CSCE 428/828 HW 3

due in class Monday 3/13/2017
1. (15 pts) Design CFG’s for the following languages (10 pts). Briefly describe the strings derived by each variable in the grammars (5 pts).

\[ L_1 = \{ a^i b^j \mid 3i < j, i \geq 0, j \geq 0 \} \]
\[ L_2 = \{ a^i b^j \mid i \leq j \leq 3i, i \geq 0, j \geq 0 \} \]
\[ L_3 = \{ a^i b^j \mid i < j < 3i, i \geq 0, j \geq 0 \} \]

**CFG\(\text{(}L_1\text{)}\) =**
- \( S \rightarrow YB \)
- \( Y \rightarrow aYbb \mid \epsilon \)
- \( B \rightarrow bB \mid b \)

Variable \( Y \) generates \( \{ a^n b^{3n} \mid n \geq 0 \} \). Variable \( B \) generates \( b^+ \).

**CFG\(\text{(}L_2\text{)}\) =**
- \( S \rightarrow aSb \mid aSbb \mid aSbbb \mid \epsilon \)

Basic idea: Let’s consider a string in \( L \) (e.g., \( \epsilon, ab, abb, abbb \)). We can see that for each additional \( a \), there should be at least one and at most three additional \( b \)'s. The special case is the empty string.

**CFG\(\text{(}L_3\text{)}\) =**
- \( S \rightarrow aSb \mid aSbb \mid aSbbb \mid abb \)

Basic idea: Let’s consider a string in \( L \) (e.g., \( abb, aabb, aabbb, aabbbb \)). We can see that for each additional \( a \), there should be at least one and at most three additional \( b \)'s. The special case is the \( abb \).
2. (10 points) Design a **CFG** for the following language (5 points). Briefly describe the strings derived by each variable of your grammar (5 points).

\[ L = \{ a^i b^j \mid i \neq j \text{ and } 3i \neq j, \ i \geq 0, \ j \geq 0 \} \]

**SOLUTION:**

The following context-free grammar generates language \( L \), where

- \( S_1 \) generates language \( \{ a^i b^j \mid j < i \} \),
- \( S_2 \) generates language \( \{ a^i b^j \mid i < j < 3i \} \), and
- \( S_3 \) generates language \( \{ a^i b^j \mid 3i < j \} \).

\[
\begin{align*}
S & \rightarrow S_1 | S_2 | S_3 \\
S_1 & \rightarrow AX \\
A & \rightarrow aA | a \\
X & \rightarrow aXb | \varepsilon \\
S_2 & \rightarrow aS_2b | aS_2bb | aS_2bbb | abb \\
S_3 & \rightarrow YB \\
Y & \rightarrow aYbbb | \varepsilon \\
B & \rightarrow bB | b
\end{align*}
\]

Variable \( A \) generates \( a^+ \). Variable \( X \) generates \( \{a^n b^n \mid n \geq 0 \} \). Variable \( Y \) generates \( \{a^n b^{3n} \mid n \geq 0 \} \). Variable \( B \) generates \( b^+ \).
3. (10 pts) Convert the following CFG to an **equivalent PDA**.

\[
\begin{align*}
S & \rightarrow aSb \\
S & \rightarrow bSa \\
S & \rightarrow SS \\
S & \rightarrow \epsilon \\
\end{align*}
\]

![Diagram](image)
4. (10 points) Design a PDA recognizing the following language (5 points). Briefly describe how your PDA works (5 points). Directly design a PDA, and a PDA that is simply converted from a CFG is not allowed.

\[ L = \{a^i b^j \mid i \leq j \leq 3i, \ i \geq 0, \ j \geq 0 \} \]

**SOLUTION 1:**

Basic idea: Let’s consider a string in L (e.g., \( \epsilon, ab, abb, abbb \)). We can see that for each additional \( a \), there should be at least one and at most three additional \( b \)’s. The special case is the empty string.

The transition from state 1 to 2 pushes the dollar sign to mark the bottom of the stack. The loopback transition at state 2 counts the number of symbol \( a \) using the stack. The transition from state 2 to 3 makes sure that there is at least one symbol \( a \). The top loopback transition at state 3 corresponds to the case where a single \( a \) maps to a single \( b \), the bottom two transitions at state 4 correspond to the case where a single \( a \) maps to two \( b \)s, the bottom three transitions at states 5 and 6 correspond to the case where a single \( a \) maps to three \( b \)s. The transition from state 3 to 7 makes sure that the constraints are satisfied.

![Figure 1: Solution 1](image1)

**SOLUTION 2:** Another slightly different but equivalent PDA is

\[
\begin{align*}
a, \epsilon &\rightarrow A \\
a, \epsilon &\rightarrow AA \\
a, \epsilon &\rightarrow A AA \\
b, A &\rightarrow \epsilon
\end{align*}
\]

![Figure 2: Solution 2](image2)
5. (10 points) Design a **PDA** recognizing the following language (5 points). Briefly describe how your PDA works (5 points). Directly design a PDA, and a PDA that is simply converted from a CFG is not allowed.

\[ L = \{a^i b^j \mid i < j < 3i, \ i \geq 0, \ j \geq 0\} \]

**SOLUTION 1:**
Similar to the PDA for previous question but with 2 extra states \( p, q \) to satisfy the strict less than \( i < j < 3i \).

**SOLUTION 2:** Another slightly different but equivalent PDA is

\[
\begin{align*}
  a, \varepsilon &\rightarrow A \\
  a, \varepsilon &\rightarrow AA \\
  a, \varepsilon &\rightarrow AAA \\
  b, A &\rightarrow \varepsilon
\end{align*}
\]
6. (10 points) Design a **PDA** for the following language. Briefly describe how your PDA works. Directly design a PDA, and a PDA that is simply converted from a CFG is not allowed.

\[ \{a^i b^j c^k d^l \mid (2i + l) \leq (k - 3j), \ i \geq 0, \ j \geq 0, \ k \geq 0, \ l \geq 0\} \]

**SOLUTION:**

We have the following PDA for \( L \). We calculate the value of \( 2i + 3j - k + l \) by using \( X \) to denote +1 and using \( Y \) to denote -1. There are three possibilities in the stack:

- Only \( X \) (one or multiple): the value is greater than 0, and is the number of \( X \)'s.
- Only \( Y \) (one or multiple): the value is less than 0, and is the negative number of \( Y \)'s.
- Only \( \$ \) (just one): the value is 0.

Finally, we check whether \( 2i + 3j - k + l \leq 0 \), that is, whether we have \( Y \) in the stack or an empty stack.
7. (10 points) Let $L$ be the language of all palindromes over $\{0, 1\}$ containing an equal number of 0s and 1s. Show that $L$ is not context free using the **pumping lemma**.

**SOLUTION:**

String $0^p1^p0^p \in L$. Let’s consider the decomposition of $s = uvxyz$ with $|vy| > 0$ and $|vxy| \leq p$. Since $|vxy| \leq p$, it cannot contain both 0’s before 1’s and after 1’s at the same time. Let’s consider the following three cases:

- **$vy$ contains some 0’s before 1’s, and might contain some 1’s.** String $uv^2xy^2z$ contains more 0’s before 1’s than after 1’s, so it is not a member of $L$.
- **$vy$ contains some 0’s after 1’s, and might contain some 1’s.** String $uv^2xy^2z$ contains more 0’s after 1’s than before 1’s, so it is not a member of $L$.
- **$vy$ contains only 1’s.** String $uv^2xy^2z$ contains more 1’s than 0’s, so it is not a member of $L$.

Because $0^p1^p0^p$ cannot be pumped without violating the pumping lemma conditions, $L$ is not context free.
8. (10 points) Prove that the following language is not context free using pumping lemma.

\[ \{x_1#x_2#\ldots#x_n \mid n \geq 3, \text{ each } x_i \in (a \cup b)^*, \text{ and } x_j = x_k \text{ for some } j \neq k \} \]

Examples: strings \(aab#aab#\), \(ab#aab#a#b\), \(a##b\), and \(#a#\) belong to the language.

**PROOF:**

String \(s = a^pb^p##a^pb^p\) in the language. Let’s consider the decomposition of \(s = uvxyz\) with \(|vy| > 0\) and \(|vxy| \leq p\). Let’s consider the following two cases:

- **Case 1:** \(vy\) does not contain any \#.
  - Case 1.1: \(vy\) contains only some symbols before the first \#.
  - Case 1.2: \(vy\) contains some symbol \(b\)'s before the first \# and some symbol \(a\)'s after the second \#.
  - Case 1.3: \(vy\) contains only some symbols after the second \#.

- **Case 2:** \(vy\) contains at least one \# and possibly some other symbols. String \(uxz\) contains less than two \#’s, so it is not a member of the language. Note that \(uv^2xy^2z\) does not work in this case.

Because \(a^pb^p##a^pb^p\) cannot be pumped without violating the pumping lemma conditions, the language is not context free.

The following strings also work for the proof.

- strings \(a^pb^p##a^pb^p\), \(#a^pb^p##a^pb^p\)

**common mistakes:** the following strings do not work for the proof.

- strings \(a^pb^p##a##a^pb\), \(a^pb^p##a^pb^p\)
- strings \(a^p##a^p\), \(a^p##a^p\), \(a^p##a##a^p\)
- strings \(a^pb##a^pb\), \(a^pb##a^pb\), \(a^pb##a##a^pb\)
- any string with more than two \#’s, such as string \(a^pb^p#a^pb^p#a\#\)
9. (EXTRA CREDIT: 15 points) Prove or disprove that \( L \) is a CFL. If prove, then use either a CFG or PDA (and as before, provide some brief descriptions). If disprove, then use the Pumping Lemma.

\[ L = \text{the complement of } \{a^n b^n \mid n \geq 0\} \text{ over alphabet } \{a, b\} \]

**SOLUTION:**

Observe that \( L \) is the union of the following three languages.

\[
\begin{align*}
L_1 &= \{a^i b^j \mid i > j \geq 0\} \\
L_2 &= \{a^i b^j \mid 0 \leq i < j\} \\
L_3 &= \Sigma^* ba \Sigma^*
\end{align*}
\]

We have the following the context free grammar for language \( L \).

- \( S \to S_1 \mid S_2 \mid S_3 \)
- \( S_1 \to AT \)
- \( S_2 \to TB \)
- \( S_3 \to EbaE \)
- \( A \to aA \mid a \)
- \( B \to bB \mid b \)
- \( T \to aTb \mid \varepsilon \)
- \( E \to aE \mid bE \mid \varepsilon \)

\( S_1 \) generates all strings in language \( L_1 \). \( S_2 \) generates all strings in language \( L_2 \). \( S_3 \) generates all strings in language \( L_3 \). \( A \) generates all non-empty strings with only symbol \( a \). \( B \) generates all non-empty strings with only symbol \( b \). \( T \) generates \( a^n b^n \) with \( n \geq 0 \). Finally, \( E \) generates all strings over alphabet \( \{a, b\} \).