Selected solutions week 3 - IN2080

The following are solutions to some of the weekly tasks which we discuss in the group sessions.

2.6

b.

A string in the complement language of $L_b = \{a^n b^n | n \geq 0\}$ will contain an unequal amount of as and bs, or have them in an incorrect order , i.e. an a occurs after a b (or both, of course). It is natural then to split these two cases into two branches, essentially taking the union of two languages.

$$
\begin{aligned}
&\text{CFG}_{L_b}: \\
&\qquad S \to OR \mid EQ \\
&\qquad OR \to XbaX \\
&\qquad EQ \to aEQb \mid A \mid B \\
&\qquad X \to aX \mid bX \mid \varepsilon \\
&\qquad A \to aA \mid \varepsilon \\
&\qquad B \to bB \mid \varepsilon
\end{aligned}
$$

d.

This language is the language of strings over {a,b} separated by #. Additionally, for each string over {a,b} occurring in a sequence $w_1 \# w_2 ... \# w_n$, the reverse also occurs in the same sequence (at least that is my interpretation). Note that palindromes are its own reverse, and that w_k can be ε, for some ks.

The definition is a little ambiguous, but let us assume that the $k = 1$ case means a word $x \in \{a, b\}^*$ that is its own inverse (a palindrome) *not* followed by a #.

Then our language consists of words that are either:

1. a single palindrome (ε is a palindrome),

2. some w and its inverse w^R separated by another valid word between #s, or
3. several valid words concatenated.

We construct the grammar:

$$S \to P \mid R \mid RS \mid SR$$
$$P \to aPa \mid bPb \mid \varepsilon$$
$$R \to aRa \mid bRb \mid \#S\# \mid \#$$

Here P generates palindromes and R generates w and w^R separated by either # or a valid word between #s. At the top level, S generates either a single palindrome, a single par of $w\# \ldots \#w^R$, or a concatenation of valid words.

2.9

The following CFG generates the language.

$$S \to KC \mid AG$$
$$K \to aKb \mid \varepsilon$$
$$G \to bGc \mid \varepsilon$$
$$A \to aA \mid \varepsilon$$
$$C \to cC \mid \varepsilon$$

For a grammar to be ambiguous, there must be at least two different leftmost derivations for a single string. In this case, a string $a^n b^n c^n$ will satisfy this, as S can yield both KC and AG, and from there generate the same string.

2.18

$L(G)$ is the language $\{w_1 w_2 ... w_k \mid w_i \in \{a^n b^n | n \geq 1\}, k \geq 1\}$, that is, the language of strings made up of substrings on the form $a^n b^n$, where $n \geq 1$. The grammar gives the following leftmost derivations:
$S \Rightarrow SS \Rightarrow SSS \Rightarrow TSS \Rightarrow abSS \Rightarrow abTS \Rightarrow ababS \Rightarrow ababT \Rightarrow ababab$
and
$S \Rightarrow SS \Rightarrow TS \Rightarrow abS \Rightarrow abSS \Rightarrow abTS \Rightarrow \ ... \Rightarrow ababab$
and is thus ambiguous.

Let G' be the following grammar:

$$S \to TS \mid T$$
$$T \to aTb \mid ab$$

such that $L(G') = L(G)$ (we will not prove this, as the exercise does not ask us to).

To show that the grammar isn't ambiguous, we must show that no string s in $L(G')$ has two different leftmost derivations. We first make an observation: once we have a T in the leftmost position, the only option is to repeatedly apply the T-rules which will generate a word of the form $a^n b^n$ in place of the T.

Starting with S, there are two cases to consider:

1. If the derivation starts with $S \Rightarrow T \Rightarrow \ldots$, we have observed that the result is a word of the form $a^n b^n$ Since there is only ever one variable (T) to consider, this is the unique leftmost derivation.
2. If the derivation starts with $S \Rightarrow TS \Rightarrow \ldots$, then a leftmost derivation must first apply rules to the T to obtain a word of the form $a^n b^n S$. Then, we either continue $a^n b^n S \Rightarrow a^n b^n T \Rightarrow \ldots \Rightarrow a^n b^n a^m b^m$ — in which case we are finished with a unique leftmost derivation — or $a^n b^n S \Rightarrow a^n b^n TS \Rightarrow \ldots \Rightarrow a^n b^n a^m b^m S$ — in which case we are back to the beginning and can choose once again.

Recall that $L(G') = \{w_1 w_2 \ldots w_k \mid w_i \in \{a^n b^n \mid n \geq 1\}, k \geq 1\}$. The first case generates the words with $k = 1$ and repeated application of the second case ($k - 1$ times) generates the rest — in each case the leftmost derivation is unique. Intuitively, this grammar forces us to completely expand one T at the time, resulting in a unique leftmost derivation for each word in the language.