

Theory of Computation

Course note based on *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, 2nd edition, authored by Martin Davis, Ron Sigal, and Elaine J. Weyuker.

course note prepared by

Tyng-Ruey Chuang

Institute of Information Science, Academia Sinica

Department of Information Management, National Taiwan University

Week 10, Spring 2010

About This Course Note

- ▶ It is prepared for the course *Theory of Computation* taught at the National Taiwan University in Spring 2010.
- ▶ It follows very closely the book *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, 2nd edition, by Martin Davis, Ron Sigal, and Elaine J. Weyuker. Morgan Kaufmann Publishers. ISBN: 0-12-206382-1.
- ▶ It is available from Tyng-Ruey Chuang's web site:

<http://www.iis.sinica.edu.tw/~trc/>

and released under a Creative Commons
"Attribution-ShareAlike 3.0 Taiwan" license:

<http://creativecommons.org/licenses/by-sa/3.0/tw/>

Context-Free Production

Let \mathcal{V}, \mathcal{T} be a pair of disjoint alphabets. A *context-free production* on \mathcal{V}, \mathcal{T} is an expression

$$X \rightarrow h$$

where $X \in \mathcal{V}$ and $h \in (\mathcal{V} \cup \mathcal{T})^*$.

Context-Free Production

Let \mathcal{V}, T be a pair of disjoint alphabets. A *context-free production* on \mathcal{V}, T is an expression

$$X \rightarrow h$$

where $X \in \mathcal{V}$ and $h \in (\mathcal{V} \cup T)^*$.

- ▶ The elements of \mathcal{V} are called *variables*, and the elements of T are called *terminals*.
- ▶ If P stands for the production $X \rightarrow h$ and $u, v \in (\mathcal{V} \cup T)^*$, we write

$$u \Rightarrow_P v$$

to mean that there are words $p, q \in (\mathcal{V} \cup T)^*$ such that $u = pXq$ and $v = phq$.

- ▶ Productions $X \rightarrow 0$ are called *null productions*.

Context-Free Grammar

A *context-free grammar* Γ with variables \mathcal{V} and terminals T consists of a finite set of context-free productions on \mathcal{V}, T together with a designated symbol $S \in \mathcal{V}$ called the *start symbol*.

Context-Free Grammar

A *context-free grammar* Γ with variables \mathcal{V} and terminals T consists of a finite set of context-free productions on \mathcal{V}, T together with a designated symbol $S \in \mathcal{V}$ called the *start symbol*.

- ▶ Collectively, the set $\mathcal{V} \cup T$ is called the *alphabet* of Γ .
- ▶ If none of the productions of Γ is a null production, Γ is called a *positive context-free grammar*.

Derivation

If Γ is a context-free grammar with variables \mathcal{V} and terminals T , and if $u, v \in (\mathcal{V} \cup T)^*$, we write

$$u \Rightarrow_{\Gamma} v$$

to mean that $u \Rightarrow_P v$ for some production P of Γ . We write

$$u \Rightarrow_{\Gamma}^* v$$

to mean there is a sequence u_1, \dots, u_m where $u = u_1$, $u_m = v$, and

$$u_i \Rightarrow_{\Gamma} u_{i+1} \quad \text{for } 1 \leq i < m.$$

The sequence u_1, \dots, u_m is called a *derivation of v from u in Γ* .

Derivation

If Γ is a context-free grammar with variables \mathcal{V} and terminals T , and if $u, v \in (\mathcal{V} \cup T)^*$, we write

$$u \Rightarrow_{\Gamma} v$$

to mean that $u \Rightarrow_P v$ for some production P of Γ . We write

$$u \Rightarrow_{\Gamma}^* v$$

to mean there is a sequence u_1, \dots, u_m where $u = u_1$, $u_m = v$, and

$$u_i \Rightarrow_{\Gamma} u_{i+1} \quad \text{for } 1 \leq i < m.$$

The sequence u_1, \dots, u_m is called a *derivation of v from u in Γ* .

- ▶ The number m is called the length of the derivation.
- ▶ The subscript Γ in \Rightarrow_{Γ} may be omitted when no ambiguity results.

Context-Free Language

- ▶ Let Γ be a context-free grammar with terminals T and start symbol S , we define

$$L(\Gamma) = \{u \in T^* \mid S \Rightarrow^* u\}.$$

$L(\Gamma)$ is called the language *generated* by Γ .

- ▶ A Language $L \subseteq T^*$ is called *context-free* if there is a context-free grammar Γ such that $L = L(\Gamma)$.

Context-Free Language, An Example

A simple example of a context-free grammar Γ is given by $\mathcal{V} = \{S\}$, $T = \{a, b\}$, and the productions

$$S \rightarrow aSb$$

$$S \rightarrow ab$$

- ▶ Clearly, we have

$$L(\Gamma) = \{a^{[n]}b^{[n]} \mid n > 0\}.$$

- ▶ That is, the language $\{a^{[n]}b^{[n]} \mid n > 0\}$ is context-free.
- ▶ Note that $L(\Gamma)$ is not regular.
- ▶ Later we shall show that every regular language is context-free.

Positive Context-Free Grammar

- ▶ Recall that if none of the productions of a context-free grammar Γ is a null production, Γ is called a *positive context-free grammar*.
- ▶ If Γ is a positive context-free grammar, then $0 \notin L(\Gamma)$.
- ▶ The following algorithm transforms a given context-free grammar Γ into a positive context-free grammar $\bar{\Gamma}$ such that $L(\Gamma) = L(\bar{\Gamma})$ or $L(\Gamma) = L(\bar{\Gamma}) \cup \{0\}$.
 1. First we compute the *kernel* of Γ ,

$$\ker(\Gamma) = \{V \in \mathcal{V} \mid V \Rightarrow_{\Gamma}^* 0\}.$$

2. Then we obtain $\bar{\Gamma}$ by first adding all productions that can be obtained from the productions of Γ by deleting from the righthand sides one or more variables belonging to $\ker(\Gamma)$ and then deleting all null productions.

Positive Context-Free Grammar, An Example

Consider the context-free grammar Γ with productions

$$S \rightarrow XYYX, \quad S \rightarrow aX, \quad X \rightarrow 0, \quad Y \rightarrow 0.$$

We obtain a positive context-free grammar $\bar{\Gamma}$ by

1. first computing the *kernel* of Γ ,

$$\ker(\Gamma) = \{X, Y, S\}.$$

2. then obtaining the productions of $\bar{\Gamma}$ as the following:

$$S \rightarrow XYYX, \quad S \rightarrow YYX, \quad S \rightarrow XYX, \quad S \rightarrow XYY,$$

$$S \rightarrow YX, \quad S \rightarrow YY, \quad S \rightarrow XX, \quad S \rightarrow XY,$$

$$S \rightarrow X, \quad S \rightarrow Y,$$

$$S \rightarrow aX, \quad S \rightarrow a.$$

Positive Context-Free Grammar, Continued

Theorem 1.2. A language L is context-free if and only if there is a positive context-free grammar Γ such that

$$L = L(\Gamma) \quad \text{or} \quad L = L(\Gamma) \cup \{0\}.$$

Moreover, there is an algorithm that will transform a context-free grammar Δ for which $L = L(\Delta)$ into a positive context-free grammar Γ that satisfies the above equation. □

Γ -tree

Let Γ be a *positive* context-free grammar with alphabet $\mathcal{V} \cup T$, where T consists of the terminals and \mathcal{V} is the set of variables. A tree is called a Γ -tree if it satisfies the following conditions:

1. the root is labeled by a variable;
2. each vertex which is not a leaf is labeled by a variable;
3. if a vertex is labeled X and its immediate successors (i.e. children) are labeled $\alpha_1, \alpha_2, \dots, \alpha_k$ (reading from left to right), then $X \rightarrow \alpha_1 \alpha_2 \dots \alpha_k$ is a production of Γ .

Γ -tree

Let Γ be a *positive* context-free grammar with alphabet $\mathcal{V} \cup T$, where T consists of the terminals and \mathcal{V} is the set of variables. A tree is called a Γ -tree if it satisfies the following conditions:

1. the root is labeled by a variable;
2. each vertex which is not a leaf is labeled by a variable;
3. if a vertex is labeled X and its immediate successors (i.e. children) are labeled $\alpha_1, \alpha_2, \dots, \alpha_k$ (reading from left to right), then $X \rightarrow \alpha_1 \alpha_2 \dots \alpha_k$ is a production of Γ .

Let \mathcal{T} be a Γ -tree, and let v be a vertex of Γ which is labeled by the variable X . We shall speak of the *subtree* \mathcal{T}^v of \mathcal{T} *determined by* v . The vertices of \mathcal{T}^v are v , its immediate successors in \mathcal{T} , their immediate successors, and so on. Clearly, \mathcal{T}^v is itself a Γ -tree.

Derivation Tree

- ▶ If \mathcal{T} is a Γ -tree, we write $\langle \mathcal{T} \rangle$ for the word that consists of *the labels of the leaves of \mathcal{T} reading from left to right.*
- ▶ If the root of \mathcal{T} is labeled by the start symbol S of Γ and if $w = \langle \mathcal{T} \rangle$, then \mathcal{T} is called a *derivation tree for w in Γ .*
- ▶ See the tree shown in Fig. 1.1 for a derivation tree for $a^{[4]}b^{[3]}$ in the grammar shown in the same figure

Derivation Tree

- ▶ If \mathcal{T} is a Γ -tree, we write $\langle \mathcal{T} \rangle$ for the word that consists of *the labels of the leaves of \mathcal{T}* reading from left to right.
- ▶ If the root of \mathcal{T} is labeled by the start symbol S of Γ and if $w = \langle \mathcal{T} \rangle$, then \mathcal{T} is called a *derivation tree for w in Γ* .
- ▶ See the tree shown in Fig. 1.1 for a derivation tree for $a^{[4]}b^{[3]}$ in the grammar shown in the same figure

Theorem 1.3. If Γ is a positive context-free grammar, and $S \Rightarrow_{\Gamma}^* w$, then there is a derivation tree for w in Γ . □

Leftmost Derivation and Rightmost Derivation

Definition. We write $u \Rightarrow_l v$ in Γ if $u = xXy$ and $v = xzy$, where $X \rightarrow z$ is a production of Γ and $x \in T^*$. If instead, $x \in (\mathcal{V} \cup T)^*$ but $y \in T^*$, we write $u \Rightarrow_r v$. \square

- ▶ When $u \Rightarrow_l v$, it is the *leftmost* variable in u for which a substitution is made. whereas when $u \Rightarrow_r v$, it is the *rightmost* variable in u .
- ▶ A derivation

$$u_1 \Rightarrow_l u_2 \Rightarrow_l u_3 \Rightarrow_l \dots u_n$$

is called a *leftmost* derivation, and then we write $u_1 \Rightarrow_l^* u_n$. Similarly, a derivation

$$u_1 \Rightarrow_r u_2 \Rightarrow_r u_3 \Rightarrow_r \dots u_n$$

is called a *rightmost* derivation, and we write $u_1 \Rightarrow_r^* u_n$.

Leftmost Derivation and Rightmost Derivation, Examples

Consider the following positive context-free grammar

$$S \rightarrow aXbY, \quad X \rightarrow aX, \quad X \rightarrow a, \quad Y \rightarrow bY, \quad Y \rightarrow b$$

and consider the following three derivations of a^4b^3 from S :

1. $S \Rightarrow aXbY \Rightarrow a^{[2]}XbY \Rightarrow a^{[3]}XbY \Rightarrow a^{[4]}bY \Rightarrow a^{[4]}b^{[2]}Y \Rightarrow a^{[4]}b^{[3]}$
2. $S \Rightarrow aXbY \Rightarrow a^{[2]}XbY \Rightarrow a^{[2]}Xb^{[2]}Y \Rightarrow a^{[3]}Xb^{[2]}Y \Rightarrow a^{[3]}Xb^{[3]} \Rightarrow a^{[4]}b^{[3]}$
3. $S \Rightarrow aXbY \Rightarrow aXb^{[2]}Y \Rightarrow aXb^{[3]} \Rightarrow a^{[2]}Xb^{[3]} \Rightarrow a^{[3]}Xb^{[3]} \Rightarrow a^{[4]}b^{[3]}$

The first derivation is leftmost, the last is rightmost, and the second is neither.

Leftmost Derivation and Rightmost Derivation, Continued

Theorem 1.4. Let Γ be a positive context-free grammar with start symbol S and terminals T . Let $w \in T^*$. Then the following conditions are equivalent:

1. $w \in L(\Gamma)$;
2. there is a derivation tree for w in Γ ;
3. there is a leftmost derivation of w from S in Γ ;
4. there is a rightmost derivation of w from S in Γ .



Branching Context-Free Grammar

Definition. A positive context-free grammar is called *branching* if it has no productions of the form $X \rightarrow Y$, where X and Y are variables. □

Branching Context-Free Grammar

Definition. A positive context-free grammar is called *branching* if it has no productions of the form $X \rightarrow Y$, where X and Y are variables. □

Theorem 1.5. There is an algorithm that transforms a given positive context-free grammar Γ into a branching grammar Δ such that $L(\Delta) = L(\Gamma)$.

Branching Context-Free Grammar

Definition. A positive context-free grammar is called *branching* if it has no productions of the form $X \rightarrow Y$, where X and Y are variables. □

Theorem 1.5. There is an algorithm that transforms a given positive context-free grammar Γ into a branching grammar Δ such that $L(\Delta) = L(\Gamma)$.

Proof. We transform Γ into Δ in two steps. First, we eliminate from Γ all the “cycling” productions

$$X_1 \rightarrow X_2, \quad X_2 \rightarrow X_3, \quad \dots, \quad X_k \rightarrow X_1$$

and replace variables X_1, X_2, \dots, X_k in the remaining productions of Γ by a new variable X . Next, we eliminate production $X \rightarrow Y$, but add to Γ productions $X \rightarrow x$ for each word $x \in (\mathcal{V} \cup T)^*$ for which $Y \rightarrow x$ is a production of Γ .

Path in a Γ -tree

A *path* in a Γ -tree \mathcal{T} is a sequence $\alpha_1, \alpha_2, \dots, \alpha_k$ of vertices of \mathcal{T} such that α_{i+1} is an immediate successor of α_i for $u = 1, 2, \dots, k - 1$. All of the vertices on the path are called *descendants* of α_1 .

We may have two different vertices α, β lie on the same path in the derivation tree \mathcal{T} and are labeled by the same variable X . In such a case one of the vertices is a descendant of the other, say, β is a descendant of α . Therefore, \mathcal{T}^β is not only a subtree of \mathcal{T} but also of \mathcal{T}^α .

We wish to consider two important operations in the derivation tree \mathcal{T} which can be performed in this case. The two operations are called *pruning* and *splicing*.

Pruning and Splicing

- ▶ *Pruning* is the operation that removes the subtree \mathcal{T}^α from the vertex α and to graft the subtree \mathcal{T}^β in its place.
- ▶ *Splicing* is the operation that removes the subtree \mathcal{T}^β from the vertex β and to graft an exact copy of \mathcal{T}^α in its place.
- ▶ *Because α and β are labeled by the same variable, the trees obtained by pruning and splicing are themselves derivation trees.*
- ▶ See Fig. 1.3 in the textbook for illustrations of pruning and splicing.

Pruning and Splicing, Continued

Let \mathcal{T}_p and \mathcal{T}_s be trees obtained from a derivation tree \mathcal{T} in a branching grammar by pruning and splicing, respectively, where α and β are as before.

We have $\langle \mathcal{T} \rangle = r_1 \langle \mathcal{T}^\alpha \rangle r_2$ for words r_1, r_2 and $\langle \mathcal{T}^\alpha \rangle = q_1 \langle \mathcal{T}^\beta \rangle q_2$ for words q_1, q_2 . Since α, β are distinct vertices, and since the grammar is branching, q_1 and q_2 cannot both be 0. (That is, $q_1 q_2 \neq 0$.)

Also,

$$\langle \mathcal{T}_p \rangle = r_1 \langle \mathcal{T}^\beta \rangle r_2 \quad \text{and} \quad \langle \mathcal{T}_s \rangle = r_1 q_1^{[2]} \langle \mathcal{T}^\beta \rangle q_2^{[2]} r_2.$$

Since $q_1 q_2 \neq 0$, we have $|\langle \mathcal{T}^\beta \rangle| < |\langle \mathcal{T}^\alpha \rangle|$ and hence $|\langle \mathcal{T}_p \rangle| < |\langle \mathcal{T} \rangle|$.

Pruning and Splicing, Continued

Theorem 1.6. Let Γ be a branching context-free grammar, let $u \in L(\Gamma)$, and let u have a derivation tree \mathcal{T} in Γ that has two different vertices on the same path labeled by the same variable. Then there is a word $v \in L(\Gamma)$ such that $|v| < |u|$.

Proof. Since $u = \langle \mathcal{T} \rangle$, we need only take $v = \langle \mathcal{T}_p \rangle$. □

Regular Grammars

Definition. A context-free grammar is called *regular* if each of its productions has one of the two forms

$$U \rightarrow aV \quad \text{or} \quad U \rightarrow a$$

where U, V are variables and a is a terminal. □

Theorem 2.1. If L is a regular language, then there is a regular grammar Γ such that either $L = L(\Gamma)$ or $L = L(\Gamma) \cup \{0\}$. □

A Regular Grammar for Every Regular Language

Proof of Theorem 2.1. Let $L = (M)$, where M is a dfa with states q_1, \dots, q_m , alphabet $\{s_1, \dots, s_n\}$, transition function δ , and the set of accepting states F . We construct a grammar Γ with variables q_1, \dots, q_m , terminals s_1, \dots, s_n , and start symbol q_1 . The productions are

1. $q_i \rightarrow s_r q_j$ whenever $\delta(q_i, s_r) = q_j$, and
2. $q_i \rightarrow s_r$ whenever $\delta(q_i, s_r) \in F$.

Clearly the grammar Γ is regular. To show that $L(\Gamma) = L - \{0\}$ we suppose $u \in L$, $u = s_{i_1} s_{i_2} \dots s_{i_l} s_{i_{l+1}} \neq 0$. Thus, $\delta^*(q_1, u) \in F$, so that we have

$$\delta(q_1, s_{i_1}) = q_{j_1}, \quad \delta(q_{j_1}, s_{i_2}) = q_{j_2}, \quad \dots, \quad \delta(q_{j_l}, s_{i_{l+1}}) = q_{j_{l+1}} \in F.$$

A Regular Grammar for Every Regular Language, Continued

Proof of Theorem 2.1. (Continued) By construction, grammar Γ contains the productions

$$q_1 \rightarrow s_{i_1} q_{j_1}, \quad q_{j_1} \rightarrow s_{i_2} q_{j_2}, \quad \dots, \quad q_{j_{l-1}} \rightarrow s_{i_l} q_{j_l}, \quad q_{j_l} \rightarrow s_{i_{l+1}}.$$

Thus, we have in Γ

$$q_1 \Rightarrow s_{i_1} q_{j_1} \Rightarrow s_{i_1} s_{i_2} q_{j_2} \Rightarrow \dots \Rightarrow s_{i_1} s_{i_2} \dots s_{i_l} q_{j_l} \Rightarrow s_{i_1} s_{i_2} \dots s_{i_l} s_{i_{l+1}} = u$$

so that $u \in L(\Gamma)$.

Conversely, suppose that $u \in L(\Gamma)$, $u = s_{i_1} s_{i_2} \dots s_{i_l} s_{i_{l+1}}$. Then there is a derivation of u from q_1 in Γ . By construction, Γ has all the necessary productions to simulate the transition $\delta^*(q_1, u) \in F$ in the dfa \mathcal{M} . □

A Regular Language for Every Regular Grammar

Theorem 2.2. Let Γ be a regular grammar. Then $L(\Gamma)$ is a regular language.

Proof. Let Γ have the variables V_1, V_2, \dots, V_K , where $S = V_1$ is the start symbol, and terminals s_1, s_2, \dots, s_n . Since Γ is regular, its productions are of the form $V_i \rightarrow s_r V_j$ and $V_i \rightarrow s_r$. We now construct the following ndfa \mathcal{M} which accepts precisely $L(\Gamma)$.

- ▶ The states are V_1, V_2, \dots, V_K and an additional state W . V_1 is the initial state and W is the only accepting state.
- ▶ For transition functions, let

$$\begin{aligned} \delta_1(V_i, s_r) &= \{V_j \mid V_i \rightarrow s_r V_j \text{ is a production of } \Gamma\}, \\ \delta_2(V_i, s_r) &= \begin{cases} \{W\} & \text{if } V_i \rightarrow s_r \text{ is a production of } \Gamma \\ \emptyset & \text{otherwise.} \end{cases} \end{aligned}$$

Then define the transition function δ as

$$\delta(V_i, s_r) = \delta_1(V_i, s_r) \cup \delta_2(V_i, s_r).$$

A Regular Language for Every Regular Grammar

Proof of Theorem 2.2. (Continued) Now let

$u = s_{i_1}s_{i_2} \dots s_{i_l}s_{i_{l+1}} \in L(\Gamma)$. Thus we have

$$V_1 \Rightarrow s_{i_1} V_{j_1} \Rightarrow s_{i_1}s_{i_2} V_{j_2} \Rightarrow^* s_{i_1}s_{i_2} \dots s_{i_l} V_{j_l} \Rightarrow s_{i_1}s_{i_2} \dots s_{i_l}s_{i_{l+1}}$$

where Γ contains the productions

$$V_1 \rightarrow s_{i_1} V_{j_1}, \quad V_{j_1} \rightarrow s_{i_2} V_{j_2}, \quad \dots, \quad V_{j_{l-1}} \rightarrow s_{i_l} V_{j_l}, \quad V_{j_l} \rightarrow s_{i_{l+1}}$$

Thus,

$$V_{j_1} \in \delta(V_1, s_{i_1}), \quad V_{j_2} \in \delta(V_{j_1}, s_{i_2}), \quad \dots, \quad W \in \delta(V_{j_l}, s_{i_{l+1}}).$$

Thus $W \in \delta^*(V_1, u)$ and $u \in L(\mathcal{M})$.

Conversely, if $u = s_{i_1}s_{i_2} \dots s_{i_l}s_{i_{l+1}}$ is accepted by \mathcal{M} , then there must be a sequence of transitions of the form above. Hence, the productions listed above must all belong to Γ , so that there is a derivation of u from V_1 .

Every Regular Language Is Context-free

Theorem 2.3. A language L is regular if and only if there is a regular grammar Γ such that either $L = L(\Gamma)$ or $L = L(\Gamma) \cup \{0\}$. \square

Every Regular Language Is Context-free

Theorem 2.3. A language L is regular if and only if there is a regular grammar Γ such that either $L = L(\Gamma)$ or $L = L(\Gamma) \cup \{0\}$. \square

Corollary 2.4. Every regular language is context-free. \square

Right-linear Grammars

Definition. A context-free grammar is called *right-linear* if each of its productions has one of the two forms

$$U \rightarrow xV \quad \text{or} \quad U \rightarrow x,$$

where U, V are variables and $x \neq \epsilon$ is a word consisting entirely of terminals. □

Right-linear Grammars

Definition. A context-free grammar is called *right-linear* if each of its productions has one of the two forms

$$U \rightarrow xV \quad \text{or} \quad U \rightarrow x,$$

where U, V are variables and $x \neq \epsilon$ is a word consisting entirely of terminals. \square

Thus, a regular grammar is just a right-linear grammar in which $|x| = 1$.

Right-linear Grammars, Continued

Theorem 2.5. Let Γ be a right-linear grammar. Then $L(\Gamma)$ is regular.

Proof. We replace each production of Γ of the form

$$U \rightarrow a_1 a_2 \dots a_n V, \quad n > 1$$

by the productions

$$U \rightarrow a_1 Z_1, \quad Z_1 \rightarrow a_2 Z_2, \quad Z_{n-2} \rightarrow a_{n-1} Z_{n-1}, \quad Z_{n-1} \rightarrow a_n V,$$

where Z_1, \dots, Z_{n-1} are new variables. Do similar replacement for production

$$U \rightarrow a_1 a_2 \dots a_n, \quad n > 1$$



Chomsky Normal Form

Definition. A context-free grammar Γ with variables \mathcal{V} and terminals T is in *Chomsky normal form* if each of its productions has one of the forms

$$X \rightarrow YZ \quad \text{or} \quad X \rightarrow a,$$

where $X, Y, Z \in \mathcal{V}$ and $a \in T$. □

Theorem 3.1. There is an algorithm that transforms a given positive context-free grammar Γ into a Chomsky normal form grammar Δ such that $L(\Gamma) = L(\Delta)$. □

Chomsky Normal Form, Continued

Proof of Theorem 3.1. Using Theorem 1.5, we begin with a branching context-free grammar Γ with variable \mathcal{V} and terminals T . We then perform the following two steps:

1. a new variable X_a is introduced for each $a \in T$, and for each production $X \rightarrow x \in \Gamma, |x| > 1$, we replace it with $X \rightarrow x'$ where x' is obtained from x by replacing each terminal a by the corresponding new variable X_a ;
2. For productions of the form $X \rightarrow X_1 X_2 \dots X_k, k > 2$, we introduce new variables Z_1, Z_2, \dots, Z_{k-2} and replace the production with the following

$$X \rightarrow X_1 Z_1$$

$$\vdots$$

$$Z_{k-3} \rightarrow X_{k-2} Z_{k-2}$$

$$Z_{k-2} \rightarrow X_{k-1} X_k$$

Chomsky Normal Form, Examples

Consider the following branching context-free grammar

$$S \rightarrow aXbY, \quad X \rightarrow aX, \quad Y \rightarrow bY, \quad X \rightarrow a, \quad Y \rightarrow b$$

The resulting grammar, respectively, from the two steps is:

1.

$$S \rightarrow X_aXX_bY, \quad X \rightarrow X_aX, \quad Y \rightarrow X_bY, \\ X \rightarrow a, \quad X_a \rightarrow a, \quad Y \rightarrow b, \quad X_b \rightarrow b$$

2. For the production $S \rightarrow X_aXX_bY$, we replace it with the following:

$$S \rightarrow X_aZ_1 \\ Z_1 \rightarrow XZ_2 \\ Z_2 \rightarrow X_bY.$$

The resulting grammar is in Chomsky normal form.

Bar-Hillel's Pumping Lemma

An application of Chomsky normal form is in the proof of the following theorem, which is an analogy for context-free languages of the pumping lemma for regular languages.

Theorem 4.1. Let Γ be a Chomsky normal form grammar with exactly n variables, and let $L = L(\Gamma)$. Then, for every $x \in L$ for which $|x| > 2^n$, we have $x = r_1 q_1 r q_2 r_2$, where

1. $|q_1 r q_2| \leq 2^n$;
2. $q_1 q_2 \neq \epsilon$;
3. for all $i \geq 0$, $r_1 q_1^{[i]} r q_2^{[i]} r_2 \in L$.



A Small Lemma

Lemma. Let $S \Rightarrow_{\Gamma}^* u$, where Γ is a Chomsky normal form grammar. Suppose that \mathcal{T} is a derivation tree for u in Γ and that no path in \mathcal{T} contains more than k nodes. Then $|u| \leq 2^{k-2}$.

Proof. First, suppose, that \mathcal{T} has just one leaf labeled by a terminal a . Then $u = a$, and \mathcal{T} just have two nodes, S and a , and one path of length $1 < k = 2$. Clearly $|u| = 1 \leq 2^{2-2}$.

Otherwise, since Γ is in Chomsky normal form, the root of \mathcal{T} is labeled by S where $S \rightarrow XY$ for variables X and Y . Let \mathcal{T}_1 and \mathcal{T}_2 be the two trees whose roots are labeled by X and Y , respectively. In each of \mathcal{T}_1 and \mathcal{T}_2 , the longest path must contain $\leq k - 1$ nodes. Proceeding inductively, we may assume that each of the $\mathcal{T}_1, \mathcal{T}_2$ have $\leq 2^{k-3}$ leaves. Hence

$$|u| \leq 2^{k-3} + 2^{k-3} = 2^{k-2}.$$

Bar-Hillel's Pumping Lemma, Proof

Proof of Theorem 4.1. Let $x \in L$, where $|x| > 2^n$, and let \mathcal{T} be a derivation tree for x in Γ . Let $\alpha_1, \alpha_2, \dots, \alpha_m$ be the longest path in \mathcal{T} . Then $m \geq n + 2$ and α_m is a leaf. This is because, if $m \leq n + 1$, by the small lemma, $|x| \leq 2^n - 1$ is a contradiction.

Note that $\alpha_1, \alpha_2, \dots, \alpha_{m-1}$ are all labeled by variables, while α_m is labeled by a terminal. Let $\gamma_1, \gamma_2, \dots, \gamma_{n+2}$ be the path consisting of the vertices $\alpha_{m-n-1}, \alpha_{m-n-2}, \dots, \alpha_{m-1}, \alpha_m$.

Since there are only n variables in the alphabet of Γ , the pigeon-hole principle guarantees that there is a variable X that labels two different vertices: $\alpha = r_i$ and $\beta = r_j$, where $i < j$. (See Fig. 4.2.)

Bar-Hillel's Pumping Lemma, Proof

(Proof of Theorem 4.1., Continued)

Hence, the operations of *pruning* and *splicing* can be applied. Let $r = \langle \mathcal{I}^\beta \rangle$. Then we have, for example,

$$\begin{aligned}\langle \mathcal{I}_p \rangle &= r_1 r r_2, \\ \langle \mathcal{I}_s \rangle &= r_1 q_1^{[2]} r q_2^{[2]} r_2, \\ \langle (\mathcal{I}_s)_s \rangle &= r_1 q_1^{[3]} r q_2^{[3]} r_2\end{aligned}$$

That is, $r_1 q_1^i r q_2^i r_2 \in L(\Gamma), i \geq 0$. Note that the path in \mathcal{I}^α consists of $\leq n + 2$ nodes, so by the small lemma $|q_1 r q_2| = |q_1 \langle \mathcal{I}^\beta \rangle q_2| = |\langle \mathcal{I}^\alpha \rangle| \leq 2^n$. □

Bar-Hillel's Pumping Lemma, Application

Theorem 4.2. The language $L = \{a^{[n]}b^{[n]}c^{[n]} \mid n > 0\}$ is *not* context-free.

Bar-Hillel's Pumping Lemma, Application

Theorem 4.2. The language $L = \{a^{[n]}b^{[n]}c^{[n]} \mid n > 0\}$ is *not* context-free.

Proof. Suppose that L is context-free with $L = L(\Gamma)$, where Γ is a Chomsky normal form grammar with n variables. Choose k so large that $|a^{[k]}b^{[k]}c^{[k]}| > 2^n$. Then $a^{[k]}b^{[k]}c^{[k]} = r_1 q_1^{[i]} r q_2^{[j]} r_2$, where

$$x_i = r_1 q_1^{[i]} r q_2^{[j]} r_2 \in L$$

for all $i \geq 0$. As $x_2 = r_1 q_1 q_1 r q_2 q_2 r_2 \in L$, we know that q_1 and q_2 must each contain only one of the letters a, b, c . That is, one letter is missing in both q_1 and q_2 .

But as $i = 2, 3, 4, \dots$ contains more and more copies of q_1 and q_2 and since $q_1 q_2 \neq \emptyset$, it is impossible for x_i to have the same number of occurrences of a, b , and c . This contradiction shows that L is not context-free.