CpSc 421 Homework 3 **Due: Oct. 21**

NO LATE HOMEWORK ACCEPTED

1. **(30 points):** (Question 2 from Kozen Homework 5)
   Prove that the CFG:
   $$S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$$
   generates the set of all strings of $\{a, b\}$ with equally many a's and b's. (**Hint:** Characterize elements of the set in terms of the graph of the function $\#b(y) - \#a(y)$ as y ranges over prefixes of x, as we did with balanced parentheses.)

Solution:

Let A be the language of all strings over $\{a, b\}^*$ that have an equal number of a's and b's. I'll prove that $L(G) \subseteq A$ and $A \subseteq L(G)$ separately.

$L(G) \subseteq A$:

Let $w \in L(G)$. My proof in by induction on the number of steps, n, in the derivation of w.

Induction Hypothesis: $S \xrightarrow[G]{n} \alpha \;\Rightarrow\; (\#\mathsf{a}(\alpha) = \#\mathsf{b}(\alpha))$

Base case — $n = 0$:

$$\begin{array}{ll} & S \xrightarrow[G]{0} \alpha \\ \Rightarrow & \alpha = S \\ & (\#\mathsf{a}(\alpha) = 0) \wedge (\#\mathsf{b}(\alpha) = 0) \\ & \#\mathsf{a}(\alpha) = \#\mathsf{b}(\alpha) \end{array}$$

Induction step — $S \xrightarrow[G]{n} \alpha A \beta \xrightarrow[G]{1} \alpha\mu\beta$ There is a separate case for each production of $A \rightarrow \mu$ of G.

$A \rightarrow \mu \;\equiv\; S \rightarrow aSb$:

$$\begin{array}{rcll} \#\mathsf{a}(\alpha\mu\beta) & = & \#\mathsf{a}(\alpha aSb\beta), & \text{case hypothesis: } \mu = aSb \\ & = & \#\mathsf{a}(\alpha\beta) + \#\mathsf{a}(aSb), & \text{properties of addition} \\ & = & \#\mathsf{a}(\alpha S\beta) + 1, & \#\mathsf{a}(s) = 0, \#\mathsf{a}(aSb) = 1 \end{array}$$

Likewise, $\#\mathsf{b}(\alpha aSb\beta) = \#\mathsf{b}(\alpha S\beta) + 1$. $\#\mathsf{a}(\alpha S\beta) = \#\mathsf{b}(\alpha S\beta)$ from which we conclude $\#\mathsf{a}(\alpha aSb\beta) = \#\mathsf{b}(\alpha aSb\beta)$ as required.

$A \rightarrow \mu \;\equiv\; S \rightarrow aSb$:

The same argument as for the previous case (swapping a and b) applies here.

$A \rightarrow \mu \;\equiv\; S \rightarrow SS$:

As there are neither any a nor b terminals in S, we have $\#\mathsf{a}(\alpha S\beta) = \#\mathsf{a}(\alpha SS\beta)$, and likewise for the b's. Thus, $\#\mathsf{a}(\alpha SS\beta) = \#\mathsf{b}(\alpha SS\beta)$ follows directly from the induction hypothesis.

Thus, if $S \xrightarrow[G]{*} w$, then $\#\mathsf{a}(w) = \#\mathsf{b}(w)$. Therefore, $w \in A$ and I conclude $L(G) \subseteq A$.

$A \subseteq L(G)$: Let $w \in A$. I'll prove $w \in L(G)$ by induction on w. As in the proof for the balanced parenthesis language, my proof uses "strong" induction – I use apply the induction hypothesis to strings that are shorter than $|w| - 1$.

Induction Hypothesis — $((w \in A) \wedge (|w| \leq n)) \;\Rightarrow\; (w \in L(G))$:

Base step — $w = \epsilon$:

$S \rightarrow \epsilon$. Thus, $\epsilon \in L(G)$.

Induction step:

I break the proof into two cases as was done for the balanced parentheses language according to whether or not w can be divided into a non-empty prefix and suffix where each has an equal number of a's and b's.

case $\exists x, y \in \Sigma^+. \ (w = xy) \wedge (\#\mathsf{a}(x) = \#\mathsf{b}(x)) \wedge (\#\mathsf{a}(y) = \#\mathsf{b}(y))$:
Because $|x| < |w|$ and $x \in A$, the induction hypothesis applies, and I conclude $x \in L(G)$. Equivalently, $S \xrightarrow[G]{*} x$. Likewise, $S \xrightarrow[G]{*} y$. The production $S \to SS$ is in G. Thus,

$$S \xrightarrow[G]{1} S\ S \xrightarrow[G]{*} x\ S \xrightarrow[G]{*} x\ y = w$$

Thus, $w \in L(G)$.
case $\neg\exists x, y \in \Sigma^+. \ (w = xy) \wedge (\#\mathsf{a}(x) = \#\mathsf{b}(x)) \wedge (\#\mathsf{a}(y) = \#\mathsf{b}(y))$:
Because $\#\mathsf{a}(x)$ and $\#\mathsf{b}(x)$ are integers and change by 0 or $+1$ as symbols are appended to x, I conclude:

$$\forall x, y \in \Sigma^+. \ (w = xy) \Rightarrow ((\#\mathsf{a}(x) > \#\mathsf{b}(x)) \vee (\#\mathsf{a}(y) > \#\mathsf{b}(y)))$$

I'll assume that $\#\mathsf{a}(x) > \#\mathsf{b}(x)$ for all prefixes of w with lengths in $\{1 \dots |w| - 1\}$. The other case is similar. Let c_1 be the first symbol of w and $c_{|w|}$ be the last symbol of w, and choose y such that $w = c_1 \cdot y \cdot c_|w|$. By the assumption about x, $count(c_1) > \#\mathsf{b}(c_1)$ which implies that $c_1 = a$. Furthermore, $\#\mathsf{a}(c_1 y) > \#\mathsf{b}(c_1 y)$, and $\#\mathsf{a}(c_1 y c_{|w|}) = \#\mathsf{b}(c_1 y c_{|w|})$. Thus, $c_{|w|} = b$. In other words, $w = a \cdot y \cdot b$. Because

$$\#\mathsf{a}(y) = (\#\mathsf{a}(w) - 1) = (\#\mathsf{a}(w) - 1) = (\#\mathsf{b}(w) - 1) = \#\mathsf{b}(y)$$

$y \in A$. By the induction hypothesis, $y \in L(G)$. Thus, $S \xrightarrow[G]{*} y$; furthermore $S \xrightarrow[G]{1} aSb \xrightarrow[G]{*} a\,y\,b = w$. Thus $w \in L(G)$ as required.

This completes the induction proof.

This induction argument shows that $A \subseteq L(G)$.

Having shown $L(G) \subseteq A$ and $A \subseteq L(G)$, I conclude $L(G) = A$.

2. **(30 points):** (Question 2 from Kozen Homework 6)
Construct a pushdown automaton that accepts the set of strings in $\{a, b\}^*$ with equally many a's and b's. Specify all transitions.

Solution:

Let M be a PDA that accepts on empty stack with

$$\begin{array}{rcl} M & = & (\{q\}, \{a, b\}, \{\perp, a, b\}, \delta, q, \perp, \emptyset) \\ \delta & = & \{ \ (q, a, \perp) \to (q, a \perp) \\ & & \quad (q, a, a) \to (q, aa) \\ & & \quad (q, a, b) \to (q, \epsilon) \\ & & \quad (q, b, \perp) \to (q, b \perp) \\ & & \quad (q, b, b) \to (q, bb) \\ & & \quad (q, b, a) \to (q, \epsilon) \\ & & \quad (q, \epsilon, \perp) \to (q, \epsilon) \\ & & \} \end{array}$$

This machine has the charming property that

$$\begin{array}{rl} & (q, xy, \perp) \to (q, y, \alpha) \\ \Leftrightarrow & ((\#\mathsf{a}(x) - \#\mathsf{b}(x)) = (\#\mathsf{a}(\alpha) - \#\mathsf{b}(\alpha))) \wedge ((\#\mathsf{a}(\alpha) = 0) \vee (\#\mathsf{b}(\alpha) = 0)) \end{array}$$

This can be shown by induction on x. Furthermore, the machine can always consume its entire input string, as there is always a move on either input symbol that doesn't empty the stack. Thus, if w has an equal number of a's and b's, M will reach a configuration where $\#\mathsf{a}(\alpha) = \#\mathsf{b}(\alpha)$. Since at least one of $\#\mathsf{a}(\alpha)$ or $\#\mathsf{b}(\alpha)$ must be zero, they must both be zero. Thus, the stack must be either $\perp$ or ϵ. In the former case, the machine can perform the transition $(q, \epsilon, \perp) \to (q, \epsilon)$ to empty its stack. In the other case, the stack is already empty. Thus, M accepts w.

Conversely, if M accepts w, M must read all of w and empty its stack. By the property shown above, if M empties its stack, then it has read an equal number of a's and b's. Thus, w has an equal number of a's and b's.

My explanation is longer than is needed to get full credit. My goal is to make sure that everyone in the class gets their questions answered.

3. **(40 points):** Let T be the language over the alphabet $\{\texttt{[}, \texttt{][}, \texttt{]}\}$ such that every $\texttt{[}$ is followed by its matching $\texttt{][}$, and every $\texttt{][}$ is followed by its matching $\texttt{]}$, and the total number of $\texttt{[}$ symbols and the total number of $\texttt{]}$ symbols in the string are the same. More formally, let

$$\begin{array}{rclrclrcl}
\mathit{left}(\epsilon) & = & 0, & \mathit{middle}(\epsilon) & = & 0, & \mathit{right}(\epsilon) & = & 0, \\
\mathit{left}(x\,\texttt{[}) & = & \mathit{left}(x)+1, & \mathit{middle}(x\,\texttt{[}) & = & \mathit{middle}(x), & \mathit{right}(x\,\texttt{[}) & = & \mathit{right}(x), \\
\mathit{left}(x\,\texttt{][}) & = & \mathit{left}(x), & \mathit{middle}(x\,\texttt{][}) & = & \mathit{middle}(x)+1, & \mathit{right}(x\,\texttt{][}) & = & \mathit{right}(x), \\
\mathit{left}(x\,\texttt{]}) & = & \mathit{left}(x), & \mathit{middle}(x\,\texttt{]}) & = & \mathit{middle}(x), & \mathit{right}(x\,\texttt{]}) & = & \mathit{right}(x)+1
\end{array}$$

A string x is in T iff

$$\forall y, z.\ x = yz.\ (\mathit{left}(y) \geq \mathit{middle}(y)) \wedge (\mathit{middle}(y) \geq \mathit{right}(y)) \wedge (\mathit{left}(x) = \mathit{middle}(x) = \mathit{right}(x))$$

For example, the strings

$$\texttt{[\,][\,]} \qquad \texttt{[\,[\,][\,[\,][\,]\,]\,[\,][\,]\,][\,]} \qquad \texttt{[\,][\,[\,]\,][\,]}$$

are in T, but the strings

$$\texttt{[\,]\,[\,][\,][\,]} \qquad \texttt{[\,]} \qquad \texttt{[\,][\,[\,][\,]\,][\,[\,]\,]}$$

are not.

(a) **(20 points):** Prove that T is not a context-free language.

Solution:

If T were context-free, it would have a pumping lemma constant, k. Let $z = \texttt{[}^k\,\texttt{][}^k\,\texttt{]}^k$. Clearly, $w \in T$. Consider any strings u, v, w, x, y with $uvxwy = z$, $|vx| \geq 1$, and $|vwx| \leq k$. If vwx is contained in the prefix $\texttt{[}^k\,\texttt{][}^k$, then pumping it will result in having a different number of $\texttt{[}$ or $\texttt{][}$ symbols than $\texttt{]}$ symbols. Likewise, if vwx is a substring of $\texttt{][}^k\,\texttt{]}^k$, then pumping it will produce a string with a different number of $\texttt{][}$ or $\texttt{]}$ symbols than $\texttt{[}$ symbols. In either case, a string that is not in T is produced. It is not possible for vwx to be contain symbols in both $\texttt{[}^k$ and $\texttt{]}^k$ because it would have to have a length of at least $k+1$. Thus, T doesn't satisfy the conditions of the pumping lemma. Therefore, it is not context-free.

(b) **(15 points):** Give the grammars for two CFLs, A_1 and A_2 such that $T = A_1 \cap A_2$.

Solution:

Let

$$\begin{array}{rcll}
G_1 & = & (\{S_1, R\}, \{\texttt{[}, \texttt{][}, \texttt{]}\}, P_1, S_1), & \text{grammar for } A_1 \\
P_1 & = & \{\ S_1 \to R S_1 R \mid \texttt{[}\,S_1\,\texttt{][} \mid S_1 S_1 \mid \epsilon & \\
 & & \quad R \to \texttt{]} \mid \epsilon & \\
 & & \} & \\
G_2 & = & (\{S_2, L\}, \{\texttt{[}, \texttt{][}, \texttt{]}\}, P_2, S_2), & \text{grammar for } A_2 \\
P_2 & = & \{\ S_2 \to L S_2 L \mid \texttt{][}\,S_2\,\texttt{]} \mid S_2 S_2 \mid \epsilon & \\
 & & \quad L \to \texttt{[} \mid \epsilon & \\
 & & \} &
\end{array}$$

The grammar G_1 accepts any strings where left and middle parentheses match properly, and allows right parentheses to appear anywhere. The grammar G_2 accepts any strings where middle and right parentheses match properly, and allows left parentheses to appear anywhere. Thus, any string that is in both $L(G_1)$ and $L(G_2)$ has every left parentheses matched by a subsequent middle parentheses, and every middle parentheses matched by a subsequent right parentheses. This is the language T.

(c) **(5 points):** Are context-free languages closed under intersection? Give a *short* justification for your answer.

Solution:

No. As shown above, G_1 and G_2 are context-free grammars. $L(G_1) \cap L(G_2) = T$, and T is not context-free.

(d) **(5 points):** Are context-free languages closed under complement? Give a *short* justification for your answer.

Solution:

As noted in the newsgroup, I had meant to write "intersection" instead of "complement". A short answer is that CFLs are not closed under complement as shown in the text book. For example, the language $\{x|\ x = ww\}$ is not a CFL, but its complement is.

Here's another proof that uses what we've just shown. CFLs are closed under union. Let A_1 and A_2 be two CFLs over the same alphabet. Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, P_2, S_2)$ be CFGs for A_1 and A_2 respectively. We can assume that N_1 and N_2 are disjoint and that neither contains the symbol S (this can be achieved by renaming non-terminals in one set or the other if needed). Let

$$\begin{array}{rcl} G & = & (N, \Sigma, P, S) \\ N & = & N_1 \cup N_2 \cup \{S\} \\ P & = & P_1 \cup P_2 \cup \{S \rightarrow S_1 \mid S_2\} \end{array}$$

It is straightforward to show that $L(G) = A_1 \cup A_2$. Thus, CFLs are closed under union.

Now, if CFLs were closed under both union and complement, they would also be closed under intersection by De Morgan's Law:

$$A_1 \cap A_2 \quad = \quad \sim((\sim A_1) \cup (\sim A_2))$$

We have shown that CFLs are closed under union but not closed under intersection. Therefore, they are not closed under complement.