 does not respect the pumping lemma and is therefore not a regular language.

Question 2

(10 pts) Give context-free grammar for each of the following languages.

(a) $\{a^h b^k a^m b^n : h + k = m + n\}$

(b) $\{a^i b^j a^k : (i = j \; and \; k \geq 0) \; or \; (i \geq 0 \; and \; j > k)\}$

Answer

(a) The following context-free-grammar describes the language $L = \{a^h b^k a^m b^n : h + k = m + n\}$

$$\begin{aligned} S &\rightarrow T|U|V|W|\lambda \\ T &\rightarrow aSb|\lambda \\ U &\rightarrow aUa|bVa|\lambda \\ V &\rightarrow bVa|\lambda \\ W &\rightarrow bWb|bVa|\lambda \end{aligned}$$

(b) In order to find a context-free-grammar corresponding to the language $L = \{a^i b^j a^k : (i = j \; and \; k \geq 0) \; or \; (i \geq 0 \; and \; j > k)\}$, it helps to divide the problem into two smaller problems.
Let's say we have to find a grammar for $L_1 = \{a^i b^j a^k : i = j \; and \; k \geq 0\}$ and another for $L_2 = \{a^i b^j a^k : i \geq 0 \; and \; j > k\}$
For $L_1 = \{a^i b^j a^k : i = j \; and \; k \geq 0\}$ we have:

$$\begin{aligned} S_1 &\rightarrow S_2A|\lambda \\ S_2 &\rightarrow aS_2b|\lambda \\ A &\rightarrow aA|\lambda \end{aligned}$$

For $L_2 = \{a^i b^j a^k : i \geq 0 \; and \; j > k\}$ we have:

$$\begin{aligned} S_3 &\rightarrow A_2bS_4 \\ A_2 &\rightarrow aA_2|\lambda \\ S_4 &\rightarrow bS_4a|bS_4|\lambda \end{aligned}$$

Combining these two grammars yields a grammar corresponding to $\{a^i b^j a^k : (i = j \; and \; k \geq 0) \; or \; (i \geq 0 \; and \; j > k)\}$:

$$\begin{aligned} S &\rightarrow S_1|S_3 \\ S_1 &\rightarrow S_2A|\lambda \\ S_2 &\rightarrow aS_2b|\lambda \\ A &\rightarrow aA|\lambda \\ S_3 &\rightarrow A_2bS_4 \\ A_2 &\rightarrow aA_2|\lambda \\ S_4 &\rightarrow bS_4a|bS_4|\lambda \end{aligned}$$

Question 3

(15 pts) Let CFG G be defined by production $S \rightarrow aS|Sb|a|b$

(a) (10 pts) Prove by an induction of number of derivations steps that no string $w \in L(G)$ has ba as substring.

(b) (5 pts) Describe $L(G)$ formally.

Answer

(a) As a basis, we note that indeed, no string w that can be derived in one step has ba as a substring. The strings that can be derived from G in one step are a and b.
We assume that any string w obtained from n derivation steps does not have ba as a substring. Now, any w_1 derivable in $n+1$ steps is of the form:

$$
\begin{aligned}
S &\rightarrow aS \\
S &\rightarrow Sb \\
S &\rightarrow a \\
S &\rightarrow b
\end{aligned}
$$

We then know that the the next step (n + 1) can have 4 possible outcomes. First, if $S \rightarrow aS$ is used, an a is added at the left of the string w. The resulting string aw cannot contain ba as a substring. If $S \rightarrow Sb$ is used, a b is added at the right of the string w. The resulting string wb cannot contain ba as a substring. Therefore, it is impossible to build a string from G that contains ba as a substring.

(b) $L(G) = \{a^i b^j : i + j > 0\}$

Question 4

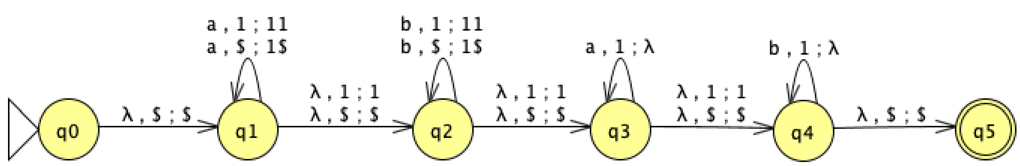
(10 pts) Design a PDA to accept each of the following languages. You may design your PDA to accept either by final state or empty stack, whichever is more convenient.

(a) $\{a^h b^k a^m b^n : h + k = m + n\}$

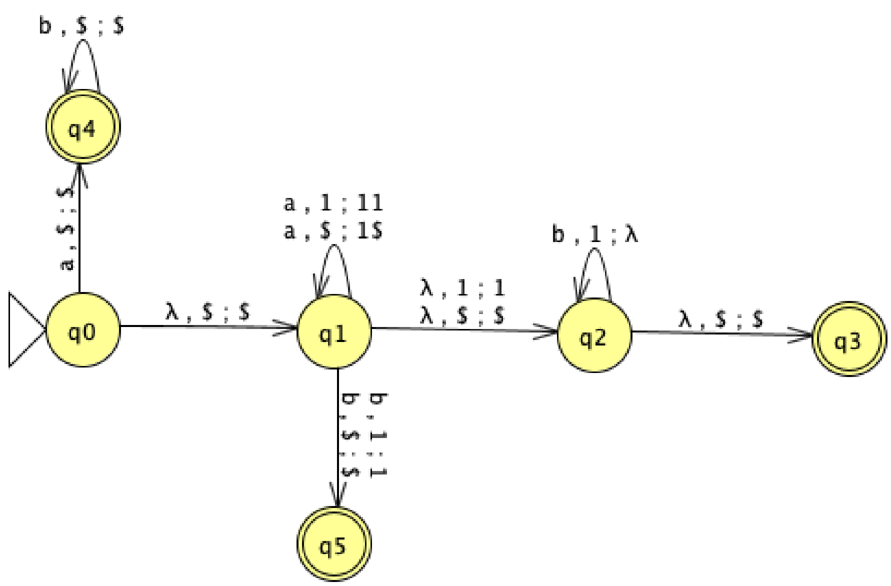
(b) $\{a^n b : n \geq 0\} \cup \{ab^n \geq 0\} \cup \{a^n b^n : n \geq 0\}$

Answer

(a) PDA corresponding to $\{a^h b^k a^m b^n : h + k = m + n\}$ is:



(b) PDA corresponding to $L = \{a^n b : n \geq 0\} \cup \{ab^n \geq 0\} \cup \{a^n b^n : n \geq 0\}$



Question 5

(10 pts) Convert the following grammars into Chomsky Normal Form.

(a) $S \rightarrow ASB|\lambda$, $A \rightarrow aAS|a$, $B \rightarrow SbS|A|bb$

(b) $S \rightarrow 0A0|1B1|BB,\ A \rightarrow C,\ B \rightarrow S|A,\ C \rightarrow S|\lambda$

Answer

(a) The grammar has the following productions:

$$
\begin{aligned}
S &\rightarrow ASB|\lambda \\
A &\rightarrow aAS|a \\
B &\rightarrow SbS|A|bb
\end{aligned}
$$

We first eliminate the start symbol S from RHS of productions. We do that by creating a new start symbol S_0 and new production $S_0 \rightarrow S$. It yields the following grammar:

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASB|\lambda \\
A &\rightarrow aAS|a \\
B &\rightarrow SbS|A|bb
\end{aligned}
$$

Second, we will eliminate λ-productions. Since the language described by this grammar accepts the empty string, we must keep a lambda production with the start variable S_0. It yields the following grammar:

$$
\begin{aligned}
S_0 &\rightarrow S|\lambda \\
S &\rightarrow ASB|\lambda \\
A &\rightarrow aAS|a \\
B &\rightarrow SbS|A|bb
\end{aligned}
$$

We now have to remove the λ-production $S \rightarrow \lambda$. By doing so, we obtain the following grammar:

$$
\begin{aligned}
S_0 &\rightarrow S|\lambda \\
S &\rightarrow ASB| \\
A &\rightarrow aAS|aA|a \\
B &\rightarrow SbS|Sb|bS|A|bb|b
\end{aligned}
$$

We then want to remove unit-productions. This grammar contains two unit-productions namely $S_0 \to S$ and $B \to A$. We replace S and A respectively by the right-hand-side of their productions (where they are on the left-hand-side).

It yields the following grammar:

$$
\begin{aligned}
S_0 &\to ASB|\lambda \\
S &\to ASB \\
A &\to aAS|aA|a \\
B &\to SbS|Sb|bS|aAS|aA|a|bb|b
\end{aligned}
$$

CNF does not accept production with both terminals and variables on their right-hand-side. The productions $A \to aAS$, $A \to aA$, $B \to SbS$, $B \to Sb$, $B \to bS$, $B \to aAS$, and $B \to aA$ must be changed. We do so by introducing two new variables and two new productions namely $X \to a$ and $Y \to b$. It yields the following grammar:

$$
\begin{aligned}
S_0 &\to ASB|\lambda \\
S &\to ASB \\
A &\to XAS|XA|a \\
X &\to a \\
Y &\to b \\
B &\to SYS|SY|YS|XAS|XA|a|bb|b
\end{aligned}
$$

CNF does not accept productions with more than one terminal value on the right-hand-side. The production $B \to bb$ must then be replaced. We do so by using the newly introduced production $Y \to b$. It yields the following grammar:

$$
\begin{aligned}
S_0 &\to S|\lambda \\
S &\to ASB \\
A &\to XAS|a \\
X &\to a \\
Y &\to b \\
B &\to SYS|SY|YS|XAS|XA|YY|a|b
\end{aligned}
$$

CNF does not accept productions with more than two variables on the rigth-hand-side. We can easily get ride of such productions by introducing new variables and productions. It yields the following grammar:

$$
\begin{aligned}
S_0 &\to PB|\lambda \\
P &\to AS \\
S &\to PB \\
A &\to QS|a \\
Q &\to XA \\
X &\to a \\
Y &\to b \\
B &\to RS|SY|YS|QS|XA|YY|a|b \\
R &\to SY
\end{aligned}
$$

Since no useless-productions are included, this production is now in Chomsky Normal Form.

(b) The grammar has the following productions:

$$
\begin{aligned}
S &\to 0A0|1B1|BB \\
A &\to C \\
B &\to S|A \\
C &\to S|\lambda
\end{aligned}
$$

We start by removing the λ-productions. We have $C \to \lambda$. We remove it, which introduces a new λ-production. $A \to \lambda$. Removing it yields the following grammar:

$$
\begin{aligned}
S &\to 0A0|00|1B1|BB \\
A &\to C \\
B &\to S|\lambda \\
C &\to S
\end{aligned}
$$

Removing the λ-transition $B \to \lambda$ yields the following grammar:

$$
\begin{aligned}
S &\to 0A0|00|1B1|BB|B|11|\lambda \\
A &\to C \\
B &\to S \\
C &\to S
\end{aligned}
$$

Then we have to remove unit-productions. There are 4 unit-productions in this grammar namely $A \to C$, $B \to S$, $S \to B$, and $C \to S$. Starting with

$C \to S$, we replace S with all productions that have S on the left-hand-side which yields the new productions $C \to 0A0|00|1B1|BB$. Doing the same with all other cases yields the following grammar:

$$
\begin{aligned}
S &\to 0A0|00|1B1|BB|11|\lambda \\
A &\to 0A0|00|1B1|BB|11| \\
B &\to 0A0|00|1B1|BB|11| \\
C &\to 0A0|00|1B1|BB|11|
\end{aligned}
$$

We then have to remove useless-productions. $C \to 0A0|00|1B1|BB$ since C can never be accessed. It is therefore removed as an useless-production. This yields the following grammar:

$$
\begin{aligned}
S &\to 0A0|00|1B1|BB|11|\lambda \\
A &\to 0A0|00|1B1|BB|11| \\
B &\to 0A0|00|1B1|BB|11|
\end{aligned}
$$

CNF does not accept productions with both terminals and variables on their right-hand-side. The productions $S \to 0A0$, $S \to 1B1$, $A \to 0A0$, $A \to 1B1$, $B \to 0A0$, and $B \to 1B1$ must be changed. We can do so by introducing new variables and corresponding new productions which yields the following language:

$$
\begin{aligned}
S &\to DAD|00|EBE|BB|11|\lambda \\
A &\to DAD|00|EBE|BB|11 \\
B &\to DAD|00|EBE|BB|11 \\
D &\to 0 \\
E &\to 1
\end{aligned}
$$

CNF does not accept productions with more than one terminal value on the right-hand-side. The productions $S \to 00$, $A \to 00$, and $B \to 00$ must be replaced. We can do so using the productions introduced in the previous step which yields the following grammar:

$$
\begin{aligned}
S &\to DAD|DD|EBE|BB|EE|\lambda \\
A &\to DAD|DD|EBE|BB|EE \\
B &\to DAD|DD|EBE|BB|EE \\
D &\to 0 \\
E &\to 1
\end{aligned}
$$

Finally, CNF does not accept productions with more than 2 variables on the right-hand-side. We replace such productions by introducing new productions and corresponding variables. This yields the following CNF grammar:

$$
\begin{aligned}
S &\to FD|DD|GE|BB|EE|\lambda \\
A &\to FD|DD|GE|BB|EE \\
B &\to FD|DD|GE|BB|EE \\
F &\to DA \\
G &\to EB \\
D &\to 0 \\
E &\to 1
\end{aligned}
$$