CS 350: Prog. Lang. Design

Parse trees are a hierarchical depiction of a derivation

Every internal node of a parse tree is labeled with a non-terminal

Every leaf will be labeled with a terminal

Every subtree defines one instance of an abstraction within the derivation

A grammar is ambiguous if and only if it generates a sentential form that has two or more distinct parse trees

If a language has more than one permutation for parse trees then the meaning of that tree can’t be determined uniquely

Reasons for this are operator precedence and associativity, can we re-arrange the X + Y in a parse tree to be Y + X? Do we need to re-order or can we re-order our parse tree?

Typically and ambiguous grammar can be corrected to be clearer.

* Add new non-terminals, rules, etc. All to represent operands and force different operators into different levels within our parse tree

Commutativity – X + Y = Y + X

Associative – 2 \* 3 \* X = (2x \* 3x) = (6 \* 2x) = (2 \* 3) \* x = 2 \* (3 \* x)

The key to these principles is that the result be the same despite moving the operands around

To conquer dangling-else, add brackets as terminals to the language