Lecture 10
Ambiguity
Instructor: Hossein Hojjat
February 13, 2017
• Context-free grammar is a 4-tuple \( G = (T, N, S, R) \)

• Parse trees are trees where
  - root is labeled with the start symbol \( S \)
  - internal nodes are labeled with symbols \( \in N \)
  - leaf nodes are labeled with symbols \( \in T \cup \{\epsilon\} \)
  - if \( v \) is a node with label \( X \) and its child nodes \( v_1, \cdots, v_n \) are labeled with \( X_1, \cdots, X_n \) then
    \[ X \rightarrow X_1 \cdots X_n \] is a production rule \( \in R \)

**Example.**

Grammar: \( G = (\{(\), \}\), \{S\}, S, R) \) where

\[
R = \left\{ S \rightarrow SS \mid (S) \mid () \right\}
\]

Derivation:

\[
S \Rightarrow SS \Rightarrow (S)S \Rightarrow (())S \Rightarrow (())()
\]
With this grammar there is a choice of variables to expand

Sample derivation:

\[ S \Rightarrow SS \Rightarrow SSS \Rightarrow S(S)S \Rightarrow S()() \Rightarrow ()()() \]

Leftmost derivation: always expand the leftmost variable first

\[ S \Rightarrow SS \Rightarrow SSS \Rightarrow ()SS \Rightarrow ()()S \Rightarrow ()()() \]

Rightmost derivation: always expand the rightmost variable first

\[ S \Rightarrow SS \Rightarrow SSS \Rightarrow SS() \Rightarrow S()() \Rightarrow ()()() \]
Ambiguous Grammars

- Ambiguous CFG:
  there is a word in the language that has two or more parse trees
- Example:

  $$S \rightarrow SS \mid (S) \mid ()$$

  Two parse trees for \((())()()\)
Ambiguity, Left- and Rightmost Derivations

- To show that a grammar is ambiguous:
  1) Give two different parse trees for a word, or
  2) Give two different leftmost derivations for a word, or
  3) Give two different rightmost derivations for a word

- One leftmost and one rightmost derivation for a word is not sufficient
- Leftmost and rightmost derivations might correspond to the same parse tree
Ambiguity is Bad

- Sometimes ambiguity in grammar can leave meaning of some programs ill-defined

Example: \[ <\text{cmd}> ::= \text{if } <\text{bool}> \text{ then } <\text{cmd}> \]
\[ \mid \text{if } <\text{bool}> \text{ then } <\text{cmd}> \text{ else } <\text{cmd}> \]

- Do not know if else clause is paired with the outermost or with the innermost then

\[
\text{if } (x > 0) \text{ then } \\
\quad \text{if } (y > 0) \text{ then } \\
\quad \text{print}(1) \\
\text{else } \\
\text{print}(2)
\]
Ambiguity

- Ambiguity is a property of grammars not languages
- For the balanced parentheses language, here is another CFG which is unambiguous:

\[
B \rightarrow (RB \mid \epsilon \\
R \rightarrow ) \mid (RR)
\]

- Start symbol \( B \) generates balanced strings
- \( R \) generates strings that have one more right parentheses than left
Example: Unambiguous Grammar

\[ B \rightarrow (RB \mid \epsilon) \]
\[ R \rightarrow ) \mid (RR \]

- This grammar constructs a unique leftmost derivation for a given balanced string of parentheses
- When scanning the input string from left to right:
  - If we need to expand \( B \):
    - If the next symbol is ( then use \( B \rightarrow (RB \)
    - If it is at the end then use \( B \rightarrow \epsilon \)
  - If we need to expand \( R \):
    - If the next symbol is ) then use \( R \rightarrow ) \)
    - If the next symbol is ( then use \( R \rightarrow (RR \)
Theorem
The problem of deciding whether a given CFG is ambiguous is undecidable.

- Bad news: There is no general algorithm to remove ambiguity from a CFG.
- More bad news: Some CFL's have only ambiguous CFG's.
- **CFL** $L$ is inherently ambiguous if all grammars for $L$ are ambiguous.
- There are heuristics that can be used to remove ambiguity from a grammar.
Inherent Ambiguity

- $L = \{0^i1^j2^k \mid i = j \text{ or } j = k\}$
- Intuitively strings of the form $0^n1^n2^n$ can be generated by two different parse trees:
  - one based on checking the 0’s and 1’s,
  - the other based on checking the 1’s and 2’s
One Possible Ambiguous Grammar for $L = \{0^i1^j2^k \mid i = j \text{ or } j = k\}$

$S \rightarrow AB \mid CD$

$A \rightarrow 0A1 \mid 01$

$B \rightarrow 2B \mid 2$

$C \rightarrow 0C \mid 0$

$D \rightarrow 1D2 \mid 12$

- $A$ generates equal numbers 0’s and 1’s
- $B$ generates any number of 2’s
- $C$ generates any number of 0’s.
- $D$ generates equal numbers 1’s and 2’s
One Possible Ambiguous Grammar for

\[ L = \{0^i1^j2^k \mid i = j \text{ or } j = k\} \]

- \[ S \rightarrow AB \mid CD \]
- \[ A \rightarrow 0A1 \mid 01 \]
- \[ B \rightarrow 2B \mid 2 \]
- \[ C \rightarrow 0C \mid 0 \]
- \[ D \rightarrow 1D2 \mid 12 \]

- There are two derivations of every string with equal numbers of 0’s, 1’s and 2’s

\[ S \Rightarrow AB \Rightarrow 01B \Rightarrow 012 \]
\[ S \Rightarrow CD \Rightarrow 0D \Rightarrow 012 \]
Question
Show that the following grammar is ambiguous:

\[ A \rightarrow BC \]
\[ B \rightarrow 1B1 \mid 1 \]
\[ C \rightarrow 1C1 \mid \epsilon \]
Ambiguity Exercise

Question
Show that the following grammar is ambiguous:

\[
\begin{align*}
A & \rightarrow BC \\
B & \rightarrow 1B1 \mid 1 \\
C & \rightarrow 1C1 \mid \epsilon
\end{align*}
\]

Answer
Two different leftmost derivations for 111

- \[A \Rightarrow BC \Rightarrow 1C \Rightarrow 11C1 \Rightarrow 111\]
- \[A \Rightarrow BC \Rightarrow 1B1C \Rightarrow 111C \Rightarrow 111\]
Chomsky Normal Form

• Consider the grammar $G_\epsilon = (\emptyset, \{S\}, S, R)$ with the following production rules

$$S \rightarrow SSSSSS \mid \epsilon$$

• Grammar is obviously ambiguous

• It has infinitely many parse trees which can be arbitrarily large!
Chomsky Normal Form

- Bad news: we cannot eliminate ambiguity from CFGs in general
- Good news: we can at least eliminate the possibility to have infinitely many parse trees for a given string
- There is an equivalent grammar in Chomsky Normal Form (CNF) for any context-free grammar
- Grammar in CNF guarantees
  - every string has a finite number of parse trees
  - every parse tree for a given string has the same size (binary tree)
A CFG is in Chomsky Normal Form if each rule is of the form

\[ A \rightarrow BC \]
\[ A \rightarrow a \]

where

- \( a \) is any terminal
- \( A, B, C \) are non-terminals
- \( B, C \) cannot be start variable

We allow the rule \( S \rightarrow \varepsilon \) if \( \varepsilon \in L \)
• For the balanced parentheses language,

\[ S \rightarrow SS \mid (S) \mid () \]

• Equivalent Chomsky Normal Form (CNF) grammar:

\[
S \rightarrow SS \\
S \rightarrow LA \\
A \rightarrow SR \\
S \rightarrow LR \\
L \rightarrow ( \\
R \rightarrow )
\]