

# *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

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/>

## 1 Context-Free Languages (10)

### 1.1 Context-Free Grammars and Their Derivation Trees (10.1)

#### 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  $X \rightarrow \epsilon$  are called *null productions*.

### Context-Free Grammar

A *context-free grammar*  $\Gamma$  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  $\Gamma$ .
- If none of the productions of  $\Gamma$  is a null production,  $\Gamma$  is called a *positive context-free grammar*.

### Derivation

If  $\Gamma$  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  $\Gamma$ . 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  $\Gamma$* .

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

### Context-Free Language

- Let  $\Gamma$  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  $\Gamma$ .

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

## Context-Free Language, An Example

A simple example of a context-free grammar  $\Gamma$  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  $\Gamma$  is a null production,  $\Gamma$  is called a *positive context-free grammar*.
- If  $\Gamma$  is a positive context-free grammar, then  $0 \notin L(\Gamma)$ .
- The following algorithm transforms a given context-free grammar  $\Gamma$  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  $\Gamma$ ,

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

2. Then we obtain  $\bar{\Gamma}$  by first adding all productions that can be obtained from the productions of  $\Gamma$  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  $\Gamma$  with productions

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

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

1. first computing the *kernel* of  $\Gamma$ ,

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

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

$$\begin{aligned} S &\rightarrow XYYX, \quad S \rightarrow YYX, \quad S \rightarrow XYX, \quad S \rightarrow XYY, \\ S &\rightarrow YX, \quad S \rightarrow YY, \quad S \rightarrow XX, \quad S \rightarrow XY, \\ S &\rightarrow X, \quad S \rightarrow Y, \\ S &\rightarrow aX, \quad S \rightarrow a. \end{aligned}$$

### 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  $\Gamma$  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  $\Delta$  for which  $L = L(\Delta)$  into a positive context-free grammar  $\Gamma$  that satisfies the above equation.  $\square$

### $\Gamma$ -tree

Let  $\Gamma$  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  $\Gamma$ -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  $\Gamma$ .

Let  $\mathcal{T}$  be a  $\Gamma$ -tree, and let  $v$  be a vertex of  $\Gamma$  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  $\Gamma$ -tree.

### Derivation Tree

- If  $\mathcal{T}$  is a  $\Gamma$ -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  $\Gamma$  and if  $w = \langle \mathcal{T} \rangle$ , then  $\mathcal{T}$  is called a *derivation tree for  $w$  in  $\Gamma$* .
- 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  $\Gamma$  is a positive context-free grammar, and  $S \Rightarrow_{\Gamma}^* w$ , then there is a derivation tree for  $w$  in  $\Gamma$ .  $\square$

## Leftmost Derivation and Rightmost Derivation

**Definition.** We write  $u \Rightarrow_l v$  in  $\Gamma$  if  $u = xXy$  and  $v = xzy$ , where  $X \rightarrow z$  is a production of  $\Gamma$  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  $a^{[4]}b^{[3]}$  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  $\Gamma$  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  $\Gamma$ ;
3. there is a leftmost derivation of  $w$  from  $S$  in  $\Gamma$ ;
4. there is a rightmost derivation of  $w$  from  $S$  in  $\Gamma$ .

$\square$

## 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.  $\square$

**Theorem 1.5.** There is an algorithm that transforms a given positive context-free grammar  $\Gamma$  into a branching grammar  $\Delta$  such that  $L(\Delta) = L(\Gamma)$ . *Proof.* We transform  $\Gamma$  into  $\Delta$  in two steps. First, we eliminate from  $\Gamma$  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  $\Gamma$  by a new variable  $X$ . Next, we eliminate production  $X \rightarrow Y$ , but add to  $\Gamma$  productions  $X \rightarrow x$  for each word  $x \in (\mathcal{V} \cup T)^*$  for which  $Y \rightarrow x$  is a production of  $\Gamma$ .  $\square$

## Path in a $\Gamma$ -tree

A *path* in a  $\Gamma$ -tree  $\mathcal{T}$  is a sequence  $\alpha_1, \alpha_2, \dots, \alpha_k$  of vertices of  $\mathcal{T}$  such that  $\alpha_{i+1}$  is an immediate successor of  $\alpha_i$  for  $i = 1, 2, \dots, k-1$ . All of the vertices on the path are called *descendants* of  $\alpha_1$ . We may have two different vertices  $\alpha, \beta$  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,  $\beta$  is a descendant of  $\alpha$ . Therefore,  $\mathcal{T}^\beta$  is not only a subtree of  $\mathcal{T}$  but also of  $\mathcal{T}^\alpha$ . 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  $\mathcal{T}^\alpha$  from the vertex  $\alpha$  and to graft the subtree  $\mathcal{T}^\beta$  in its place.
- *Splicing* is the operation that removes the subtree  $\mathcal{T}^\beta$  from the vertex  $\beta$  and to graft an exact copy of  $\mathcal{T}^\alpha$  in its place.
- *Because  $\alpha$  and  $\beta$  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  $\alpha$  and  $\beta$  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  $\alpha, \beta$  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  $\Gamma$  be a branching context-free grammar, let  $u \in L(\Gamma)$ , and let  $u$  have a derivation tree  $\mathcal{T}$  in  $\Gamma$  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  $v = \langle \mathcal{T}_p \rangle$ .  $\square$