COT 5310: Theory of Automata and Formal Languages

Lecture 9

Florida State University
Department of Computer Science

Slides Credit: Dr. Michael Mascagni, CS FSU

Context-Free Production

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

$$X \to h$$

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

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

  $$u \Rightarrow_P v$$

  to mean that there are words $p, q \in (\mathscr{V} \cup T)^*$ such that $u = pXq$ and $v = phq$.
- Productions $X \to 0$ are called *null productions*.

Context-Free Grammar

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

- Collectively, the set $\mathscr{V} \cup T$ 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 $\mathscr{V}$ and terminals T, and if $u, v \in (\mathscr{V} \cup T)^*$, 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 $u_1, \ldots, u_m$ where $u = u_1$, $u_m = v$, and

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

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

- The number m is called the length of the derivation.
- The subscript Γ in $\Rightarrow_\Gamma$ 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* is 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 $\mathscr{V} = \{S\}$, $T = \{a, b\}$, 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,
  $$\ker(\Gamma) = \{V \in \mathscr{V} \mid V \Rightarrow_{\Gamma}^{*} 0\}.$$
  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

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

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:

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

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

$$S \to X, \quad S \to Y,$$

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

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 $\mathscr{V} \cup T$, where T consists of the terminals and $\mathscr{V}$ 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 $\alpha_1, \alpha_2, \ldots, \alpha_k$ (reading from left to right), then $X \to \alpha_1\alpha_2 \ldots \alpha_k$ is a production of Γ.

Let $\mathscr{T}$ 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* $\mathscr{T}^v$ *of* $\mathscr{T}$ *determined by* v. The vertices of $\mathscr{T}^v$ are v, its immediate successors in $\mathscr{T}$, their immediate successors, and so on. Clearly, $\mathscr{T}^v$ is itself a Γ-tree.

Derivation Tree

- If $\mathscr{T}$ is a Γ-tree, we write $\langle \mathscr{T} \rangle$ for the word that consists of *the labels of the leaves of* $\mathscr{T}$ reading from left to right.
- If the root of $\mathscr{T}$ is labeled by the start symbol symbol S of Γ and if $w = \langle \mathscr{T} \rangle$, then $\mathscr{T}$ 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, $x \in (\mathscr{V} \cup T)^*$ but $y \in T^*$, we write $u \Rightarrow_r v$. □

- 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

$$u_1 \Rightarrow_l u_2 \Rightarrow_l u_3 \Rightarrow_l \ldots u_n$$

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

$$u_1 \Rightarrow_r u_2 \Rightarrow_r u_3 \Rightarrow_r \ldots u_n$$

  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

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

and consider the following three derivations of $a^{[4]}b^{[3]}$ 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 $X \to Y$, 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

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

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 $X \to Y$, but add to Γ productions $X \to x$ for each word $x \in (\mathscr{V} \cup T)^*$ for which $Y \to x$ is a production of Γ. □

Path in a Γ-tree

A *path* in a Γ-tree $\mathscr{T}$ is a sequence $\alpha_1, \alpha_2, \ldots, \alpha_k$ of vertices of $\mathscr{T}$ such that α_{i+1} is an immediate successor of α_i for $i = 1, 2, \ldots, k-1$. 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 $\mathscr{T}$ 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, $\mathscr{T}^{\beta}$ is not only a subtree of $\mathscr{T}$ but also of $\mathscr{T}^{\alpha}$.

We wish to consider two important operations in the derivation tree $\mathscr{T}$ 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 $\mathscr{T}^\alpha$ from the vertex α and to graft the subtree $\mathscr{T}^\beta$ in its place.
- *Splicing* is the operation that removes the subtree $\mathscr{T}^\beta$ from the vertex β and to graft an exact copy of $\mathscr{T}^\alpha$ 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 $\mathscr{T}_p$ and $\mathscr{T}_s$ be trees obtained from a derivation tree $\mathscr{T}$ in a branching grammar by pruning and splicing, respectively, where α and β are as before.

We have $\langle \mathscr{T} \rangle = r_1 \langle \mathscr{T}^\alpha \rangle r_2$ for words r_1, r_2 and $\langle \mathscr{T}^\alpha \rangle = q_1 \langle \mathscr{T}^\beta \rangle q_2$ 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,

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

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

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

Regular Grammars

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

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

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 $L = L(\mathscr{M})$, where $\mathscr{M}$ 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. $q_i \to s_r q_j$ whenever $\delta(q_i, s_r) = q_j$, and
2. $q_i \to s_r$ 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

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

Thus, we have in Γ

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

so that $u \in L(\Gamma)$.
Conversely, suppose that $u \in L(\Gamma), u = s_{i_1} s_{i_2} \ldots s_{i_l} s_{i_{l+1}}$. 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 $\mathscr{M}$. □

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

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

$$\delta_1(V_i, s_r) = \{V_j \mid V_i \to s_r V_j \text{ is a production of } \Gamma\},$$

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

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

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

where Γ contains the productions

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

Thus,

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

Thus $W \in \delta^*(V_1, u)$ and $u \in L(\mathscr{M})$.
Conversely, if $u = s_{i_1} s_{i_2} \ldots s_{i_l} s_{i_{l+1}}$ is accepted by $\mathscr{M}$, 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 $L = L(\Gamma) \cup \{0\}$. □

Corollary 2.4. Every regular language is context-free. □

Right-linear Grammars

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

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

where U, V are variables and $x \neq 0$ is a word consisting entirely of terminals. □

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

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

by the productions

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

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

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

□

Chomsky Normal Form

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

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

where $X, Y, Z \in \mathscr{V}$ 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 $\mathscr{V}$ 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 $X \to x \in \Gamma, |x| > 1$, we replace it with $X \to x'$ where x' is obtained from x by replacing each terminal a by the corresponding new variable X_a;
2. For productions of the form $X \to X_1 X_2 \ldots X_k, k > 2$, we introduce new variables $Z_1, Z_2, \ldots, Z_{k-2}$ and replace the production with the following

$$\begin{aligned} X &\to X_1 Z_1 \\ &\vdots \\ Z_{k-3} &\to X_{k-2} Z_{k-2} \\ Z_{k-2} &\to X_{k-1} X_k . \end{aligned}$$

Chomsky Normal Form, Examples

Consider the following branching context-free grammar

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

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

1. $$S \to X_a X X_b Y, \quad X \to X_a X, \quad Y \to X_b Y,$$
   $$X \to a, \quad X_a \to a, \quad Y \to b, \quad X_b \to b$$
2. For the production $S \to X_a X X_b Y$, we replace it with the following:

$$\begin{array}{rcl} S & \to & X_a Z_1 \\ Z_1 & \to & X Z_2 \\ Z_2 & \to & X_b Y. \end{array}$$

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

□

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 $a^{[k]}b^{[k]}c^{[k]} = r_1q_1rq_2r_2$, where

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

for all $i \geq 0$. As $x_2 = r_1q_1q_1rq_2q_2r_2 \in L$, 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 $i = 2, 3, 4, \ldots$ contains more and more copies of q_1 and q_2 and since $q_1q_2 \neq 0$, 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. □