

CMSC330: CFG Practice Problem Solutions

Disclaimer: Please let the TAs know if you find any problem or have any questions regarding the solutions below.

1) $L = \{a^n b^n \mid n \text{ is even and } n \geq 0\}$

In words: A sequence of an even number of a 's followed by the same number of b 's.

2)

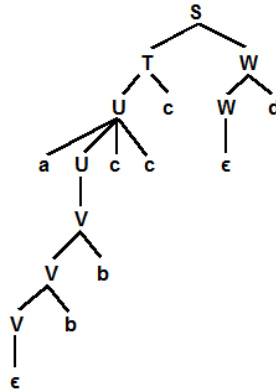
a) One correct grammar is:

$$\begin{aligned} S &\rightarrow TW \\ T &\rightarrow Uc \\ U &\rightarrow aUcc \mid V \\ V &\rightarrow Vb \mid \epsilon \\ W &\rightarrow Wd \mid \epsilon \end{aligned}$$

b) *Leftmost Derivation:* $S \rightarrow TW \rightarrow UcW \rightarrow aUcccW \rightarrow aVcccW \rightarrow aVbccW \rightarrow aVbbcccW \rightarrow abbcccW \rightarrow abbcccWd \rightarrow abbcccd$

Rightmost Derivation: $S \rightarrow TW \rightarrow TWd \rightarrow Td \rightarrow Ucd \rightarrow aUcccd \rightarrow aVcccd \rightarrow aVbcccd \rightarrow abbcccd$

c) The parse tree is:



3)

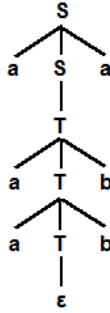
a) Rewrite the language as $L = \{a^{n-m} a^m b^m a^{n-m}\}$

$$\begin{aligned} S &\rightarrow aSa \mid T \\ T &\rightarrow aTb \mid \epsilon \end{aligned}$$

b) The left and rightmost derivations are identical because there is never more than one variable to replace.

Derivation: $S \rightarrow aSa \rightarrow aTa \rightarrow aaTba \rightarrow aaaTbba \rightarrow aaabba$

c) The parse tree is:



4)

a) **NOTE:** This exercise has been updated: The grammar does not have to be unambiguous.

A good strategy here is to make a separate production rule for each regular expression operator.

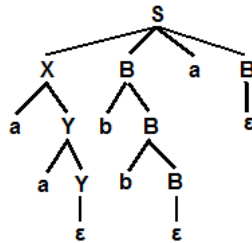
$$\begin{aligned}
 S &\rightarrow XBaB \\
 X &\rightarrow aY \mid bY \\
 Y &\rightarrow aY \mid bY \mid \epsilon \\
 B &\rightarrow bB \mid \epsilon
 \end{aligned}$$

b) The grammar in (a) is ambiguous. Therefore, the following derivations and parse trees are only one instance of several possibilities.

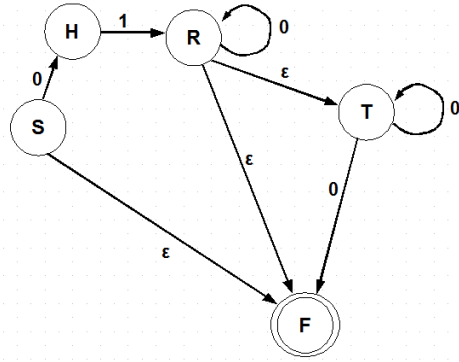
Leftmost Derivation: $S \rightarrow XBaB \rightarrow aYBaB \rightarrow aaYBaB \rightarrow aaaYBaB \rightarrow aaaBaB \rightarrow aaabBaB \rightarrow aaabbBaB \rightarrow aaabbaB \rightarrow aaabba$

Rightmost Derivation: $S \rightarrow XBaB \rightarrow XBa \rightarrow XbBa \rightarrow XbbBa \rightarrow Xbba \rightarrow aYbba \rightarrow aaYbba \rightarrow aaabba$

c) The parse tree is:



5) An NFA for the grammar is:



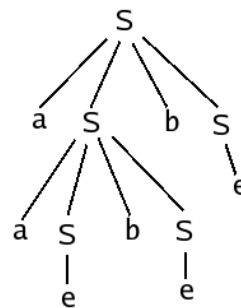
6) **NOTE:** this exercise and exercise 7 were updated: it is no longer required that the grammars be unambiguous.

a) $S \rightarrow aSbS \mid bSaS \mid \epsilon$

b) Let us give a leftmost derivation for $aabb$:

$$S \rightarrow aSbS \rightarrow aaSbSbS \rightarrow aabSbS \rightarrow aabbS \rightarrow aabb$$

The parse tree corresponding to the previous derivation is:



(the e 's above should be ϵ 's)

7) **NOTE:** this exercise and exercise 6 were updated: it is no longer required that the grammars be unambiguous.

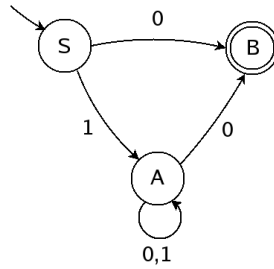
a) $S \rightarrow aSaSbS \mid aSbSaS \mid bSaSaS \mid \epsilon$

b) We can add an extra a to a string represented by the non-terminal S , to get the desired number of a 's and b 's. One way to do this is to add the following rule to the grammar defined above:

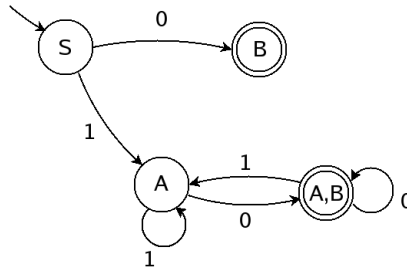
$$T \rightarrow SaS$$

8)

a) An NFA for the regular expression is:



We can then build the corresponding DFA:



b) A grammar corresponding to the regular expression is:

$$\begin{aligned} S &\rightarrow T0 \\ T &\rightarrow 1U \mid \epsilon \\ U &\rightarrow 1U \mid 0U \mid \epsilon \end{aligned}$$

c) All strings of the language end in 0. We can either have a string containing only the 0, or have it preceded by the $1(0|1)^*$. So, in case the string is not 0, we must have a 1 as initial symbol; then we're allowed to have anything else in between (represented by $(0|1)^*$). Examples of strings in this language are 0, 10, 100, 110, 1010, 1100, 1110, ... This can be thought of as the binary representations of even numbers.

9)

a) To help us in designing the grammar, we can re-write $a^m b^n a^{m+n}$ as $a^m b^n a^n a^m$. Remembering that we must have $m \geq 0$ and $n \geq 1$, a resulting grammar for the language is:

$$\begin{aligned} S &\rightarrow aSa \mid T \\ T &\rightarrow bTa \mid ba \end{aligned}$$

b) One way to think of this problem is to consider all possible terminal symbols for the first character read and then derive each corresponding rule. For example, if the first symbol read is an a , to balance out the expression (as to maintain $m + n = p + q$), we could require a c or d at the end of the string. Similar reasoning for the case in which b is the first symbol read provides us with the rule:

$$S \rightarrow aSd \mid aTc \mid bUd \mid bVc \mid \epsilon$$

Each non-terminal describes a different situation. For example, for T , we can still have both a 's and b 's in the beginning of the string it represents, but we can only have c 's at its end (no more d 's), since a c has been previously consumed at the end of the string. With this in mind, we come up with the remaining rules for the grammar:

$$\begin{aligned} T &\rightarrow aTc \mid bVc \mid \epsilon \\ U &\rightarrow bUd \mid bVc \mid \epsilon \\ V &\rightarrow bVc \mid \epsilon \end{aligned}$$

c) If we reason in the same fashion as in problem, 9)a), by considering all of the possible initial symbols of the string and the corresponding rules, we can start out with the rule S :

$$S \rightarrow aA \mid bS \mid \epsilon$$

Note that if we have a b as initial terminal symbol, the next symbol can be anything, therefore the derivation $S \rightarrow bS$ contained in the rule above. On the other hand, if our string is of the form aA , we can't have a b as the initial terminal in A (otherwise we would have an ab sequence), so:

$$A \rightarrow aA \mid \epsilon$$

d) $S \rightarrow aSa \mid bSb \mid aa \mid bb$

10) The union operation in the languages can be mapped to the OR (\mid) operation in the rules description. So let us describe $\{a^n b^n a^m b^m \mid m, n \geq 1\}$ using the non-terminal T , and $\{a^n b^m a^m b^n \mid m, n \geq 1\}$, using the non-terminal V as below:

$$\begin{aligned} S &\rightarrow T \mid V \\ T &\rightarrow UU \\ U &\rightarrow aUb \mid ab \\ V &\rightarrow aVb \mid aWb \\ W &\rightarrow bWa \mid ba \end{aligned}$$

11)

a) We can show that the grammar is ambiguous by finding a string that has two leftmost or rightmost derivations. Equivalently, we could also provide a string that has two different parse trees. Consider the string (ϵ) ; below are three different leftmost derivations for it using the provided grammar:

1. $S \rightarrow SS \rightarrow (S)S \rightarrow (\epsilon)S \rightarrow (\epsilon)\epsilon$

2. $S \rightarrow SS \rightarrow \epsilon S \rightarrow \epsilon(S) \rightarrow \epsilon(\epsilon)$

3. $S \rightarrow (S) \rightarrow (\epsilon)$

The ϵ 's are noted above only for clarity; usually they are simply omitted.

b) A nonambiguous grammar for balanced parentheses is

$$S \rightarrow (S)S \mid \epsilon$$