Formal languages and Automata
形式语言与自动机

Chapter3 REGULAR LANGUAGES AND REGULAR GRAMMARS

Beijing University of Posts and Telecommunications

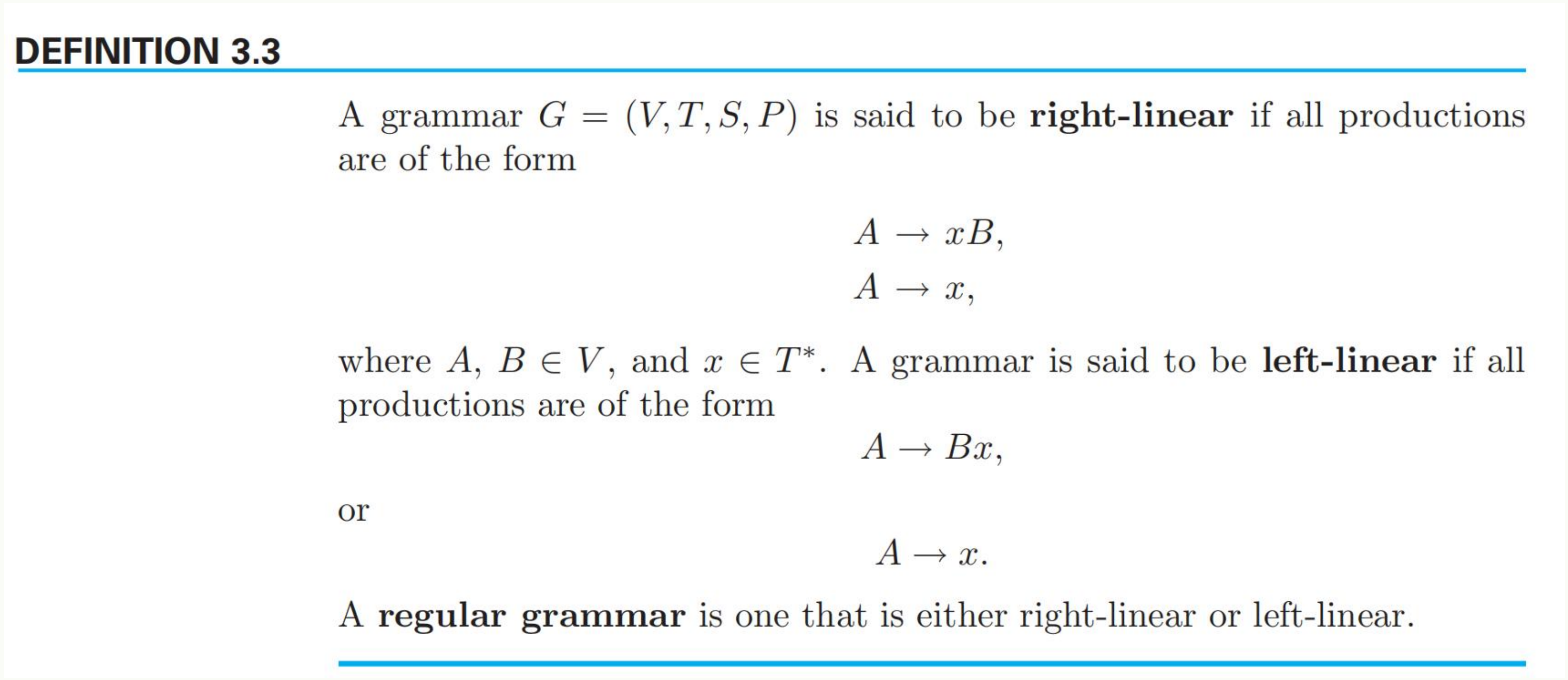
北京邮电大学

刘咏彬 liuyb@bupt.edu.cn

2023.10

3.3 REGULAR GRAMMARS

- **Right- and Left-Linear Grammars**



DEFINITION 3.3

A grammar $G = (V, T, S, P)$ is said to be **right-linear** if all productions are of the form

$$A \to xB,$$
$$A \to x,$$

where A, $B \in V$, and $x \in T^*$. A grammar is said to be **left-linear** if all productions are of the form

$$A \to Bx,$$

or

$$A \to x.$$

A **regular grammar** is one that is either right-linear or left-linear.

- At most one variable appears on the right side of any production
- Furthermore, that variable must consistently be either the rightmost or leftmost symbol of the right side of any pro duction.

EXAMPLE 3.13

The grammar $G_1 = (\{S\}, \{a,b\}, S, P_1)$, with P_1 given as

$$S \rightarrow abS|a$$

is right-linear. The grammar $G_2 = (\{S, S_1, S_2\}, \{a,b\}, S, P_2)$, with productions

$$\begin{aligned} S &\rightarrow S_1ab, \\ S_1 &\rightarrow S_1ab|S_2, \\ S_2 &\rightarrow a, \end{aligned}$$

is left-linear. Both G_1 and G_2 are regular grammars.

The sequence

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$$

is a derivation with G_1. From this single instance it is easy to conjecture that $L(G_1)$ is the language denoted by the regular expression $r = (ab)^* a$. In a similar way, we can see that $L(G_2)$ is the regular language $L\left(aab(ab)^*\right)$.

EXAMPLE 3.14

The grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\rightarrow A, \\ A &\rightarrow aB|\lambda, \\ B &\rightarrow Ab, \end{aligned}$$

is not regular. Although every production is either in right-linear or left-linear form, the grammar itself is neither right-linear nor left-linear, and therefore is not regular. The grammar is an example of a **linear grammar**. A linear grammar is a grammar in which at most one variable can occur on the right side of any production, without restriction on the position of this variable. Clearly, a regular grammar is always linear, but not all linear grammars are regular.

Right-linear grammar->nfa

- Right-linear grammars generate regular languages
- Now we construct an nfa that mimics the derivations of a right-linear grammar.
  - A step in a derivation
    $$ab \cdots cD \Rightarrow ab \cdots cdE,$$
    by using a production $D \to dE$.
  - The corresponding nfa can imitate this step by going from state D to state E when a symbol d is encountered.
  - The state of the automaton corresponds to the variable in the sentential form, while the part of the string already processed is identical to the terminal prefix of the sentential form（句型的终极符前缀）.

THEOREM 3.3

THEOREM 3.3

Let $G = (V, T, S, P)$ be a right-linear grammar. Then $L(G)$ is a regular language.

Proof: We assume that $V = \{V_0, V_1, ...\}$, that $S = V_0$, and that we have productions of the form $V_0 \to v_1 V_i, V_i \to v_2 V_j, ...$ or $V_n \to v_l, ...$ If w is a string in $L(G)$, then because of the form of the productions

$$\begin{aligned} V_0 &\Rightarrow v_1 V_i \\ &\Rightarrow v_1 v_2 V_j \\ &\overset{*}{\Rightarrow} v_1 v_2 \cdots v_k V_n \\ &\Rightarrow v_1 v_2 \cdots v_k v_l = w. \end{aligned} \tag{3.4}$$

The automaton to be constructed will reproduce the derivation by consuming each of these v's in turn. The initial state of the automaton will be labeled V_0, and for each variable V_i there will be a nonfinal state labeled V_i. For each production

$$V_i \to a_1 a_2 \cdots a_m V_j,$$

the automaton will have transitions to connect V_i and V_j that is, δ will be defined so that

$$\delta^* (V_i, a_1 a_2 \cdots a_m) = V_j.$$

THEOREM 3.3

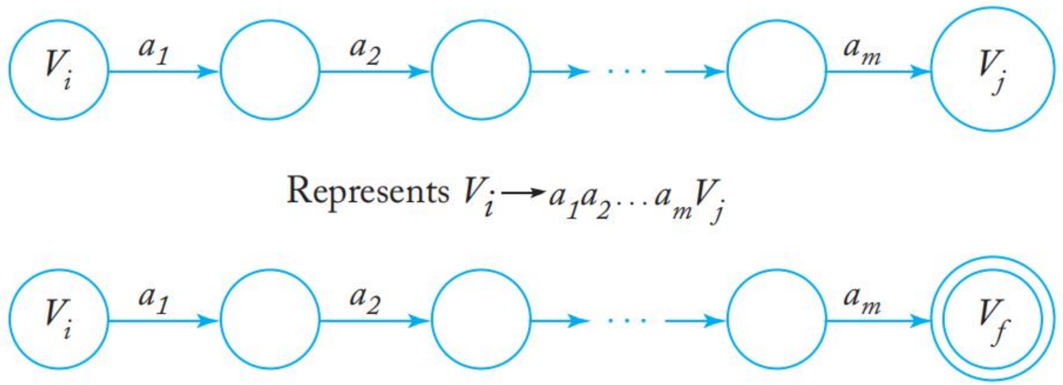
For each production

$$V_i \rightarrow a_1 a_2 \cdots a_m,$$

the corresponding transition of the automaton will be

$$\delta^* (V_i, a_1 a_2 \cdots a_m) = V_f,$$

where V_f is a final state. The intermediate states that are needed to do this are of no concern and can be given arbitrary labels. The general scheme is shown in Figure 3.16. The complete automaton is assembled from such individual parts.

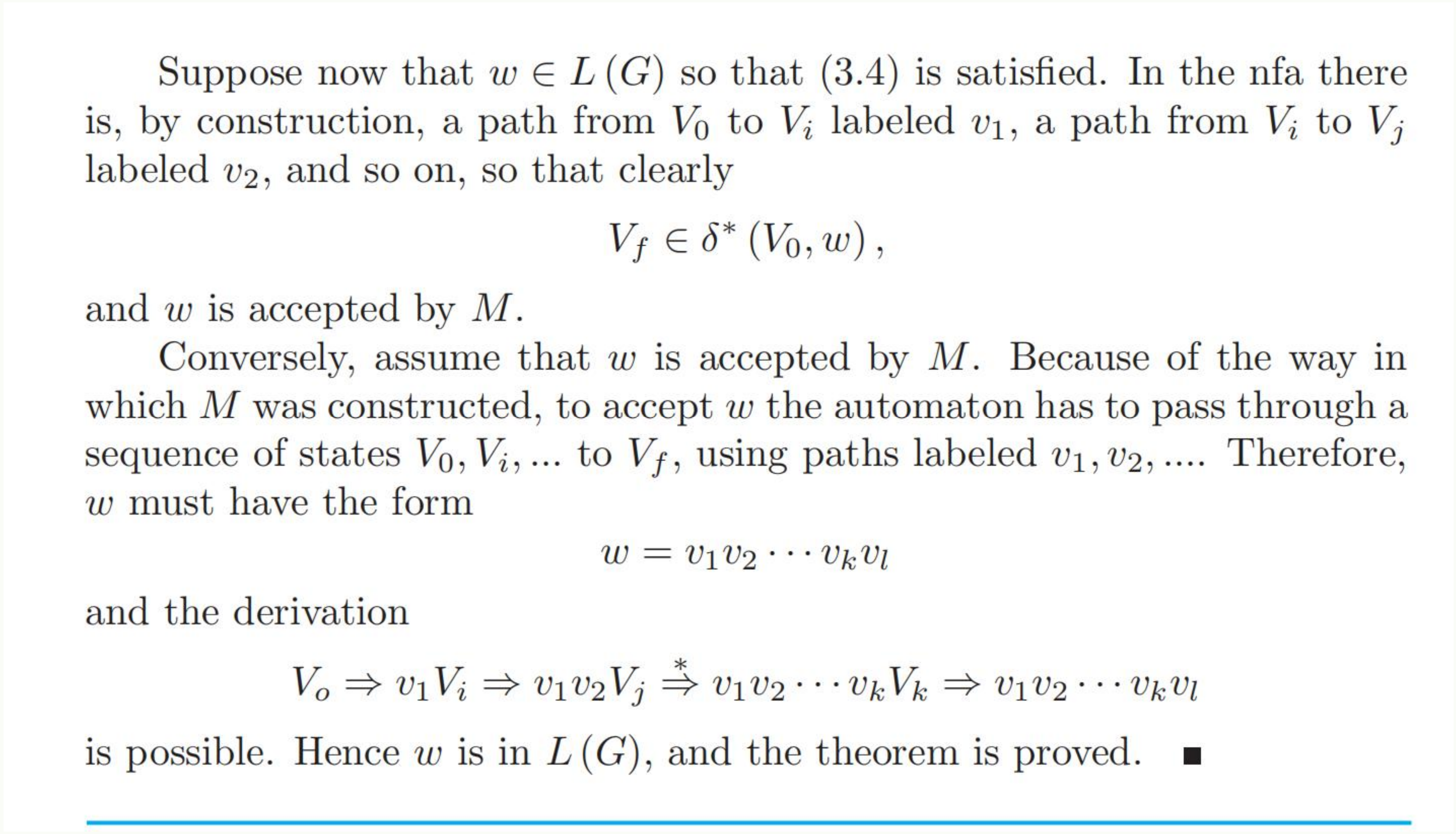


Represents $V_i \rightarrow a_1 a_2 \ldots a_m V_j$

Represents $V_i \rightarrow a_1 a_2 \ldots a_m$

FIGURE 3.16

THEOREM 3.3 (equivalence)



Suppose now that $w \in L(G)$ so that (3.4) is satisfied. In the nfa there is, by construction, a path from V_0 to V_i labeled v_1, a path from V_i to V_j labeled v_2, and so on, so that clearly

$$V_f \in \delta^* (V_0, w),$$

and w is accepted by M.

Conversely, assume that w is accepted by M. Because of the way in which M was constructed, to accept w the automaton has to pass through a sequence of states $V_0, V_i, \ldots$ to V_f, using paths labeled $v_1, v_2, \ldots$. Therefore, w must have the form

$$w = v_1 v_2 \cdots v_k v_l$$

and the derivation

$$V_o \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \cdots v_k V_k \Rightarrow v_1 v_2 \cdots v_k v_l$$

is possible. Hence w is in $L(G)$, and the theorem is proved. ■

EXAMPLE 3.15

Construct a finite automaton that accepts the language generated by the grammar

$$\begin{aligned} V_0 &\rightarrow aV_1, \\ V_1 &\rightarrow abV_0|b, \end{aligned}$$

where V_0 is the start variable. We start the transition graph with vertices V_0, V_1, and V_f. The first production rule creates an edge labeled a between V_0 and V_1. For the second rule, we need to introduce an additional vertex so that there is a path labeled ab between V_1 and V_0. Finally, we need to add an edge labeled b between V_1 and V_f, giving the automaton shown in Figure 3.17. The language generated by the grammar and accepted by the automaton is the regular language $L\left((aab)^* \, ab\right)$.

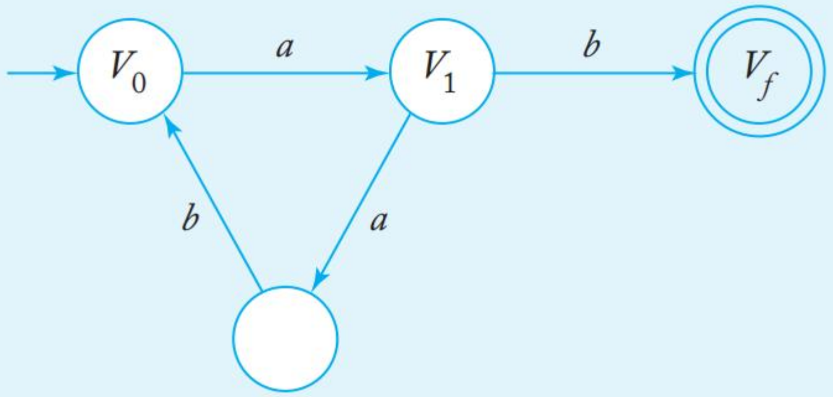
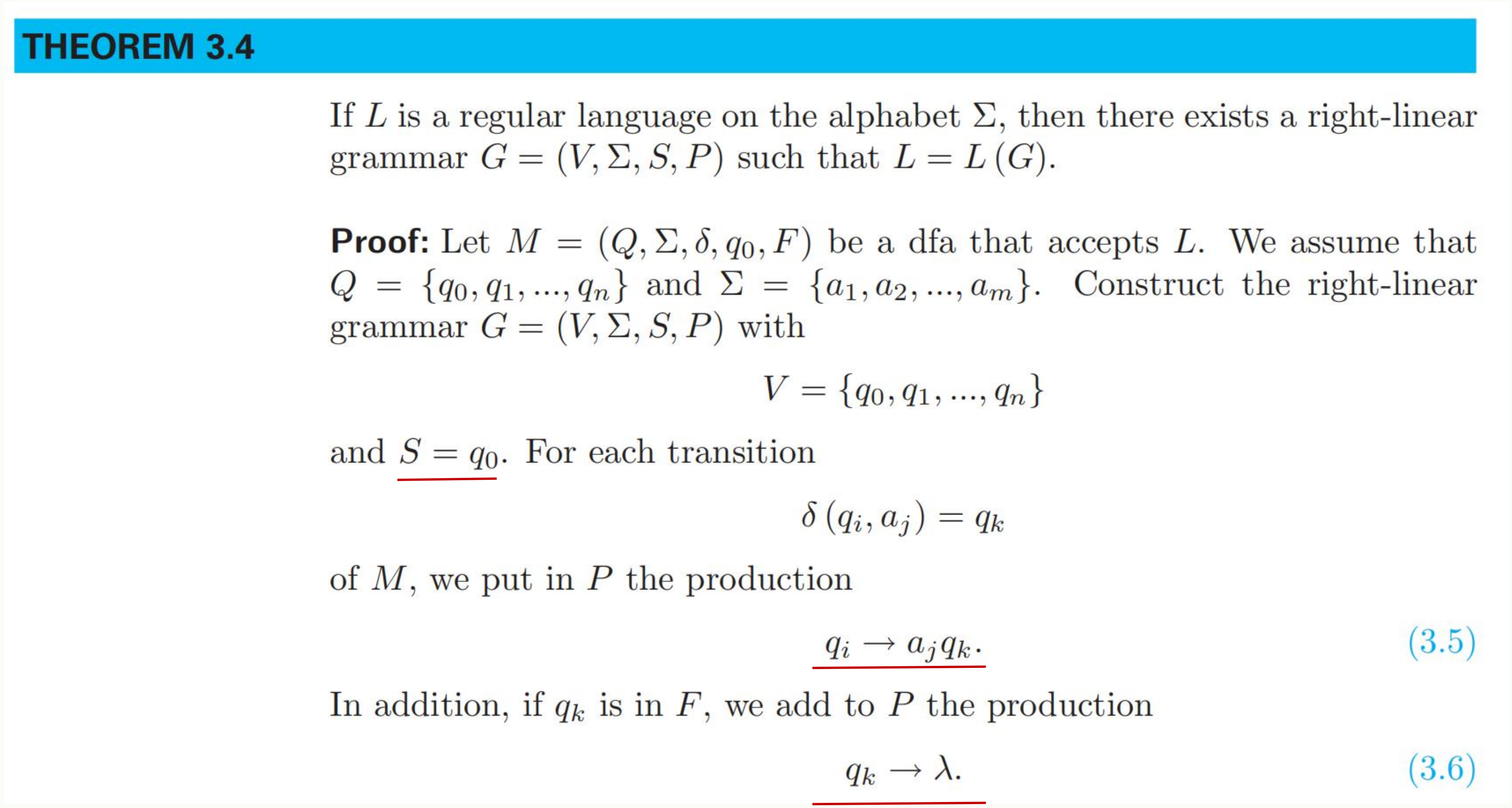


FIGURE 3.17

$A \rightarrow B \mid \lambda$

Dfa-> right-linear grammar

- Right-Linear Grammars for Regular Languages



THEOREM 3.4

If L is a regular language on the alphabet Σ, then there exists a right-linear grammar $G = (V, \Sigma, S, P)$ such that $L = L(G)$.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a dfa that accepts L. We assume that $Q = \{q_0, q_1, ..., q_n\}$ and $\Sigma = \{a_1, a_2, ..., a_m\}$. Construct the right-linear grammar $G = (V, \Sigma, S, P)$ with

$$V = \{q_0, q_1, ..., q_n\}$$

and $\underline{S = q_0}$. For each transition

$$\delta(q_i, a_j) = q_k$$

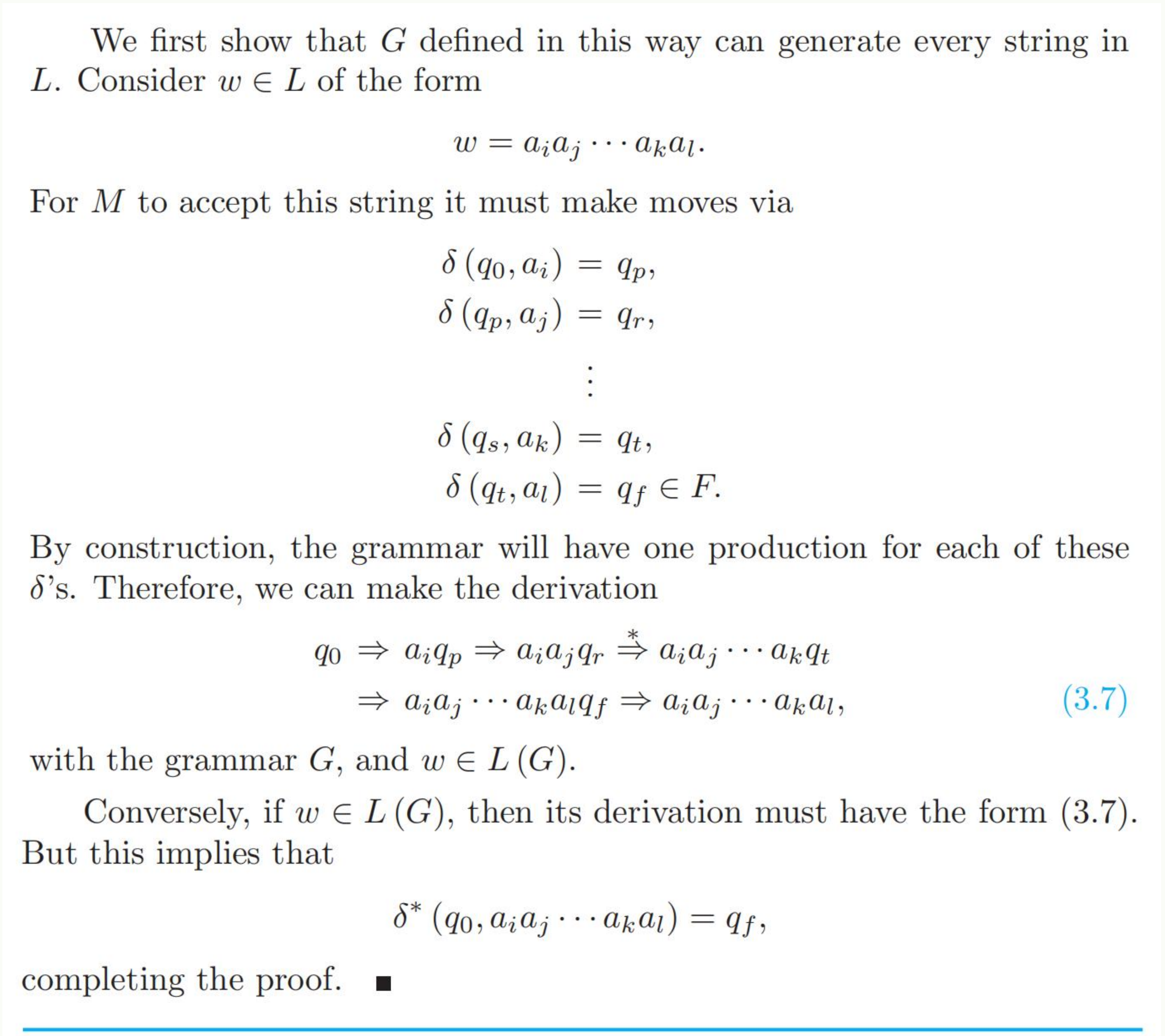
of M, we put in P the production

$$\underline{q_i \to a_j q_k}. \tag{3.5}$$

In addition, if q_k is in F, we add to P the production

$$\underline{q_k \to \lambda}. \tag{3.6}$$

THEOREM 3.4 (equivalence)



We first show that G defined in this way can generate every string in L. Consider $w \in L$ of the form

$$w = a_i a_j \cdots a_k a_l.$$

For M to accept this string it must make moves via

$$\begin{aligned} \delta(q_0, a_i) &= q_p, \\ \delta(q_p, a_j) &= q_r, \\ &\vdots \\ \delta(q_s, a_k) &= q_t, \\ \delta(q_t, a_l) &= q_f \in F. \end{aligned}$$

By construction, the grammar will have one production for each of these δ's. Therefore, we can make the derivation

$$\begin{aligned} q_0 &\Rightarrow a_i q_p \Rightarrow a_i a_j q_r \overset{*}{\Rightarrow} a_i a_j \cdots a_k q_t \\ &\Rightarrow a_i a_j \cdots a_k a_l q_f \Rightarrow a_i a_j \cdots a_k a_l, \end{aligned} \tag{3.7}$$

with the grammar G, and $w \in L(G)$.

Conversely, if $w \in L(G)$, then its derivation must have the form (3.7). But this implies that

$$\delta^*(q_0, a_i a_j \cdots a_k a_l) = q_f,$$

completing the proof. ■

EXAMPLE 3.16

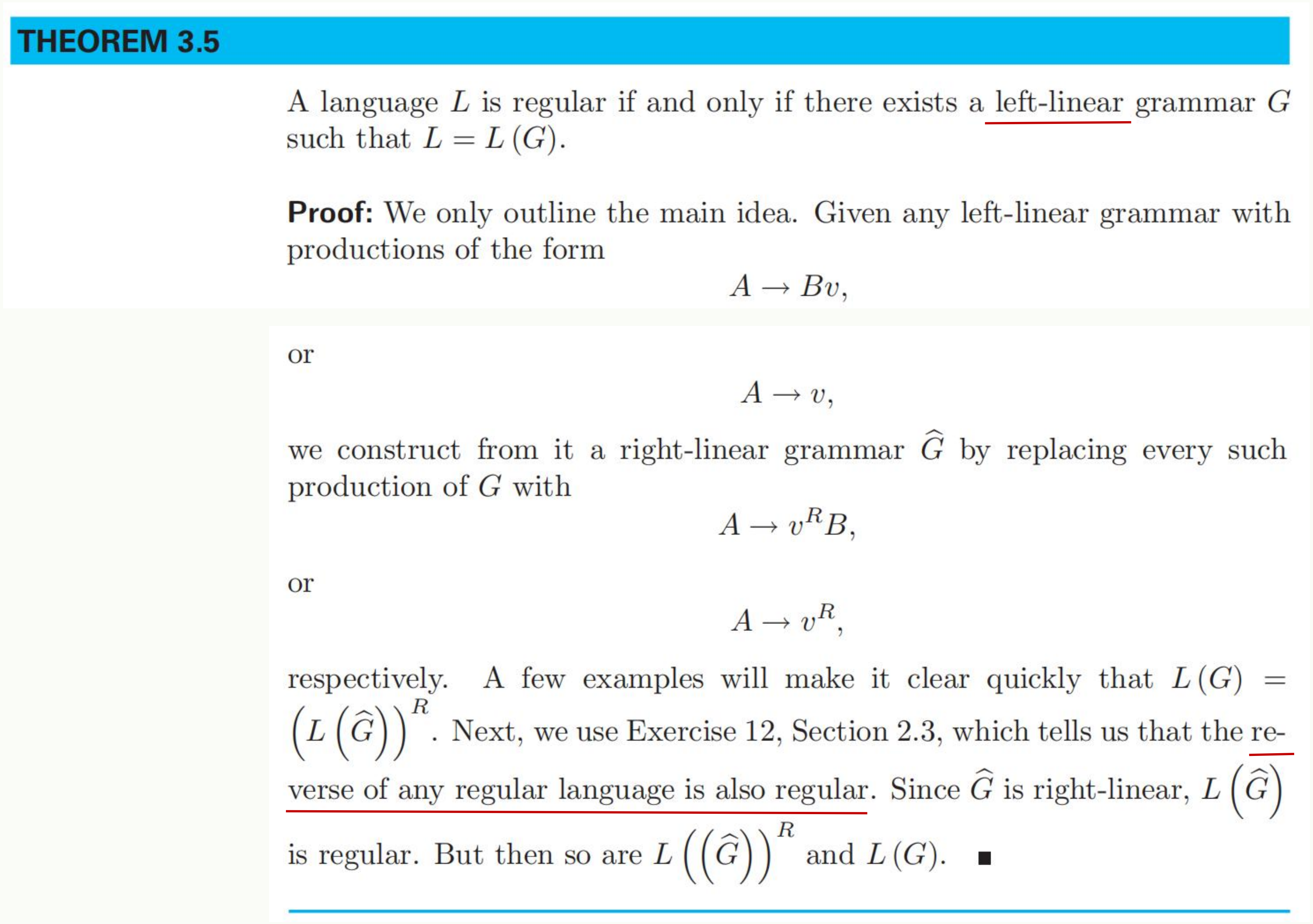
Construct a right-linear grammar for $L(aab^*a)$. The transition function for an nfa, together with the corresponding grammar productions, is given in Figure 3.18. The result was obtained by simply following the construction in Theorem 3.4. The string *aaba* can be derived with the constructed grammar by

$$q_0 \Rightarrow aq_1 \Rightarrow aaq_2 \Rightarrow aabq_2 \Rightarrow aabaq_f \Rightarrow aaba.$$

$\delta(q_0, a) = \{q_1\}$	$q_0 \rightarrow aq_1$
$\delta(q_1, a) = \{q_2\}$	$q_1 \rightarrow aq_2$
$\delta(q_2, b) = \{q_2\}$	$q_2 \rightarrow bq_2$
$\delta(q_2, a) = \{q_f\}$	$q_2 \rightarrow aq_f$
$q_f \in F$	$q_f \rightarrow \lambda$

FIGURE 3.18

Equivalence of Regular Languages and Regular Grammars



THEOREM 3.5

A language L is regular if and only if there exists a left-linear grammar G such that $L = L(G)$.

Proof: We only outline the main idea. Given any left-linear grammar with productions of the form

$$A \to Bv,$$

or

$$A \to v,$$

we construct from it a right-linear grammar $\widehat{G}$ by replacing every such production of G with

$$A \to v^R B,$$

or

$$A \to v^R,$$

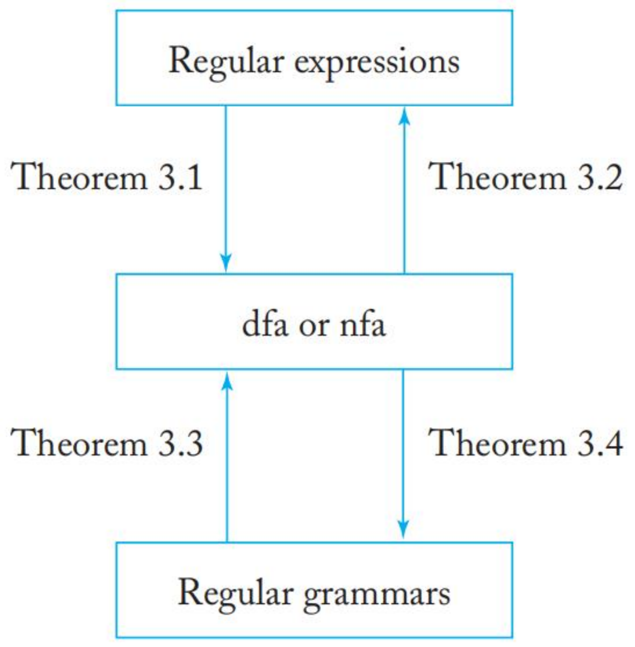
respectively. A few examples will make it clear quickly that $L(G) = \left(L\left(\widehat{G}\right)\right)^R$. Next, we use Exercise 12, Section 2.3, which tells us that the reverse of any regular language is also regular. Since $\widehat{G}$ is right-linear, $L\left(\widehat{G}\right)$ is regular. But then so are $L\left(\left(\widehat{G}\right)\right)^R$ and $L(G)$. ■

Define RL with Regular Grammar

THEOREM 3.6

A language L is regular if and only if there exists a regular grammar G such that $L = L(G)$.

We now have several ways of describing regular languages: dfa's, nfa's, regular expressions, and regular grammars. While in some instances one or the other of these may be most suitable, they are all equally powerful.



(图上作业法)

FIGURE 3.19

END