COSE215: Theory of Computation

Lecture 13 — Properties of Context-Free Languages (1)

Hakjoo Oh
2019 Spring

Properties of CFLs

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

Chomsky Normal Form

Definition

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

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

Theorem

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

Preliminary Simplications

1. Elimination of *useless symbols*
2. Elimination of ϵ*-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 $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, where $w \in T^*$. Otherwise, X is *useless*.

Eliminating Useless Symbols

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

Example

$$\begin{array}{rcl} S & \rightarrow & AB \mid a \\ A & \rightarrow & b \end{array}$$

1. Find generating symbols:
2. Remove non-generating symbols:
3. Find reachable symbols:
4. Remove non-reachable symbols:

Correctness of Useless Symbol Elimination

Theorem

Let $G = (V, T, S, P)$ be a CFG and assume that $L(G) \neq \emptyset$. Let G_2 be the grammar obtained by running the following procedure:

1. *Eliminate non-generating symbols and all productions involving those symbols. Let $G_2 = (V_2, T_2, S, P_2)$ be this new grammar.*
2. *Eliminate all symbols that are not reachable in the grammar G_2. Let G_1 be the result.*

Then, G_1 has no useless symbols, and $L(G) = L(G_1)$.

Finding Generating and Reachable Symbols

1. The sets of generating and reachable symbols are defined inductively.
2. We can compute inductive sets via an 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

An iterative fixed point algorithm:

$$
\begin{array}{l}
Y := T \\
repeat \\
\quad Y' := Y \\
\quad Y := Y \cup \{A \mid (A \to \alpha) \in P, Y \text{ includes every symbol of } \alpha\} \\
until\ \ Y = Y'
\end{array}
$$

Example

$$\begin{array}{rcl} S & \rightarrow & AB \mid a \\ A & \rightarrow & b \end{array}$$

- The fixed point iteration for finding generating symbols:

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 $A \rightarrow X_1 \dots X_k$, then the set includes $X_1, \dots, X_k$.

$$
\begin{array}{l}
Y := \{S\} \\
repeat \\
\quad Y' := Y \\
\quad Y := Y \cup \{X_1, \dots, X_k \mid A \in Y, (A \rightarrow X_1, \dots, X_k) \in P\} \\
until \;\; Y = Y'
\end{array}
$$

Example

$$\begin{array}{rcl} S & \rightarrow & AB \mid a \\ A & \rightarrow & b \end{array}$$

- The fixed point iteration for finding reachable symbols:

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

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 $A \Rightarrow^* \epsilon$.

Nullable Variables

Definition

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

Definition (Inductive version)

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

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

$$
\begin{array}{l}
Y := \{A \mid (A \to \epsilon) \in P\} \\
\mathit{repeat} \\
\quad Y' := Y \\
\quad Y := Y \cup \{B \mid (B \to C_1 \dots C_k) \in P, C_i \in Y \text{ for every } i\} \\
\mathit{until} \;\; Y = Y'
\end{array}
$$

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 $A \to X_1 X_2 \ldots X_k$ of P, where $k \geq 1$

1. Put $A \to X_1 X_2 \ldots X_k$ 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's are nullable, do not put $A \to \epsilon$ into P_1.

Example

$$\begin{array}{rcl} S & \rightarrow & AB \\ A & \rightarrow & aAA \mid \epsilon \\ B & \rightarrow & bBB \mid \epsilon \end{array}$$

- The set of nullable symbols:
- The new grammar without ϵ-productions:

Eliminating Unit Productions

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

$$\begin{array}{lcl} S & \to & A \\ A & \to & a \mid b \end{array}$$

Eliminating Unit Productions

Given $G = (V, T, S, P)$,

1. Find all *unit pairs* of variables (A, B) such that $A \Rightarrow^* B$ using a sequence of unit productions only.
2. Define $G_1 = (V, T, S, P_1)$ as follows. For each unit pair (A, B), add to P_1 all the productions $A \to \alpha$ where $B \to \alpha$ is a non-unit production in P.

E.g.,

$$\begin{array}{rcl} S & \to & A \\ A & \to & a \mid b \end{array}$$

Example

$$
\begin{array}{rcl}
S & \rightarrow & Aa \mid B \\
B & \rightarrow & A \mid bb \\
A & \rightarrow & a \mid bc \mid B
\end{array}
$$

- Unit pairs:
- The grammar without unit productions:

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.

$$
\begin{array}{l}
Y := \{ \qquad\qquad\qquad\qquad \} \\
repeat \\
\quad Y' := Y \\
\quad Y := Y \cup \{ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \} \\
until \;\; Y = Y'
\end{array}
$$

Example

$$\begin{array}{rcl} S & \rightarrow & Aa \mid B \\ B & \rightarrow & A \mid bb \\ A & \rightarrow & a \mid bc \mid B \end{array}$$

The fixed point computation proceeds as follows:

$$\begin{array}{l} \{(S,S),(A,A),(B,B)\}, \\ \{(S,S),(A,A),(B,B),(S,B),(B,A),(A,B)\}, \\ \{(S,S),(A,A),(B,B),(S,B),(B,A),(A,B),(S,A)\} \\ \{(S,S),(A,A),(B,B),(S,B),(B,A),(A,B),(S,A)\} \end{array}$$

Putting them together

Apply them in the following order:

1. Eliminate ϵ-productions
2. Eliminate unit productions
3. Eliminate useless symbols

Theorem

If G is a CFG generating a language that contains at least one string other than ϵ, then there is another CFG G_1 such that $L(G_1) = L(G) - \{\epsilon\}$, and G_1 has no useless symbols, ϵ-productions, or unit-productions.

Proof.

$\square$

Chomsky Normal Form

Definition (Chomsky Normal Form)

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

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

Further, G has no useless symbols.

Putting CFG in CNF

1. Start with a grammar without useless symbols, ϵ-productions, and unit productions.
2. Each production of the grammar is either of the form $A \rightarrow a$, which is already in a form allowed by CNF, or it has a body of length 2 or more. Do the following:
   1. Arrange that all bodies of length 2 or more consist only of variables. To do so, if terminal a appears in a body of length 2 or more, replace it by a new variable, say A and add $A \rightarrow a$.
   2. Break bodies of length 3 or more into a cascade of productions, each with a body consisting of two variables. To do so, we break production $A \rightarrow B_1B_2 \ldots B_k$ into a set of productions

$$
\begin{aligned}
&A \rightarrow B_1C_1, \\
&C_1 \rightarrow B_2C_2, \\
&\ldots, \\
&C_{k-3} \rightarrow B_{k-2}C_{k-2}, \\
&C_{k-2} \rightarrow B_{k-1}B_k
\end{aligned}
$$

Summary

- Every CFG can be transformed into a CFG in CNF
- To do so,
    1. Apply ϵ-production, unit production, useless symbols eliminations
    2. Arrange and break remaining productions.