COMPSCI 501: Formal Language Theory

Lecture 7: Context-Free Grammars

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

6 February 2019

Grammars describe programming languages

(6.8.4) *selection-statement:*
  if (*expression*) *statement*
  if (*expression*) *statement* **else** *statement*
  switch (*expression*) *statement*

(6.8.5) *iteration-statement:*
  while (*expression*) *statement*
  do *statement* **while** (*expression*) ;
  for (*expression*$_{opt}$; *expression*$_{opt}$; *expression*$_{opt}$) *statement*
  for (*declaration* *expression*$_{opt}$; *expression*$_{opt}$) *statement*

Grammars describe natural language

A good student reads books.

S → NP VP noun phrase + verb phrase
NP → Noun
NP → Det NP determiner: article, adjective
VP → Verb
VP → Verb NP verb + (direct) object
Det → a | good
Noun → books | student
Verb → reads

Terminology

$A \to 0A1$
$A \to B$
$B \to \#$

- Grammar = set of **productions** (substitution **rules**)
- Left-hand side: **variable** (**nonterminal**)
- Right-hand side: string of symbols (variables and **terminals**)
- A **start variable**

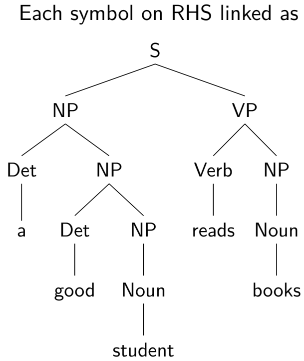
Derivation: apply substitution rules from start variable, until no variables remain.

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

Or represent as **parse tree**.

Parse Tree for a Sentence

Hierarchical representation of derivation
Each symbol on RHS linked as child of LHS variable



A good student reads books.

S → NP VP

NP → Det NP VP → Verb NP

NP → Det NP NP → Noun

NP → Noun

Sentence: terminals (leaves) left to right

Formal Definition

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

- V is a finite set, called **variables** (also called nonterminals)
- Σ is a finite set of **terminals**; $\Sigma \cap V = \emptyset$
- R is a finite set of **rules**, $R \subseteq V \times (\Sigma \cup V)^*$
- $S \in V$ is the start variable

Denote $uAv \Rightarrow uwv$ (**yields**) if $A \to w$ is a rule

u **derives** v ($u \stackrel{*}{\Rightarrow} v$) if u yields v in ≥ 0 steps, i.e.,
 $u = v$ or $u \Rightarrow u_1 \Rightarrow u_2 \ldots \Rightarrow u_k \Rightarrow v$

Language of a grammar: all strings (of **terminals**) generated from start symbol $L(G) = \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$

Context-free language: generated by a context-free grammar

Designing Grammars

$0^n 1^n$

"count" by matching individual symbols, use nonterminals for rest

Match *first* 0 with *last* 1
  expose the same (but smaller) pattern : recursion
  (direct or indirect: circular dependencies between variables)

$S \to 0S1|\varepsilon$

Balanced Parentheses:
  first symbol is (and must be matched by) somewhere
  string in between and string after are balanced

$S \to (S)S|\varepsilon$

Designing Grammars

$L = \{ww^R | w \in \Sigma^*\}$

match first symbol with last symbol

$S \to \varepsilon|aSa|bSb|\dots$ for all symbols in Σ

Other tips:
  split into sublanguages, use union
  recognize recursive structures

Regular Languages are Context-Free

Can we convert a regular expression to a grammar ?

Grammars have concatenation
  and union (multiple right-hand sides)
  Kleene star? $A\_star \to \varepsilon|AA\_star$

Can also easily convert a DFA to a CFG:

- make variable R_i for each state q_i
- for transition $q_i \xrightarrow{a} q_j$, add rule $R_i \to aR_j$
- for each accept state q_i, add rule $R_i \to \varepsilon$
- starting state q_0 gives start variable R_0

Ambiguity

Can we have multiple derivations for a parse tree?
Yes, order of expanding children ($S \to AB$, $A \to a$, $B \to b$)
Not a problem.

Leftmost derivation: replace leftmost variable at each step
  unique for each parse tree

Multiple **parse trees** for one string = **ambiguity**.
(= different **rules** to derive the same sentence)
Is a problem: meaning usually associated with production rule.

"the girl touches *the boy with the flower*"
"the girl *touches* the boy *with the flower*"
  (propositional phrase part of noun phrase or verb phrase)

Def: A string is derived **ambiguously** if it has more than one leftmost derivation.

Ambiguity: If-Then-Else

(Backus-Naur-Form: mix of grammar and regex notation)

Stmt ::= ExpStmt | IfStmt | WhileStmt | Block

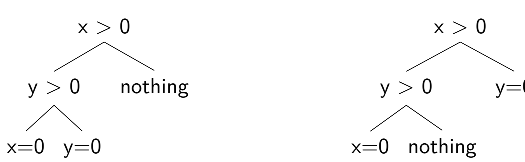
ExpStmt ::= expr ;

IfStmt ::= `if` (expr) Stmt `else` Stmt | `if` (expr) Stmt

WhileStmt ::= `while` (expr) Stmt Block ::= { Stmt* }

Problem: which `if` is matched by the `else` ?

```
if (x > 0) if (y > 0) x = 0; else y = 0;
```



Disambiguating If-Then-Else

Redesign grammar; distinguish between:
balanced `if` (with `else`)
unbalanced `if` (without `else`)

Other statements are included in the "balanced" category

Stmt ::= BalancedStmt | UnBalancedIf

BalancedStmt ::= ExpStmt | WhileStmt | Block | BalancedIf

ExpStmt ::= expr ;

WhileStmt ::= `while` (expr) Stmt Block ::= { Stmt* }

BalancedIf ::= `if` (expr) BalancedStmt `else` Stmt

UnBalancedIf ::= `if` (expr) Stmt

Disambiguating Expressions

Expr → Expr + Expr | Expr * Expr | (Expr) | num

Two parse trees for: num + num * num

Disambiguate: introduce one nonterminal for each precedence level

Expr → Term | Expr + Term

Term → Factor | Term * Factor

Factor → (Expr) | num

Chomsky Normal Form

A CFG is in **Chomsky normal form** if every rule is of the form

$$A \to BC$$
$$A \to a$$
$$S \to \varepsilon$$

where B, C may not be the start variable

Theorem: Any context-free language can be generated by a CFG in Chomsky Normal Form

Proof: By Construction

Conversion to Chomsky Normal Form

- Add new start variable $S_0 \to S$, if S appears on RHS.
- Eliminate rules $A \to \epsilon$:
  for each rule with A on RHS, add a copy eliminating A
  must do for each occurrence:
  $R \to AuA$ will yield $R \to uA|Au|u$
  for $R \to A$ we add $R \to \varepsilon$ unless it was previously removed
  (ensures termination)
- Eliminate unit rules $A \to B$:
  for any $B \to u$, add $A \to u$, unless this is a previously removed unit rule
- Eliminate rules with ≥ 3 symbols on RHS:
  for $SA \to u_1 u_2 \dots u_k$, add rules:
  $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $\dots A_{k-2} \to u_{k-1} u_k$.
  If any u_i are terminals, add rule $U_i \to u_i$

Outlook: Chomsky Hierarchy

Type-3: **regular** grammar: generate regular languages
$A \to a$, $A \to \varepsilon$, $A \to aB$ (right-regular), OR
$A \to a$, $A \to \varepsilon$, $A \to Ba$ (left-regular); **don't** combine!
recognized by deterministic automata

Type-2: **context-free** grammar
$A \to \gamma$ left: nonterminal; right: any string
recognized by (nondeterministic) pushdown automata

Type-1: **context-sensitive** grammar
$\alpha A \beta \to \alpha \gamma \beta$ with $\gamma \neq \varepsilon$
(also $S \to \varepsilon$)
recognized by linear-bounded nondeterministic Turing machine

Type-0: **recursively enumerable** languages
$\alpha \to \beta$ (α contains some nonterminal)
recognized by Turing machine