COMS W3261
Computer Science Theory
Lecture 7
Context-Free Grammars

Alexander Roth

2014 – 09 – 24

Outline

- Review
- Definition of a context-free grammar
- Derivations
- Leftmost and rightmost derivations
- Parse trees
- Ambiguity

1 Overview

- We now begin the study of context-free languages, the next family of languages in the Chomsky hierarchy that properly includes the class of regular languages.
- A context-free language is defined by a context-free grammar, a formalism that generates the strings in the language of the grammar.
- Context-free grammars are a key formalism for describing the syntactic structure of programming languages. They are also useful in the study of natural languages.
- In this lecture we define what a context-free grammar is and show how it defines a language.

2 Definition of a Context-Free Grammar (CFG)

- A CFG is a formalism for defining a language.
- A CFG has four components (V, T, P, S) where
  - V is a finite set of variables called nonterminals, sometimes called syntactic categories. Each variable represents a language.
  - T is a finite set of symbols called terminals. The set of terminals is the alphabet of the language defined by the grammar.
  - P is a finite set of productions, rewrite rules of the $A \rightarrow \alpha$ where A is a nonterminal and α is a string (possibly empty) of nonterminals and terminals.
  - S is a nonterminal, called the start symbol.
- Example grammar G1:
  1. $V = \{s\}$
  2. $T = \{(\,,)\}$
  3. P is the set with two productions

$$s \rightarrow s\,(\,s\,)$$

$$s \rightarrow \epsilon$$

     We shall see that G1 generates the language consisting of all strings of balanced parentheses.

Class Notes

Example of a CFG:
`<sentence>` $\rightarrow$ `<noun phrase><verb phrase>`
`<noun phrase>` $\rightarrow$ girl
`<noun phrase>` $\rightarrow$ cat
`<verb phrase>` $\rightarrow$ `<verb><noun phrase>`
`<verb>` $\rightarrow$ likes

1. V is the finite set of variables called nonterminals.
2. T is the alphabet for strings. Can be called a sentence.
3. P is the finite set of rewrite rules

3 Derivations

- A grammar is used to define a language.
- Example of a derivation of ( )( ) from S in G1:

$$
\begin{aligned}
s &\Longrightarrow s\,(\,s\,) \\
&\Longrightarrow s\,(\,s\,)\,(\,s\,) \\
&\Longrightarrow (\,s\,)\,(\,s\,) \\
&\Longrightarrow (\,)\,(\,s\,) \\
&\Longrightarrow (\,)\,(\,)
\end{aligned}
$$

- This derivation show that ( )( ) is string in the language defined by G1. In each step of the derivation a nonterminal symbol s in a sentential form is replaced by the string on the right hand side of a production that has s on the left hand side.
- $L(G)$, the set of all strings of terminals that can be derived from the start symbol of a grammar G, is the language defined by G.
- We often call a string in $L(G)$ a sentence of $L(G)$.
- A string of terminals and nonterminals that can be derived from the start symbol of a grammar is called a sentential form.

Class Notes

`<sentence>`
$\rightarrow$ cat `<verb phrase>`
$\rightarrow$ cat `<verb>` `<noun phrase>`
$\rightarrow$ cat likes `<noun phrase>`
$\rightarrow$ cat likes girl
Example:
$G: \quad S \rightarrow aSa\,|bSb\,|\epsilon$
$S \rightarrow$ aSa
$\rightarrow$ ab$S$ba
$\rightarrow$ abb$S$bba
$\rightarrow$ abb bba

4 Leftmost and Rightmost Derivations

- A derivation in which at each step we replace the leftmost nonterminal by one of its production bodies is called a leftmost derivation.

  - The derivation above is a leftmost derivation of ( )( ) from s in $G1$.

- A rightmost derivation is one in which at each step we replace the rightmost nonterminal by one of its production bodies.

  - Here is a rightmost derivation of ( )( ) from s in G1:

$$
\begin{aligned}
S &\Longrightarrow s\,(\,s\,) \\
&\Longrightarrow s\,(\,) \\
&\Longrightarrow s\,(\,s\,)\,(\,) \\
&\Longrightarrow s\,(\,)\,(\,) \\
&\Longrightarrow (\,)\,(\,)
\end{aligned}
$$

Class Notes

Every parse tree has a unique leftmost and rightmost derivation.

5 Parse Trees

- A derivation can be represented by a parse tree.

- Let $G = (V, T, P, S)$ be a CFG. A parse tree for G is a tree in which:

  - Each interior node is labeled by a nonterminal in V.
  - Each leaf is labeled by a nonterminal, or a terminal, or ϵ
  - If an interior node is labeled by a nonterminal A and its children are labeled $X_1, X_2, \ldots, X_k$, then $A \to X_1 X_2 \ldots X_k$ is a production in P.

- The *yield* of a parse tree is the string obtained by concatenating the labels of the leaves from the left.

- Derivations, parse trees, leftmost derivations, rightmost derivations, and recursive inference are equivalent.

- A parser for a grammar G is a program that takes as input a string and produces as output a parse tree for the string or a message saying that the string cannot be generated by G.

- A parser generator is a program that takes as input a grammar G and produces as output a parser for G. YACC is a widely used parser generator.

6 Ambiguity

- A grammar G is ambiguous if there is a sentence in $L(G)$ with two or more distinct parse trees.

- The following grammar $G2$ for arithmetic expressions is ambiguous because $a + a * a$ has two parse trees.

$$E \rightarrow E + E | E * E | (E) | a$$

- We can remove the ambiguity by specifying the associativity and precedence of the $+$ and $*$.

- The grammar $G3$ below is unambiguous and makes $*$ have higher precedence than $+$ and makes both $*$ and $+$ left associative.

$$\begin{aligned} E &\rightarrow E + T \quad | \, T \\ T &\rightarrow T * F \quad | \, F \\ F &\rightarrow (E) \quad | \, a \end{aligned}$$

- A context-free language L is inherently ambiguous if it cannot be generated by an unambiguous grammar.