CS3063 Theory of Computing

Semester 4 (20 Intake), Feb – Jun 2023

Lecture 6

Context-Free Languages: Session 1

Sanath Jayasena

Announcements

- Assignment 1: due 24th April
- Quiz 5 (based on this Lecture, L6): 24th April
- Next 2 weeks: Semester Break
  - Next lecture L7 will be in the week of 24 – 28 April
- Mid-semester Test: 4th May
  - Will be based on lectures L1 – L7
- No Quiz in the week of 1 – 5 May
  - Quiz 6 (based on L7) will be on 8th May

Today's Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- CFL: Definition and Examples
- CFGs and Regular Languages
- Derivation Trees
- Ambiguity in CFGs

PART 1

Today's Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- CFL: Definition and Examples
- CFGs and Regular Languages
- Derivation Trees
- Ambiguity in CFGs

Context-free Grammars

- Definition: A context-free grammar (CFG) is a 4-tuple $G = (V, \Sigma, S, P)$ where V and Σ are disjoint finite sets, $S \in V$ and P is a finite set of formulas of the form $A \rightarrow \alpha$ where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$
  - V is a set of *variables* or *non-terminal* symbols
  - Σ are terminal symbols or *terminals*
  - S is the *start symbol*
  - P is the set of *grammar rules* or *productions*

Example 1

- Can we describe natural languages?

<sentence> → <noun phrase><verb phrase>

<noun phrase> → <adjective><noun phrase>

<noun phrase> → <noun>

Non-terminals

<noun> → dog

<noun> → cat

Terminals

……

……

Example 2

$S \rightarrow \Lambda$

$S \rightarrow Sa$

$S \rightarrow Sb$

Non-terminal: S

Terminals: Λ ,a, b

- Use symbol "|" to mean "or"

$S \rightarrow \Lambda \mid Sa \mid Sb$

- Write "α⇒β" to mean β can be obtained by applying one of the rules to α

$S \Rightarrow Sa \Rightarrow Sba \Rightarrow Sbba \Rightarrow \Lambda bba = bba$

Derivations

- CFGs generally contain recursive definitions; we obtain (or derive) strings by applying productions
- To indicate a derivation is w.r.t a CFG, *G*, we write "$\alpha \Rightarrow_G \beta$" (generally, "$\alpha \Rightarrow \beta$")
  - This means string β can be obtained from string α by replacing some non-terminal on the LHS of a production in G
  - That is: $\alpha=\alpha_1 A \alpha_2$, $\beta=\alpha_1 \theta \alpha_2$ since $A \rightarrow \theta$ in *G*

Derivations ...contd

- With “$\alpha \Rightarrow_G \beta$” (or, “$\alpha \Rightarrow \beta$”) we say α derives β, or β is derived from α, in one step

- We write “$\alpha \Rightarrow^*_G \beta$” (or, “$\alpha \Rightarrow^* \beta$”) if α derives β in zero or more steps

Derivations ...contd

- Suppose at some point in a derivation we have obtained a string $\alpha = \alpha_1 A \alpha_2$ containing the non-terminal A
- Suppose we have the production $A \rightarrow \theta$

- We may continue by substituting θ for A, *independent of the context* (~ *context-free*) which means no matter what α_1 and α_2 are

PART 2

Today's Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- **CFL: Definition and Examples**
- CFGs and Regular Languages
- Derivation Trees
- Ambiguity in CFGs

Context-free Languages

- Definition: Let $G = (V, \Sigma, S, P)$ be a CFG. The language generated by G is

$$L(G) = \{x \in \Sigma^* | S \Longrightarrow_G^* x\}$$

- A language L is a *context-free language* (CFL) if there is a CFG G so that $L = L(G)$.

Example 1

- The language $\{a^n b^n \mid n \geq 0\}$

$$S \rightarrow \Lambda \mid aSb$$

- Whenever a is added to a string b is added simultaneously
  - Recall: this is not a regular language

Example 2

- The language *pal* of palindromes over $\Sigma = \{a, b\}$
  - Λ is in *pal*
  - For any a in Σ, a is in *pal*
  - For any a in Σ and x in *pal*, axa is in *pal*
  - Nothing else can be in *pal*

- The following CFG defines *pal*

$$S \rightarrow \Lambda \mid a \mid b \mid aSa \mid bSb$$

Example 3

- Language of simple arithmetic expressions
  - Consider only: +, -, *, /, (,) and identifier a
- Can you write the set of productions?

$$S \rightarrow S+S \mid S-S \mid S*S \mid S/S \mid (S) \mid a$$

Example 4

- Syntax of programming languages
- Can you formulate grammar rules to specify a legal statement in C/Java/Python?

<statement> → … | <if_statement> |
<for_statement> | …

<if_statement> → if (<expr>) <statement>

<for_statement> → for (<expr>;<expr>;…)
<statement>

Properties of CFLs

- Theorem: If L_1 and L_2 are CFLs, then the languages $L_1 \cup L_2$, L_1L_2 and L_1^* are also CFLs

- Corollary: Every regular language is a CFL

PART 3

Today's Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- CFL: Definition and Examples
- **CFGs and Regular Languages**
- Derivation Trees
- Ambiguity in CFGs

CFG for a Regular Language

- Example: obtain a CFG equivalent to the regular language (011|1)*(01)*

$A \rightarrow 011 \mid 1$ (we get {011, 1})

$B \rightarrow AB \mid \Lambda$ (we get {011, 1}*)

$D \rightarrow 01$

$C \rightarrow DC \mid \Lambda$ (we get {01}*)

$S \rightarrow BC$ (we get {011,1}*{01}*)

(S is the start symbol)

CFG from an FA

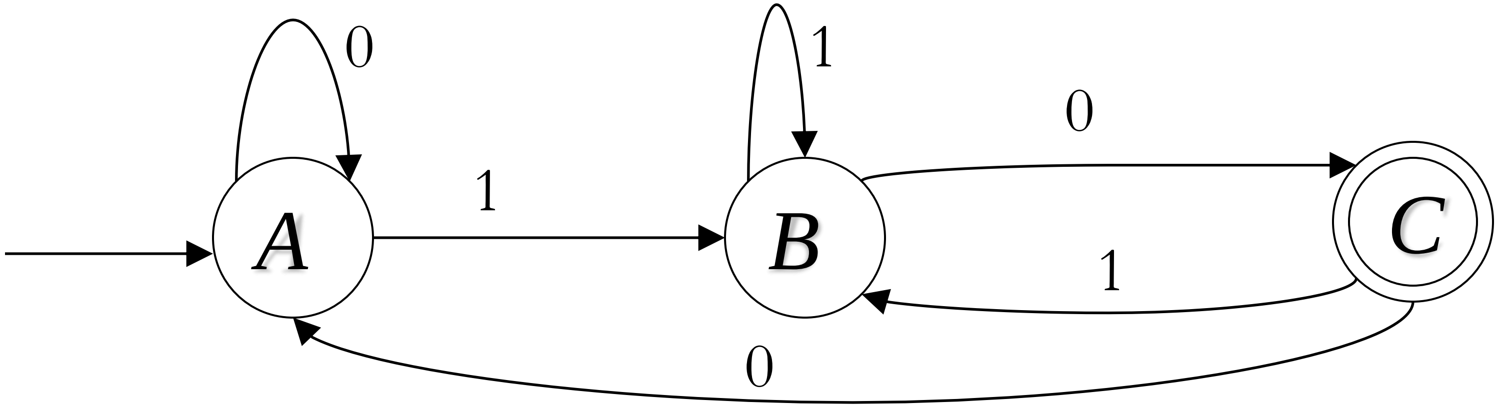
- Suppose we have an FA that accepts L
- We can get the CFG from the FA
  - Productions have a simple form (they track the transitions in the FA)
  - Reversible construction (can get the FA from the CFG)
  - Include productions of the form $P \rightarrow aQ$ where

$$P \xrightarrow{a} Q$$

is a transition in the FA

Example

$(0|1)^*(10)$



- $A \rightarrow 1B$, $A \rightarrow 0A$, $B \rightarrow 1B$, $B \rightarrow 0C$, $C \rightarrow 0A$, $C \rightarrow 1B$
- To terminate, add $B \rightarrow 0$
  - General form $P \rightarrow a$ which says that the FA goes to an accepting state from P with input a

Regular Grammars

- A grammar G is regular if every production takes one of the two following forms:

$$B \rightarrow aC$$

$$B \rightarrow a$$

(B, C are non-terminals, a is a terminal)

PART 4

Today's Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- CFL: Definition and Examples
- CFGs and Regular Languages
- **Derivation Trees**
- Ambiguity in CFGs

Derivation Trees

- Given a CFG, interpreting a string correctly requires finding a correct derivation of the string in the grammar
- Structure of derivation can be shown by a derivation tree (or parse tree)
  - *Root*: non-terminal with which derivation starts
  - *Internal nodes*: non-terminals that appear in the derivation
  - *Leaf nodes*: terminals appearing in derivation

Example

- Given the grammar:

  $S \rightarrow S+S \mid S - S \mid S*S \mid S/S \mid (S) \mid a$

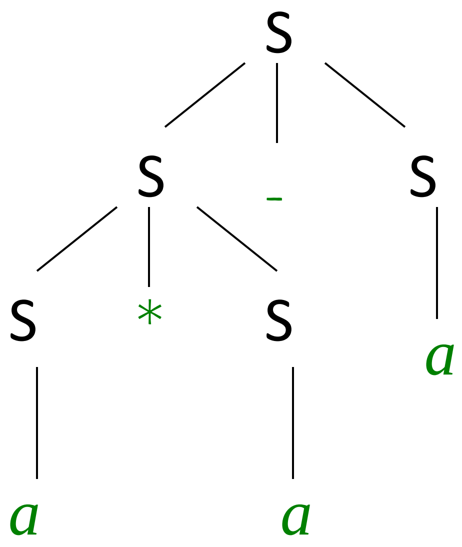
  show derivation trees for

  (i) $a * a - a$

  (ii) $a - a/a$

Solution

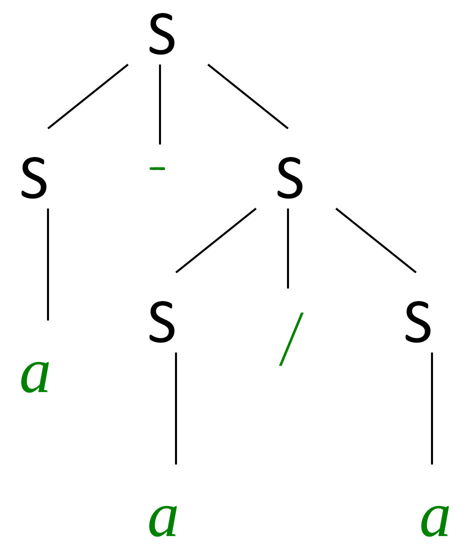
(i) $S \Rightarrow S - S \Rightarrow S{*}S - S \Rightarrow a * S - S \Rightarrow a * a - S \Rightarrow a * a - a$



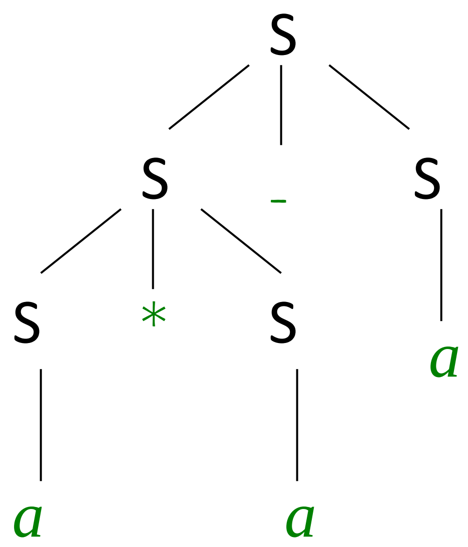
The derived string is extracted from the derivation tree by a left-to-right scan of the leaves

Solution ...contd

(ii) $S \Rightarrow S - S \Rightarrow S - S/S \Rightarrow a - S/S \Rightarrow a - a/S \Rightarrow a - a/a$



Derivation Trees ...contd



- Consider the derivation tree for $a * a - a$
  - What derivation corresponds to it?
  - Is there only one derivation?
- Generally speaking
  - A derivation tree can correspond to more than one derivation
  - Yet **a derivation tree has only one leftmost derivation (and vice versa)**

    (i.e., a 1-to-1 correspondence)

Derivation Trees ...contd

- *Leftmost derivation*
  - Always replace the leftmost non-terminal
- *Rightmost derivation*
  - Always replace the rightmost non-terminal
- Exercise
  - Given the CFG $S \rightarrow S+S \mid S - S \mid S*S \mid S/S \mid (S) \mid a$ and the derivation tree in last slide (for the string $a * a - a$), show the leftmost, rightmost and another derivation.

Derivation Trees ...contd

- Derivation trees
  - specifies productions used
  - temporal order not specified
- *Leftmost derivations corresponding to **different** derivation trees are **different***
- A string of terminals has more than one derivation tree iff it has more than one leftmost derivation
  - [Same for rightmost, symmetrically]

PART 5

Today’s Outline:

Lecture 6

Context-free Languages (CFLs) - 1

- Context-free Grammars (CFGs)
- Derivations
- CFL: Definition and Examples
- CFGs and Regular Languages
- Derivation Trees
- **Ambiguity in CFGs**

Ambiguity

Definitions

- A **string** of terminals is said to be **ambiguous** (or **ambiguously derived**) if it has more than one derivation tree

- A **CFG**, *G*, is **ambiguous** if there is at least one string in $L(G)$ having two or more derivation trees

Ambiguity ...contd

- Ambiguity in natural languages?
- Can you give an example?
  - "They are flying airplanes"
  - "Disabled fly to see the President"
    - S → <collective_noun><verb>…
    - S → <adjective><noun> …

Ambiguity ...contd

- Given the grammar:

  $$S \rightarrow S+S \mid S - S \mid S*S \mid S/S \mid (S) \mid a$$

  string "$a+a-a$" has two leftmost derivations:

(i) $S \Rightarrow S+S \Rightarrow a+S \Rightarrow a+S-S \Rightarrow a+a-S \Rightarrow a+a-a$

(ii) $S \Rightarrow S-S \Rightarrow S+S-S \Rightarrow a+S-S \Rightarrow a+a-S \Rightarrow a+a-a$

- Can you draw the derivation trees?

Ambiguity ...contd

- The "*Dangling Else*" ambiguity:

<stmt> → if (<expr>) <stmt> |
if (<expr>) <stmt> else <stmt> | <other_stmt>

- Consider the statement:

if (expr1) if (expr2) f( ); else g( );

- Give two different derivation trees for this

Ambiguity ...contd

- Given statement:

  if (expr1) if (expr2) f( ); else g( );

- Two different derivations correspond to:

  (i) if (expr1) { if (expr2) f( ); } else g( );

  (ii) if (expr1) { if (expr2) f( ); else g( ); }

- (ii) is used (imposing the rule: "*else associates with the closest else-less if* ")

Unambiguous Grammars

- Examples

$S \rightarrow aSb \mid ab$ $\{a^n b^n \mid n \geq 1\}$

$S \rightarrow aSa \mid bSb \mid c$ $\{wcw^R \mid w \in \{a,b\}^*\}$

Ambiguity ...contd

- Ambiguity comes from the grammar
  - (not really a property of the language)

- Given an ambiguous CFG, usually possible (and desirable) to find an equivalent unambiguous CFG

Example

- Suppose we have the ambiguous CFG:

  $$S \rightarrow S + S \mid S * S \mid (S) \mid a$$

- Avoid $S \rightarrow S + S$ and $S \rightarrow S * S$ because these produce ambiguity
- Also possible to impose rules of order and operator precedence
- Homework: obtain an unambiguous CFG equivalent to the given ambiguous CFG

Conclusion

- We started new topic: CFGs and CFLs
  - Basics of CFGs, CFLs
  - Regular grammars
  - Derivations
  - Ambiguity