Lecture 14 — Properties of Context-Free Languages (1)

CSE205: Theory of Computation

Minseok Jeon

2026 Fall

Properties of CFLs

- Normal forms for CFGs
- Pumping lemma for CFLs
- Closure properties for CFLs

This lecture focuses on *normal forms* for context-free grammars, in particular the Chomsky Normal Form and the simplifications needed to obtain it.

Chomsky Normal Form

Definition

A CFG is in *Chomsky Normal Form* (CNF) if all of its productions are of the form

$$A \to BC \qquad \text{or} \qquad A \to a$$

where A, B, C are variables and a is a terminal.

Theorem

Every CFL (without ε) has a CFG in CNF.

Preliminary Simplifications

To put a grammar in CNF, we first apply three simplifications:

1. Elimination of useless symbols
2. Elimination of $\boldsymbol{\varepsilon}$-productions
3. Elimination of unit productions

Useless Symbols

Definition (Useful / Useless Symbols)

A symbol X is *useful* for a grammar $G = (V, T, S, P)$ if there is some derivation of the form

$$S \Rightarrow^* \alpha X \beta \Rightarrow^* w, \qquad w \in T^*.$$

Otherwise, X is *useless*.

Eliminating Useless Symbols

1. Identify *generating* and *reachable* symbols.
   - $\boldsymbol{X}$ is *generating* if $\boldsymbol{X} \Rightarrow^* \boldsymbol{w}$ for some terminal string $\boldsymbol{w}$.
   - $\boldsymbol{X}$ is *reachable* if $\boldsymbol{S} \Rightarrow^* \boldsymbol{\alpha X \beta}$ for some $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.
2. Remove non-generating symbols, and *then* non-reachable symbols.

Example

$$S \to AB \mid a$$
$$A \to b$$

1. Find generating symbols: $\{a, b, A, S\}$ (B is non-generating).
2. Remove non-generating symbols: $S \to a,\ A \to b$.
3. Find reachable symbols: $\{S, a\}$.
4. Remove non-reachable symbols: $S \to a$.

The resulting grammar has no useless symbols and generates $\{a\}$.

Correctness of Useless Symbol Elimination

Theorem

Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$ be a CFG and assume $\boldsymbol{L}(\boldsymbol{G}) \neq \emptyset$. Let $\boldsymbol{G}_1$ be obtained as follows:

1. *Eliminate non-generating symbols and all productions involving those symbols, giving $\boldsymbol{G}_2 = (\boldsymbol{V}_2, \boldsymbol{T}_2, \boldsymbol{S}, \boldsymbol{P}_2)$.*
2. *Eliminate all symbols that are not reachable in $\boldsymbol{G}_2$.*

Then $\boldsymbol{G}_1$ has no useless symbols, and $\boldsymbol{L}(\boldsymbol{G}_1) = \boldsymbol{L}(\boldsymbol{G})$.

Finding Generating and Reachable Symbols

1. The sets of generating and reachable symbols are defined *inductively*.
2. We can compute these inductive sets via the iterative *fixed-point* algorithm.

Inductive Definition of Generating Symbols

Definition (Generating Symbols)

Let $G = (V, T, S, P)$ be a grammar. The set of generating symbols of G is defined as follows:

- **Basis:** The set includes every symbol of T.
- **Induction:** If there is a production $A \to \alpha$ and the set includes every symbol of α, then the set includes A.

Note that the definition is non-constructive.

Computing the Set of Generating Symbols

1. Represent the inductive definition by a function $F \in 2^{V \cup T} \to 2^{V \cup T}$:

$$F(X) = T \cup \{A \mid A \to \alpha,\ \alpha \in X^*\}.$$

2. Apply the iterative fixed-point algorithm:

$$\begin{array}{l} fix(F) = \\ \quad S := \emptyset \\ \quad \textbf{repeat} \\ \quad\quad S' := S \\ \quad\quad S := S \cup F(S) \\ \quad \textbf{until } S = S' \\ \quad \textbf{return } S \end{array}$$

Example: Generating Symbols

$$S \to AB \mid a$$
$$A \to b$$

The fixed-point iteration for finding generating symbols:

$$S_0 = \emptyset$$
$$S_1 = \{a, b\}$$
$$S_2 = \{a, b, A, S\}$$
$$S_3 = \{a, b, A, S\} = S_2 \quad \text{(fixed point)}$$

B is not generating.

Inductive Definition of Reachable Symbols

Definition (Reachable Symbols)

Let $G = (V, T, S, P)$ be a grammar. The set of reachable symbols of G is defined as follows:

- **Basis:** The set includes S.
- **Induction:** If the set includes A and there is a production $A \to X_1 \cdots X_k$, then the set includes $X_1, \ldots, X_k$.

The function $F \in 2^{V \cup T} \to 2^{V \cup T}$:

$$F(X) = \{S\} \cup \bigcup_{A \in X} \{X_1, \ldots, X_k \mid A \to X_1 \cdots X_k\}.$$

Example: Reachable Symbols

$$S \to AB \mid a$$
$$A \to b$$

The fixed-point iteration for finding reachable symbols (in the grammar already reduced to generating symbols, $S \to a$):

$$S_0 = \emptyset$$
$$S_1 = \{S\}$$
$$S_2 = \{S, a\}$$
$$S_3 = \{S, a\} = S_2 \quad \text{(fixed point)}$$

Eliminating ε-Productions ($A \to \varepsilon$)

1. Find *nullable* variables.
2. Construct a new grammar, where nullable variables are replaced by ε in all possible combinations.

Nullable Variables

Definition

A variable A is *nullable* if $A \Rightarrow^* \varepsilon$.

Definition (Inductive version)

Let $G = (V, T, S, P)$ be a grammar. The set of nullable variables is defined as follows:

- **Basis:** If $A \to \varepsilon$ is a production of G, then the set includes A.
- **Induction:** If there is a production $B \to C_1 \cdots C_k$ where every C_i is in the set, then the set includes B.

Eliminate ε-Productions

Let $G = (V, T, S, P)$ be a grammar. Construct a new grammar (V, T, S, P_1) where P_1 is defined as follows.

For each production $A \to X_1 X_2 \cdots X_k$ of P with $k \geq 1$:

1. Put $A \to X_1 X_2 \cdots X_k$ into P_1.
2. Put into P_1 all those productions generated by replacing nullable variables by ε in all possible combinations. If all X_i are nullable, do *not* put $A \to \varepsilon$ into P_1.

Example: Eliminating ε-Productions

$$
\begin{aligned}
S &\to AB \\
A &\to aAA \mid \varepsilon \\
B &\to bBB \mid \varepsilon
\end{aligned}
$$

The set of nullable symbols: $\{A, B, S\}$.

The new grammar without ε-productions:

$$
\begin{aligned}
S &\to AB \mid A \mid B \\
A &\to aAA \mid aA \mid aA \mid a \\
B &\to bBB \mid bB \mid bB \mid b
\end{aligned}
$$

(i.e. $A \to aAA \mid aA \mid a$ and $B \to bBB \mid bB \mid b$ after removing duplicates).

Eliminating Unit Productions

A *unit production* is of the form $A \to B$, e.g.,

$$S \to A$$
$$A \to a \mid b$$

Eliminating Unit Productions

Given $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P})$:

1. Find all *unit pairs* of variables $(\boldsymbol{A}, \boldsymbol{B})$ such that $\boldsymbol{A} \Rightarrow^* \boldsymbol{B}$ using a sequence of unit productions only.
2. Define $\boldsymbol{G}_1 = (\boldsymbol{V}, \boldsymbol{T}, \boldsymbol{S}, \boldsymbol{P}_1)$ as follows. For each unit pair $(\boldsymbol{A}, \boldsymbol{B})$, add to $\boldsymbol{P}_1$ all productions $\boldsymbol{A} \to \boldsymbol{\alpha}$ where $\boldsymbol{B} \to \boldsymbol{\alpha}$ is a *non-unit* production in $\boldsymbol{P}$.

E.g.,

$$\begin{array}{l} \boldsymbol{S} \to \boldsymbol{A} \\ \boldsymbol{A} \to \boldsymbol{a} \mid \boldsymbol{b} \end{array} \quad \Longrightarrow \quad \begin{array}{l} \boldsymbol{S} \to \boldsymbol{a} \mid \boldsymbol{b} \\ \boldsymbol{A} \to \boldsymbol{a} \mid \boldsymbol{b} \end{array}$$

Example: Unit Productions

$$S \to Aa \mid B$$
$$B \to A \mid bb$$
$$A \to a \mid bc \mid B$$

Unit pairs: $(S, S), (A, A), (B, B), (S, B), (B, A), (A, B)$.

The grammar without unit productions:

$$S \to Aa \mid bb \mid a \mid bc$$
$$B \to bb \mid a \mid bc$$
$$A \to a \mid bc \mid bb$$

Eliminating Unit Productions

Theorem (Correctness)

If grammar G_1 is constructed from grammar G by the algorithm for eliminating unit productions, then $L(G_1) = L(G)$.

Finding Unit Pairs

Definition (Unit Pairs)

Let $G = (V, T, S, P)$ be a grammar. The set of unit pairs is defined as follows:

- **Basis:** (A, A) is a unit pair for any variable A.
- **Induction:** Suppose we have determined that (A, B) is a unit pair, and $B \to C$ is a production where C is a variable. Then (A, C) is a unit pair.

$$F(X) = \{(A, A) \mid A \in V\} \cup \{(A, C) \mid (A, B) \in X,\ B \to C,\ C \in V\}.$$

Example: Computing Unit Pairs

$$
\begin{aligned}
S &\to Aa \mid B \\
B &\to A \mid bb \\
A &\to a \mid bc \mid B
\end{aligned}
$$

The fixed-point computation proceeds as follows:

$$
\begin{aligned}
&\emptyset, \\
&\{(S,S),(A,A),(B,B)\}, \\
&\{(S,S),(A,A),(B,B),(S,B),(B,A),(A,B)\}, \\
&\{(S,S),(A,A),(B,B),(S,B),(B,A),(A,B)\}.
\end{aligned}
$$

Putting Them Together

Apply the simplifications in the following order:

1. Eliminate $\boldsymbol{\varepsilon}$-productions
2. Eliminate unit productions
3. Eliminate useless symbols

Theorem

If $\boldsymbol{G}$ is a CFG generating a language that contains at least one string other than $\boldsymbol{\varepsilon}$, then there is another CFG $\boldsymbol{G}_1$ such that $\boldsymbol{L}(\boldsymbol{G}_1) = \boldsymbol{L}(\boldsymbol{G}) - \{\boldsymbol{\varepsilon}\}$, and $\boldsymbol{G}_1$ has no useless symbols, $\boldsymbol{\varepsilon}$-productions, or unit productions.

Chomsky Normal Form

Definition (Chomsky Normal Form)

A grammar G is in CNF if all productions in G are either

1. $A \to BC$, where A, B, and C are variables, or
2. $A \to a$, where A is a variable and a is a terminal.

Further, G has no useless symbols.

Putting a CFG in CNF

1. Start with a grammar without useless symbols, $\boldsymbol{\varepsilon}$-productions, and unit productions.
2. Each production is either of the form $\boldsymbol{A \to a}$ (already allowed by CNF) or has a body of length 2 or more. Do the following:
   - 2.1 Arrange that all bodies of length 2 or more consist only of variables. If a terminal $\boldsymbol{a}$ appears in such a body, replace it by a new variable $\boldsymbol{A}$ and add $\boldsymbol{A \to a}$.
   - 2.2 Break bodies of length 3 or more into a cascade of productions, each with a body of two variables:

$$A \to B_1 B_2 \cdots B_k$$

   becomes

$$A \to B_1 C_1,\ C_1 \to B_2 C_2,\ \ldots,\ C_{k-2} \to B_{k-1} B_k.$$

Summary

- Every CFG can be transformed into a CFG in CNF.
- To do so:
  1. Apply $\boldsymbol{\varepsilon}$-production, unit production, and useless symbol eliminations.
  2. Arrange and break the remaining productions.