

## Discussion #2: Context-Free Languages

### 1 Disassembling Grammars

Describe the following context-free grammars in English.

1.

$$S \rightarrow aSb \mid SS \mid \epsilon$$

The language of balanced a's and b's (the empty string, which is included in this set, is by definition balanced). If you substitute a with "(" and b with ")", this would be the language of balanced parenthesis.

2.

$$\begin{aligned} R &\rightarrow XRX \mid S \\ S &\rightarrow aTb \mid bTa \\ T &\rightarrow XTX \mid X \mid \epsilon \\ X &\rightarrow a \mid b \end{aligned}$$

The language of strings with a centered substring that starts with  $a$  and ends in  $b$ , or starts with  $b$  and ends in  $a$ . By centered I mean the string has an equal and non-zero number of characters before and after the substring.

*If you have a more accurate/concise description, please let me know.*

### 2 Ambiguous Grammars

Show that the following grammars are ambiguous. In other words, show that there are multiple parse trees for some string generated by the grammar. Then rewrite the grammar to be unambiguous. In other words, rewrite so that *no* string generated by the grammar has multiple parse trees.

1.

$$E \rightarrow E + E \mid E * E \mid (E) \mid int$$

There are two parse trees for the string  $int * int + int$ :  $(int * int) + int$  and  $int * (int + int)$ . Another string that results in multiple parse trees is  $int + int + int$ :  $(int + int) + int$  and  $int + (int + int)$ .

All ambiguities can be eliminated by rewriting as follows:

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * int \mid int \mid (E) \end{aligned}$$

Observe that this grammar enforces precedence of  $*$  over  $+$  and left-associativity of  $+$  and  $*$ .

2.

$$E \rightarrow -E \mid E - E \mid int$$

There are two parse trees for the string  $int - int - int$ :  $(int - int) - int$  and  $int - (int - int)$ . There are also two parse trees for the string  $int - -int - int$ :  $int - (-int) - int$  and  $int - -(int - int)$ .

All ambiguities can be eliminated by rewriting as follows:

$$\begin{aligned} E &\rightarrow E - T \mid T \\ T &\rightarrow int \mid -int \end{aligned}$$

Observe that this grammar binds the  $-$  to the integer immediately following it, thereby eliminating *one* of the ambiguities. The other ambiguity, that of associativity, is eliminated by forcing left associativity with the use of a new production  $T$ .

3.

$$\begin{aligned} \text{Stmt} &\rightarrow \text{Assign} \mid \text{If-then} \mid \text{If-then-else} \mid \text{Begin-end} \\ \text{If-then} &\rightarrow \text{if condition then Stmt} \\ \text{If-then-else} &\rightarrow \text{if condition then Stmt else Stmt} \\ \text{Begin-end} &\rightarrow \text{begin Stmt-list end} \\ \text{Stmt-list} &\rightarrow \text{Stmt-list Stmt} \mid \text{Stmt} \\ \text{Assign} &\rightarrow a = 1 \end{aligned}$$

There are two parse-trees for the string “if condition then if condition then a = 1 else a = 1”. The “else” can be associated with the first if statement or the second if statement. This is the dangling-else problem.

To disambiguate, force the “else” to bind to one of the if statements. Binding to the closest unmatched if statement seems most natural (although this choice is arbitrary), so one way to rewrite the grammar is as follows:

$$\begin{aligned} \text{Stmt} &\rightarrow \text{Assign} \mid \text{MIF} \mid \text{UIF} \mid \text{Begin-end} \\ \text{MIF} &\rightarrow \text{if condition then MIF else MIF} \mid \text{Stmt} \\ \text{UIF} &\rightarrow \text{if condition then Stmt} \mid \text{if condition then MIF else UIF} \\ \text{If-then-else} &\rightarrow \text{if condition then Stmt else Stmt} \\ \text{Begin-end} &\rightarrow \text{begin Stmt-list end} \\ \text{Stmt-list} &\rightarrow \text{Stmt-list Stmt} \mid \text{Stmt} \\ \text{Assign} &\rightarrow a = 1 \end{aligned}$$

### 3 Bonus

Provide an unambiguous grammar for this language.

$$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and either } i = j \text{ or } j = k\}$$

This language is inherently ambiguous. That is, this language can be generated only by ambiguous grammars. This was a trick question to help you think hard about the grammar disambiguation process. Can you prove that it is inherently ambiguous?