| | |
|---|---|
| Introduction to Formal Language, Fall 2016 | **Due: 21-Apr-2016 (Thursday)** |

# Homework 4 Solutions

*Instructor: Prof. Wen-Guey Tzeng* *Scribe: Yi-Ruei Chen*

1. Find context-free grammars for the language $L = \{a^n b^m : n \neq 2m\}$ with $n \geq 0$, $m \geq 0$.
**Answer.**
[Solution 1] Parse $L$ as $L = L_1 \cup L_2$, where $L_1 = \{a^n b^m : n > 2m\}$ and $L_2 = \{a^n b^m : n < 2m\}$. Then construct productions for $L_1$ and $L_2$, respectively. A context-free grammar for $L$ is $G = (\{S, S_1, S_2, A, B\}, \{a, b\}, S, P)$ with the productions
$$\begin{aligned} S &\to S_1 | S_2, \\ S_1 &\to aaS_1b | A, A \to a | aA, \\ S_2 &\to aaS_2b | B, B \to b | Bb | ab. \end{aligned}$$
[Solution 2] Produce $L' = \{a^n b^m : n = 2m\}$ then add extra $a$'s or $b$'s. A context-free grammar for $L$ is $G = (\{S, A, B\}, \{a, b\}, S, P)$ with the productions
$$\begin{aligned} S &\to aaSb | A | B, \\ A &\to a | aA, \\ B &\to b | Bb | ab. \end{aligned}$$
□

2. Find context-free grammars for the language $L = \{a^n b^m c^k : k \neq n+m\}$. (with $n \geq 0$, $m \geq 0$, $k \geq 0$)
**Answer.**
Parse $L$ as $L = L_1 \cup L_2$, where $L_1 = \{a^n b^m c^k : k > n+m\}$ and $L_2 = \{a^n b^m c^k : k < n+m\}$. Then construct productions for $L_1$ and $L_2$, respectively. A context-free grammar for $L$ is $G = (\{S, S_1, S_2, T_1, T_2, A, B, C\}, \{a, b, c\}, S, P)$ with the productions
$$\begin{aligned} S &\to S_1 | S_2, \\ S_1 &\to aS_1c | T_1, T_1 \to bT_1c | C, C \to cC | c, \\ S_2 &\to aS_2c | T_2 | AB | A | B, T_2 \to bT_2c | B, A \to aA | a, B \to bB | b. \end{aligned}$$
□

3. Show that $L = \{w \in \{a, b, c\}^* : |w| = 3n_a(w)\}$ is a context-free language.
**Answer.**
A context-free grammar for $L$ is $G = (\{S, T\}, \{a, b, c\}, S, P)$ with the productions
$$\begin{aligned} S &\to SaSTSTS | STSaSTS | STSTSaS | \lambda, \\ T &\to b | c. \end{aligned}$$
□

4. Let $L = \{a^n b^n : n \geq 0\}$. Show that $\overline{L}$ and $L^*$ are context-free.

**Answer.**

(1) $\overline{L} = \{a^m b^k : m \neq k\} \cup \{(a+b)^* ba(a+b)^*\}$. A context-free grammar for $\overline{L}$ is $G = (\{S, S_1, S_2, A, B, T\}, \{a, b\}, S, P)$ with the productions

$$
\begin{aligned}
&S \to S_1 | S_2, \\
&S_1 \to aS_1b | A | B, A \to a | aA, B \to b | bB, \\
&S_2 \to TbaT, T \to aT | bT | \lambda.
\end{aligned}
$$

(2) $L^* = \{(a^n b^n)^m : n, m \geq 0\}$. A context-free grammar for $\overline{L}$ is $G = (\{S, S_1, S_2, A, B, T\}, \{a, b\}, S, P)$ with the productions

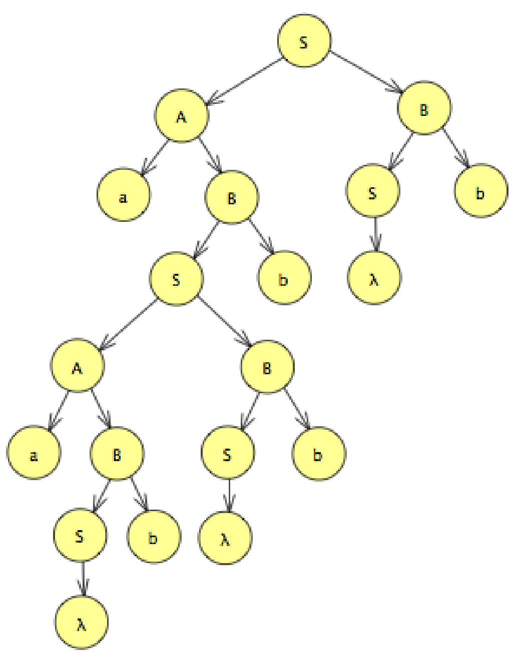
$$
\begin{aligned}
S &\to SS_1 | \lambda, \\
S_1 &\to aS_1b | \lambda.
\end{aligned}
$$

□

5. Show a derivation tree for the string *aabbbb* with the grammar

$$
\begin{aligned}
S &\to AB | \lambda, \\
A &\to aB, \\
B &\to Sb.
\end{aligned}
$$

**Answer.**



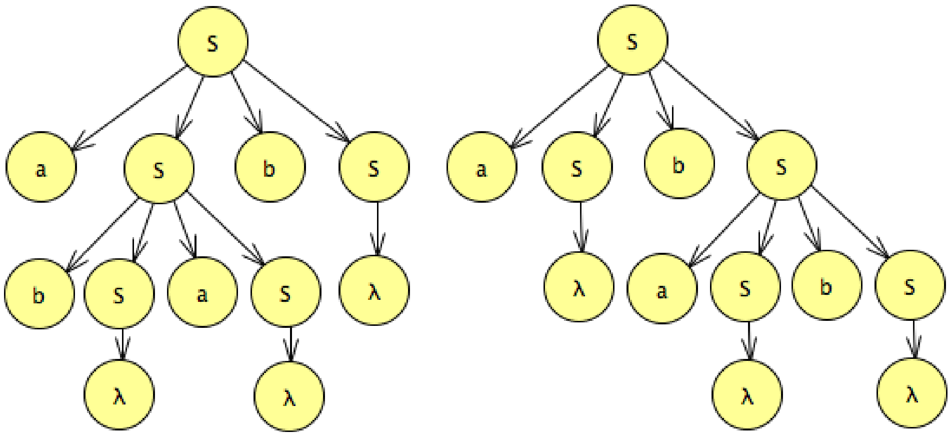

□

6. Define what one might mean by properly nested parenthesis structures involving two kinds of parentheses, say ( ) and [ ]. Intuitively, properly nested strings in this situation are ([ ]), ([[ ]])[( )], but not ([ )] or (( ]]. Using your definition, give a context-free grammar for generating all properly nested parentheses.
**Answer.**
A context-free grammar for generating all properly nested parentheses is $G = (\{S\}, \{(, ), [, ]\}, S, P)$ with production
$$S \to [S]|(S)|\lambda.$$
□

7. Find an s-grammar for $L = \{a^n b^{n+1} : n \geq 2\}$.
**Answer.**
An s-grammar for $\overline{L}$ is $G = (\{S, S_1, S_2, B\}, \{a, b\}, S, P)$ with the productions
$$\begin{aligned} S &\to aS_1B, \\ S_1 &\to aS_2B, \\ S_2 &\to aS_2B|b, \\ B &\to b. \end{aligned}$$
□

8. Construct an unambiguous grammar equivalent to the following grammar.
$$\begin{aligned} S &\to AB|aaB, \\ A &\to a|Aa, \\ B &\to b. \end{aligned}$$
**Answer.**
This grammar generates the strings $a^+b$. A desirable grammar is $G = (\{S, A\}, \{a, b\}, S, P)$ with the productions
$$\begin{aligned} S &\to aS|b, \\ A &\to aA|a. \end{aligned}$$
□

9. Show that the following grammar is ambiguous.
$$S \to aSbS|bSaS|\lambda.$$
**Answer.**
The string $w = abab$ has the following two derivation trees: □




10. Eliminate useless productions from

$$\begin{aligned} S &\to a|aA|B|C, \\ A &\to aB|\lambda, \\ B &\to Aa, \\ C &\to cCD, \\ D &\to ddd. \end{aligned}$$

**Answer.**

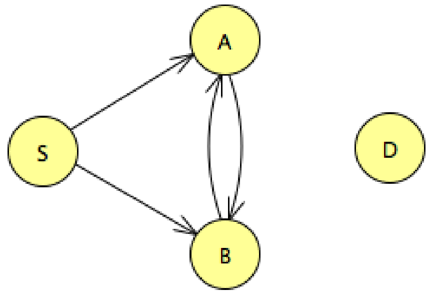
There are two cases for useless variables

- Case 1: Variables that cannot generate strings in $T^*$.
  - $V_1 = \{\}$, $(T \cup V_1)^* = \{a, b, c, d\}^*$;
  - Since $S \to a$, $A \to \lambda$, and $D \to ddd$, add $S$, $A$, and $D$ to $V_1$;
  - $V_1 = \{S, A, D\}$, $(T \cup V_1)^* = (\{a, b, c, d, S, A, D\})^*$;
  - Since $S \to aA$ and $B \to Aa$, add $S$ and $B$ to $V_1$;
  - $V_1 = \{S, A, B, D\}$, $(T \cup V_1)^* = (\{a, b, c, d, S, A, B, D\})^*$;
  - Since $S \to B$ and $A \to aB$, $B \to Aa$, the algorithm stops since there is no new rules can be added to $V_1$;

  Thus, we have $V_1 = \{S, A, B, D\}$. After removing the related useless productions, we have:

$$\begin{aligned} S &\to a|aA|B, \\ A &\to aB|\lambda, \\ B &\to Aa, \\ D &\to ddd. \end{aligned}$$

- Case 2: Variables that cannot be reached from $S$.
  - The dependency graph of the result grammar in Case 1 is as follows.




- $D$ is unreachable from $S$;

Thus, after removing the related useless productions, we have:

$$\begin{aligned} S &\to a|aA|B, \\ A &\to aB|\lambda, \\ B &\to Aa. \end{aligned}$$

□

11. Eliminate all $\lambda$-productions from

$$\begin{aligned} S &\to AaB|aBB, \\ A &\to \lambda, \\ B &\to bbA|\lambda. \end{aligned}$$

**Answer.**

A procedure of removing all $\lambda$-productions is as follows.

- Find the nullable variable set $V_N = \{A, B\}$;
- The $\lambda$-production $A \to \lambda$ can be removed after adding new productions obtained by substituting $\lambda$ for $A$ where it occurs on the right:

$$\begin{aligned} S &\to AaB|aBB|aB, \\ B &\to bbA|\lambda|bb; \end{aligned}$$

- The $\lambda$-production $B \to \lambda$ can be removed after adding new productions obtained by substituting $\lambda$ for $B$ where it occurs on the right:

$$\begin{aligned} S &\to AaB|Aa|aBB|aB|a, \\ B &\to bbA|bb; \end{aligned}$$

□

12. Eliminate all unit-productions in Question 10.
**Answer.**

From the dependency graph of the grammar, we add the new rules $S \to Aa|cCD$ to the non-unit productions

$$\begin{aligned} S &\to a|aA|Aa|cCD, \\ A &\to aB|\lambda, \\ B &\to Aa, \\ C &\to cCD, \\ D &\to ddd. \end{aligned}$$

□

13. Transform the grammar with production

$$\begin{aligned} S &\to abAB, \\ A &\to bAa|\lambda, \\ B &\to BAa|A|\lambda \end{aligned}$$

into Chomsky normal form.
**Answer.**

The transform procedure is as follows.

- Removing $\lambda$-productions:
  - Removing $A \to \lambda$: $S \to abAB|abB$, $A \to bAa|ba$, $B \to BAa|A|\lambda|Ba$.
  - Removing $B \to \lambda$: $S \to abAB|abB|abA|ab$, $A \to bAa|ba$, $B \to BAa|A|Ba|Aa|a$.
- Removing unit-production $B \to A$: $S \to abAB|abB|abA|ab$, $A \to bAa|ba$, $B \to BAa|bAa|ba|Ba|Aa|a$.
- Removing useless productions: No useless productions.
- Convert the grammar into Chomsky normal form:
  - Introduce new variables $S_x$ for each $x \in T$:

$$\begin{aligned} S &\to S_aS_bAB|S_aS_bB|S_aS_bA|S_aS_b, \\ A &\to S_bAS_a|S_bS_a, \\ B &\to BAS_a|S_bAS_a|BS_a|AS_a|a, \\ S_a &\to a, \\ S_b &\to b. \end{aligned}$$

  - Introduce additional variables to get the first two productions into normal form and we get the final result

$$
\begin{aligned}
S &\to S_aU|S_aX|S_aY|S_aS_b, \\
A &\to S_bW|S_bS_a, \\
B &\to BZ|S_bV|S_bB|S_bA|S_b|BS_a|AS_a|a, \\
U &\to S_bV, \\
V &\to AB, \\
X &\to S_bB, \\
Y &\to S_bA, \\
W &\to AS_a, \\
Z &\to AS_a, \\
S_a &\to a, \\
S_b &\to b.
\end{aligned}
$$

□

14. Convert the grammar with production

$$
\begin{aligned}
S &\to ABb|a, \\
A &\to aaA|B, \\
B &\to bAb|\lambda
\end{aligned}
$$

into Greibach normal form.

**Answer.**

We introduce new variables $X$ and $Y$:

$$
\begin{aligned}
S &\to ABX|Y, \\
A &\to YYA|B, \\
B &\to XAX|\lambda, \\
X &\to b, \\
Y &\to a
\end{aligned}
$$

Then, by using the substitution, we immediately get the equivalent grammar

$$
\begin{aligned}
S &\to aYABX|bAXBX|bAXX|b|a, \\
A &\to aYA|B, \\
B &\to bAX|\lambda, \\
X &\to b, \\
Y &\to a
\end{aligned}
$$

□

**Example 6.11**

Determine whether the string $w = aabbb$ is in the language generated by the grammar

$$\begin{aligned} S &\rightarrow AB, \\ A &\rightarrow BB|a, \\ B &\rightarrow AB|b. \end{aligned}$$

First note that $w_{11} = a$, so $V_{11}$ is the set of all variables that immediately derive a, that is, $V_{11} = \{A\}$. Since $w_{22} = a$, we also have $V_{22} = \{A\}$ and, similarly,

$$V_{11} = \{A\}, V_{22} = \{A\}, V_{33} = \{B\}, V_{44} = \{B\}, V_{55} = \{B\}.$$

Now we use (6.8) to get

$$V_{12} = \{A : A \rightarrow BC, B \in V_{11}, C \in V_{22}\}.$$

Since $V_{11} = \{A\}$ and $V_{22} = \{A\}$, the set consists of all variables that occur on the left side of a production whose right side is $AA$. Since there are none, $V12$ is empty. Next,

$$V_{23} = \{A : A \rightarrow BC, B \in V_{22}, C \in V_{33}\},$$

so the required right side is $AB$, and we have $V_{23} = \{S, B\}$. A straightforward argument along these lines then gives

$$\begin{aligned} V_{12} &= \varnothing, V_{23} = \{S, B\}, V_{34} = \{A\}, V_{45} = \{A\}, \\ V_{13} &= \{S, B\}, V_{24} = \{A\}, V_{35} = \{S, B\}, \\ V_{14} &= \{A\}, V_{25} = \{S, B\}, \\ V_{15} &= \{S, B\}, \end{aligned}$$

so that $w \in L(G)$.

15. Use the CYK algorithm to determine whether the strings $aabb$, $aabba$, and $abbbb$ are in the language generated by the grammar in Example 6.11.

**Answer.**

(1) For the string $aabb$:

Firstly, we have

$$V_{1,1} = \{A\}, V_{2,2} = \{A\}, V_{3,3} = \{B\}, V_{4,4} = \{B\}.$$

Then, by using the equation

$$V_{ij} = \bigcup_{k \in \{i, i+1, \ldots, j-1\}} \{A \to BC, \text{ with } B \in V_{ik}, C \in V_{k+1,j}\},$$

we have

$$\begin{aligned}
V_{1,2} &= \{A \to BC, B \in V_{1,1}, C \in V_{2,2}\} = \{\}, \\
V_{2,3} &= \{A \to BC, B \in V_{2,2}, C \in V_{3,3}\} = \{S, B\}, \\
V_{3,4} &= \{A \to BC, B \in V_{3,3}, C \in V_{4,4}\} = \{A\},
\end{aligned}$$

$$\begin{aligned}
V_{1,3} &= \{A \to BC, B \in V_{1,1}, C \in V_{2,3}\} \cup \{A \to BC, B \in V_{1,2}, C \in V_{3,3}\} = \{S, B\}, \\
V_{2,4} &= \{A \to BC, B \in V_{2,2}, C \in V_{3,4}\} \cup \{A \to BC, B \in V_{2,3}, C \in V_{4,4}\} = \{A\},
\end{aligned}$$

$$\begin{aligned}
V_{1,4} = &\{A \to BC, B \in V_{1,1}, C \in V_{2,4}\} \cup \{A \to BC, B \in V_{1,2}, C \in V_{3,4}\} \cup \\
&\{A \to BC, B \in V_{1,3}, C \in V_{4,4}\} = \{A\}.
\end{aligned}$$

Thus, we have

| | | | |
|---|---|---|---|
| $V_{1,4} = \{A\}$ | | | |
| $V_{1,3} = \{S, B\}$ | $V_{2,4} = \{A\}$ | | |
| $V_{1,2} = \{\}$ | $V_{2,3} = \{S, B\}$ | $V_{3,4} = \{A\}$ | |
| $V_{1,1} = \{A\}$ | $V_{2,2} = \{A\}$ | $V_{3,3} = \{B\}$ | $V_{4,4} = \{B\}$ |

Because $V_{1,4} = \{A\}$, $S \notin V_{1,4}$, we conclude that $aabb$ is not in the language generated by the given grammar, i.e., $aabb \notin L(G)$, by using the CYK algorithm.

(2) For the string $aabba$: Similarly, we have

| | | | | |
|---|---|---|---|---|
| $V_{1,5} = \{\}$ | | | | |
| $V_{1,4} = \{A\}$ | $V_{2,5} = \{\}$ | | | |
| $V_{1,3} = \{S, B\}$ | $V_{2,4} = \{A\}$ | $V_{3,5} = \{\}$ | | |
| $V_{1,2} = \{\}$ | $V_{2,3} = \{S, B\}$ | $V_{3,4} = \{A\}$ | $V_{4,5} = \{\}$ | |
| $V_{1,1} = \{A\}$ | $V_{2,2} = \{A\}$ | $V_{3,3} = \{B\}$ | $V_{4,4} = \{B\}$ | $V_{5,5} = \{A\}$ |

Because $V_{1,5} = \{\}$, we conclude that $aabba$ is not in the language generated by the given grammar, i.e., $aabba \notin L(G)$, by using the CYK algorithm.

(3) For the string $abbbb$: Similarly, we have

| | | | | |
|---|---|---|---|---|
| $V_{1,5} = \{A\}$ | | | | |
| $V_{1,4} = \{S, B\}$ | $V_{2,5} = \{A\}$ | | | |
| $V_{1,3} = \{A\}$ | $V_{2,4} = \{S, B\}$ | $V_{3,5} = \{S, B\}$ | | |
| $V_{1,2} = \{S, B\}$ | $V_{2,3} = \{A\}$ | $V_{3,4} = \{A\}$ | $V_{4,5} = \{A\}$ | |
| $V_{1,1} = \{A\}$ | $V_{2,2} = \{B\}$ | $V_{3,3} = \{B\}$ | $V_{4,4} = \{B\}$ | $V_{5,5} = \{B\}$ |

Because $V_{1,5} = \{A\}$, $S \notin V_{1,5}$, we conclude that $abbbb$ is not in the language generated by the given grammar, i.e., $abbbb \notin L(G)$, by using the CYK algorithm. $\square$