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 12, Spring 2008

About This Course Note

- It is prepared for the course *Theory of Computation* taught at the National Taiwan University in Spring 2008.
- 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 2.5 Taiwan" license:

  `http://creativecommons.org/licenses/by-sa/2.5/tw/`

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

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*.

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*.

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

$$\begin{aligned} S &\to aSb \\ S &\to ab \end{aligned}$$

- 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.
- Late 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

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

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:

$$S \to XYYYX, \quad S \to YYYX, \quad S \to XYYX, \quad S \to XYYY,$$

$$S \to YYX, \quad S \to YYY, \quad S \to XXX, \quad S \to XXY,$$

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

$$S \to YXX, \quad S \to YXY,$$

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

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

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

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

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.

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

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 $X \to z$ 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. □

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 $L(\Delta) = L(\Gamma)$.

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

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

and replace variables $X_1, X_2, \ldots, X_k$ 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 $u = 1, 2, \ldots, k-1$. 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 $v = \langle \mathscr{T}_p \rangle$. □