Lab Four - Analyzing Grammars

Murray Coueslant

murray.coueslant1@marist.edu

October 17, 2019

1 The Dragon Book

1.1 Problem 4.4.3

Problem Compute FIRST and FOLLOW for the following grammar
Grammar:

1. S -> S S +
2. | S S *
3. | a

Solution

Production	First	Follow
S	{S, a}	{S, +, *}

2 Crafting A Compiler

2.1 Problem 4.9

Problem Compute First and Follow sets for the nonterminals of the following grammar.

1. S -> a S e
2. | B
3. B -> b B e
4. | C
5. C -> c C e
6. | d

Solution

Production	First	Follow
S	{a, b, c, d}	{ }
B	{b, c, d}	{ }
C	{c, d}	{ }

2.2 Problem 5.10

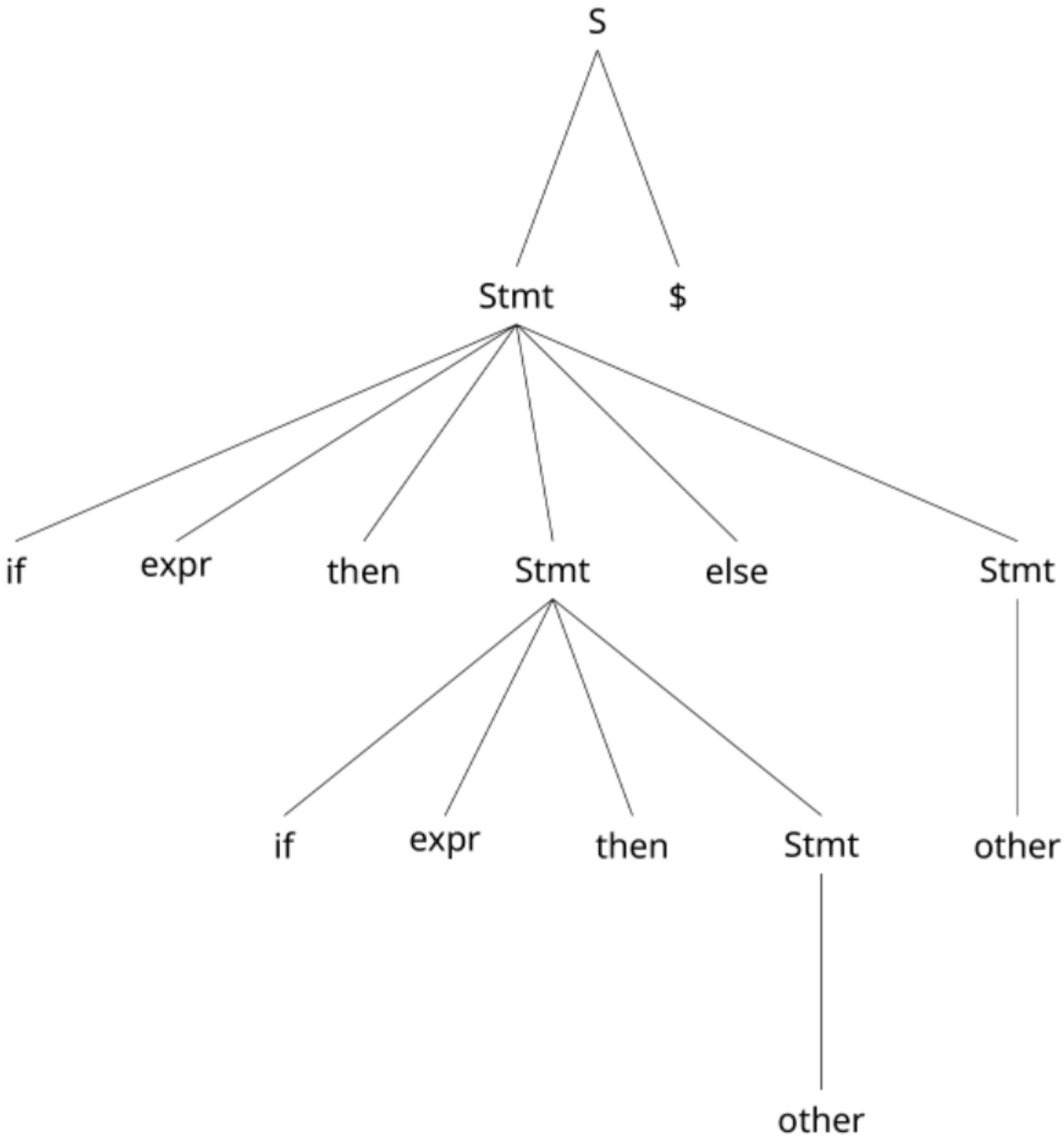
Problem Show the two distinct parse trees that can be constructed for:
if expr then if expr then other else other$
using the grammar:

1. S -> Stmt $
2. Stmt -> if expr then Stmt else Stmt
3. | if expr then Stmt
4. | other

For each parse tree, explain the correspondence of then and else.

Solution

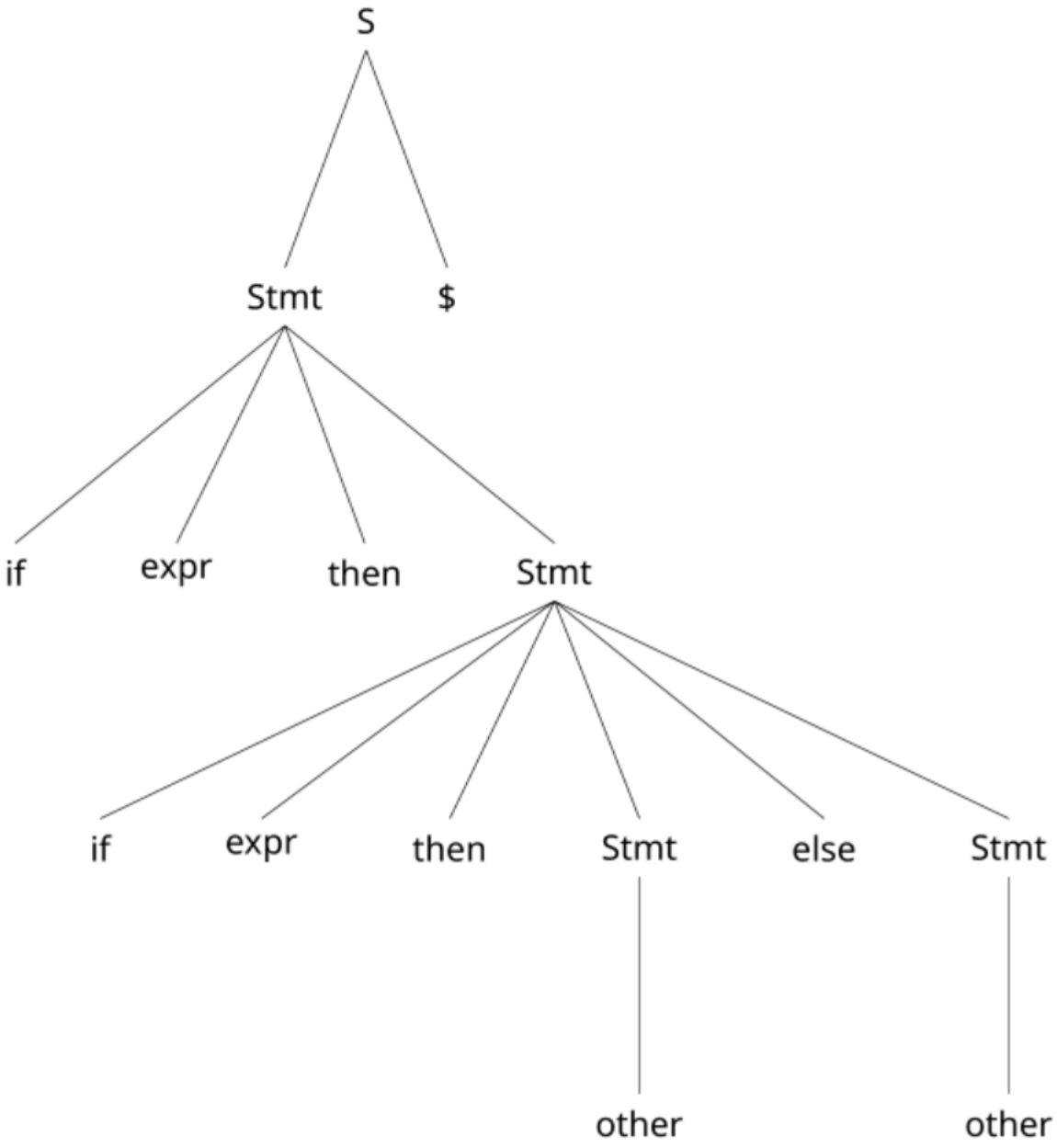
1. Tree One



a)

b) In this case, the final dangling 'else' is attributed to the first then encountered in the input stream. The ambiguity introduced by the final else clause appears in this grammar because it is not strongly bound to a specific then clause. This means that there exists more than one way to construct a valid parse tree for a single input stream. The final else being attributed to the first then encountered is one way to handle it.

2. Tree Two



a)

b) In the case of the second tree, the final else is handled the opposite way. The final dangling else is attributed to the last occurrence of then in the input stream. This causes the middle two levels of the tree to be largely flipped from the first valid parse tree.