

Chapter 3

Regular Languages and Regular Grammars

何宗易

Tsung-Yi Ho

tyho@cs.nctu.edu.tw

<http://idea.cs.nctu.edu.tw>

Department of Computer Science

National Chiao Tung University

Hsinchu, Taiwan



Outline



Regular Expressions (RE)



Connection Between REs and Regular Languages



Regular Grammars

Specifying Language

How do we specify languages?

- If language is finite, you can list all of its strings.
 - $L = \{a, aa, aba, aca\}$
- Descriptive:
 - $L = \{x \mid n_a(x) = n_b(x)\}$
- Using basic Language operations
 - $L = \{aa, ab\}^* \cup \{b\}\{bb\}^*$
- Regular languages are described using the last method

Regular Expressions

Regular expressions describe regular languages and the notation involves a combination of:

- Strings of symbols from some alphabet Σ
- Parentheses $()$
- Operators $+$, \cdot , $*$

Regular Expressions

Important thing to remember

- A regular expression is **not** a language
- A regular expression is used to **describe** a language.
- It is incorrect to say that for a language L ,
 $L = (a + b + c)^*$
- But it's okay to say that L is described by
 $(a + b + c)^*$

Regular Expressions

All finite languages can be described by regular expressions

Example: $(a + b \cdot c)^* \longleftrightarrow \{\{a\} \cup \{bc\}\}^*$

describes the language

$$\{a, bc\}^* = \{\lambda, a, bc, aa, abc, bca, \dots\}$$

Definition 3.1

Let Σ be a given alphabet. Then

1. ϕ , λ , and $a \in \Sigma$ are all regular expressions. These are called **primitive regular expressions**.
2. If r_1 and r_2 are regular expressions, so are r_1+r_2 , $r_1 \cdot r_2$, r_1^* and (r_1) .
3. A string is a regular expression **iff** it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

Example 3.1

A regular expression: $(a + b \cdot c)^* \cdot (c + \emptyset)$

Not a regular expression: $(a + b +)$
 a^n
 a^+

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L((a + b \cdot c)^*) = \{\lambda, a, bc, aa, abc, bca, \dots\}$$

Definition

- For primitive regular expressions:

$$L(\emptyset) = \emptyset \quad (1)$$

$$L(\lambda) = \{\lambda\} \quad (2)$$

$$L(a) = \{a\} \quad (3)$$

Definition (continued)

For regular expressions r_1 and r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2) \quad (4)$$

$$L(r_1 \cdot r_2) = L(r_1) L(r_2) \quad (5)$$

$$L(r_1^*) = (L(r_1))^* \quad (6)$$

$$L((r_1)) = L(r_1) \quad (7)$$

Example 3.2

Regular expression: $(a + b) \cdot a^*$

$$\begin{aligned} L((a + b) \cdot a^*) &= L((a + b)) L(a^*) \\ &= L(a + b) L(a^*) \\ &= (L(a) \cup L(b)) (L(a))^* \\ &= (\{a\} \cup \{b\}) (\{a\})^* \\ &= \{a, b\} \{\lambda, a, aa, aaa, \dots\} \\ &= \{a, aa, aaa, \dots, b, ba, baa, \dots\} \end{aligned}$$

Priority of Operators

- Regular expression: $r = a \cdot b + c$
 $r_1 = a \cdot b$ $r_2 = c$ or $r_1 = a$ $r_2 = b + c$
 $L(r) = \{ab, c\} \neq \{ab, ac\}$
- Star closure (*) precedes
concatenation (·) precedes
union (+)

Example 3.3

$$\Sigma = \{a, b\}$$

- Regular expression $r = \underline{(a + b)^*} (a + bb)$

Stands for any string of a's and b's



$$L(r) = \{a, bb, aa, abb, ba, bbb, \dots\}$$

$L(r)$ is the set of all strings on $\{a, b\}$, terminated by either an **a** or a **bb**

Example 3.4

- Regular expression $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m+1} : n, m \geq 0\}$$

$L(r)$ is the set of all strings with an even number of **a's**
followed by an odd number of **b's**

Example 3.5

- For $\Sigma = \{0, 1\}$, give a regular expression r such that

$L(r) = \{ w \in \Sigma^* : w \text{ has at least one pair of consecutive } 0 \}$

00

Example 3.6

$L(r) = \{ \text{all strings with no pairs of consecutive 0s} \}$

- Regular expression

$$r = (1 + 01)^* (0 + \lambda)$$

$$r = (1^* 0 1^*)^*$$

Diagram illustrating the components of the regular expression $r = (1^* 0 1^*)^*$:

- 1^* : Add 1 immediately after a 0
- 0 : String ending in 0
- 1^* : String with all 1's

There are an unlimited number of REs for any given language!

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2


are **equivalent** if $L(r_1) = L(r_2)$

Example

$L = \{ \text{all strings without two consecutive 0} \}$

$$r_1 = (1 + 01)^* (0 + \lambda)$$

$$r_2 = (1^* 0 1 1^*)^* (0 + \lambda) + 1^* (0 + \lambda)$$

$L(r_1) = L(r_2) = L$  r_1 and r_2
are equivalent
regular expressions

More Examples

- $L_1 = \{a, aa, aba, aca\}$
- $L_1 = \{a\} \cup \{aa\} \cup \{aba\} \cup \{aca\}$
- Regular expression describing L_1 :
 $(a + aa + aba + aca)$

More Examples

- $L_2 = \{x \in \{0,1\}^* \mid |x| \text{ is even}\}$
- $L_2 = \{00, 01, 10, 11\}^*$
- Regular expressions describing L_2 :
 $(00 + 01 + 10 + 11)^*$
 $((0 + 1)(0 + 1))^*$

More Examples

- $L_3 = \{x \in \{0,1\}^* \mid x \text{ does not end in } 01 \}$
If x does not end in 01, then either
 x ends in 00, 10, or 11
- A regular expression that describes L_3 is:
 $(0 + 1)^*(00 + 10 + 11)$

More Examples

- $L_4 = \{x \in \{0,1\}^* \mid x \text{ contains an odd number of 0s} \}$

Express $x = yz$

y is a string of the form $y = 1^i 0 1^j$

In z , there must be an even number of 0's

$$z = (01^k 01^m)^*$$

- A regular expression that describes L_4 is:

$$(1^* 0 1^*)(01^* 0 1^*)^*$$

Short Quiz

- Give regular expressions for the following language on $\Sigma = \{a, b, c\}$.
 - All strings containing exactly one a

Outline



Regular Expressions (RE)



Connection Between REs and Regular Languages



Regular Grammars

Theorem

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Described by} \\ \text{Regular Expressions} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

For every regular language there is a regular expression
For every regular expression there is a regular language

Kleene Theorem:

Regular expressions and Finite Automata
are equivalent (w.r.t. the languages they
describe/accept)



Theorem - Part 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Described by} \\ \text{Regular Expressions} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

1. For any regular expression r
the language $L(r)$ is regular

■ Theorem 3.1

Theorem - Part 2

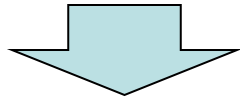
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Described by} \\ \text{Regular Expressions} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

2. For any regular language L there is a regular expression r with $L(r) = L$

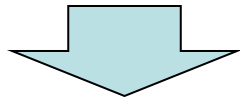
■ Theorem 3.2

Proof - Part 1 $\left\{ \begin{array}{l} \text{Languages} \\ \text{Described by} \\ \text{Regular Expressions} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$

1. For any regular expression r
the language $L(r)$ is regular



If we have any regular expression r ,
we can construct an NFA(DFA) that accepts $L(r)$

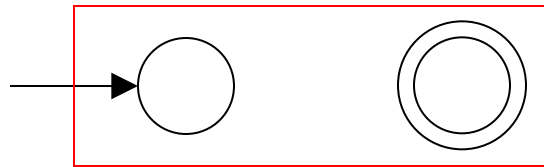


Proof by induction on the size of r

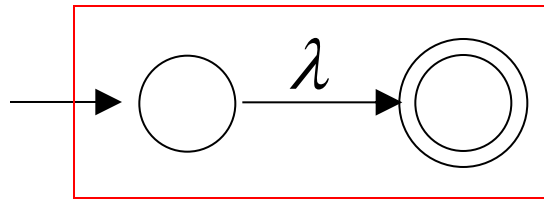
Induction Basis

- Primitive Regular Expressions: \emptyset , λ , a

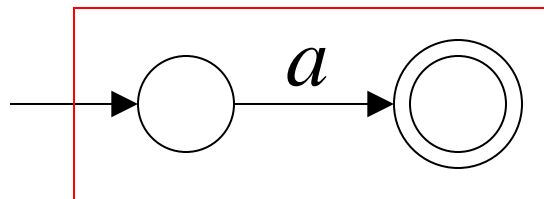
NFAs



$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{\lambda\} = L(\lambda)$$



$$L(M_3) = \{a\} = L(a)$$

regular
languages

Inductive Hypothesis

Assume

for regular expressions r_1 and r_2

that

$L(r_1)$ and $L(r_2)$ are regular languages

Inductive Step

\therefore REs are derived from these four rules:

$$L(r_1 + r_2)$$

$$L(r_1 \cdot r_2)$$

$$L(r_1^*)$$

$$L((r_1))$$

We will prove:

Are regular
Languages

- By definition of regular expressions:

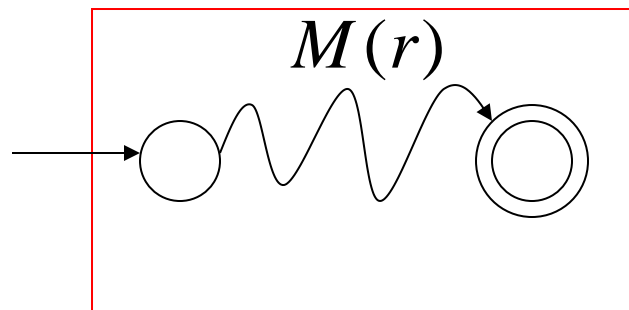
$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1) L(r_2)$$

$$L(r_1^*) = (L(r_1))^*$$

$$L((r_1)) = L(r_1)$$

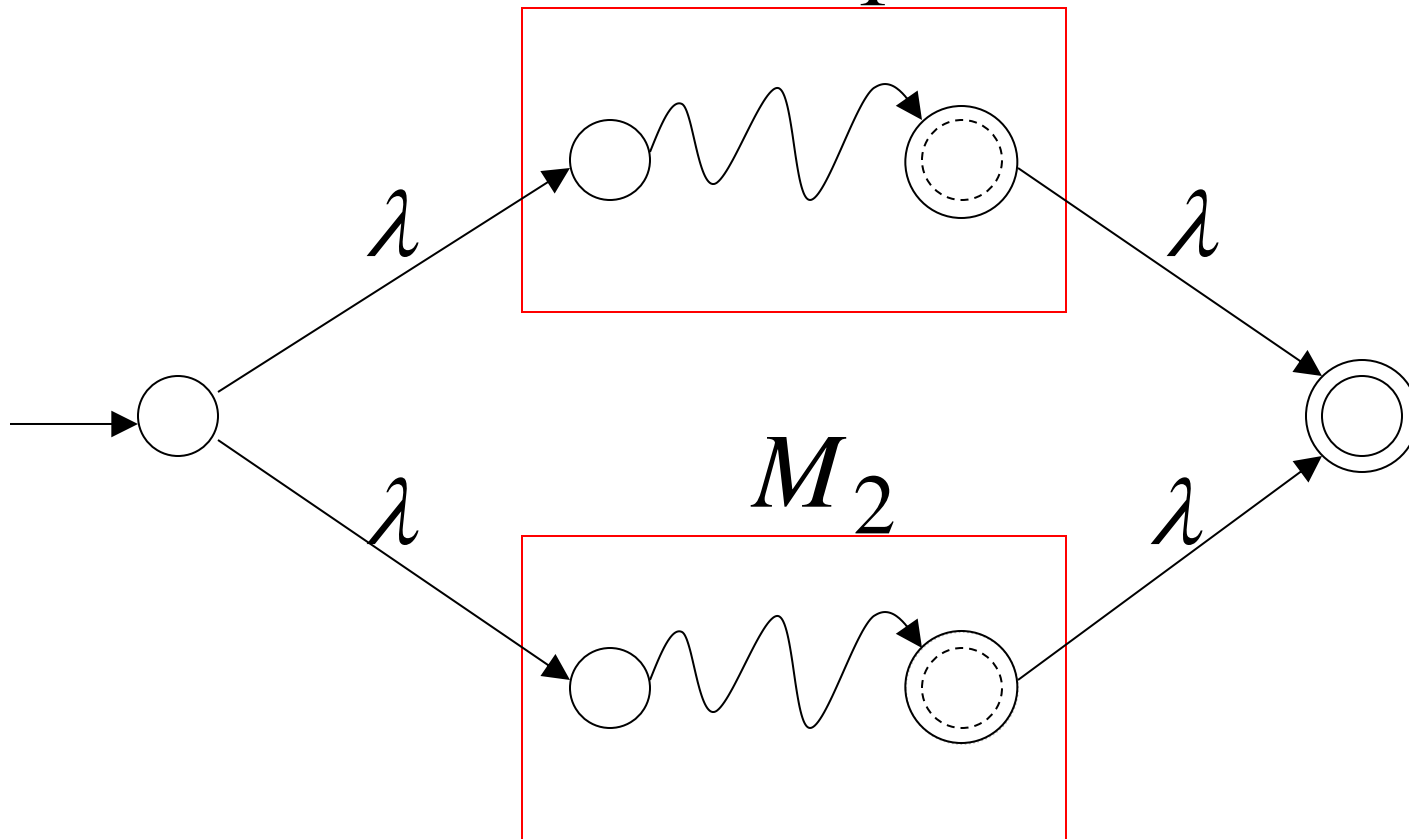
Schematic representation of an NFA ($M(r)$) accepting $L(r)$



We can claim that for every NFA there is only one final state (by exercise 7, section 2.3)

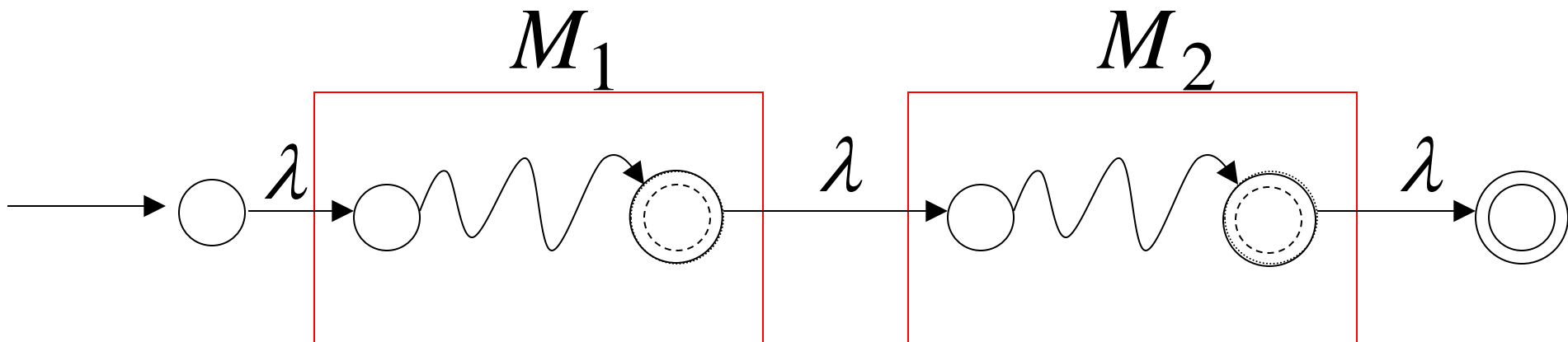
Union

- NFA for $L(r_1 + r_2)$ M_1



Concatenation

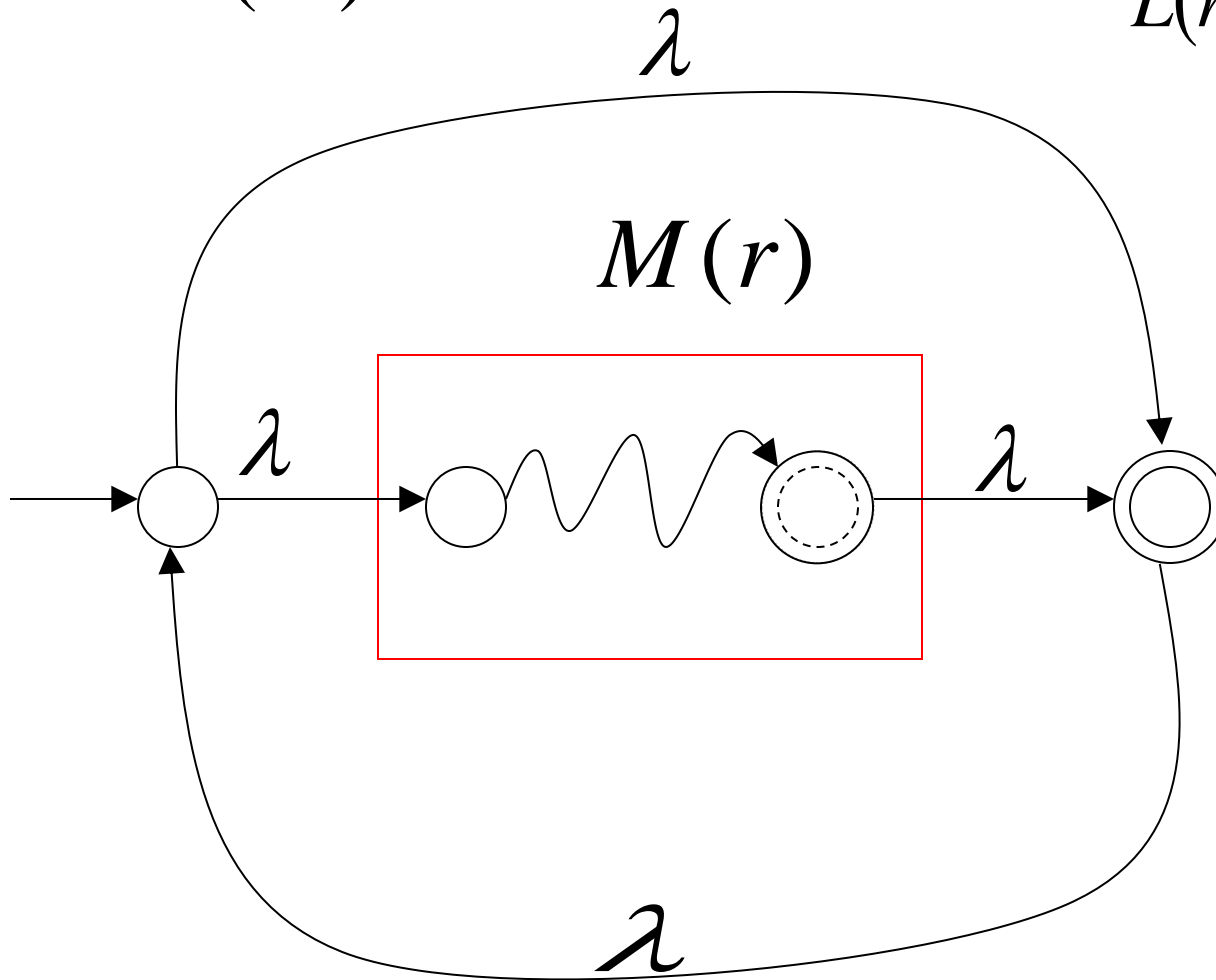
- NFA for $L(r_1 r_2)$



Star Operation

- NFA for $L(r^*)$

$$L(r_1^*) = (L(r_1))^*$$



By inductive hypothesis we know:

$L(r_1)$ and $L(r_2)$ are regular languages

We also know:

Regular languages are closed under:

Union $L(r_1) \cup L(r_2)$

Concatenation $L(r_1) L(r_2)$

Star $(L(r_1))^*$

Therefore:

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1) L(r_2)$$

$$L(r_1^*) = (L(r_1))^*$$

Are regular
languages

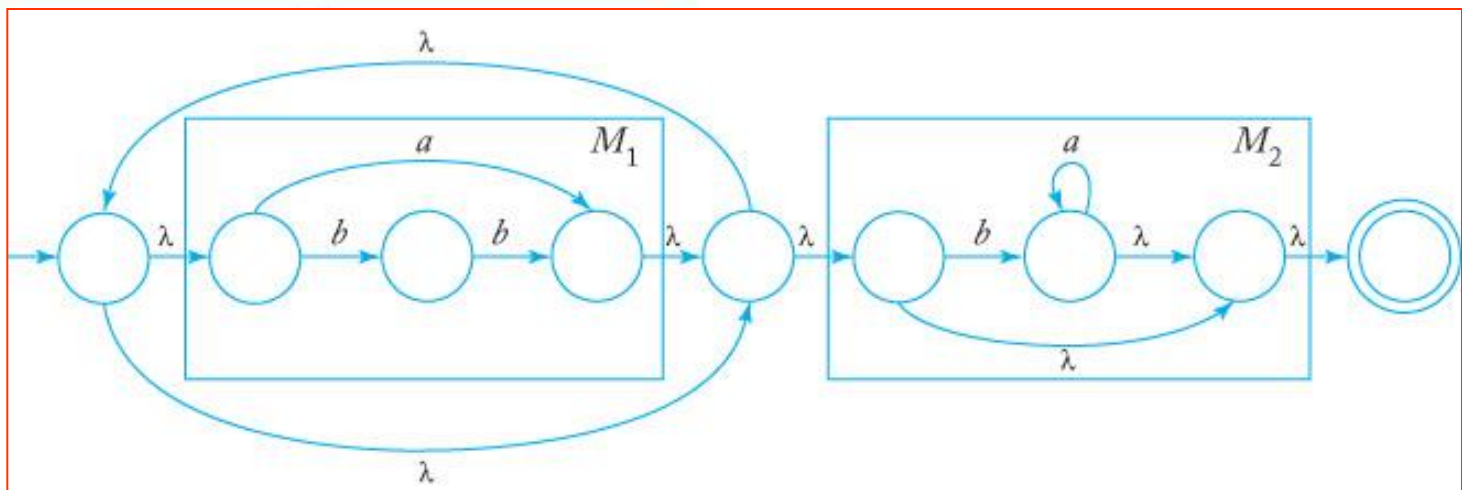
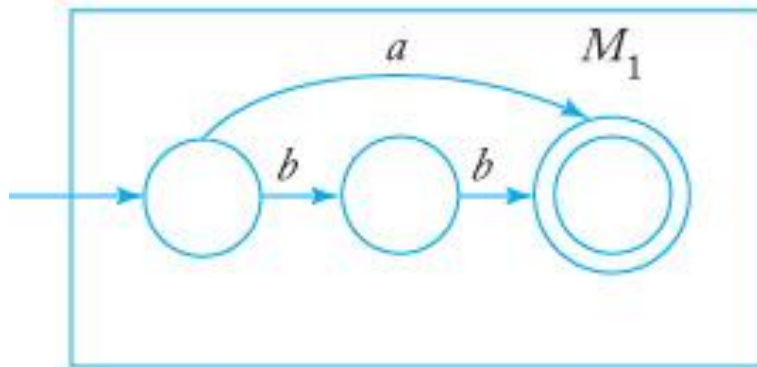
And trivially:

$L((r_1))$ is a regular language

\therefore For any regular expression r
the language $L(r)$ is regular

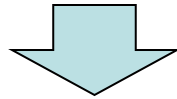
Example 3.7

- Find an NFA that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$

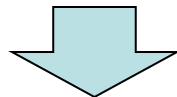


Proof – Part 2 $\left\{ \begin{array}{l} \text{Languages} \\ \text{Described by} \\ \text{Regular Expressions} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$

2. For any regular language L there is a regular expression r with $L(r) = L$



Since any regular language has an associated **NFA** and hence a **transition graph**, all we need to do is to find a **regular expression** capable of generating the labels of **all the walks from q_0 to any final state**.



Proof by construction of regular expression

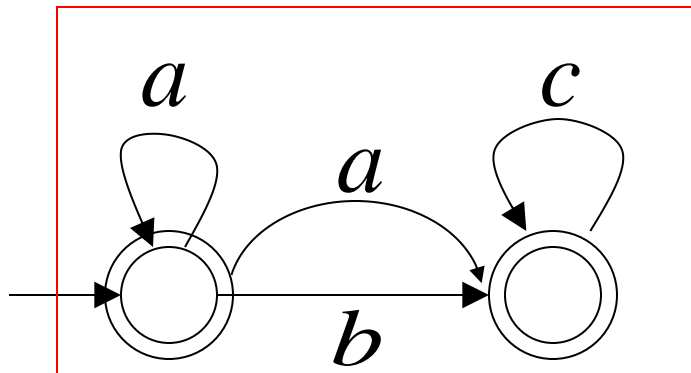
Generalized Transition Graphs (GTG)

From M construct the equivalent

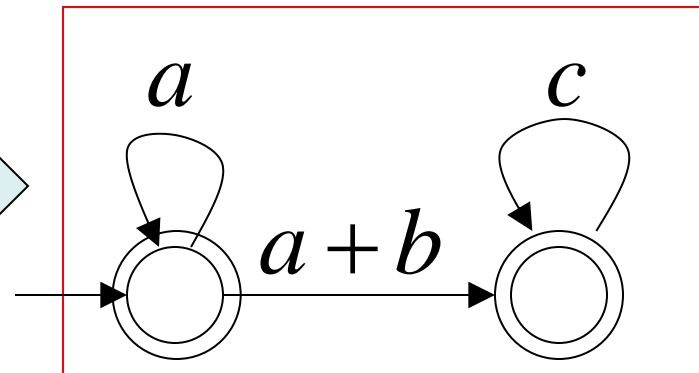
Generalized Transition Graph

in which transition labels are regular expressions

Example: M

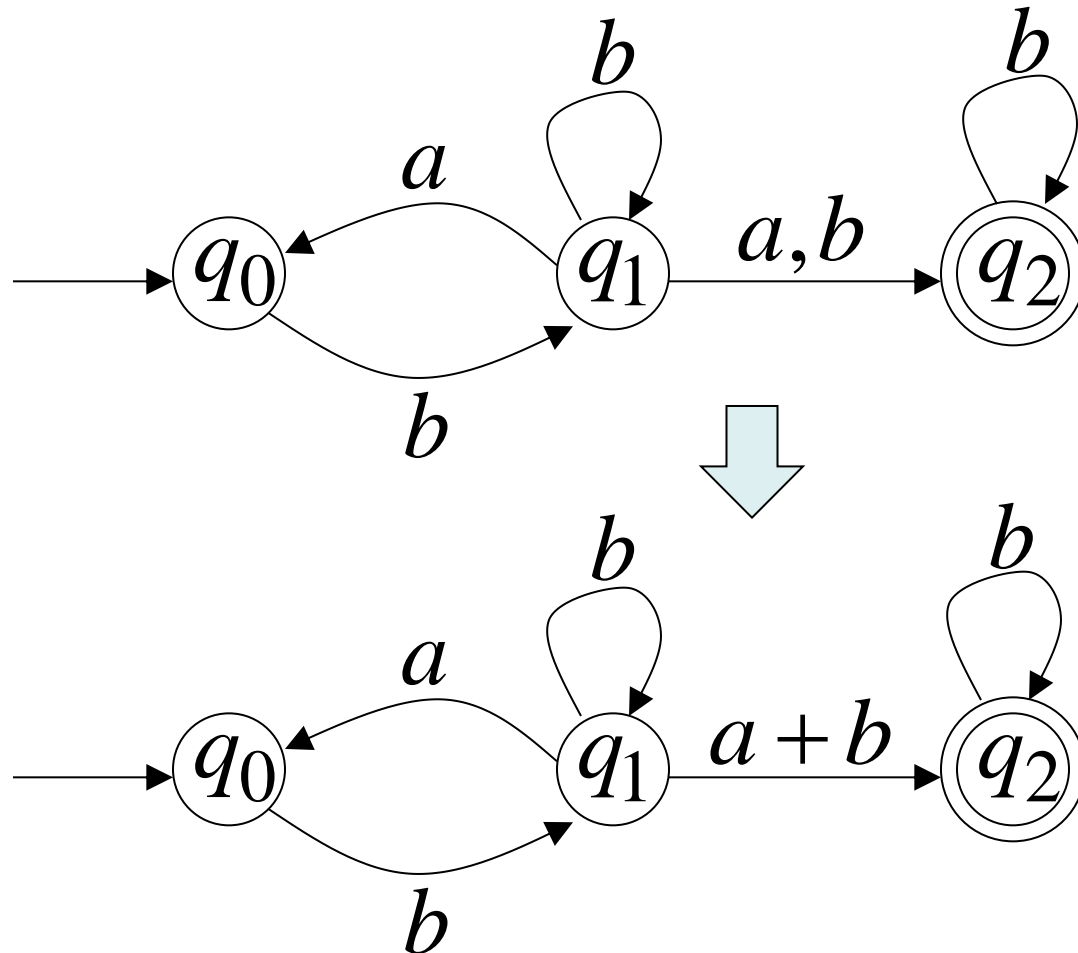


$L(a^* + a^*(a + b)c^*)$



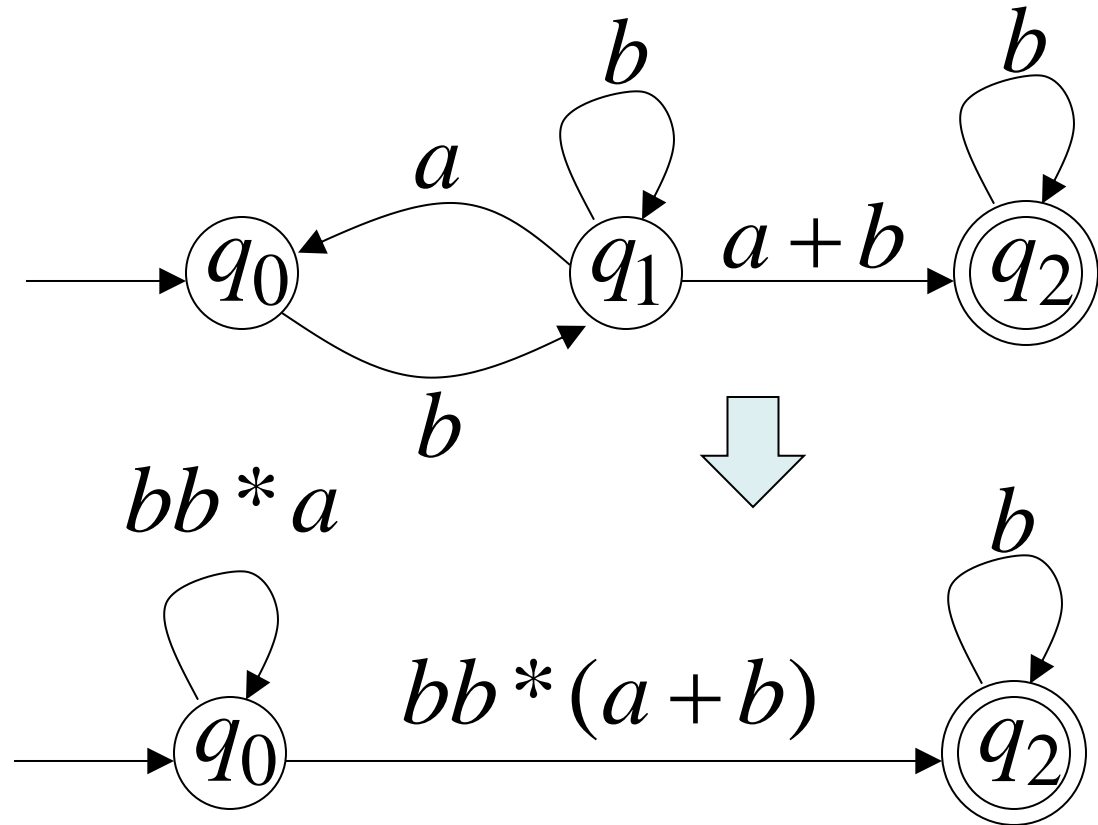
GTG may have many states

Enumerating all walks is time-consuming



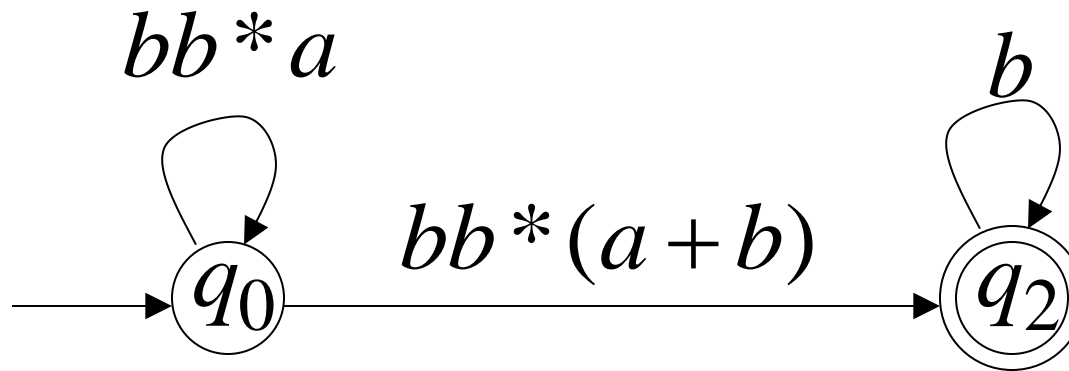
Reducing the states:

Ex. reduce q_1



Simple two-state GTG

Resulting Regular Expression:

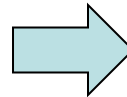
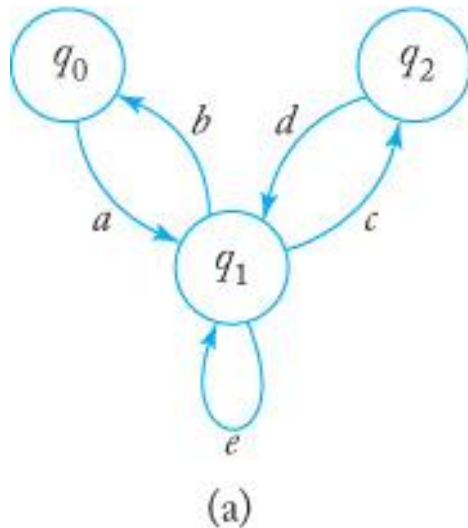


$$r = (bb^*a)^*bb^*(a+b)b^*$$

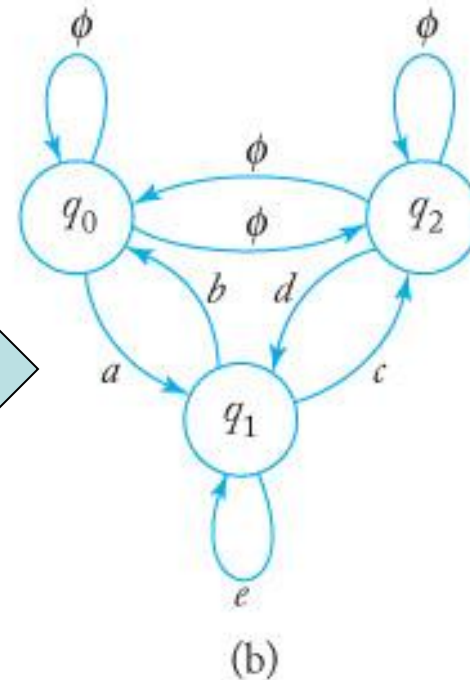
$$L(r) = L(M) = L$$

Complete GTG

GTG

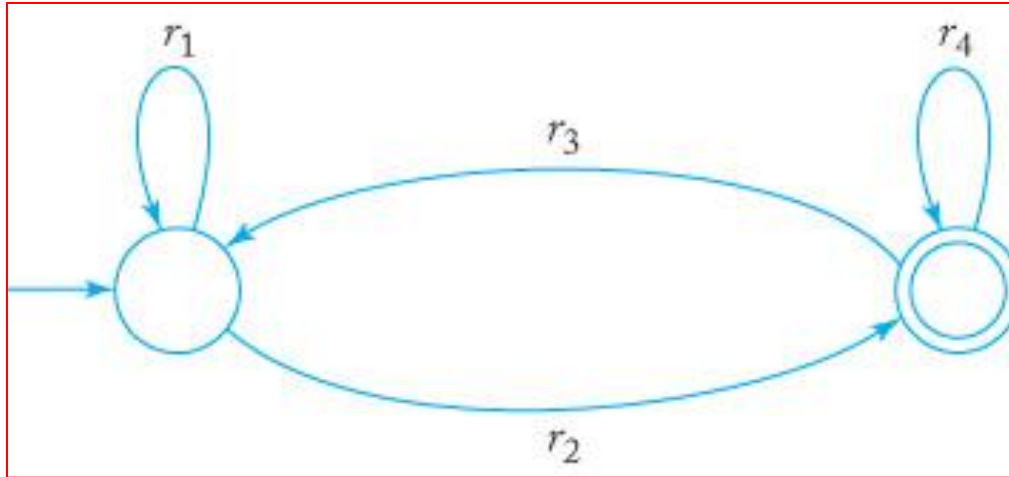


Complete GTG



- If a GTG, after conversion from an NFA, has some edges missing, we put them in and label them with ϕ
- A complete GTG with $|V|$ vertices has exactly $|V|^2$ edges

Example 3.9



RE?

$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

How about a GTG with more than two states?

We can find an equivalent graph by removing one state at a time

Example 3.10

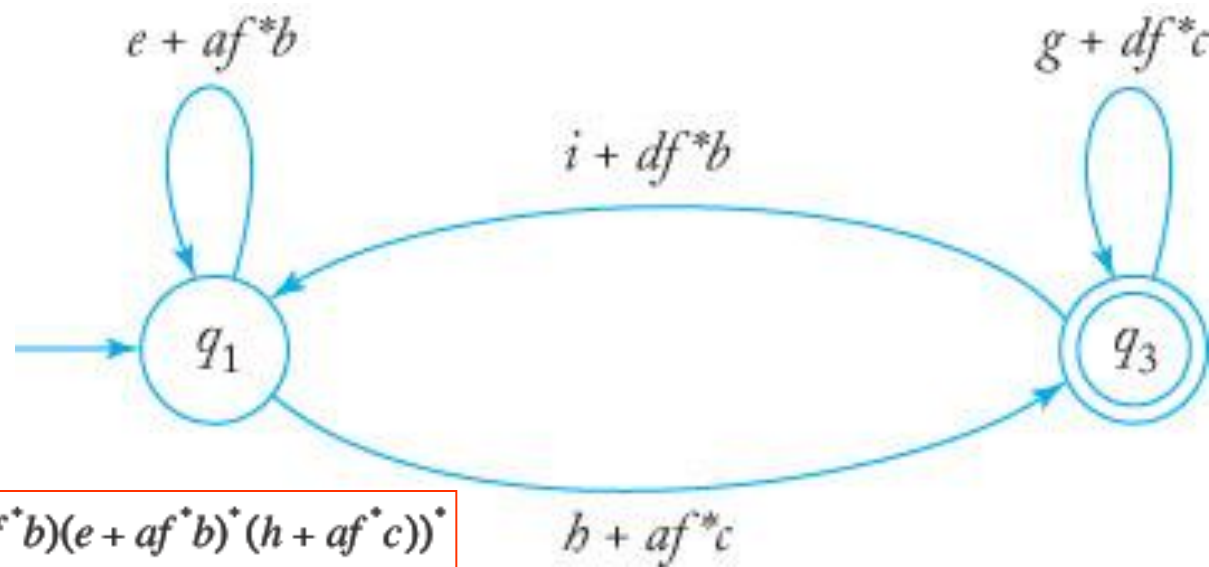
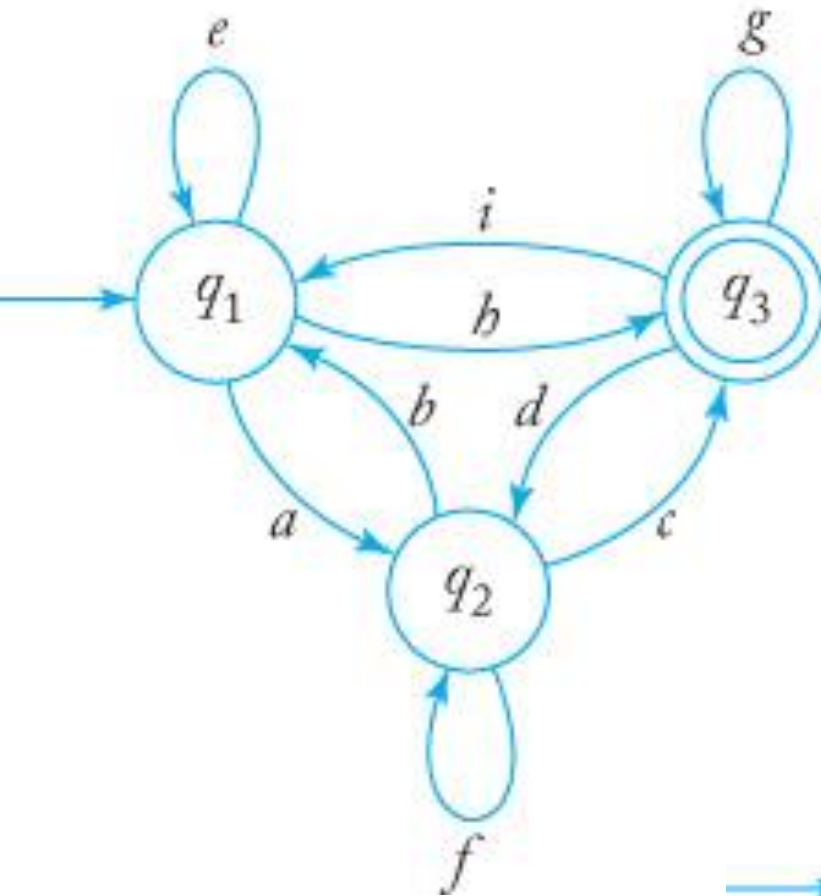
To remove q_2 , we create edges as follows:

$$\overrightarrow{q_1 q_1} \rightarrow$$

$$\overrightarrow{q_1 q_3} \rightarrow$$

$$\overrightarrow{q_3 q_1} \rightarrow$$

$$\overrightarrow{q_3 q_3} \rightarrow$$



$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

$$(e + af^*b)^*(h + af^*c)((g + df^*c) + (i + df^*b)(e + af^*b)^*(h + af^*c))^*$$

NFA \rightarrow RE

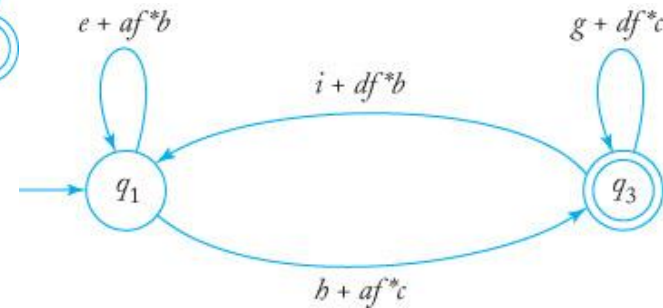
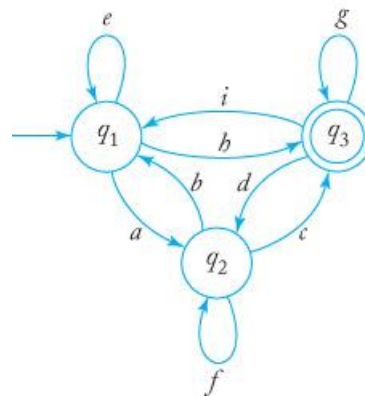
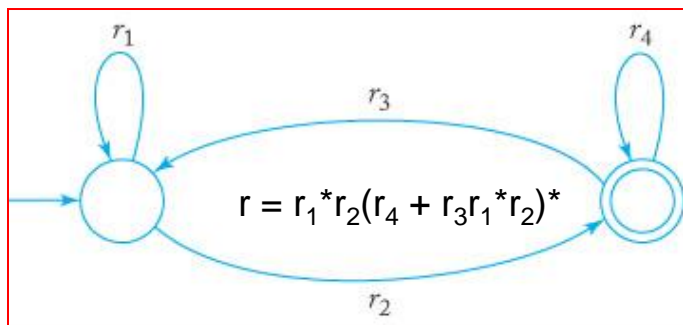
- ➔ 1. Convert the NFA (with single final state) into a complete GTG. Let r_{ij} stand for the label of the edge from q_i to q_j .
2. If the GTG has only two states with $q_i \in q_0$ and $q_j \in F$, as its associated RE is:

$$r = r_{ii}^* r_{ij} (r_{jj} + r_{ji} r_{ii}^* r_{ij})^*$$

3. If the GTG has three states with $q_i \in q_0$, $q_j \in F$, and $q_k \in Q$, introduce new edges, labeled:

$$r_{pq} + r_{pk} r_{kk}^* r_{kq}$$

for $p = i, j$, $q = i, j$. When this is done, remove vertex q_k and its associated edges.



NFA \rightarrow RE

4. If the GTG has four or more states, pick a state q_k to be removed. Apply rule 3 for all pairs of states (q_i, q_j) , $i \neq k$, $j \neq k$. At each step apply the simplifying rules

$$r + \phi = r, \quad r \cdot \phi = \phi, \quad \phi^* = \lambda$$

wherever possible. When this is done, remove state q_k .

5. Repeat step 2 to 4 until the correct RE is obtained.

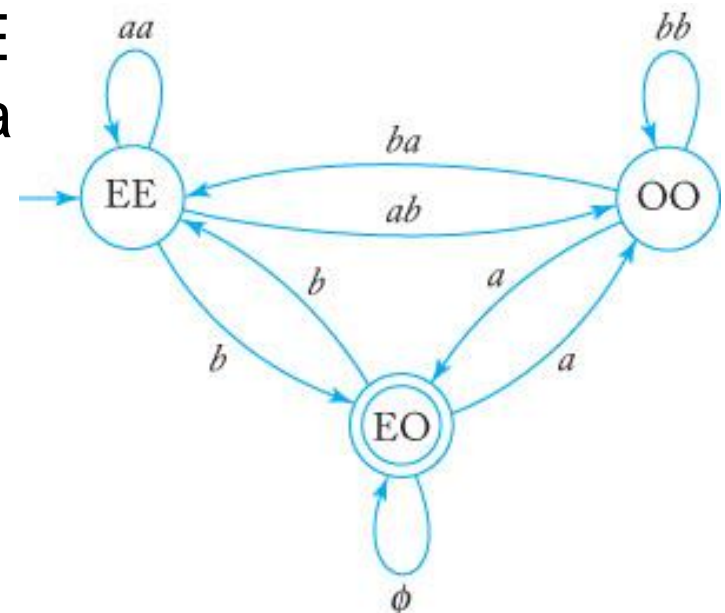
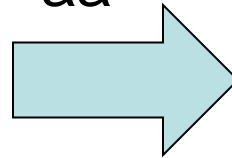
Example 3.11

- Find a RE for the language

$$L = \{w \in \{a,b\}^* : n_a(w) \text{ is even and } n_b(w) \text{ is odd}\}.$$

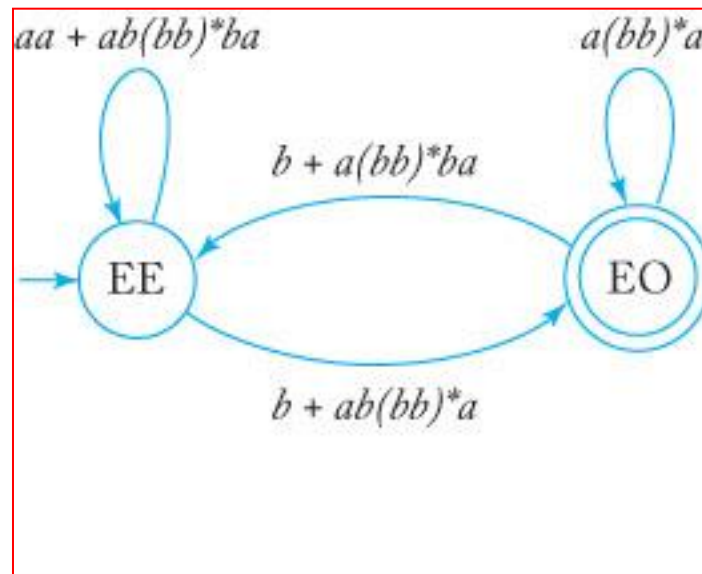
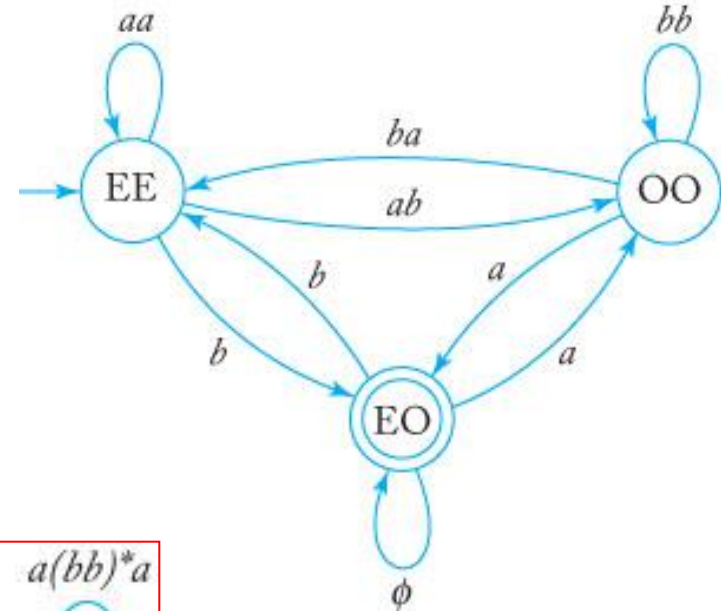
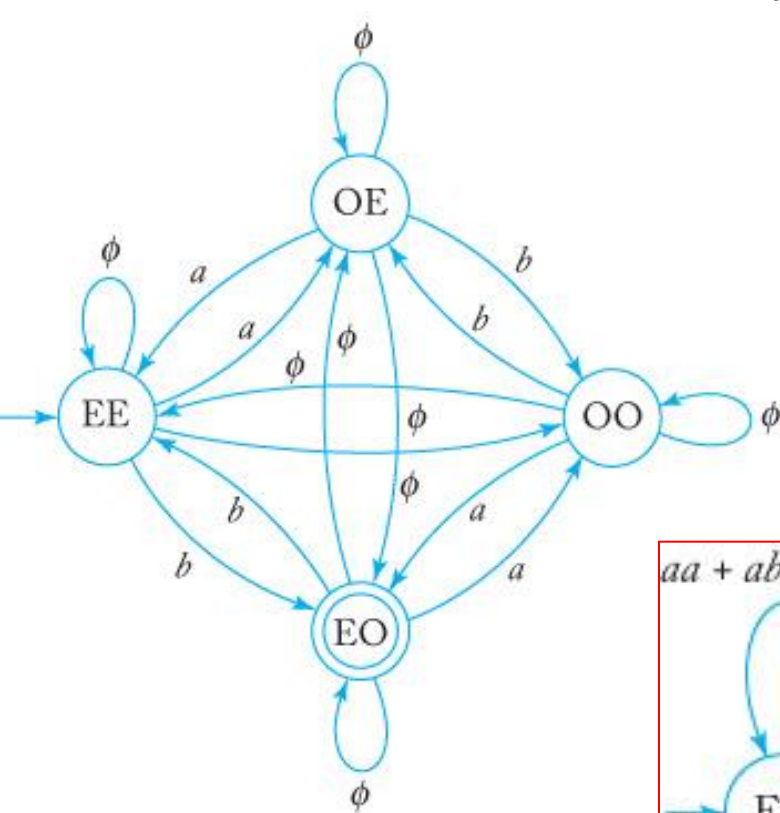
Construct NFA first!

To remove OE
 $r_{EE} = \phi + a \phi^* a$
 $= aa$



Example 3.11

$L = \{w \in \{a, b\}^* : n_a(w) \text{ is even and } n_b(w) \text{ is odd}\}.$



Outline



Regular Expressions (RE)



Connection Between REs and Regular Languages



Regular Grammars

Grammar Recap

- A grammar G is defined as a 4-tuple:

$$G = (V, T, S, P)$$

where

- V is a finite set of **variables**
- T is a finite set of **terminals**
- $S \in V$, called **start variable**
- P is a finite set of **production rules**

Grammar Recap

- Let $G = (V, T, S, P)$ be a grammar. Then the set

$$L(G) = \{w \in T^*: S \Rightarrow^* w\}$$

is the language generated by G

- If $w \in L(G)$, then the sequence

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w_n \Rightarrow w$$

is a derivation of the sentence w .

- S, w_1, w_2, \dots, w_n are called **sentential forms**

Linear Grammars

Grammars with
at most one variable at the right side
of a production

Examples: $S \rightarrow aSb$
 $S \rightarrow \lambda$

$S \rightarrow Ab$
 $A \rightarrow aAb$
 $A \rightarrow \lambda$

Another Linear Grammar

Grammar G :

$$S \rightarrow A$$
$$A \rightarrow aB \mid \lambda$$
$$B \rightarrow Ab$$

$$L(G) = \{a^n b^n : n \geq 0\}$$

A Non-Linear Grammar

Grammar G :

$$S \rightarrow SS$$
$$S \rightarrow \lambda$$
$$S \rightarrow aSb$$
$$S \rightarrow bSa$$

$$L(G) = \{w : n_a(w) = n_b(w)\}$$



Number of a in string w

Right-Linear Grammars

- All productions have form: $A \rightarrow xB$

or

$$A \rightarrow x$$



- Example: $S \rightarrow abS$

$$S \rightarrow a$$

string of
terminals

Left-Linear Grammars

- All productions have form: $A \rightarrow Bx$

or

$$A \rightarrow x$$



- Example:
 $S \rightarrow Aab$
 $A \rightarrow Aab \mid B$
 $B \rightarrow a$

string of
terminals

Regular Grammars


A **regular grammar** is either right-linear or left-linear grammar

Examples:

G_1 

$S \rightarrow abS$


$S \rightarrow a$

G_2 

$S \rightarrow Aab$

$A \rightarrow Aab \mid B$

$B \rightarrow a$

G_3 

$S \rightarrow A$

$A \rightarrow aB \mid \lambda$

$B \rightarrow Ab$

Observation

Regular grammars generate regular languages

A regular grammar is always linear, but
not all linear grammars are regular.

G_3 is linear grammar
but not regular grammar

G_3

$S \rightarrow A$

$A \rightarrow aB \mid \lambda$

$B \rightarrow Ab$

Example 3.13

Regular grammars generate regular languages

G_2

$$S \rightarrow Aab$$

$$A \rightarrow Aab \mid B$$

$$B \rightarrow a$$

$$L(G_2) = aab(ab)^*$$

G_1

$$S \rightarrow abS$$

$$S \rightarrow a$$

$$L(G_1) = (ab)^*a$$

Theorem

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Grammars} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Theorem - Part 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Grammars} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Any regular grammar generates
a regular language

■ Theorem 3.3

Theorem - Part 2

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Grammars} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Any regular language is generated by a regular grammar

■ Theorem 3.4

Proof – Part 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Grammars} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

The language $L(G)$ generated by any regular grammar G is regular

The case of Right-Linear Grammars

Let G be a right-linear grammar

We will prove: $L(G)$ is regular

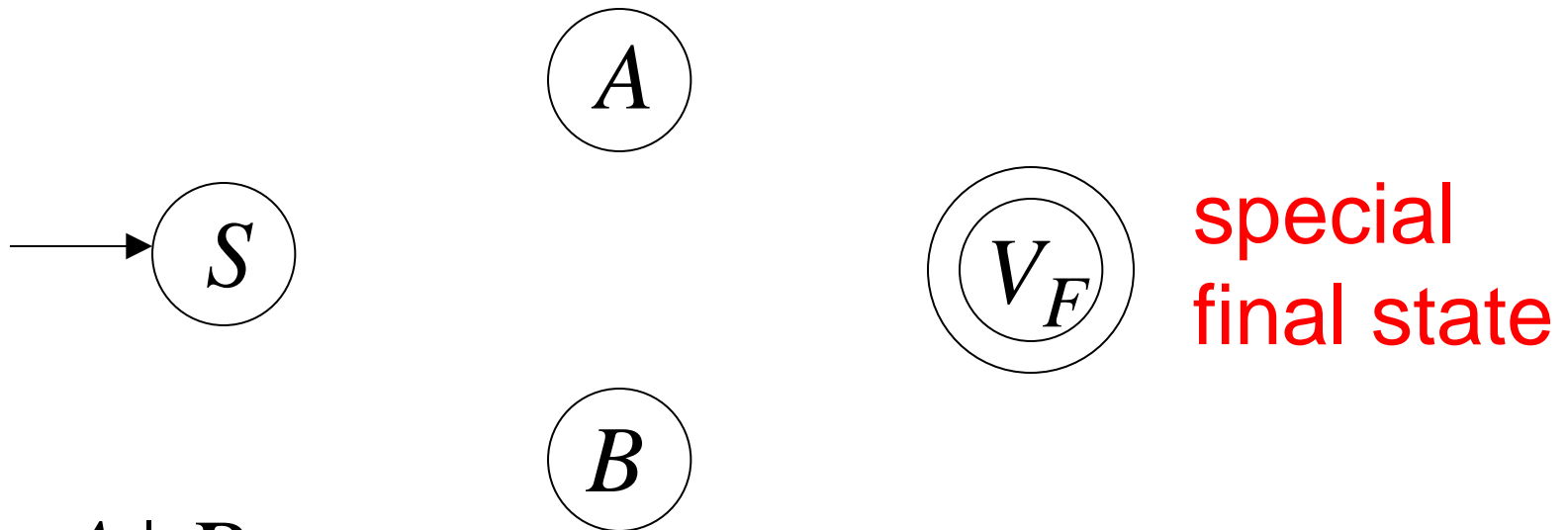
Proof idea: We will construct NFA M
with $L(M) = L(G)$

- Grammar G is right-linear

Example: $S \rightarrow aA \mid B$

$$A \rightarrow aa B$$
$$B \rightarrow b B \mid a$$

Construct NFA M such that
every state is a grammar variable:

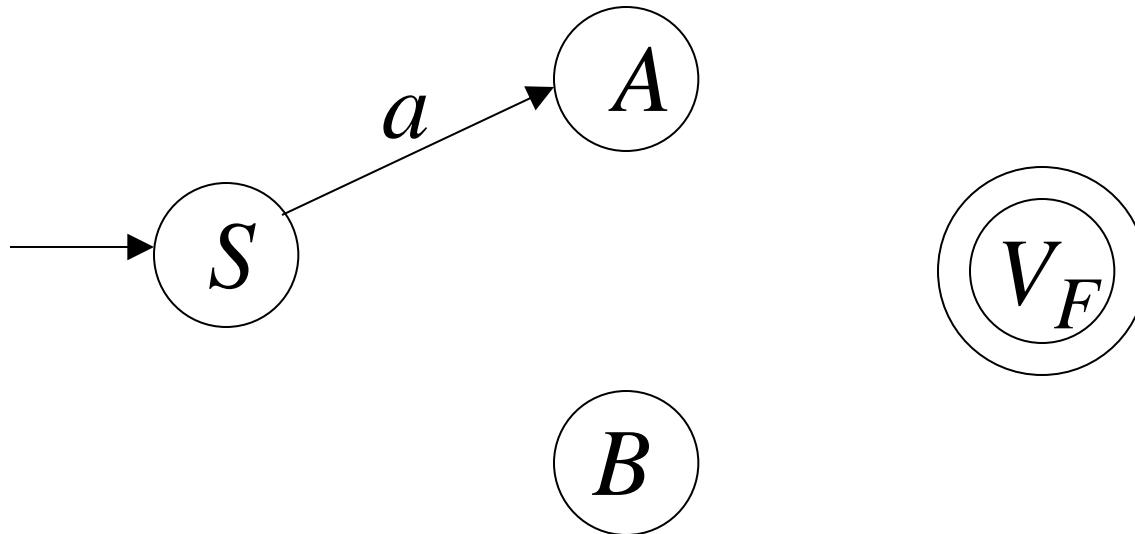


$$S \rightarrow aA \mid B$$

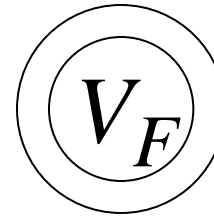
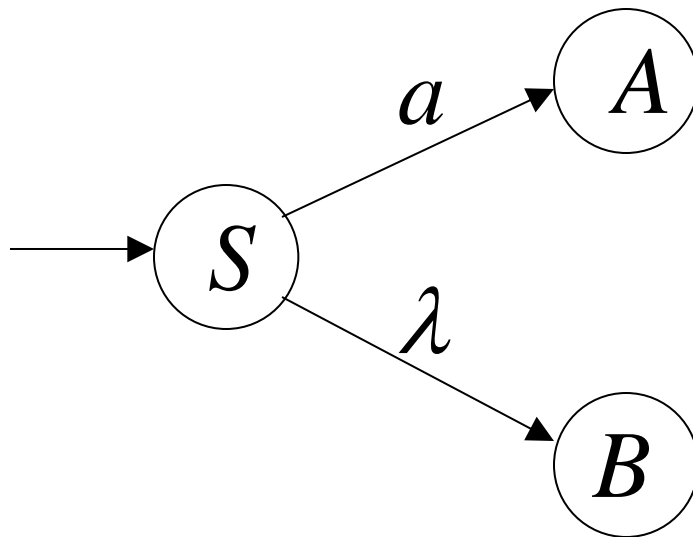
$$A \rightarrow aa B$$

$$B \rightarrow b B \mid a$$

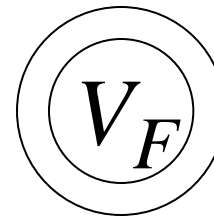
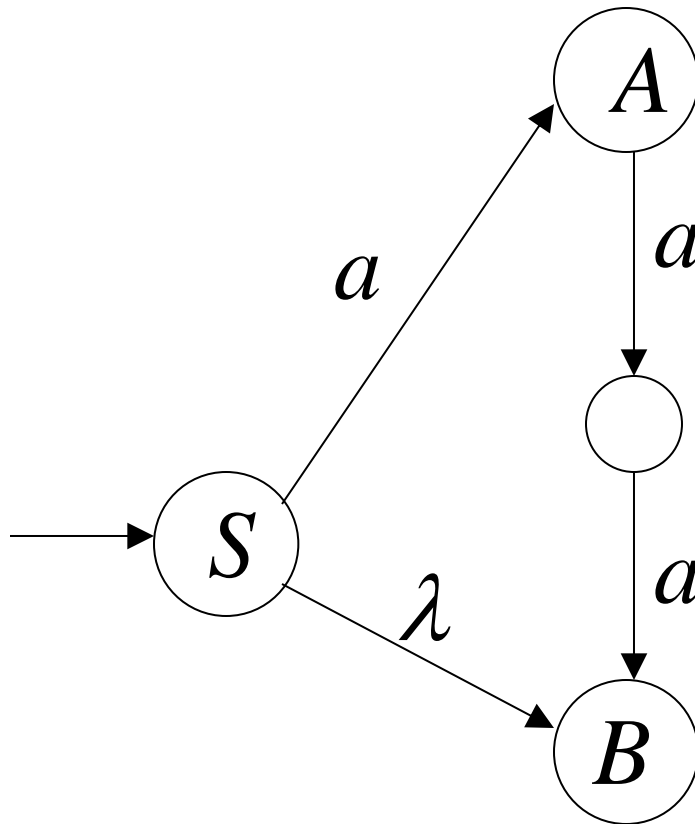
- Add edges for each production:



$$S \rightarrow aA$$

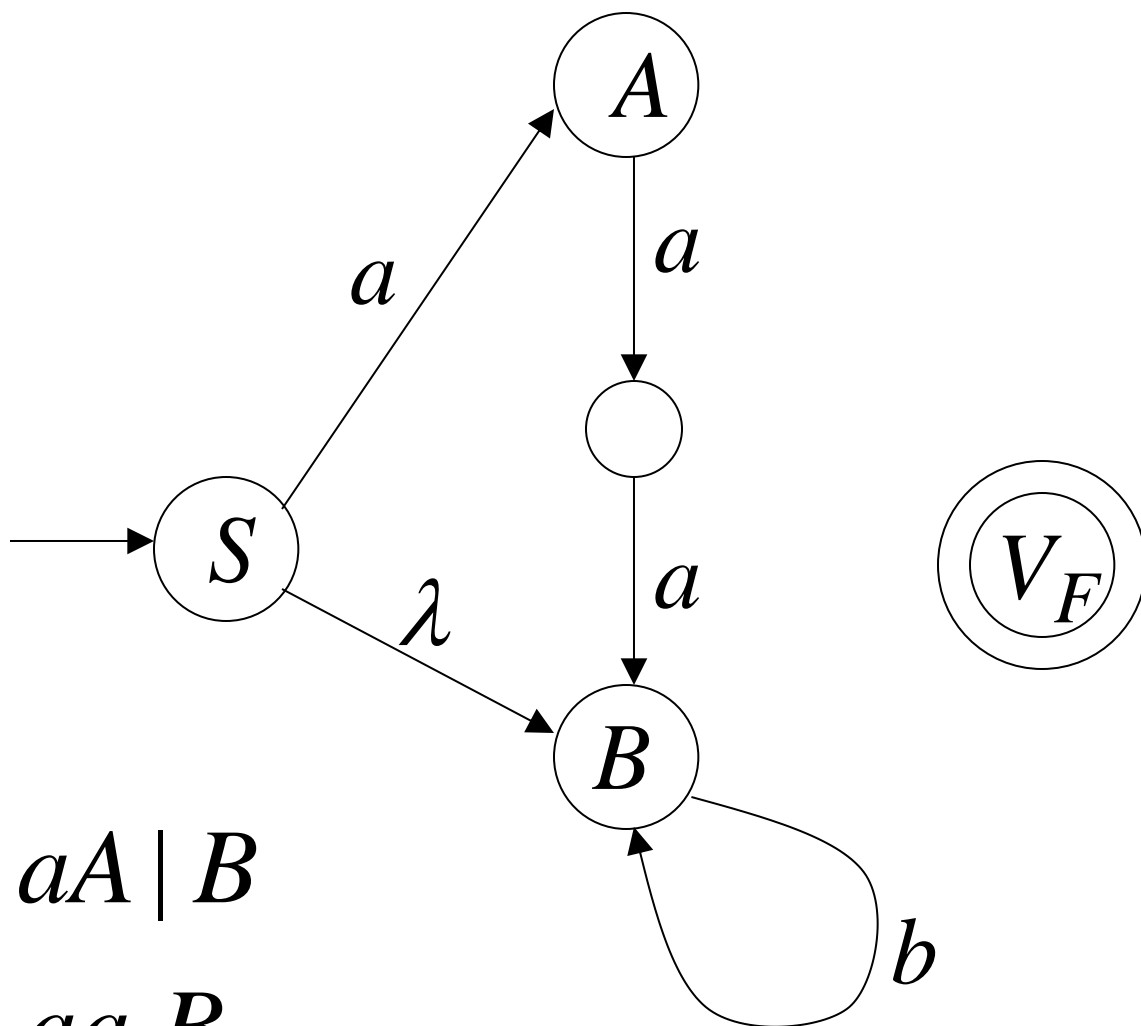


$$S \rightarrow aA \mid B$$



$$S \rightarrow aA \mid B$$

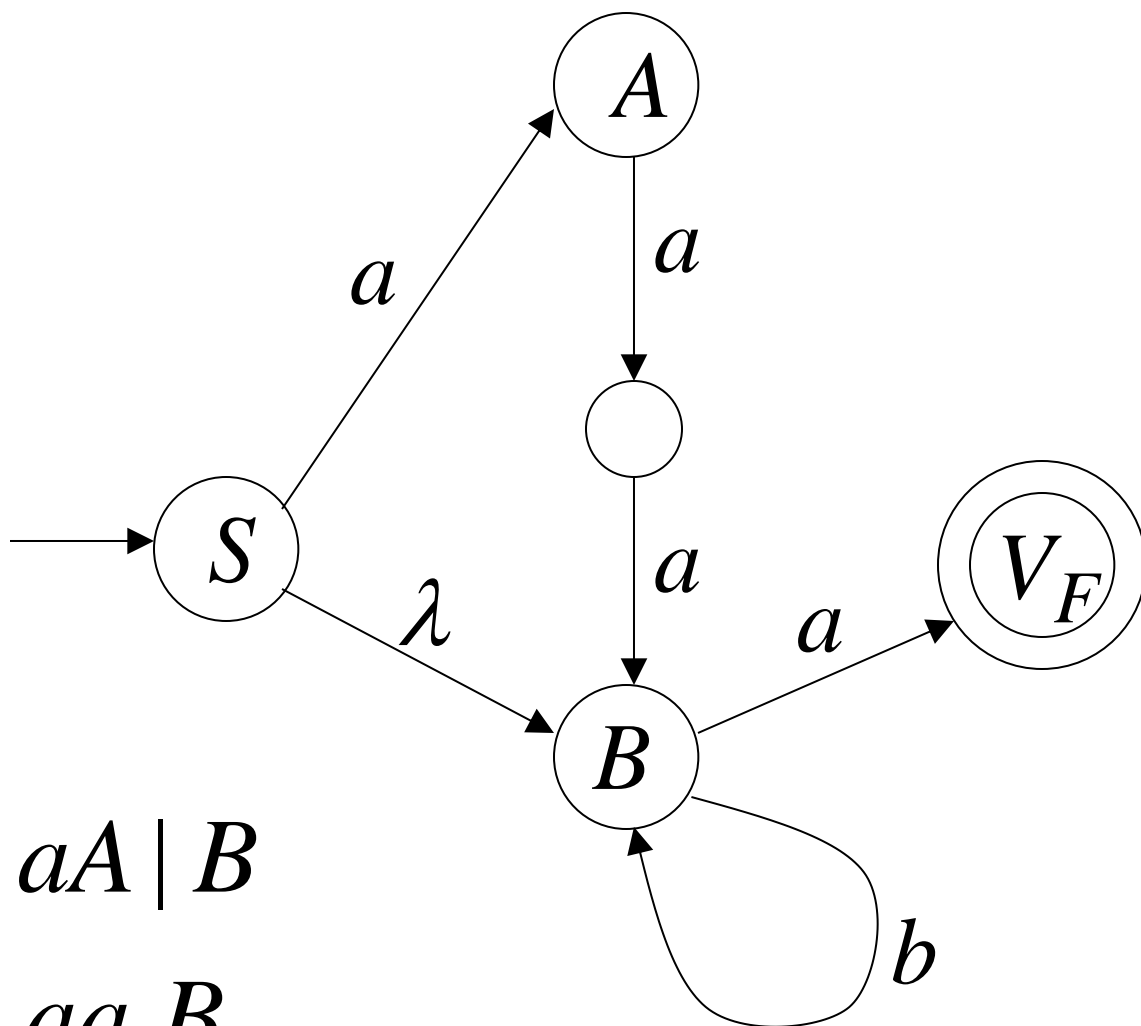
$$A \rightarrow aa B$$



$$S \rightarrow aA \mid B$$

$$A \rightarrow aa B$$

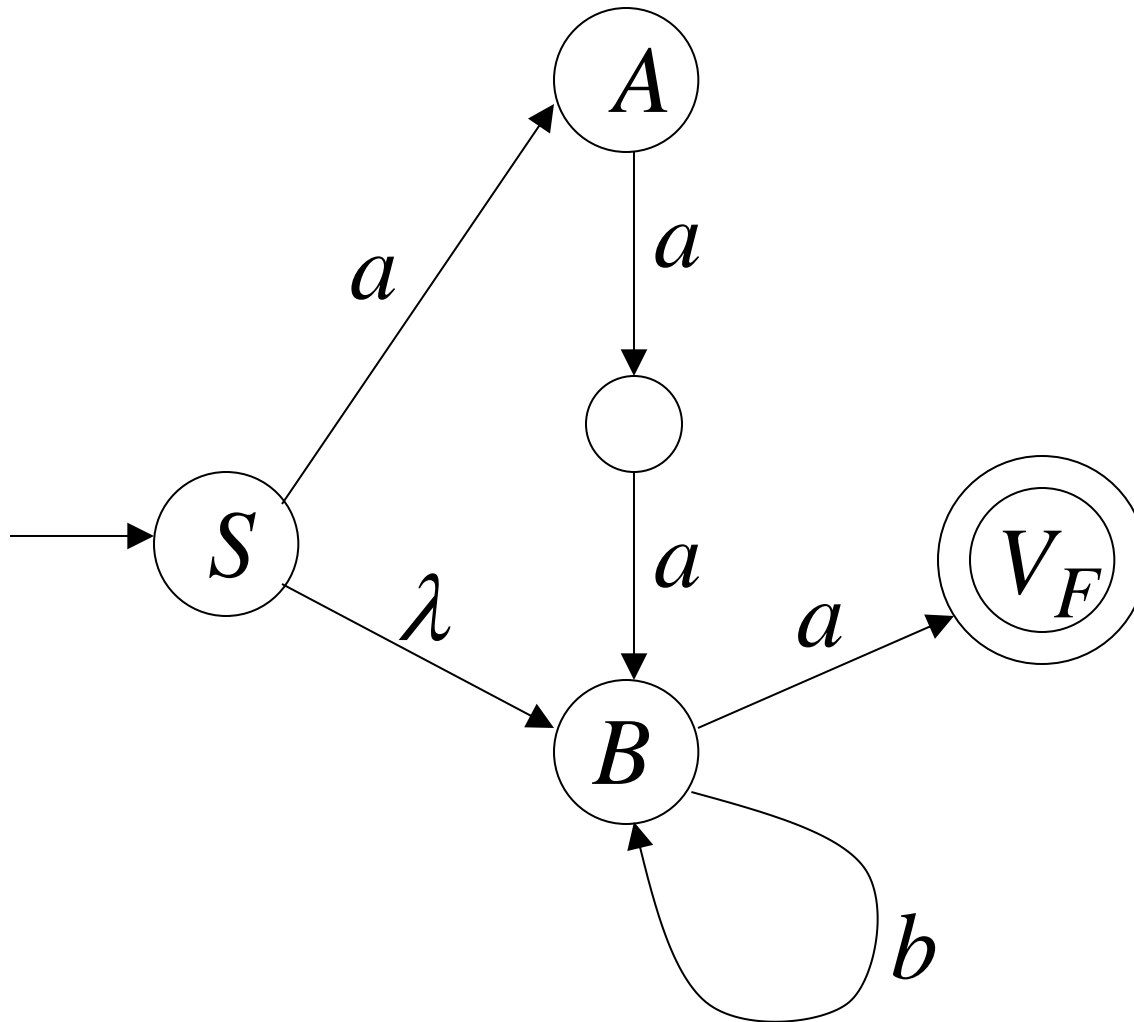
$$B \rightarrow bB$$



$$S \rightarrow aA \mid B$$

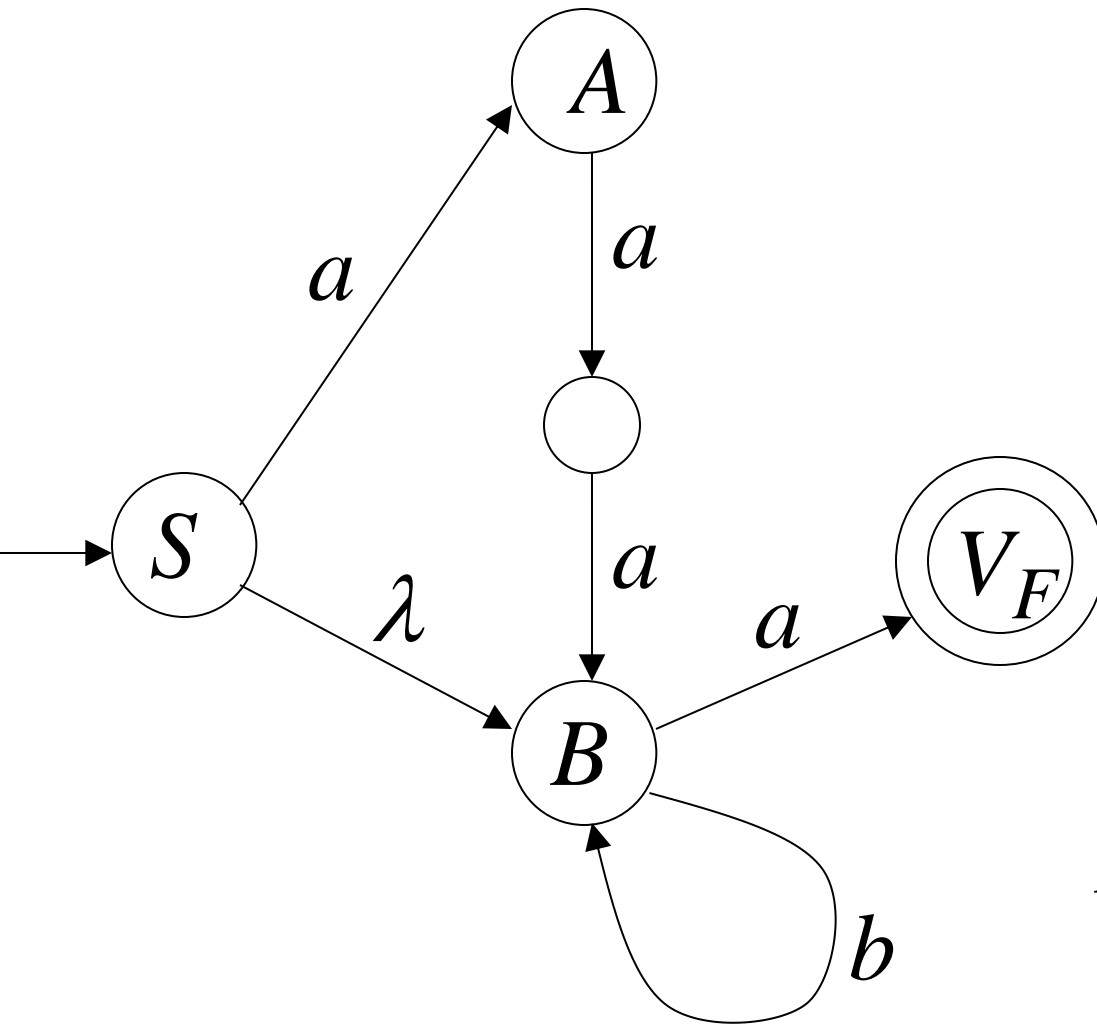
$$A \rightarrow aa B$$

$$B \rightarrow bB \mid a$$



$$S \Rightarrow aA \Rightarrow aaaB \Rightarrow aaabB \Rightarrow aaaba$$

NFA M



Grammar
 G

$$S \rightarrow aA \mid B$$

$$A \rightarrow aa B$$

$$B \rightarrow bB \mid a$$

$$L(M) = L(G) =$$

In General

A right-linear grammar G

has variables: V_0, V_1, V_2, \dots

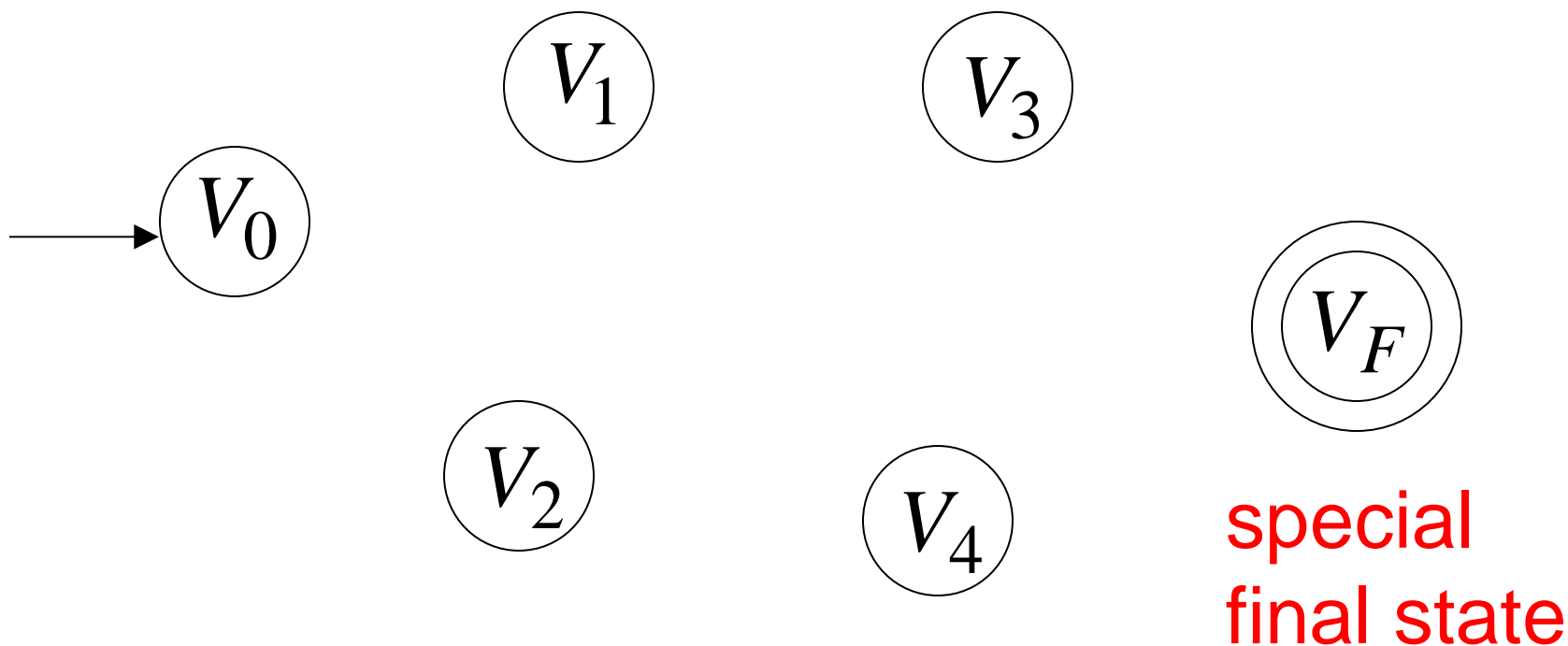
and productions: $V_i \rightarrow a_1 a_2 \cdots a_m V_j$

or

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

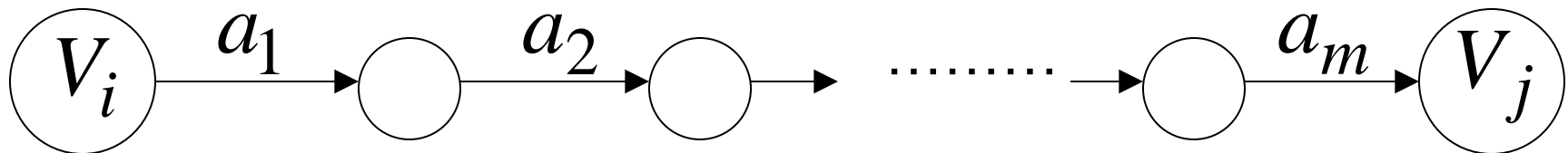
We construct the NFA M such that:

each variable V_i corresponds to a node:



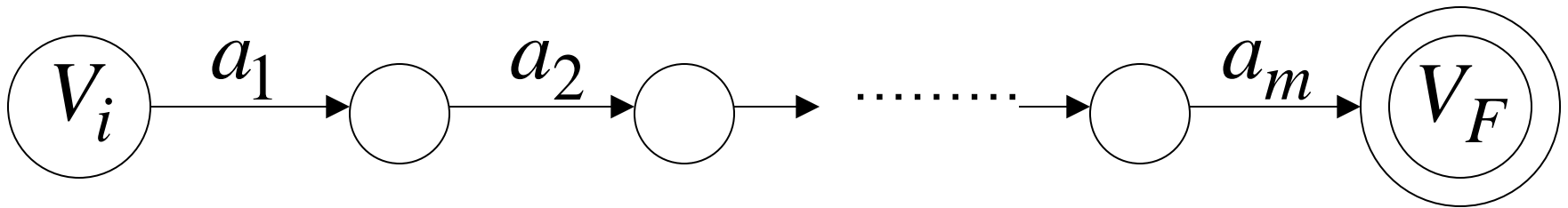
For each production: $V_i \rightarrow a_1 a_2 \cdots a_m V_j$

we add transitions and intermediate nodes



For each production: $V_i \rightarrow a_1 a_2 \cdots a_m$

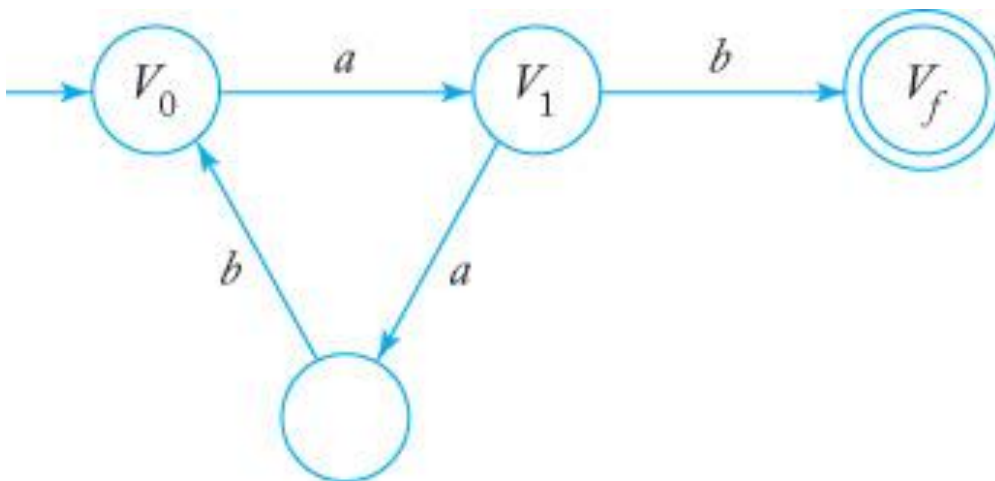
we add transitions and intermediate nodes



Example 3.15

- Construct a FA that accepts the language generated by the grammar

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



$L(G)=$

The case of Left-Linear Grammars

Let G be a left-linear grammar

We will prove: $L(G)$ is regular

Proof idea:

We will construct a right-linear grammar G' with $L(G) = L(G')^R$

Since G is left-linear grammar
the productions look like:

$$A \rightarrow Ba_1a_2 \cdots a_k$$

$$A \rightarrow a_1a_2 \cdots a_k$$

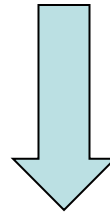
- Construct right-linear grammar G'

Left
linear

G

$$A \rightarrow Ba_1a_2 \cdots a_k$$

$$A \rightarrow Bv$$



Right
linear

G'

$$A \rightarrow a_k \cdots a_2a_1B$$

$$A \rightarrow v^R B$$

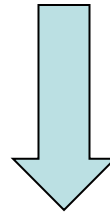
- Construct right-linear grammar G'

Left
linear

G

$$A \rightarrow a_1 a_2 \cdots a_k$$

$$A \rightarrow v$$



Right
linear

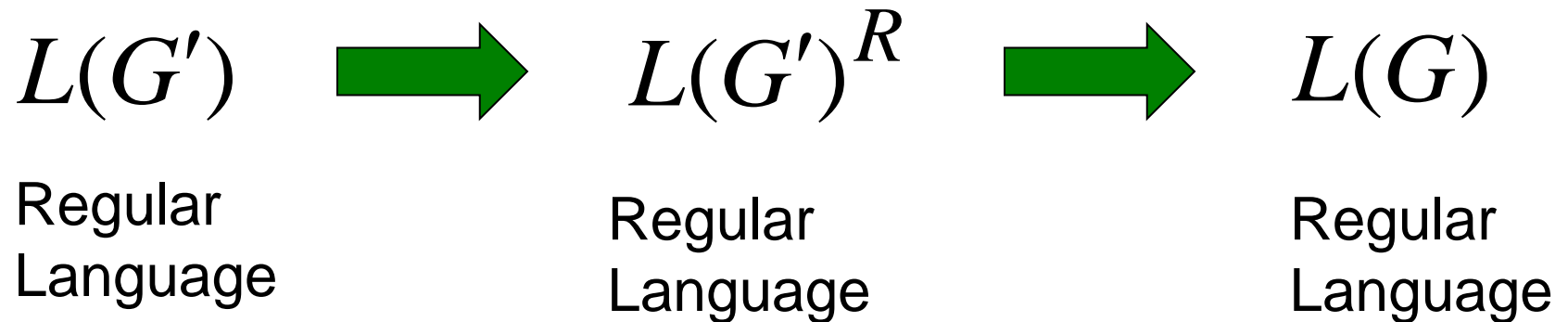
G'

$$A \rightarrow a_k \cdots a_2 a_1$$

$$A \rightarrow v^R$$

It is easy to see that: $L(G) = L(G')^R$

Since G' is right-linear, we have:



Proof - Part 2

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Grammars} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Any regular language L is generated
by some regular grammar G

Any regular language L is generated
by some regular grammar G

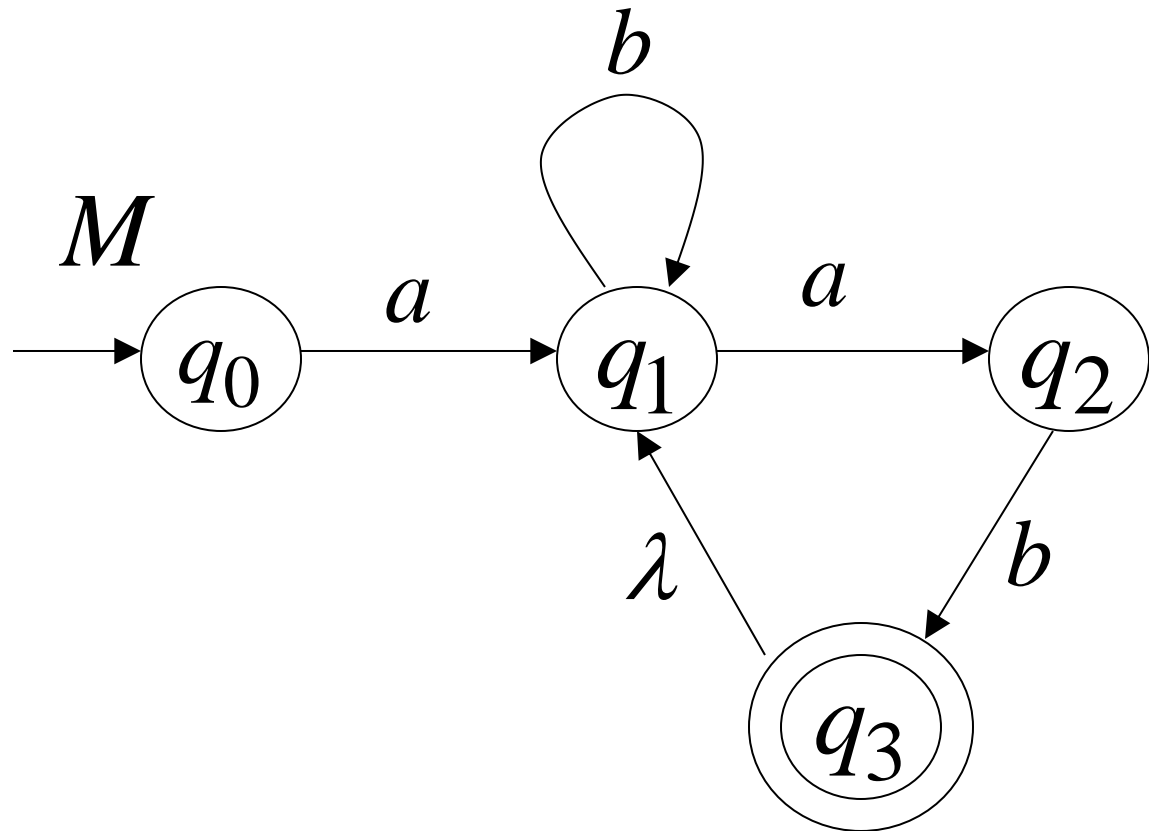
Proof idea:

Let M be the NFA with $L = L(M)$

Construct from M to a regular grammar G
such that $L(M) = L(G)$

Since L is regular
there is an NFA M such that $L = L(M)$

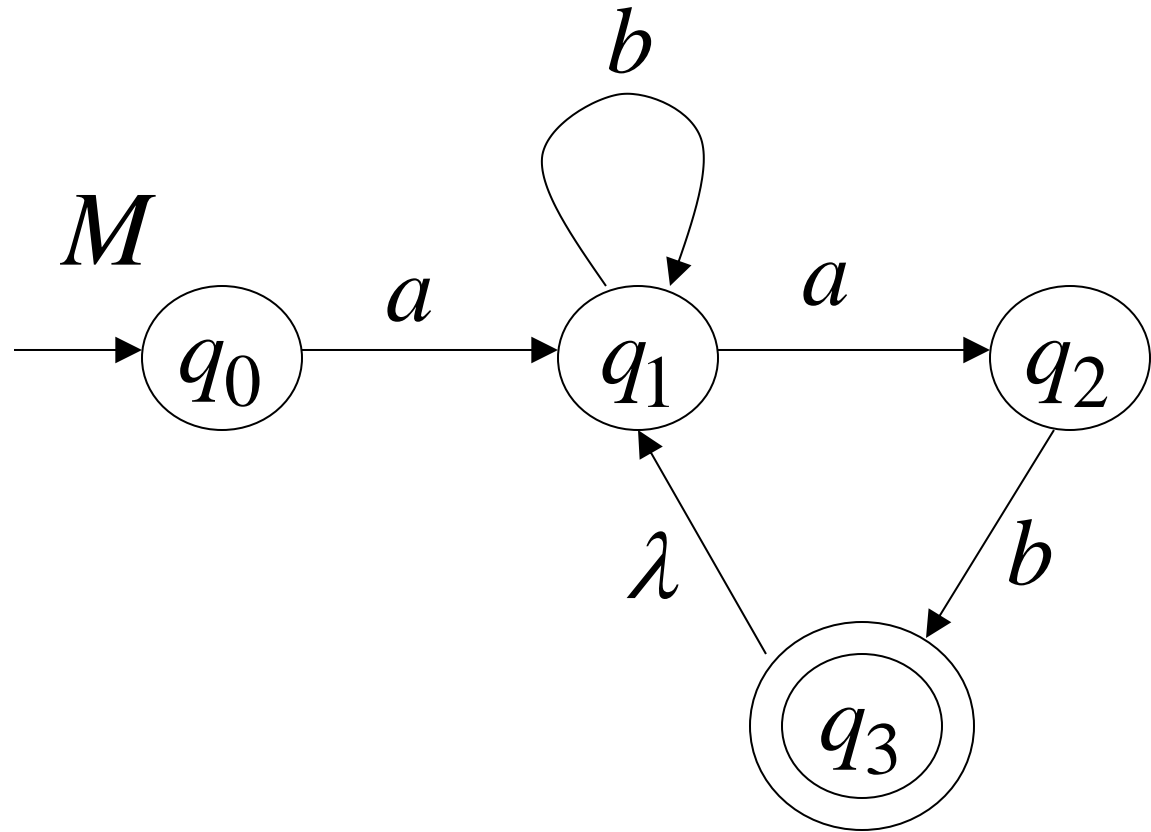
Example:



$L =$

$L = L(M)$

Convert M to a right-linear grammar
(BFS-like)

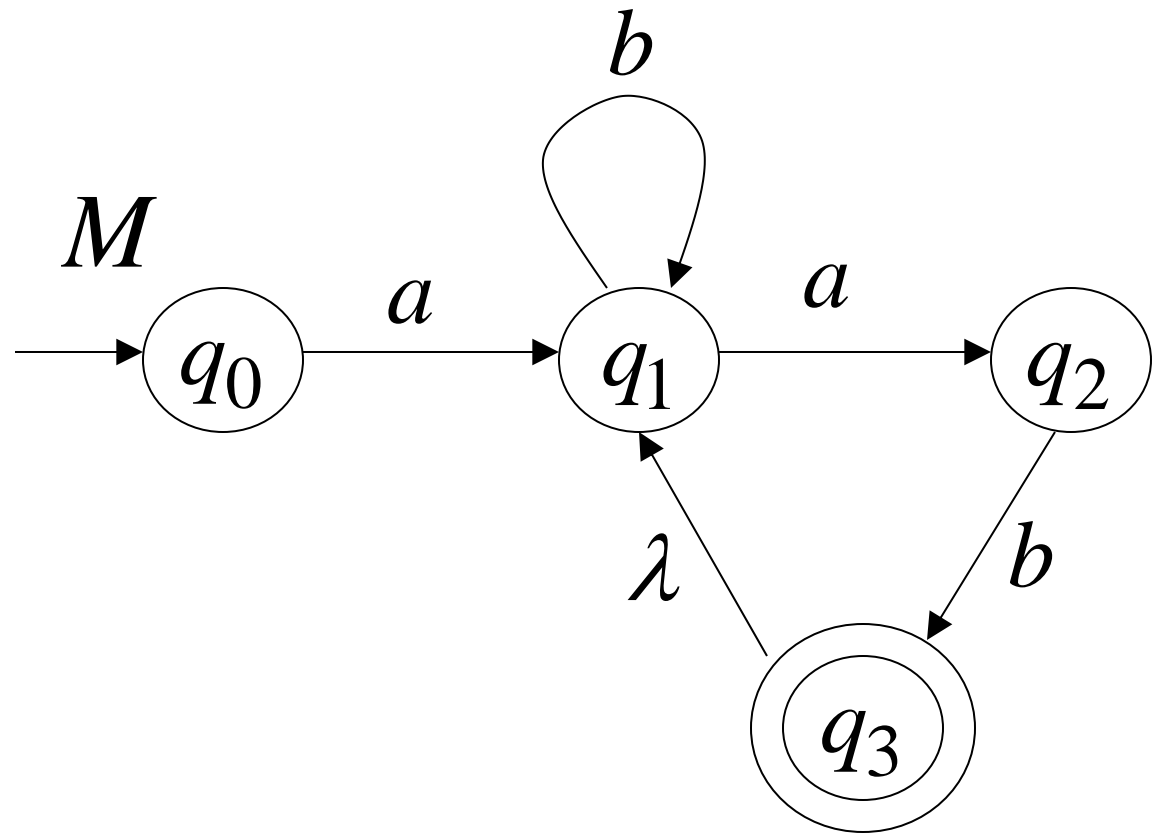


$$q_0 \rightarrow aq_1$$

$q_0 \rightarrow aq_1$

$q_1 \rightarrow bq_1$

$q_1 \rightarrow aq_2$

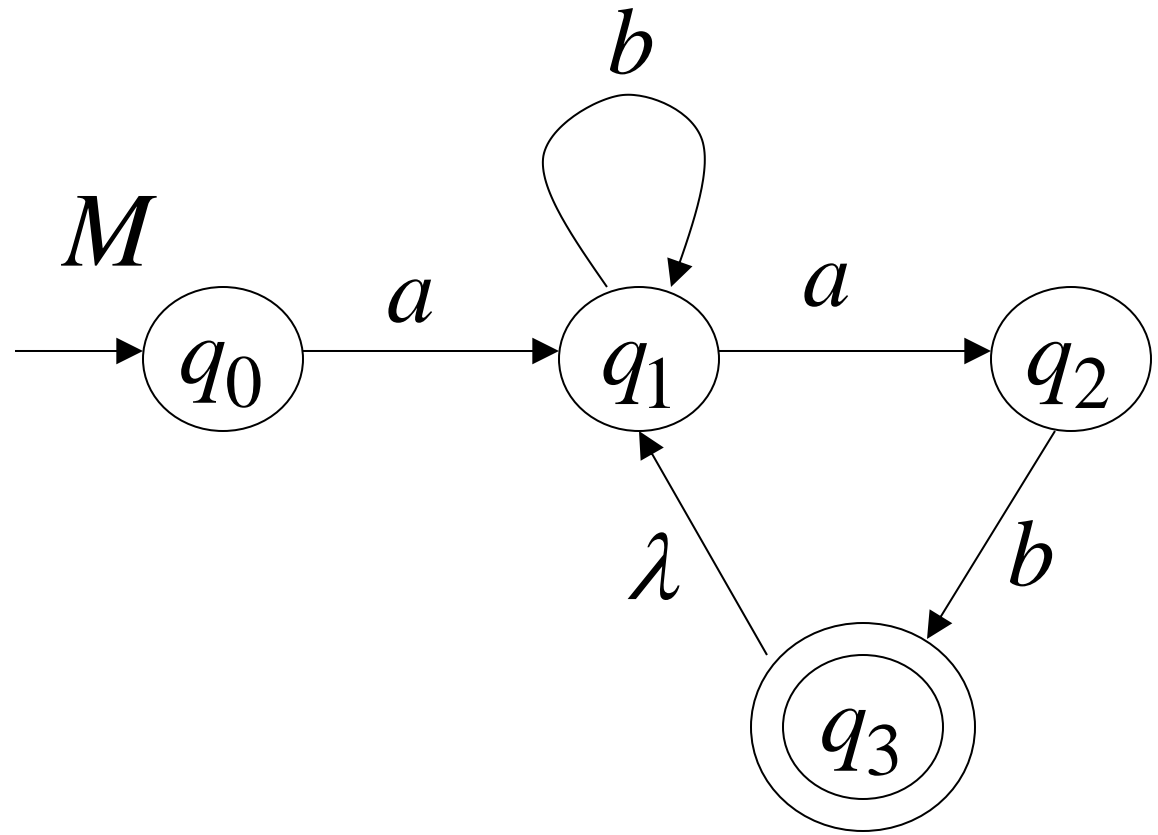


$q_0 \rightarrow aq_1$

$q_1 \rightarrow bq_1$

$q_1 \rightarrow aq_2$

$q_2 \rightarrow bq_3$



$$L(G) = L(M) = L$$

G

$q_0 \rightarrow aq_1$

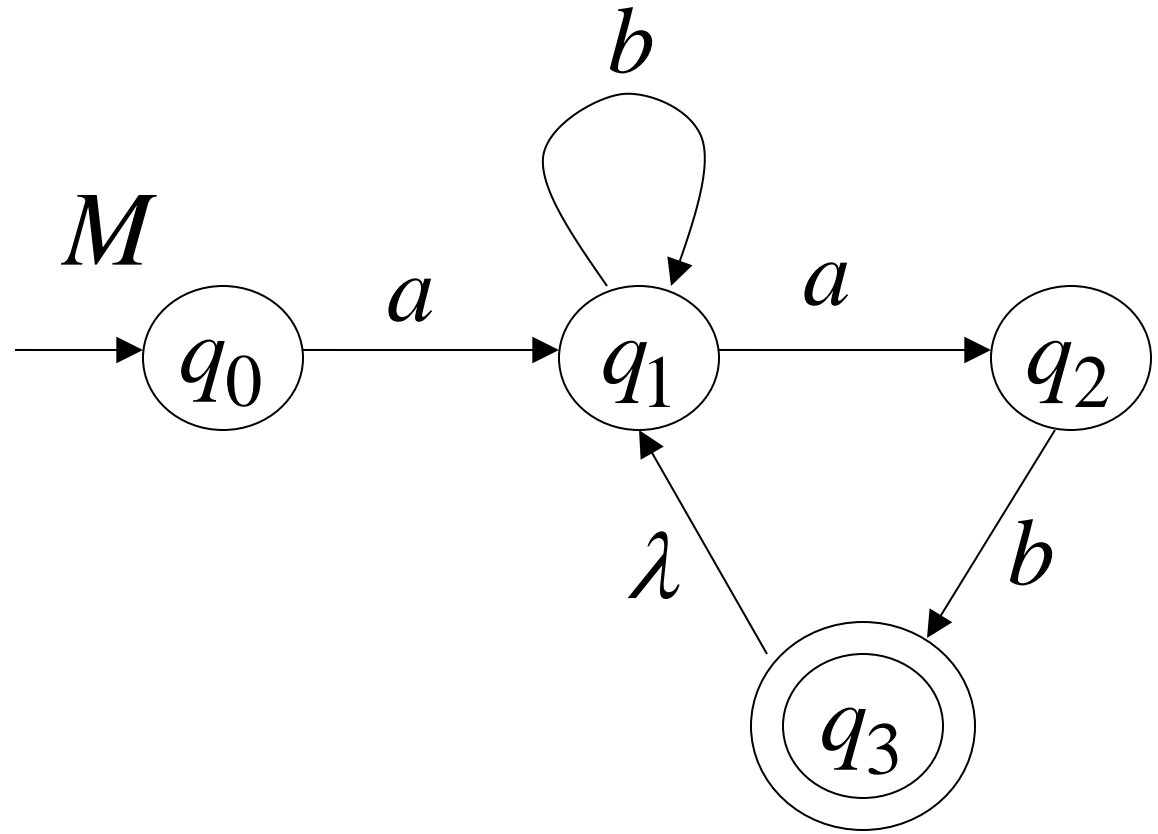
$q_1 \rightarrow bq_1$

$q_1 \rightarrow aq_2$

$q_2 \rightarrow bq_3$

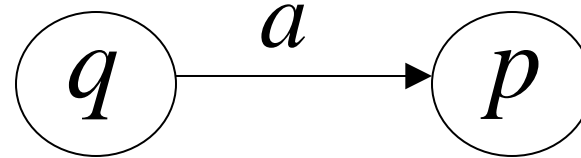
$q_3 \rightarrow q_1$

$q_3 \rightarrow \lambda$

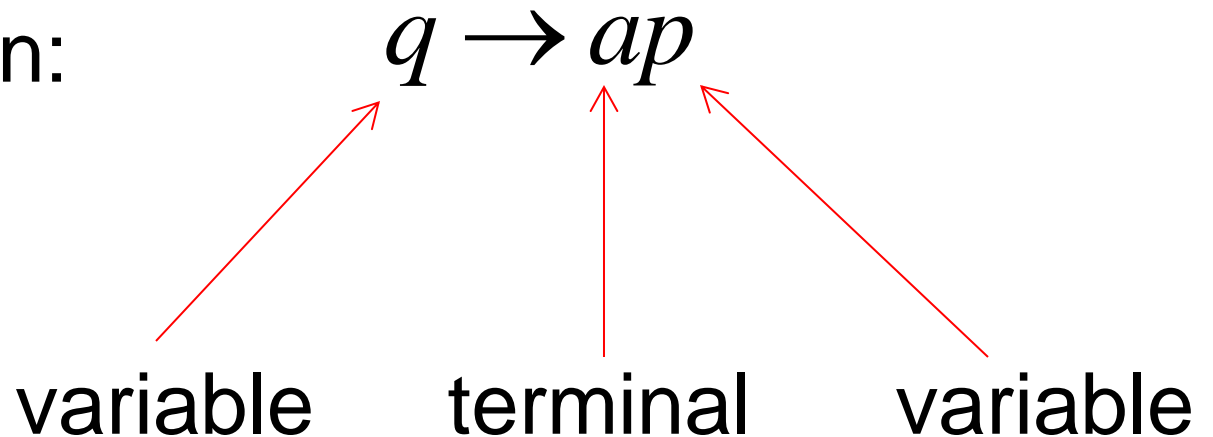


In General

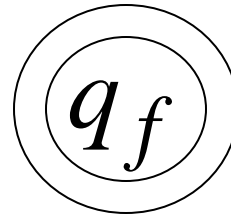
For any transition:



Add production:



For any final state:



Add production:

$$q_f \rightarrow \lambda$$

Since G is right-linear grammar

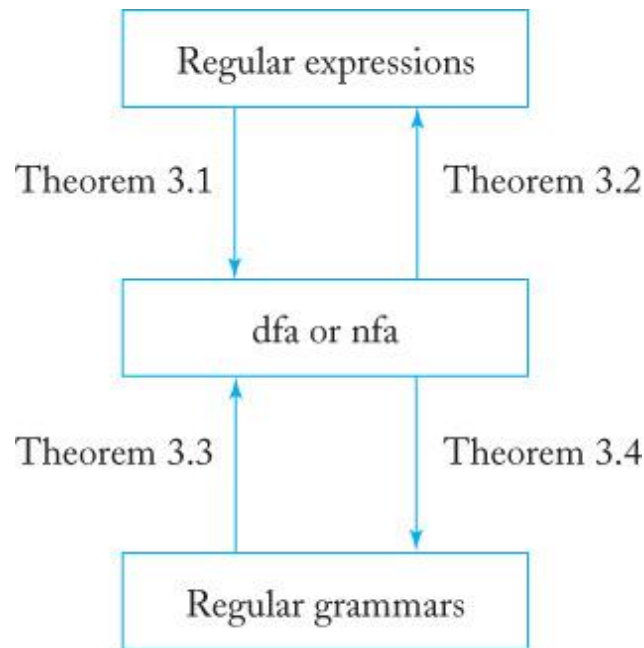
G is also a regular grammar

with $L(G) = L(M) = L$

Summary

- We now have several ways of describing **regular languages**:

- DFA
- NFA
- RE
- RG



Questions?