CS 444 - Compiler Construction **Winter 2020**

Lecture 4: January 15th, 2020

Lecturer: Ondřej Lhoták *Notes By: Harsh Mistry*

4.1 Analysis Continued

4.1.1 Parsing

- The ultimate goal is to produce a tree data structure which represents the input
- Regular expressions can express infinite languages, but not infinite levels of nesting.Therefore for parsing we use context-free languages represented by a context-free grammar

Definition 4.1 *Context Free Grammar (CFG) is a 4-tuple* $G =< N, T, R, S >$

- T - *Terminals*
  - *Words that make up the actual language*
- N - *Non-Terminals*
  - *Internal to the grammar*
- R - *Production Rules*
- S - *Start non-terminal*

- Notations in this course
  - Terminals - $a, b, c \in T$
  - Non-terminals - $A, B, C \in N$
  - Start - S
  - Symbols - $V = T \cup N \qquad X, Y, Z \in T \cup N$
  - Strings of terminals T^* - $w, x, y, z \in T^*$
  - Strings of symbols $(T \cup N)^*$ - $\alpha, \beta, \gamma, \delta \in (T \cup N)^*$
  - Production Rules - $A \to \alpha$

Definition 4.2 *"Directly Derives"* $BA\gamma \to P\alpha\gamma$ *if* $A \to a \in E$

Definition 4.3 *"Derives"* $\alpha \to^* \beta$ *if* $\alpha = \beta$ *or* $\alpha \to \gamma$ *and* $\gamma \to \beta$

Definition 4.4 $L(G) = \{x \in T^* \mid S \to *x\}$

Definition 4.5 α *is a sentential form if* $S \to^* \alpha$
$L(G) = sentential - forms(G) \cap T^*$

Definition 4.6 *Recognition is $x \in L(G)$*

- Parsing is finding a derivation for x

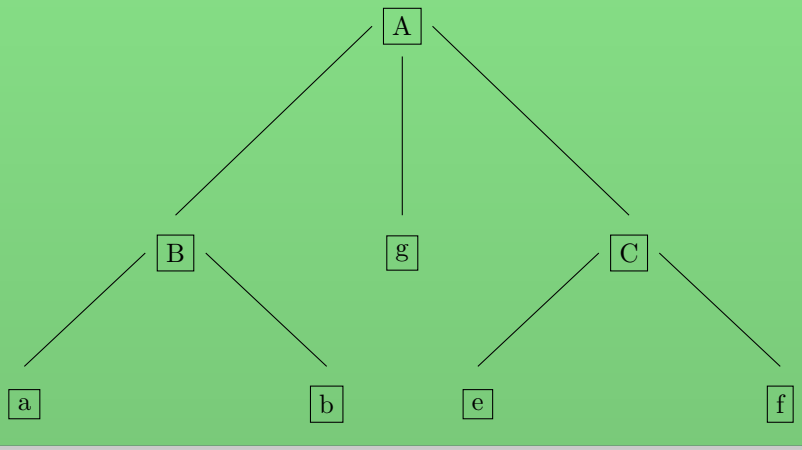
Example

Grammar:

- $A \to BgC$
- $C \to ef$
- $B \to ab$

Derivation:

- Right Canonical : $A \to BgC \to Bgef \to abgef$
- Left Canonical: $A \to BgC \to abgC \to ef$

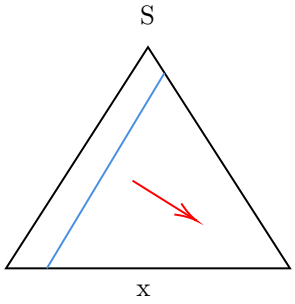


Definition 4.7 *A grammar is Ambiguous if some input has more than 1 parse tree*

4.1.1.1 Top-Down Parsing

LL(k) LL(1)

- Start with $\alpha = S$ and input string x
- Repeat till $\alpha = x$
  - Replace some non-terminal A in α with β where $A \to B \in R$



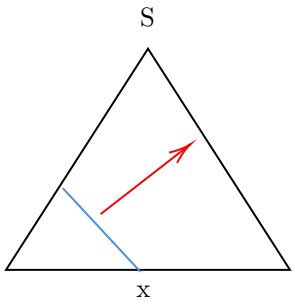
Note

$LL(k)$ grammars cannot generate left-associative trees

4.1.1.2 Bottom-Up Parsing

LR(k) LR(1)

- Start with $\alpha = x$
- Repeat till $\alpha = S$
  - Replace some "handle" β in α with A where $A \to B \in R$



Note

$LR(k)$ grammars can generate both left-associative and right-associative trees