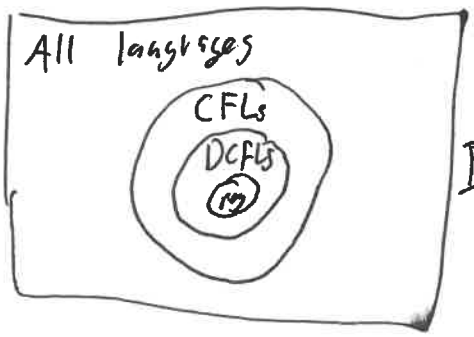
A. Exam Solutions
B. Example
C. Ambiguity
D. Fixing the example

B Real-world use of CFLs: Parsing source code
Use a restricted class of languages: Deterministic CFLs

Equiv. to DPDAs.



All inclusions are strict.
I.E. CFLs ≠ DCFLs

Consider expressions consisting of numbers, +, *, and ()

E.g. 5 + 10 * 2

Lets build a CFG for this, starting with numbers and then build up to expressions

CFG for numbers

$N \to D \mid DN$

$D \to 0 \mid 1 \mid \cdots \mid 9$

Expressions:

$E \to N \mid E+E \mid E*E \mid (E)$

Parse tree for $5 + 9 * 2$



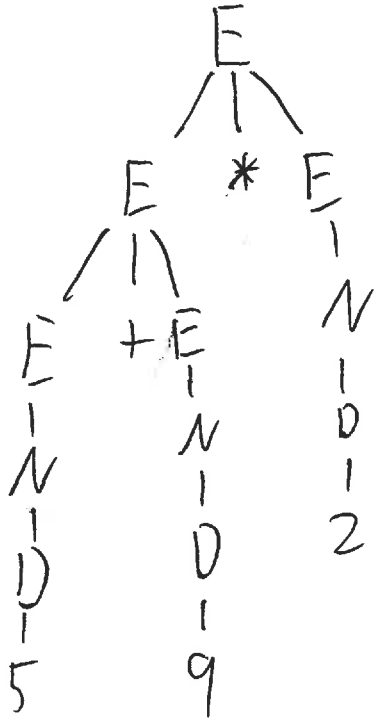
$E \Rightarrow E+E$
$\Rightarrow N+E$
$\Rightarrow D+E$
$\Rightarrow 5+E$
$\Rightarrow 5+E*E$
$\Rightarrow 5+N*E$
$\Rightarrow 5+D*E$
$\Rightarrow 5+9*E$
$\Rightarrow 5+9*2$

$\overset{*}{\Rightarrow}$ (i.e., multiple steps)

Imagine evaluating this expression by recursively evaluating children

Interpretation: $5 + (9 * 2)$ ✓

Not the only parse tree!



$E \Rightarrow E*E$
$\Rightarrow E+E*E$
$\Rightarrow N+E*E$
$\Rightarrow 5+E*E$
$\Rightarrow 5+N*E$
$\Rightarrow 5+9*E$
$\Rightarrow 5+9*N$
$\Rightarrow 5+9*2$

Interpretation: $(5+9)*2$ ✗

The parse tree encodes information about the expression

It would be great if there were only one parse tree (the first one!)

C Def. A CFG G is <u>ambiguous</u> if there exists $x \in L(G)$ s.t. x has two (or more) parse trees.

Fun fact: Parse trees correspond 1-1 with left-most derivations.

Consequence: A CFG is ambiguous if there exists $x \in L(G)$ s.t. x has two left-most derivations

Less fun fact: Some CFLs are <u>inherently ambiguous</u> meaning every CFG is ambiguous.

D Good news! The language of expressions is not inherently ambiguous.

We just need a better CFG.

Introduce terms and factors

$E \to E + T \mid T$

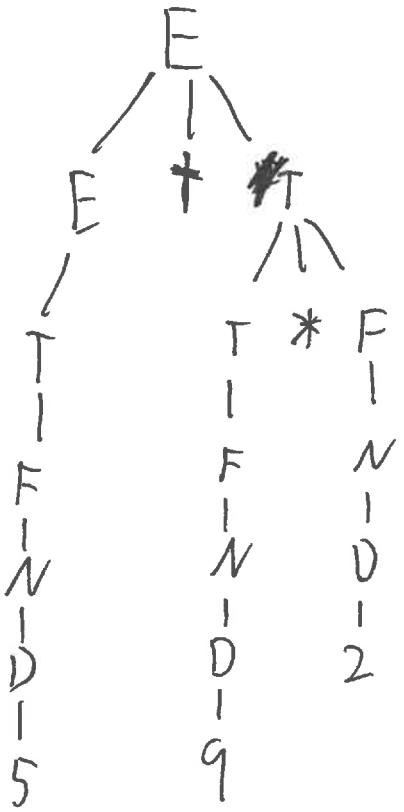
$T \to T * F \mid F$

$F \to (E) \mid N$

$N \to D \mid DN$

$D \to 0 \mid \cdots \mid 9$

Parse tree for $5 + 9 * 2$



This is the only parse tree