CSC 320 - Lecture 09

#context-free #grammars #languages #ambiguous #leftmost-derivation #CNF

Coming Up

Context-free grammars, Pushdown automata, Context-free languages.

Context-Free Grammars

They are a more powerful methods to describe languages. First used to study (describe structure of) human languages. It was invented by Noam Chomsky. Relationship of terms such as *noun*, *verb*, and *preposition*: natural recursion. *Noun* phrases may appear inside *verb* phrases and vice versa.

Compute Science Application: Specification and Compilation of Programming Languages.

- Grammar for programming language: reference for people learning syntax.
- Designing compilers and interpreters: first obtain grammar for language.
- Parser: uses grammar to extract meaning of program prior to generating compiled code.

What's a Grammar? Example!

- Grammar G
    - $A \longrightarrow 0A1$
    - $A \longrightarrow B$
    - $B \longrightarrow \#$
- G consists of
    - Productions/Rules (Substitution Rules)
        - Symbols (Variable), Arrow, String (Variables and Terminals)
    - Terminology: Use capital letters for variables!

We have 3 (substitution) rules, 2 variables: A, B, 3 terminals $0, 1, \#$, start variable: A.

What Does A Grammar Do?

Grammar G: $A \longrightarrow 0A1$, $A \longrightarrow B$, and $B \longrightarrow \#$

- G describes a language L by generating each string of L as follows...
    1. Write down the start variable
        - Normally variable on the left-hand side of the top rule.
    2. Find a variable that is written down and a rule that starts with that variable. Replace written down variable with right-hand side of that rule.
    3. Repeat step 2 until no variable remains.

Example of deriving a string from G:

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111.$

What is $L(G)$?

$L(G)$: set of all strings that can be derived from G. Thus, for the example above $000\#111 \in L(G)$.

- $A \Rightarrow B \Rightarrow \# \in L$

Note. We use $\Rightarrow$ for substitutions rules.

$L(G) = \{0^n\#1^n | n \geq 0\}$

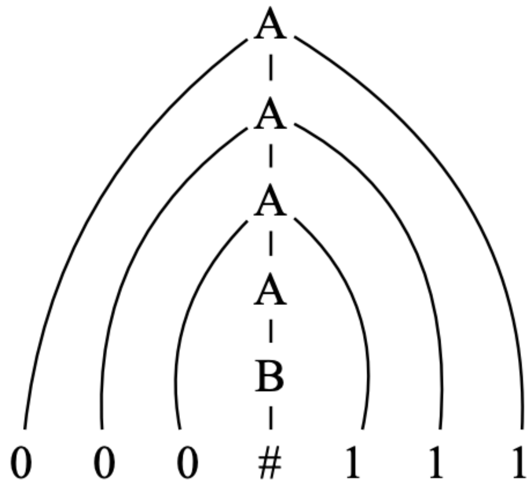
Mini Example

G: $A \longrightarrow 0A1$, $A \longrightarrow B$, and $B \longrightarrow \varepsilon$ then the language would be $L(G) = \{0^n1^n | n \geq 0\}$.

Parse Tree for Derivation of G

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

Parse Tree. Hierarchical representation of terminals and non-terminals. The leaves of a parse tree are the terminals.



Context-Free Grammars Definition

A **context-free grammar** is a 4-tuple (V, Σ, R, S).

- V: finite set of **variables**
- Σ: finite set of **terminals** (disjoint from V)
- R: finite set of (substitution) **rules**
  - each rule R: *a variable* substituted by *a string over variables and terminals*
- $S \in V$: **start variable**
- The right hand side of a rule may be ε

More Terminology

- Given grammar $G = (V, \Sigma, R, S)$
  - Let u, v and w be strings of variables and terminals, and let $A \longrightarrow w$ be a rule of G. Then
    - uAv **yields** uwv, written $uAv \Rightarrow uwv$.
    - u **derives** v, written $u \Rightarrow^* v$, if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
    - The **language of grammar** G is: $L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$

The class of languages described by context-free grammars is the class of **context-free languages**.

Note. We do $\Rightarrow^*$ when we group a bunch of rules together for the derivation.

Examples

Terminology. $S \longrightarrow (S)|SS|\varepsilon$ short for $S \longrightarrow (S)$, $S \longrightarrow SS$, $S \longrightarrow \varepsilon$.

- Given $G(V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{(,)\}$, and R is given by: $S \Rightarrow (S)|SS|\varepsilon$
- Then we can **derive** string $(((()()(()))))$ as follows...
  - ~~**Note**. The symbol replaced in next derivation step is underlined.~~

$S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (((S))) \Rightarrow (((SS))) \Rightarrow$
$((((S)S))) \Rightarrow ((((S)SS))) \Rightarrow (((()SS))) \Rightarrow (((()(S)S))) \Rightarrow (((()(S)(S)))) \Rightarrow (((()()(S)))) \Rightarrow$
$(((()()((S))))) \Rightarrow (((()()(()))))$

Examples

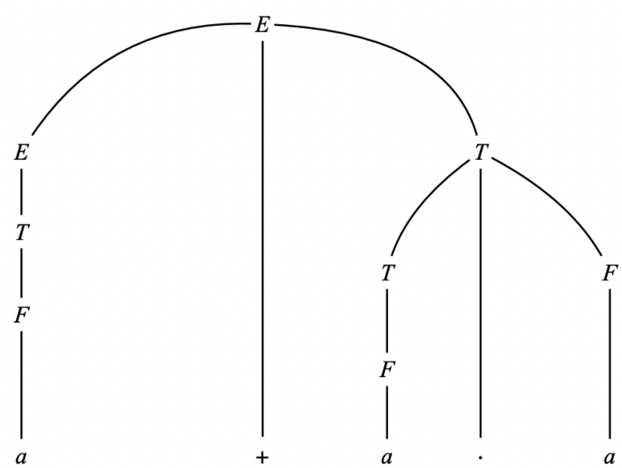
Produce a grammar for language $\{1^n0^n | n \geq 0\}$

- $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{0, 1\}$, $R : S \longrightarrow 1S0|\varepsilon$.

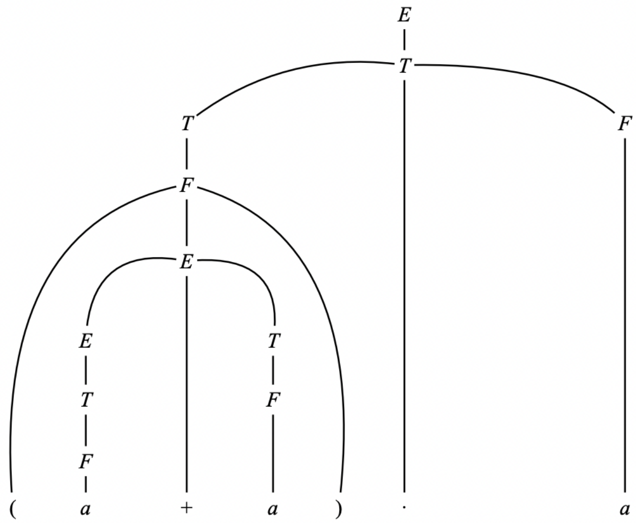
Examples

Given $G = (V, \Sigma, R, E)$ with $V = \{E, F, T\}$, $\Sigma = \{a, +, *, (,)\}$, and R is given by:
$E \longrightarrow E + T|T$, $T \longrightarrow T * F|F$, $F \longrightarrow (E)|a$.

Parse Tree For $a + a * a$



Parse Tree For $(a + a) * a$



Leftmost Derivations

We call a derivation of string w in grammar G **leftmost derivation** if at every step the leftmost remaining variable is replaced.

Ambiguous Grammars

A string w is derived **ambiguously** in context-free grammar G if it has at least two different leftmost derivations. Such a grammar is called **ambiguous**.

Example

Given $G = (V, \Sigma, R, E)$ with $V = \{E\}$, $\Sigma = \{a, +, *, (,)\}$, and R is given by:
$E \longrightarrow E + E|E * E|(E)|a$

Two leftmost derivation in $E \longrightarrow E + E|E * E|(E)|a$

- $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$
- $E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a.$

We see that $a + a * a$ is derived ambiguously in G. Therefore G is ambiguous.

We Learned

What a context-free grammars are, what a language of context-free grammar is, that ambiguous grammars exist.

Next...

Chomsky Normal Form (CNF). Helps dealing with ambiguity, constraint grammar rules.

Chomsky Normal Form

Restricted (simplified) constrains on grammar. A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky Normal Form** if every rule is of the form...

- $A \Rightarrow BC$ or $A \Rightarrow a$ where...
  - $a \in \Sigma$
  - $A, B, C \in V$
  - B, C may not be the start variable.
  - $S \longrightarrow \varepsilon$ is permitted where S is start variable. (No other ε-substitutions permitted).

Right hand side: two variables or one terminal; nothing else. Start variable not on right-hand side of rule.

Theorem

Any context-free language is generated by a context-free grammar in Chomsky Normal Form.

Proof. Any context-free language is generated by a context-free grammar in Chomsky Normal Form

Idea. Given context free grammar G, convert G into Chomsky Normal Form.

- If rule violates Chomsky Normal Form condition: replace with equivalent one that satisfies Chomsky Normal Form condition.
  - A a new start variable
  - Eliminate all ε-rules of form $A \longrightarrow \varepsilon$
  - Eliminate all *units* rules of form $A \longrightarrow B$
  - Convert remaining rules.

Goal. Given context-free grammar $G = (V, \Sigma, R, S)$, convert into context-free grammar $G' = (V', \Sigma, R', S_0)$ in Chomsky Normal Form with $L(G) = L(G')$.

Step 1. Add New Start Variable

Let $S_0 \notin V$. Add new start variable S_0 and rule $S_0 \longrightarrow S$. Start variable in G' not on right-hand side of rule.

Step 2. Eliminate All ε-Rules of Form $A \longrightarrow \varepsilon$

Repeat until all ε-rules not involving S_0 are eliminated. Let $A \longrightarrow \varepsilon$, $A \neq S_0$. For each $W \longrightarrow uAv$ and occurrence of A, with u, v strings of variables and terminals. Add new rule $W \longrightarrow uv$. For $W \longrightarrow A$, add $W \longrightarrow \varepsilon$ unless $W \longrightarrow \varepsilon$ was previously removed.

Step 3. Eliminate Unit Rules

Repeat until all units rules are eliminated. Given $A \longrightarrow B$. For each appearance of $B \longrightarrow u$ add $A \longrightarrow u$ (unless this rule was removed previously). As before, u is a string of variables and terminals.

Step 4. Convert Remaining Rules

Replace each rule $A \longrightarrow u_1 u_2 \ldots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with...

- $A \longrightarrow u_1 A_1, A_1 \longrightarrow u_2 A_2, A_2 \longrightarrow u_3 A_3, \ldots,$ and $A_{k-2} \longrightarrow u_{k-1} u_k$. The A_i's are new variables.

Replace any terminal u_i and add rule $U_i \longrightarrow u_i$.

Previous Lecture

Lecture08

Next Lecture

Lecture10a