CS241 – Week 7 Tutorial Solutions

Languages: Context-Free Grammars

Spring 2017

Context-Free Grammar Problems

Derivations

Let G be the following grammar:

$$S \to aAb$$
$$S \to bAa$$
$$A \to aSa$$
$$A \to bSb$$
$$A \to c$$

1. Show that the word $abacbbb$ is in $L(G)$ by giving a derivation.

   Solution:

   $$S \Rightarrow aAb \Rightarrow abSbb \Rightarrow abaAbbb \Rightarrow abacbbb$$

2. Which of the following are valid derivations of words in $L(G)$?

   (a) $S \Rightarrow aAb \Rightarrow aaAab \Rightarrow aacab$

   (b) $S \Rightarrow aAb \Rightarrow aaAbb \Rightarrow aacbb$

   (c) $S \Rightarrow bAa \Rightarrow bca$

   (d) $A \Rightarrow aSa \Rightarrow aaAba \Rightarrow aacba$

   (e) $S \Rightarrow aaSab \Rightarrow aaaAbab \Rightarrow aaacbab$

   Solution:

   (a) is invalid since there is no $A \to aAa$ rule in G.

   (b) is invalid since there is no $A \to aAb$ rule in G.

   (c) is valid.

   (d) is invalid since A is not the start symbol of the grammar.

   (e) is invalid since there is no $S \to aaSab$ rule. However, it would be valid if we said $S \stackrel{*}{\Rightarrow} aaSab$, since $S \Rightarrow aAb \Rightarrow aaSab$.

Designing Grammars

1. Write context-free grammars for the following languages:

   (a) The language $L = \{\text{my}, \text{name}, \text{is}, \text{inigo}, \text{montoya}\}$.

   (b) The language of all words over $\Sigma = \{a, b, c\}$ not beginning in a b.

   (c) The language defined by the regular expression $(0|1)^*((00)^*1)^*010$.

   (d) The language $L = \{a^n b^n c^m d^m : n, m \in \mathbb{N}\}$, where a^n means "n copies of a in a row."

   (e) The language of palindromes over $\Sigma = \{a, b\}$. A *palindrome* is a word or phrase which is spelled the same forwards and backwards, for example "ABBA," "racecar," or "Able was I ere I saw Elba."

   Solution:

   (a) This is just a union, which is very easy to enforce in a grammar.

   $$S \to my$$
   $$S \to name$$
   $$S \to is$$
   $$S \to inigo$$
   $$S \to montoya$$

   (b) We need to be careful here: ϵ does not begin with a b! Otherwise, we simply allow S to have an a or c followed by A, where A represents anything.

   $$S \to aA$$
   $$S \to cA$$
   $$S \to \epsilon$$
   $$A \to aA$$
   $$A \to bA$$
   $$A \to cA$$
   $$A \to \epsilon$$

   (c) It turns out every regular expression can be converted via a simple algorithm to a regular expression. We can do that to get a grammar that looks fairly similar to this one, but it may help to try to encode $a^*, aa, a|b$, and a as CFGs and then think about how you might combine them.

   $$S \to AB010$$
   $$A \to 0A$$
   $$A \to 1A$$
   $$A \to \epsilon$$
   $$B \to C1B$$
   $$B \to \epsilon$$
   $$C \to 00C$$
   $$C \to \epsilon$$

(d) This is simply two copies of the "matching parentheses" grammar, where ab and cd are our parentheses in the two parts of the problem. We can simply write down those two grammars then combine them with $S \to AB$.

$$S \to AB$$
$$A \to aAb$$
$$A \to \epsilon$$
$$B \to cBd$$
$$B \to \epsilon$$

(e) Even-length palindromes aren't too bad: we just need to enforce that we add an a or b to both the start and end at the same time. However, we need to be careful: ϵ is a palindrome, and some (odd-length) palindromes have a single copy of a or b in the exact centre. Combining these ideas we get:

$$S \to aSa$$
$$S \to bSb$$
$$S \to a$$
$$S \to b$$
$$S \to \epsilon$$

2. Argue that every regular language is context-free by describing how to convert any regular expression into a context-free grammar.

Solution:
Recall that a regular expression takes one of four forms. We will present a grammar which handles each case: then by induction on the complexity of the regular expression we see that any regular expression can be encoded in a CFG, and thus every regular language is context-free.

- $R = a$, a single symbol. The corresponding production is simply $A \to a$.
- $R = R_1 R_2$, the concatenation of two regular expressions. The corresponding production is $A \to A_1 A_2$, where A_1, A_2 are the rules corresponding to R_1, R_2.
- $R = R_1 | R_2$, the union of two regular expressions. This has two corresponding productions: $A \to A_1$ and $A \to A_2$.
- $R = R_1^*$, the Kleene closure of a regular expression. Once again, we need two productions: $A \to A_1 A$ and $A \to \epsilon$.

Since all regular expressions take one of these four forms and every regular language can be encoded by a regular expression, every regular language is context-free.

Parse Tree Problems

Let G be the following grammar:

$$\begin{array}{ll} S \to AB & S \to CD \\ A \to aAb & A \to \epsilon \\ B \to cB & B \to \epsilon \\ C \to aC & C \to \epsilon \\ D \to bDc & D \to \epsilon \end{array}$$

1. Find parse trees for the following words:

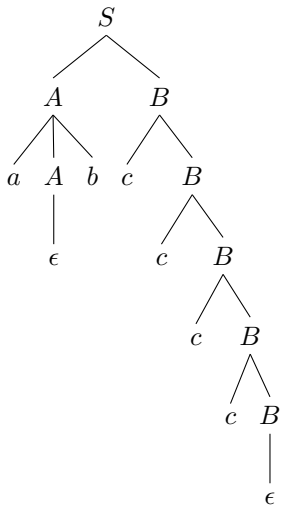
   (a) $abcccc$

   (b) $aabc$

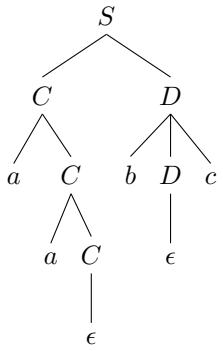
   (c) $bbbbcccc$

   Solution:
   For all of these, we simply need to find a derivation, after which the parse tree comes fairly naturally. Note that in the $S \to AB$ path we enforce that there are the same number of as and bs, but any number of cs, whereas in the $S \to CD$ path we enforce that there are the same number of bs and cs, but any number of as. As a result, we simply need to pick the appropriate starting point given the number of as, bs, and cs in the word, and after that there is only one choice at every step.

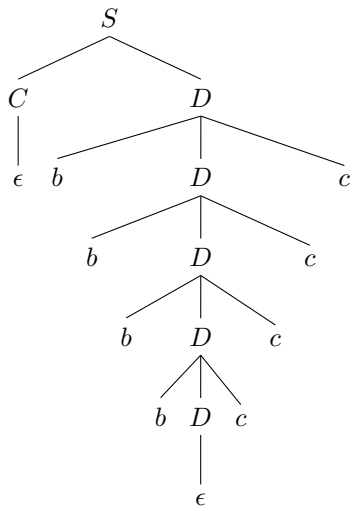
   (a)



   (b)



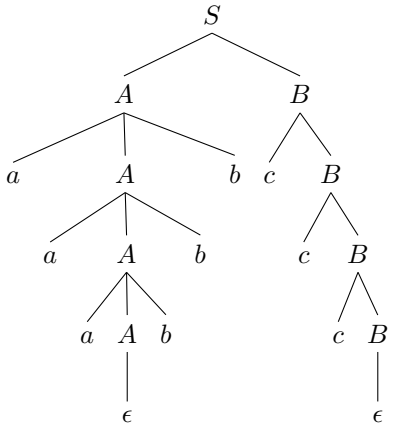
(c)



2. Prove that G is ambiguous by finding two parse trees for the word *aaabbbccc*.

Solution:
The key observation here is that $L(G) = \{a^i b^j c^k : i = j \text{ or } j = k\}$, and *aaabbbccc* has $i = j = k$, so we can choose to enforce either the $i = j$ or the $j = k$ conditions in the grammar.

- Enforcing $i = j$ we get:



- Enforcing $j = k$ we get: