91.304 Foundations of (Theoretical) Computer Science

Chapter 2 Lecture Notes (Section 2.1: Context-Free Grammars)

David Martin
dm@cs.uml.edu

With some modifications by Prof. Karen Daniels, Fall 2012

Chapter 2: CFGs, CFLs, PDAs

- We introduce our next programming model with a grammatical formulation: more like REX than DFA
- A context-free grammar (CFG) is a list of permitted substitution rules
  $S \to \varepsilon$ *S may be rewritten as ε*
  $S \to 0\ S\ 1$ *...or as 0 S 1*
- The **variables** are the substitutable things, written in UPPER CASE, sometimes called **nonterminals**
- The **terminals** are the nonsubstitutable characters
- Each **rule** has a single variable on the left of $\to$ and a list of terminals and variables on the right
- The **language generated** by a CFG is the set of all (terminal) strings that can be found by starting from S and repeatedly substituting until no variables remain.

Language generated by example

- So in
  $S \rightarrow \varepsilon$
  $S \rightarrow 0\ S\ 1$

  We can generate the strings ε, 01, 0011, 000111, etc.

  In other words, the language $\{0^n 1^n \mid n \geq 0\}$ -- which is non-regular

Formal definition

- G is a **Context Free Grammar** (CFG) if $G = (V, \Sigma, R, S)$ where

1. V is a finite set of variables
2. Σ is a finite set of terminals and $V \cap \Sigma = \emptyset$
   - It's an alphabet that can't overlap with V
3. R is a set of rules of the form $\alpha \to \beta$ where
   - $\alpha \in V$ — a single variable
   - $\beta \in (\Sigma \cup V)^*$ — a string of vars and terminals
4. $S \in V$ is the starting variable

Semantics of CFGs

- If $G=(V,\Sigma,R,S)$ is a CFG, then $\Rightarrow_G$ and $\Rightarrow_G^*$ are relations on the set $(V\cup\Sigma)^*$ as follows
  - If $\alpha,\gamma \in (V\cup\Sigma)^*$, then $\alpha \Rightarrow_G \gamma$ means that α can be rewritten as γ by using some rule from R exactly once
  - $\Rightarrow_G^*$ is the reflexive and transitive closure of $\Rightarrow_G$
    - In other words, $\alpha \Rightarrow_G^* \gamma$ means that α can be rewritten as γ by using zero or more rules
  - Pronounce them "yields" or "derives"

Language derived by a CFG

- Finally, the language **derived** by or **generated** by (or simply **of**) G is

  $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow_G^* w \}$

- Note that $w \in \Sigma^*$ — so w may *not* contain any variables from V
  - They are "nonterminal" — not done yet
  - Σ is the set of "terminals" — when you stop

Conventional form

- You can either formally specify the 4 parts of a CFG, or use the conventional notation in this example:

start

abbreviation for multiple rules

$S \to 0B \mid 0A$

$A \to BAB \mid AS \mid 1B2$

$B \to 0 \mid S \mid \varepsilon$

Here $\Sigma = \{0, 1, 2\}$

nonterminals

must be terminals

Facts about this grammar

- $S \to 0B \mid 0A$
  $A \to BAB \mid AS \mid 1B2$
  $B \to 0 \mid S \mid \varepsilon$

  $S \Rightarrow 0B \Rightarrow 0S \Rightarrow 00A \Rightarrow 00BAB$
  $\Rightarrow 000AB \Rightarrow 0001B2B$
  $\Rightarrow 000102B \Rightarrow 0001020$

  Thus $00A \Rightarrow^* 0001B2B$
  and $0001020 \in L(G)$

  Also $1A \Rightarrow 11B2 \Rightarrow 1102$ yet $1102 \not\in L(G)$

  Why?

Context-free languages

- **Definition** $L \subseteq \Sigma^*$ is a **context-free** language (CFL) if it is generated by some context free grammar (CFG).
- **Definition**
  $CFL(\Sigma) = \{ L \subseteq \Sigma^* \mid L \text{ is context free } \}$
  is the class of all CFLs
- Note that "context free" refers to the left hand side of the rules
  - $B \to 0$ Ok — "B may always be replaced by 0"
  - $2B \to 1A$ Not ok
    - such a rule is **not allowed**; this is a **context-sensitive** rule that says B may only be replaced when preceded by a 2 — **context-free** grammars can't say that

Context-free languages

- CFLs (and variants of them) are often used in compilers and interpreters in order to make sense of programs and other formal specifications
- Sample grammar:
  $E \to E + E \mid E \times E \mid (E) \mid N$
  $N \to 0 \mid 1 \mid NN$
- Then $E \Rightarrow^* 1 + (101 \times 1110)$
- See discussion of parsing and ambiguity in textbook (p. 105-106). Covered more in a compiler course.

Another example

- Let $\Sigma = \{a,b\}$ and recall REX(Σ) — the set of all regular expressions over Σ

  $S \to B \mid I$ (*base or inductive cases*)
  $B \to \underline{\emptyset} \mid \underline{\varepsilon} \mid \underline{a} \mid \underline{b}$
  $I \to \underline{(}\ S\ \underline{\cdot}\ S\ \underline{)} \mid \underline{(}\ S\ \underline{^*}\ \underline{)} \mid \underline{(}\ S\ \underline{\cup}\ S\ \underline{)}$

  The nonvariables are literal characters and are underlined here, including the symbols ∅ and ε.

Yet another example

- Let G be the grammar
  $S \to \varepsilon \mid 0S1 \mid 1S0 \mid SS$
- and $L_1 = \{ x \in \{0,1\}^* \mid n_0(x) = n_1(x) \}$
- Clearly $L(G) \subseteq L_1$. It's true but harder to see that $L_1 \subseteq L(G)$ as well...

DFA – to - CFL Conversion

- Make variable R_i for each DFA state q_i
- Add rule $R_i \rightarrow aR_j$ if $\delta(q_i, a) = q_j$ is DFA transition
- Add rule $R_i \rightarrow \varepsilon$ if q_i is a DFA accept state
- Make R_0 starting variable if DFA starting state is q_0
- Example: board work

Ambiguity

- String w is ambiguously derived in a grammar if grammar generates w in multiple different ways.
- Two derivations may differ in order of variable replacement yet be the same in their overall structure.
- Derivation is leftmost if, at each step, leftmost remaining variable is the one replaced.
- A string w is derived ambiguously in context-free grammar G if it has at least 2 different leftmost derivations.
- G is ambiguous if it generates some string ambiguously.
- Example: board work
- G is inherently ambiguous if it can only be generated by ambiguous grammars.
  - Example: $\{a^i b^j c^k \mid i = j \text{ or } j = k\}$

Chomsky Normal Form

- **Definition 2.8**: A context-free grammar is in <u>Chomsky normal form</u> if every rule is of the form:

$$A \rightarrow BC$$

$$A \rightarrow a$$

where a is any terminal and A, B, and C are any variables (except that B and C may not be the start variable). In addition, we permit the rule:

$$S \rightarrow \varepsilon$$

where S is the start variable.

Chomsky Normal Form (continued)

- **Theorem 2.9**: Any context-free language is generated by a context-free grammar in Chomsky normal form.
- **Proof Idea** (see p. 107-108 for complete proof):
  - Add a new start variable S_0 and rule: $S_0 \to S$
  - Process ε rules:
    - Remove $A \to \varepsilon$ where A is not the start variable.
    - For every occurrence of A on right-hand side of a rule, add new rule with that occurrence deleted.
    - Repeat until all ε rules not involving start variable are removed.
  - Handle all unit rules:
    - Remove unit rule: $A \to B$
    - Where rule $B \to u$ appears, add $A \to u$ unless this was unit rule previously removed.
    - Repeat until all unit rules are eliminated.
  - Convert remaining rules into the proper form:
    - Replace each rule $A \to u_1 u_2 \cdots u_k$ where $k \geq 3$ and each u_i is a variable or a terminal symbol with rules:

$$A \to u_1 A_1, \quad A_1 \to u_2 A_2, \quad A_2 \to u_3 A_3, \quad \ldots, \quad A_{k-2} \to u_{k-1} u_k$$

Chomsky Normal Form (continued)

Example 2.10

1. The original CFG G_6 is shown on the left. The result of applying the first step to make a new start variable appears on the right.

$$\begin{aligned} S &\to ASA \mid \mathtt{a}B \\ A &\to B \mid S \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned} \qquad\qquad \begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \\ A &\to B \mid S \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned}$$

2. Remove ε rules $B \to \varepsilon$, shown on the left, and $A \to \varepsilon$, shown on the right.

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathbf{a} \\ A &\to B \mid S \mid \boldsymbol{\varepsilon} \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned} \qquad\qquad \begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \boldsymbol{S} \\ A &\to B \mid S \mid \varepsilon \\ B &\to \mathtt{b} \end{aligned}$$

3a. Remove unit rules $S \to S$, shown on the left, and $S_0 \to S$, shown on the right.

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid S \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned} \qquad\qquad \begin{aligned} S_0 &\to S \mid \boldsymbol{ASA} \mid \boldsymbol{\mathtt{a}B} \mid \mathbf{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

3b. Remove unit rules $A \to B$ and $A \to S$.

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \mid \mathbf{b} \\ B &\to \mathtt{b} \end{aligned} \qquad\qquad \begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to S \mid \mathtt{b} \mid \boldsymbol{ASA} \mid \boldsymbol{\mathtt{a}B} \mid \mathbf{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\ B &\to \mathtt{b} \end{aligned}$$

4. Convert the remaining rules into the proper form by adding additional variables and rules. The final grammar in Chomsky normal form is equivalent to G_6, which follows. (Actually the procedure given in Theorem 2.9 produces several variables U_i along with several rules $U_i \to \mathtt{a}$. We simplified the resulting grammar by using a single variable U and rule $U \to \mathtt{a}$.)

$$\begin{aligned} S_0 &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ S &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ A &\to \mathtt{b} \mid AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ A_1 &\to SA \\ U &\to \mathtt{a} \\ B &\to \mathtt{b} \end{aligned}$$

Picture so far