Ambiguity in CFGs

- A CFG is said to be *ambiguous* if there exists a string which has more than one left-most derivation

Example:

S ==> AS | ε

A ==> A1 | 0A1 | 01

LM derivation #1:

S => AS
=> 0A1S
=>0A11S
=> 00111S
=> 00111

LM derivation #2:

S => AS
=> A1S
=> 0A11S
=> 00111S
=> 00111

Input string: 00111

Can be derived in two ways

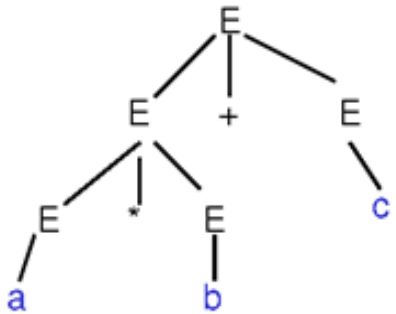
Why does ambiguity matter?

E ==> E + E | E * E | (E) | a | b | c | 0 | 1

Values are different !!!

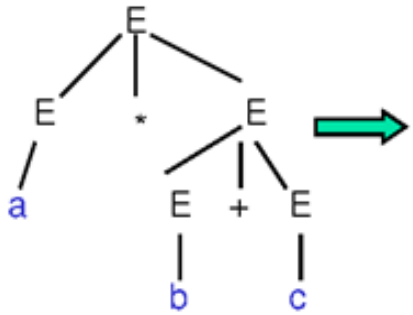
*string = a * b + c*

- LM derivation #1:
  - E => E + E => E * E + E
    ==>* a * b + c



(a*b)+c

- LM derivation #2
  - E => E * E => a * E =>
    a * E + E ==>* a * b + c



a*(b+c)

The calculated value depends on which of the two parse trees is actually used.

Removing Ambiguity in Expression Evaluations

- It MAY be possible to remove ambiguity for some CFLs
  - E.g.,, in a CFG for expression evaluation by imposing rules & restrictions such as precedence
  - This would imply rewrite of the grammar
- Precedence: (), * , +

Modified unambiguous version:

E => E + T | T
T => T * F | F
F => I | (E)
I => a | b | c | 0 | 1

How will this avoid ambiguity?

Ambiguous version:

E ==> E + E | E * E | (E) | a | b | c | 0 | 1

Inherently Ambiguous CFLs

- However, for some languages, it may not be possible to remove ambiguity

- A CFL is said to be *inherently ambiguous* if every CFG that describes it is ambiguous

Example:

- L = { $a^n b^n c^m d^m$ | n,m≥ 1} U {$a^n b^m c^m d^n$ | n,m≥ 1}
- L is inherently ambiguous
- Why?

Input string: $a^n b^n c^n d^n$