10/11

2023

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1. Regular Expressions (RE)
2. Connection Between REs and Regular Languages
3. 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 +, ·, * 0次以上

or concate

Regular Expressions

Important thing to remember

- A regular expression is not a language
  ↳ 是一個 modeling 的工具
- A regular expression is used to describe a language.
- X It is incorrect to say that for a language L, L = (a + b + c)*
- O 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

0次以上

or concate

Example: $(a \overset{\text{or}}{+} b \overset{\text{concate}}{\cdot} c)^*$ $\Longleftrightarrow$ {{a} U {bc}}*

describes the language

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

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). 以做出来的都是 regular expression
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

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

Definition 3.2

- For primitive regular expressions:

$$L(\varnothing) = \varnothing \qquad (1)$$

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

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

Definition (continued)

For regular expressions r_1 and r_2

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

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2) \qquad (5)$$

$$L(r_1 *) = (L(r_1))* \qquad (6)$$

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

Example 3.2

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

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

Priority of Operators

- Regular expression: $r = a \cdot b + c$

  ✓ $r_1 = a \cdot b \quad r_2 = c$ or ✗ $r_1 = a \quad r_2 = b + c$

  $L(r) = \{ab, c\} \neq \{ab, ac\}$

- Star closure (*) precedes concatenation (·) precedes union (+)

優先權

Example 3.3

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

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

↳ 結尾為 a or bb

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

↳ 所有 string

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

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

Example 3.4

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

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

偶a奇b

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

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

連續

$r = (0+1)^* 00 (0+1)^*$

Example 3.6

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

↳ 0的前後都要是1

- Regular expression (I) $\boxed{r = (1+01)^*(0+\lambda)}$

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

Add 1 immediately after a 0 — String ending in 0 — 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 = { all strings without two consecutive 0 }

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

$$r_2 = (1^*011^*)^*(0 + \lambda) + 1^*(0 + \lambda)$$

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

(I) • $L_2 = \{x \in \{0,1\}^* \mid |x| \text{ is even}\}$

(II) • $L_2 = \{00, 01, 10, 11\}^*$

• Regular expressions describing L_2:

(III) $(00 + 01 + 10 + 11)^*$

(IV) $((0 + 1)(0 + 1))^*$

More Examples

- $L_3 = \{x \in \{0,1\}^* \mid x \text{ does not end in } 01\}$ 以非0.1結尾

  If x does not end in 01, then either

  $|x| < 2$ or x ends in 00, 10, or 11

- A regular expression that describes L_3 is:

  $(\lambda + 0 + 1) + (0 + 1)^*(00 + 10 + 11)$

More Examples

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

  Express $x = yz$

  y is a string of the form $y=1^i01^j$

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

  $z = (01^k01^m)^*$

- A regular expression that describes L_4 is:

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

Short Quiz

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

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

Outline

1 Regular Expressions (RE)

2 Connection Between REs and Regular Languages

3 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)

↳ Regular expression ≡ DFA ≡ NFA

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)

与有NFA or DFA来描述 L(r)

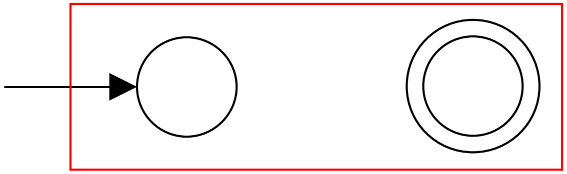
Proof by induction on the size of r

Induction Basis

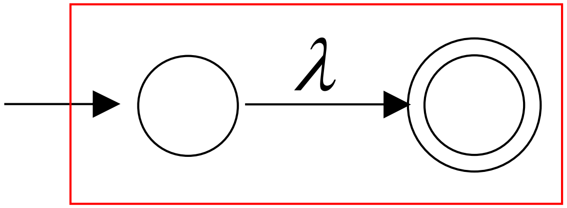
- Primitive Regular Expressions: $\emptyset, \quad \lambda, \quad a$

NFAs

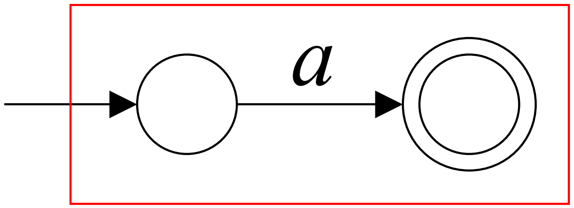
從Primitive Regular Expression 畫NFA



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



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



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

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

∵ REs are derived from these four rules:

We will prove:

$$\left.\begin{array}{l} L(r_1 + r_2) \\ \\ L(r_1 \cdot r_2) \\ \\ L(r_1 *) \\ \\ L((r_1)) \end{array}\right\}$$

Are regular Languages

- By definition of regular expressions:

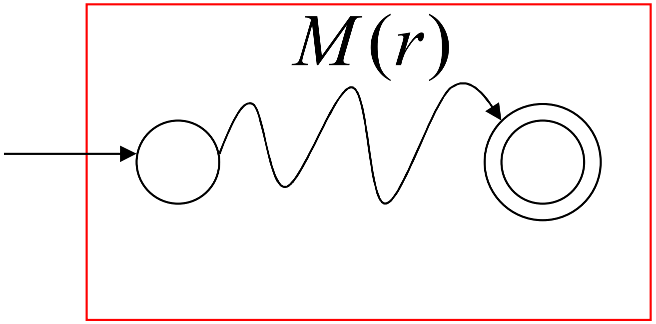
$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

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

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

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

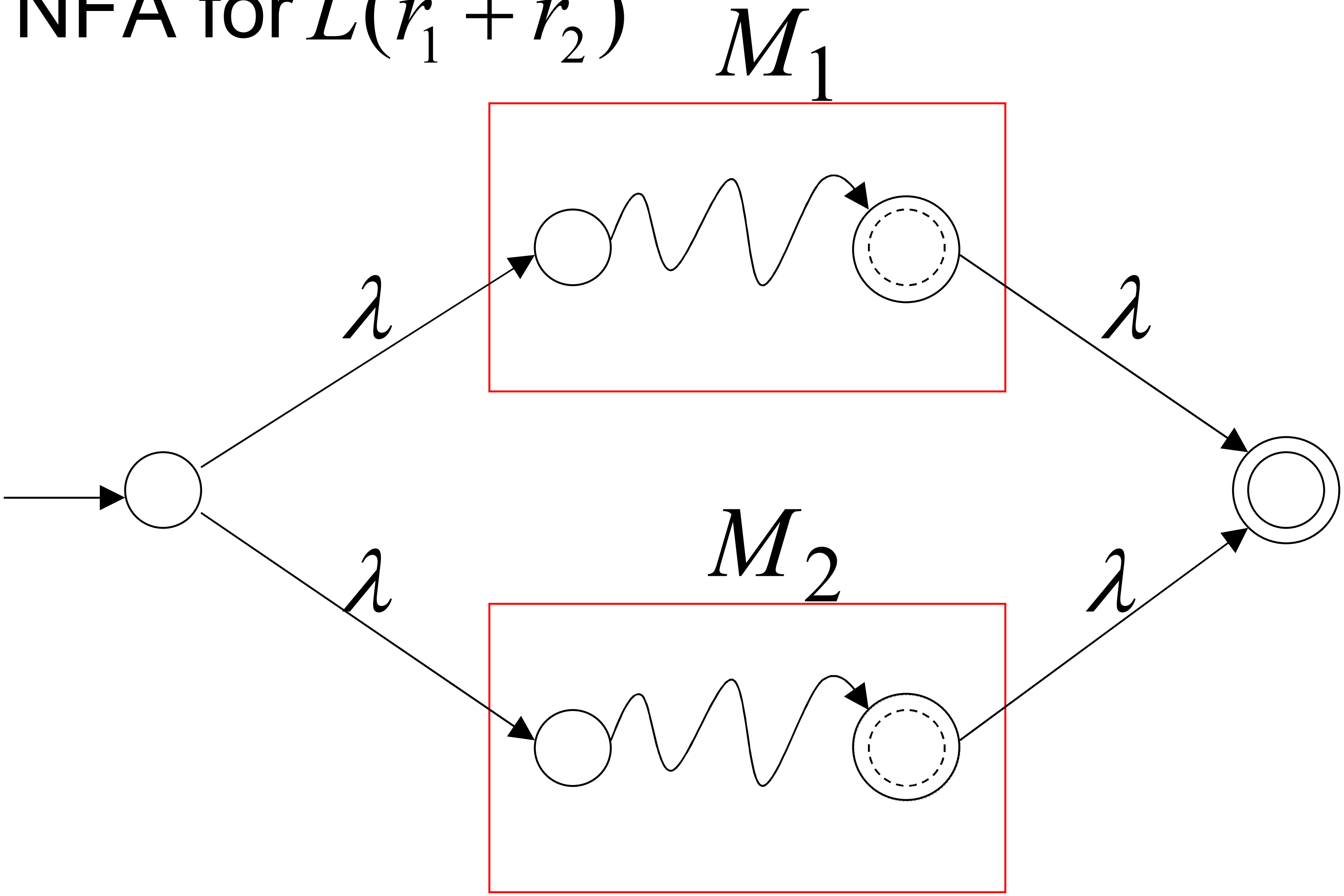
Schematic representation of an NFA (M(r)) accepting L(r)



We can claim that for every NFA there is only one final state (by exercise 7, section 2.3) ↳ NFA可畫成只有一個final

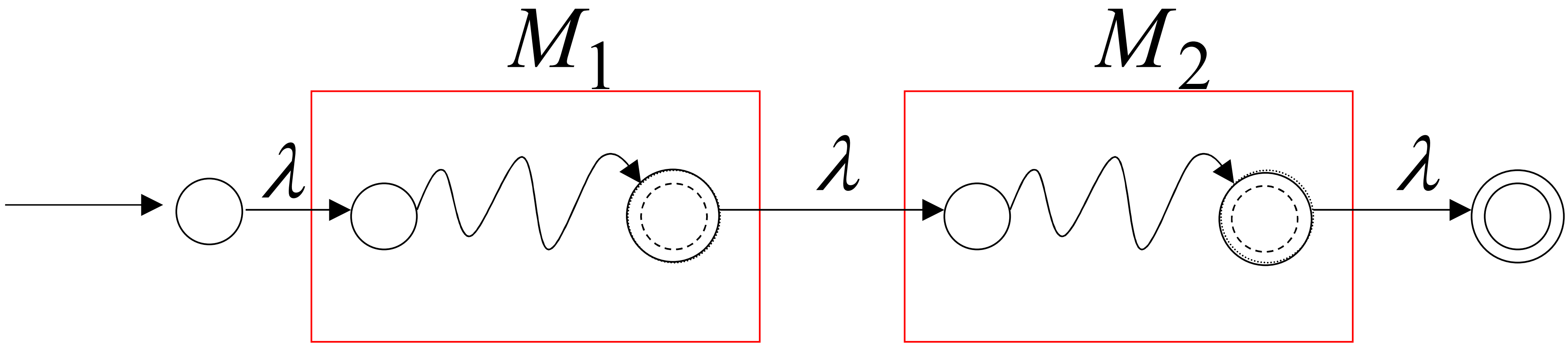
Union

- NFA for $L(r_1 + r_2)$



Concatenation

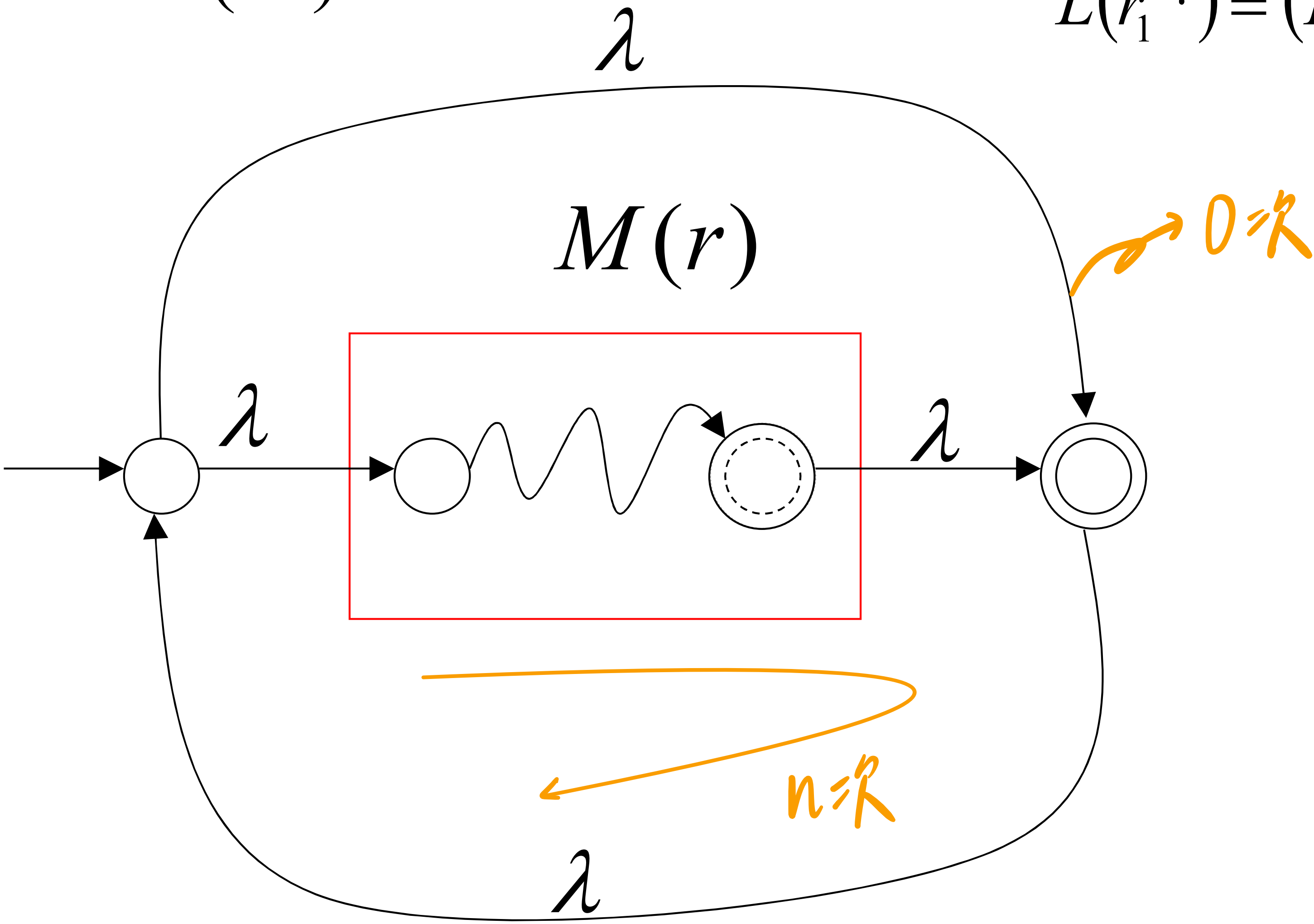
- NFA for $L(r_1 r_2)$



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:

$$\left.\begin{aligned} L(r_1 + r_2) &= L(r_1) \cup L(r_2) \\[2em] L(r_1 \cdot r_2) &= L(r_1)\, L(r_2) \\[2em] L(r_1 *) &= (L(r_1))^* \end{aligned}\right\} \text{Are regular languages}$$

And trivially:

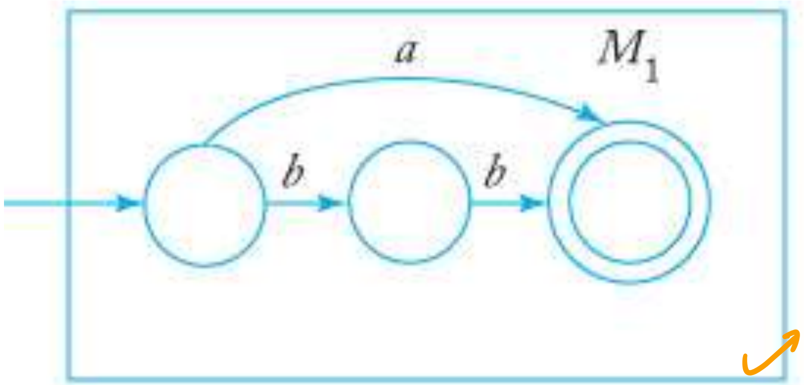
$L((r_1))$ is a regular language

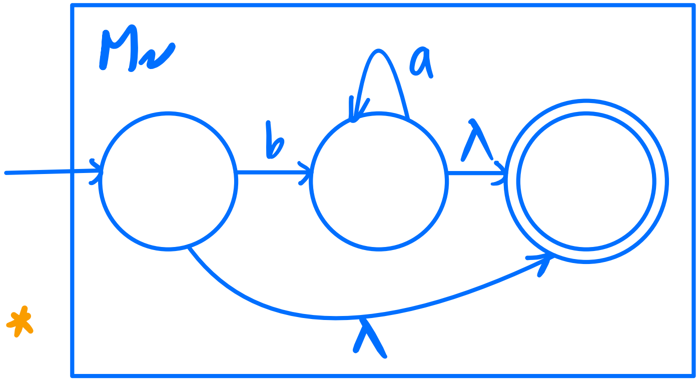
$\therefore$ For any regular expression r the language $L(r)$ is regular

Example 3.7

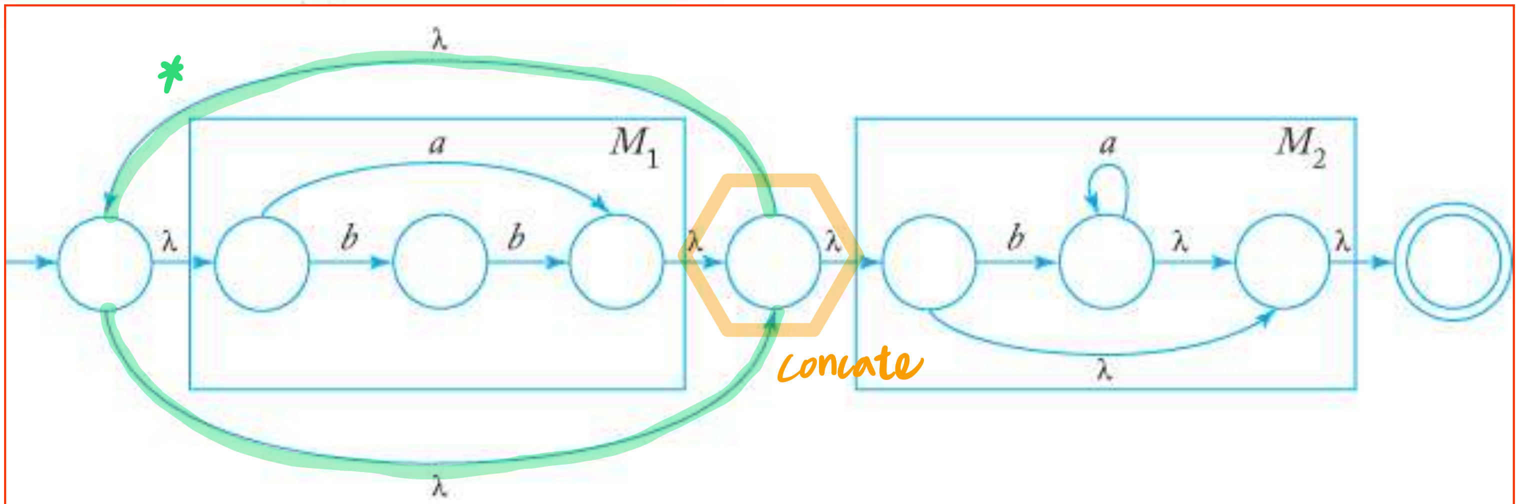
- Find an NFA that accepts L(r), where

$$r = (a + bb)^*(ba^* + \lambda)$$





未加上*



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)

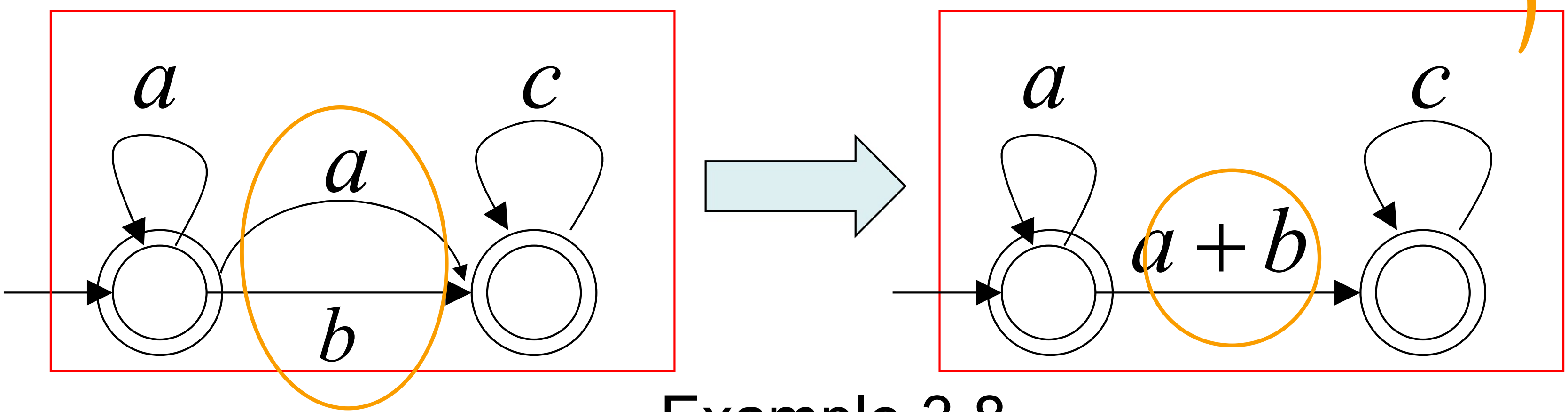
From M construct the equivalent

Generalized Transition Graph ↳ 把edge寫regular expression

in which transition labels are regular expressions

Example: M

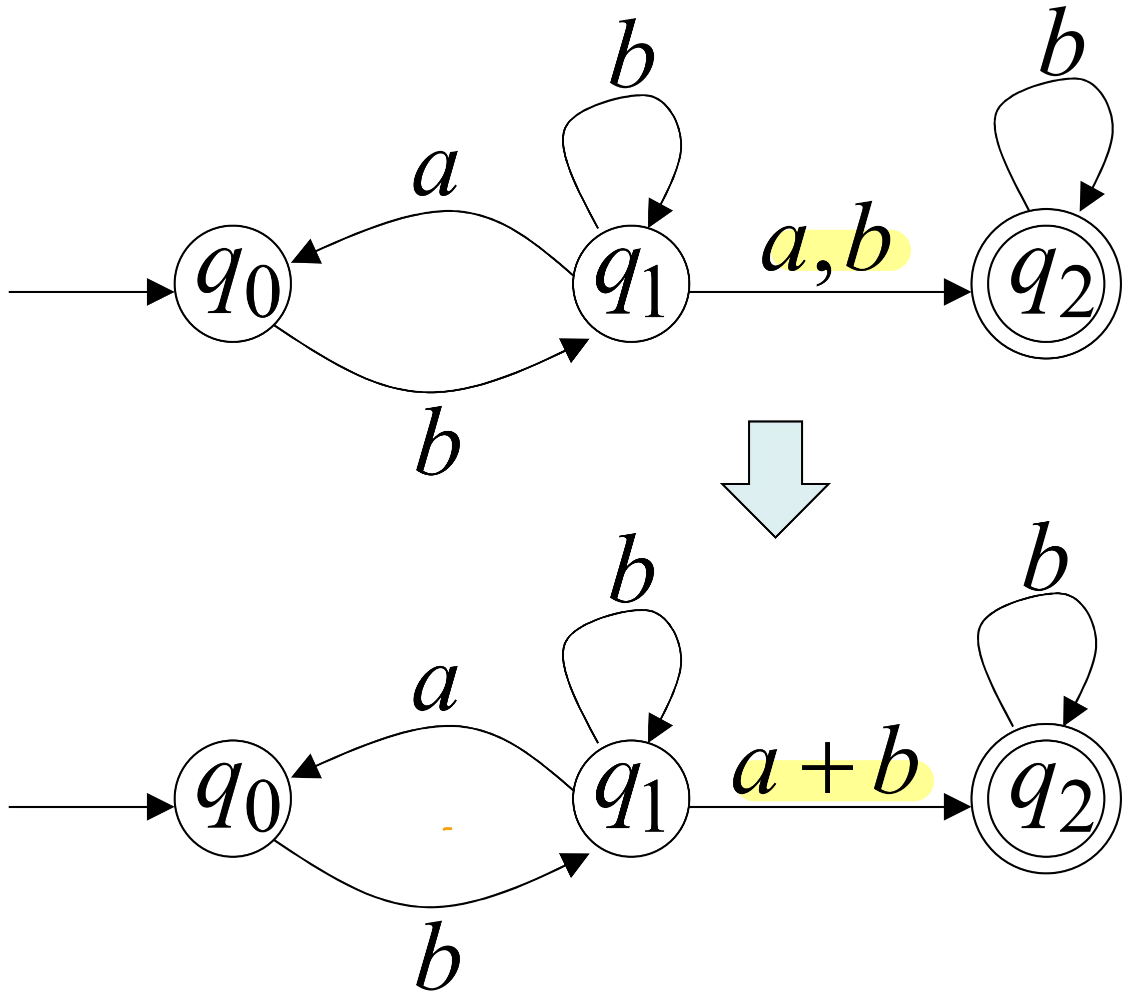
L(a* + a*(a + b)c*)



Example 3.8

GTG may have many states

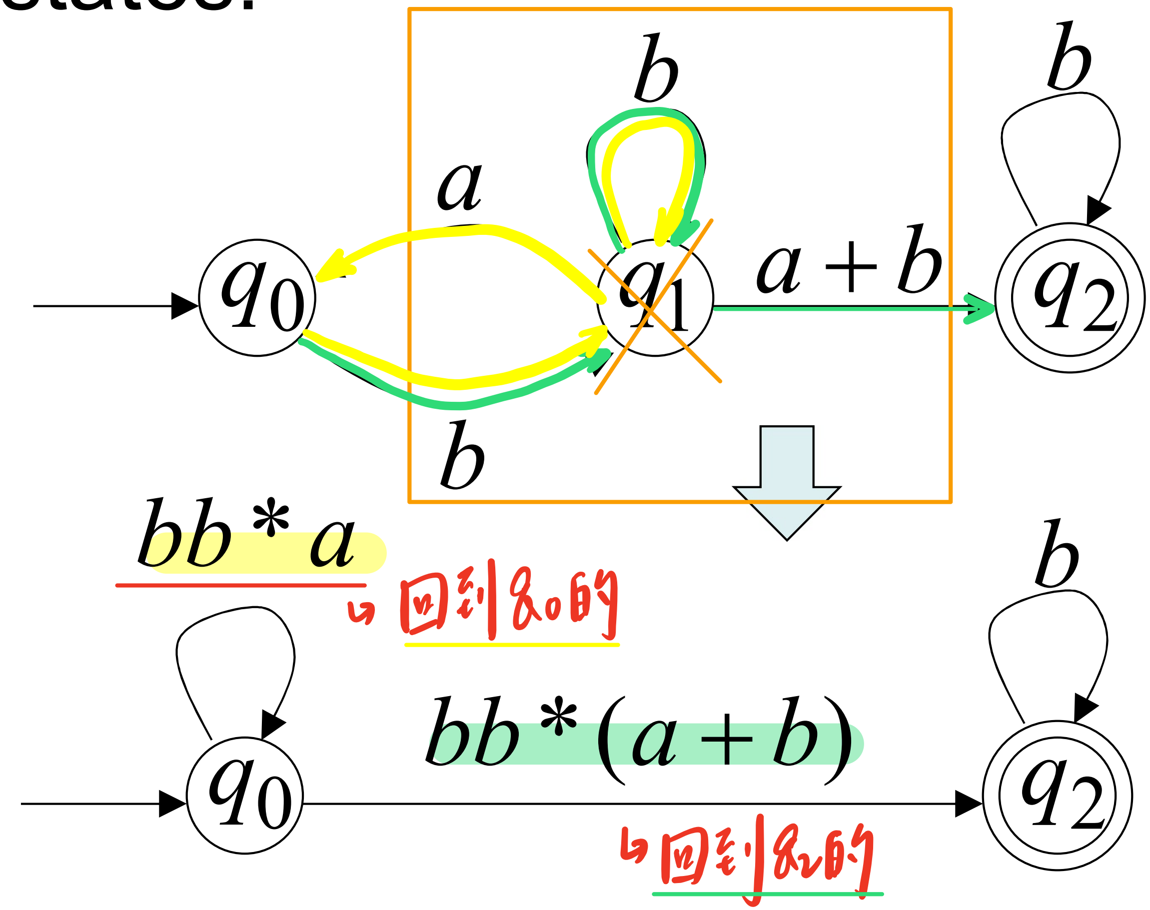
Enumerating all walks is time-consuming



Reducing the states:

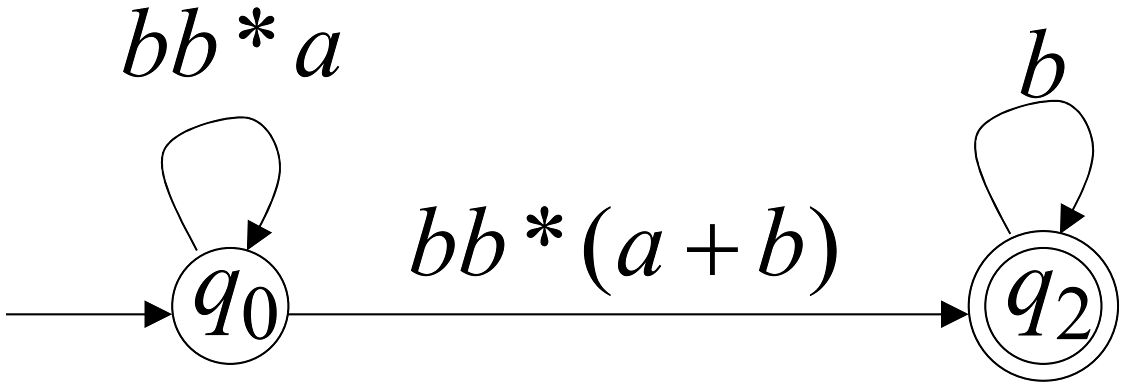
Ex. reduce q_1

分段處理



Simple two-state GTG

Resulting Regular Expression:



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

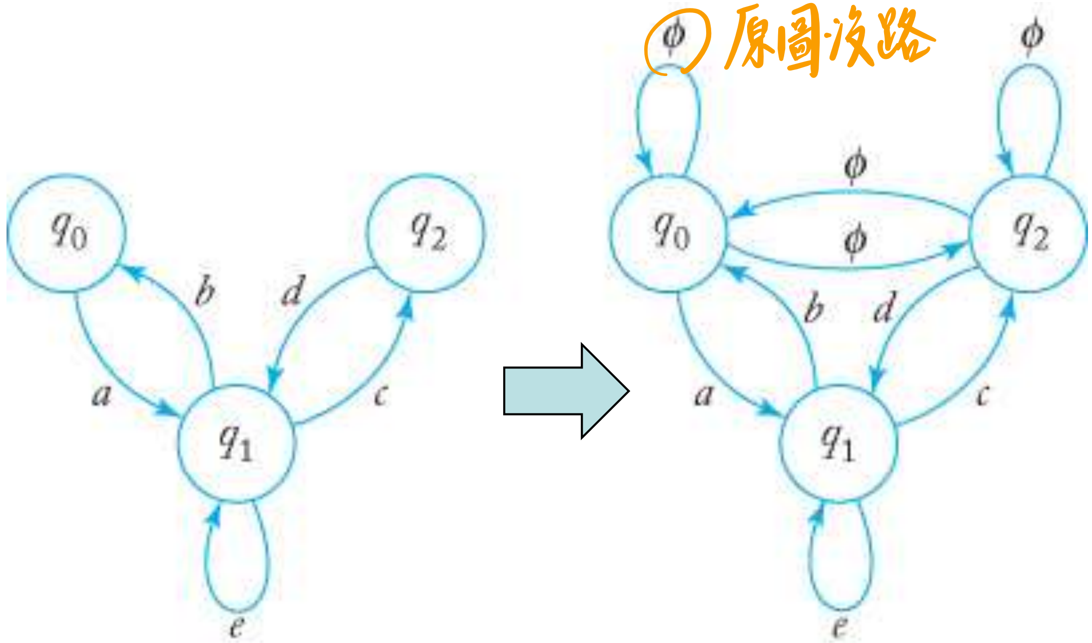
↳ reduce 到只剩 initial 和 final 即可寫出 regular expression

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

Complete GTG

GTG

Complete GTG

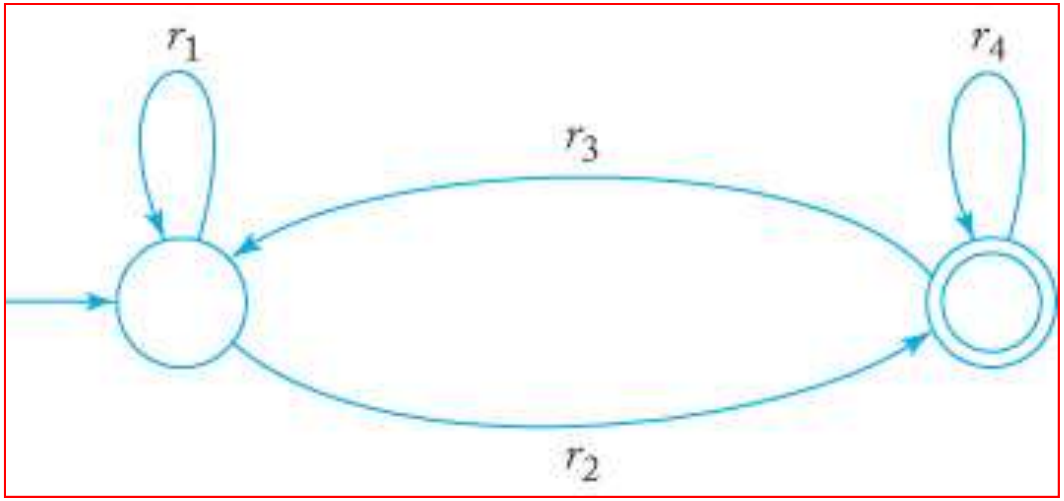


(a) (b)

補足自己走到所有人(含自己)的路徑

- 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



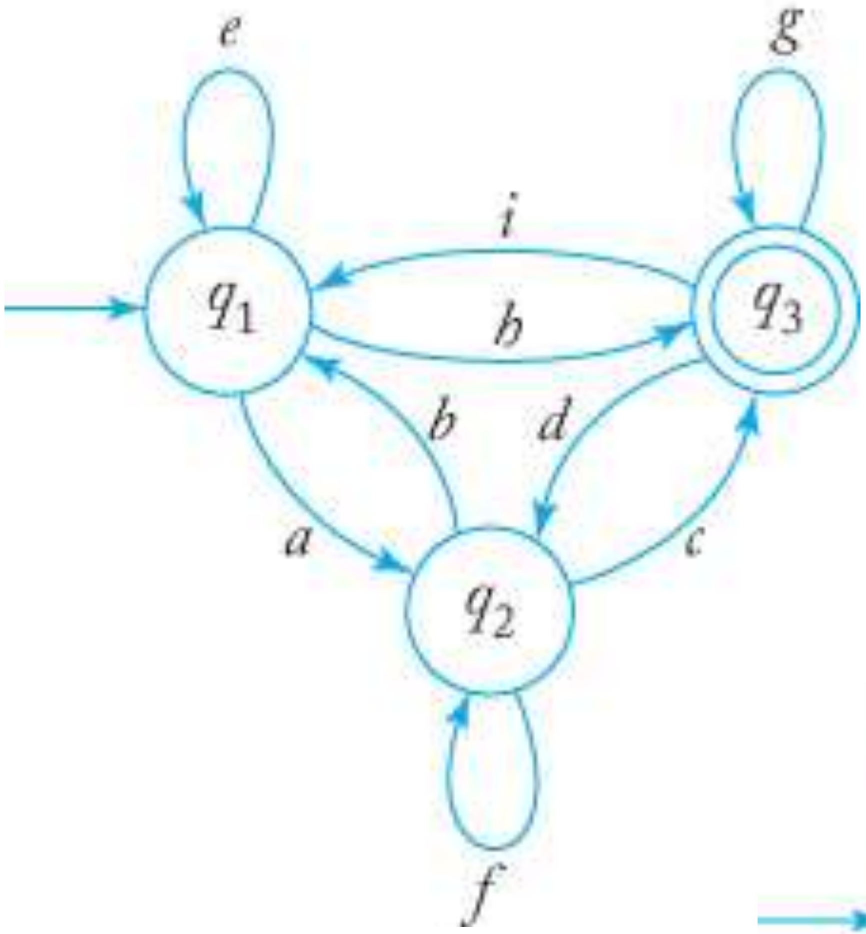
RE?

$r = r_1^*r_2(r_4 + r_3r_1^*r_2)^*$

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:

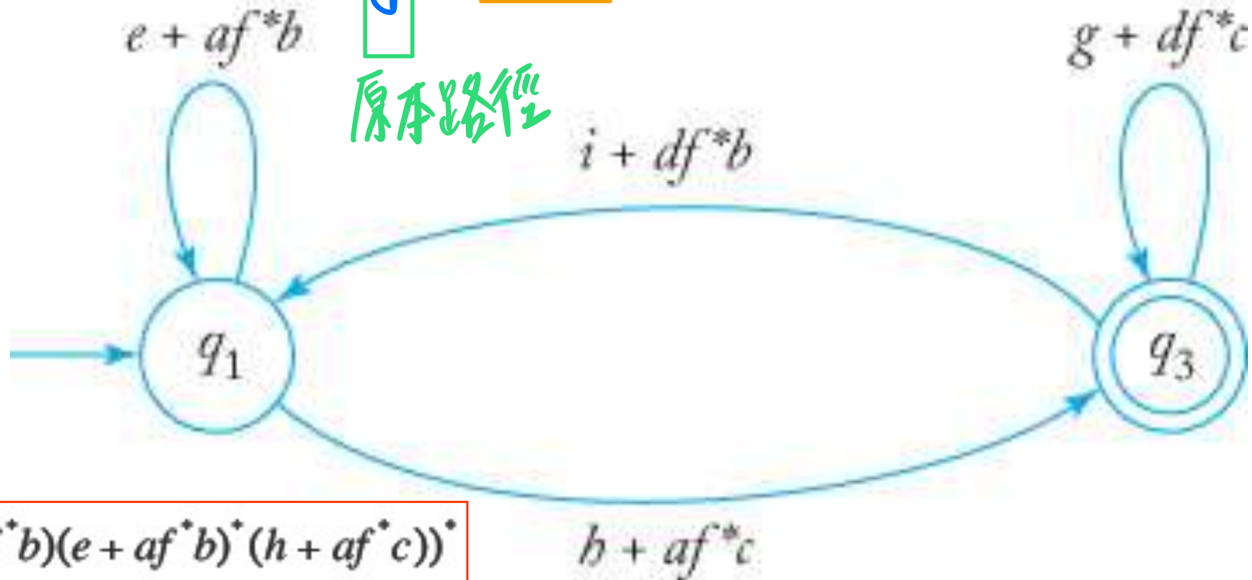
$\overrightarrow{q_1q_1} \rightarrow$ e or $+ \ af^*b$

$\overrightarrow{q_1q_3} \rightarrow$ $h + af^*c$

$\overrightarrow{q_3q_1} \rightarrow$ $i + df^*b$ 經q_2

$\overrightarrow{q_3q_3} \rightarrow$ $g + df^*c$

原本路徑



$$r = r_1^*r_2(r_4 + r_3r_1^*r_2)^*$$

$$\boxed{(e+af^*b)^*(h+af^*c)((g+df^*c)+(i+df^*b)(e+af^*b)^*(h+af^*c))^*}$$

o 10/18

NFA → 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:

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

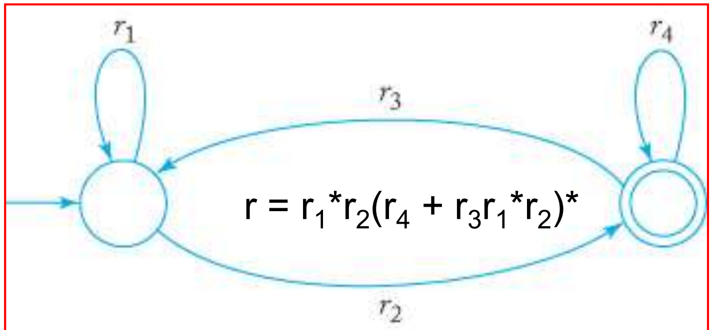
↳ 只剩2 state時使用

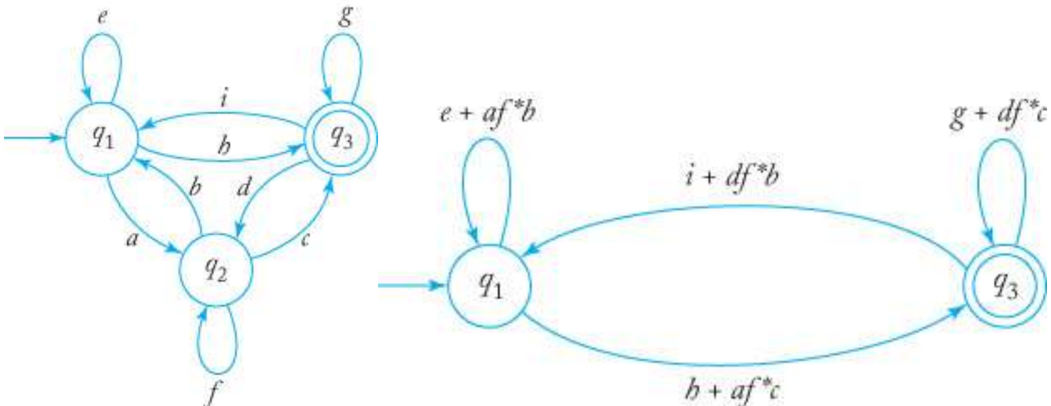
3. If the GTG has three states with $q_i \in q_0$, $q_j \in F$, and $q_k \in Q$, introduce new edges, labeled:

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

↳ 中間state消除

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





NFA → 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

$$r \overset{\text{or}}{+} \phi = r, \quad r \overset{\text{concate}}{\cdot} \phi = \phi, \quad \phi^* = \lambda$$

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

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

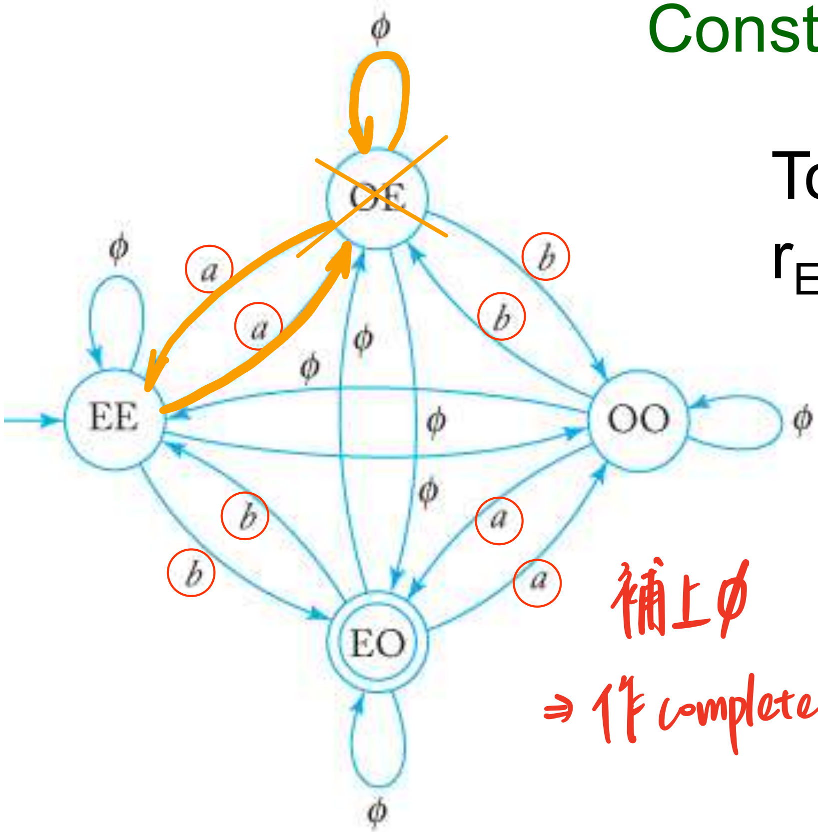
Example 3.11

- Find a RE for the language

a偶 b奇

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

Construct NFA first!



補上ϕ

⇒ 作 complete GTG

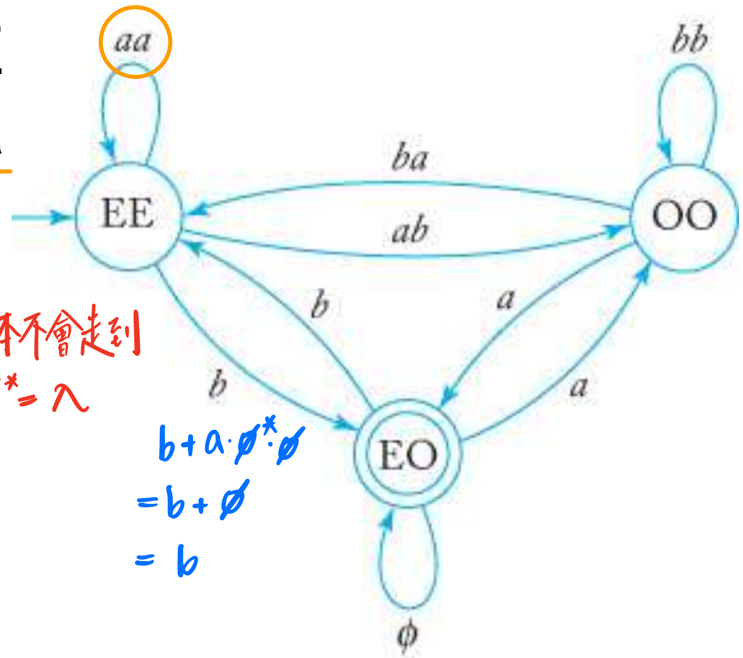
To remove OE

$r_{EE} = \phi + a\,\phi^{*}a$

$= aa$

∵根本不會走到

∴ $\phi^{*} = \lambda$



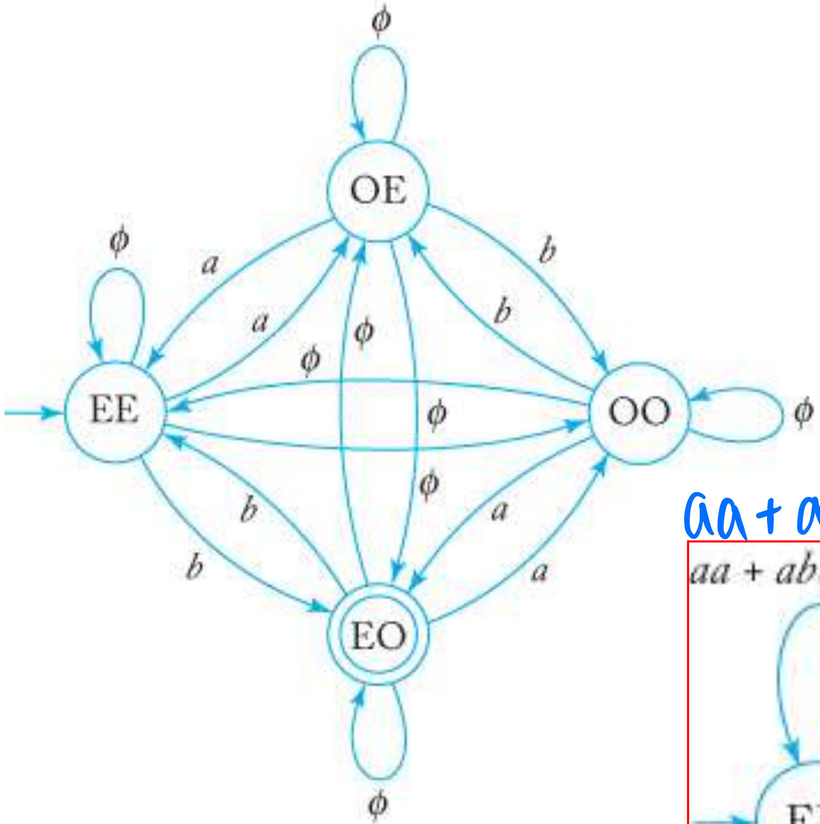
$b + a\cdot\phi^{*}\cdot\phi$

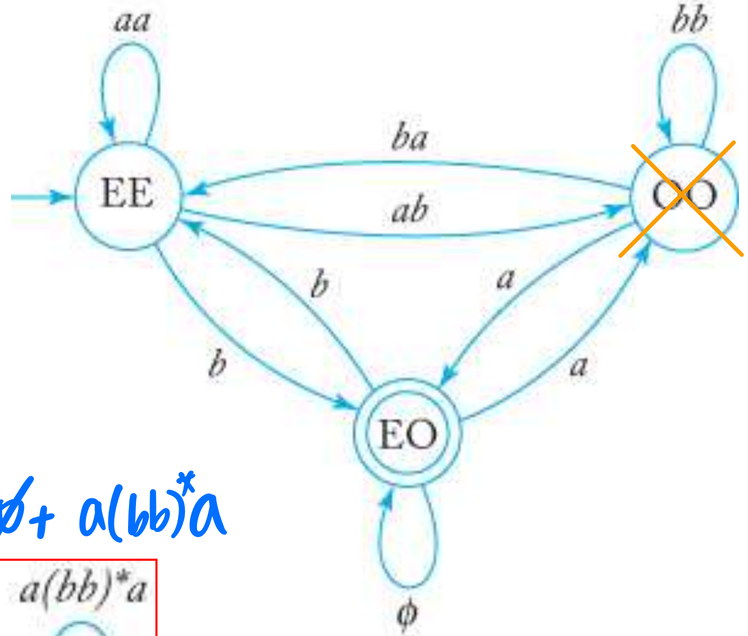
$= b + \phi$

$= b$

Example 3.11

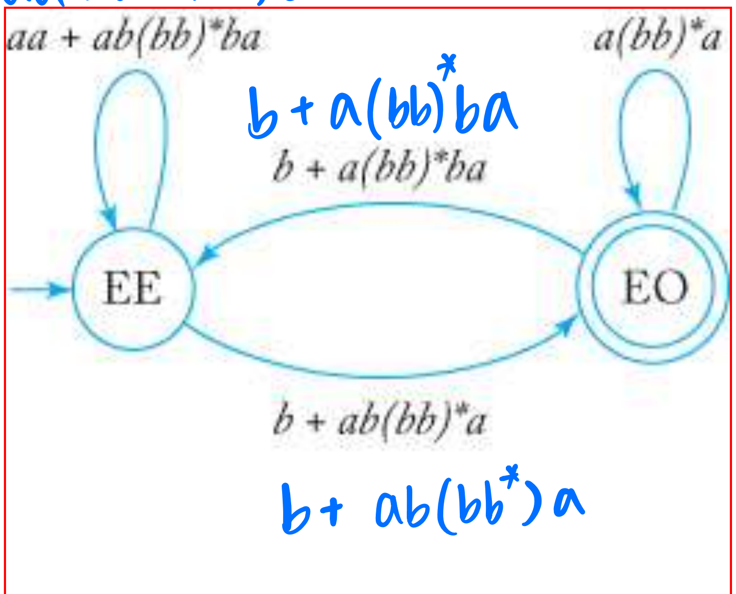
$$L = \{w \in \{a,b\}^* : n_a(w) \text{ is even and } n_b(w) \text{ is odd}\}.$$





aa + ab(bb)*ba

∅ + a(bb)*a



Outline

1. Regular Expressions (RE)
2. Connection Between REs and Regular Languages
3. 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

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

is the language generated by G

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

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

is a derivation of the sentence w.

- $S, w_1, w_2, \ldots, w_n$ are called sentential forms

  ↳至少含有一個 variable

Linear Grammars

Grammars with

at most one variable at the right side

of a production

variable : 大寫

terminal : 小寫

Examples:

$$S \to aSb$$

$$S \to \lambda$$

$$S \to Ab$$

$$A \to aAb$$

$$A \to \lambda$$

Another Linear Grammar

↳右邊只有一個 variable

Grammar G :

$$S \to A$$

or

$$A \to aB \mid \lambda$$

$$B \to Ab$$

A
aB
aAb
aaBb
aaAbb
aaλbb
aabb

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

↳非 regular：畫不出 NFA、DFA

A Non-Linear Grammar

↳ 右邊有2個variable

Grammar G:

$$S \to SS$$

$$S \to \lambda$$

$$S \to aSb$$

$$S \to bSa$$

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

Number of a in string w

Right-Linear Grammars

↳ variable 只能在最右邊 or 無 variable

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

  or

  $A \to x$

  string of terminals

- Example: $S \to abS$

  $S \to a$

Left-Linear Grammars

↳ variable 只能在最左邊 or 無 variable

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

  or

  $A \to x$

  string of terminals

- Example: $S \to Aab$

  $A \to Aab \mid B$

  $B \to a$

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

$$G_3$$

$$S \to A$$

$$A \to aB \mid \lambda$$

$$B \to Ab$$

Example 3.13

Regular grammars generate regular languages

G_1

$$S \rightarrow abS$$

$$S \rightarrow a$$

G_2

$$S \rightarrow Aab$$

$$A \rightarrow Aab \mid B$$

$$B \rightarrow a$$

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

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

Theorem <proof>

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

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 ⇒ 找NFA、DFA

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

* Grammar G is right-linear

Example:

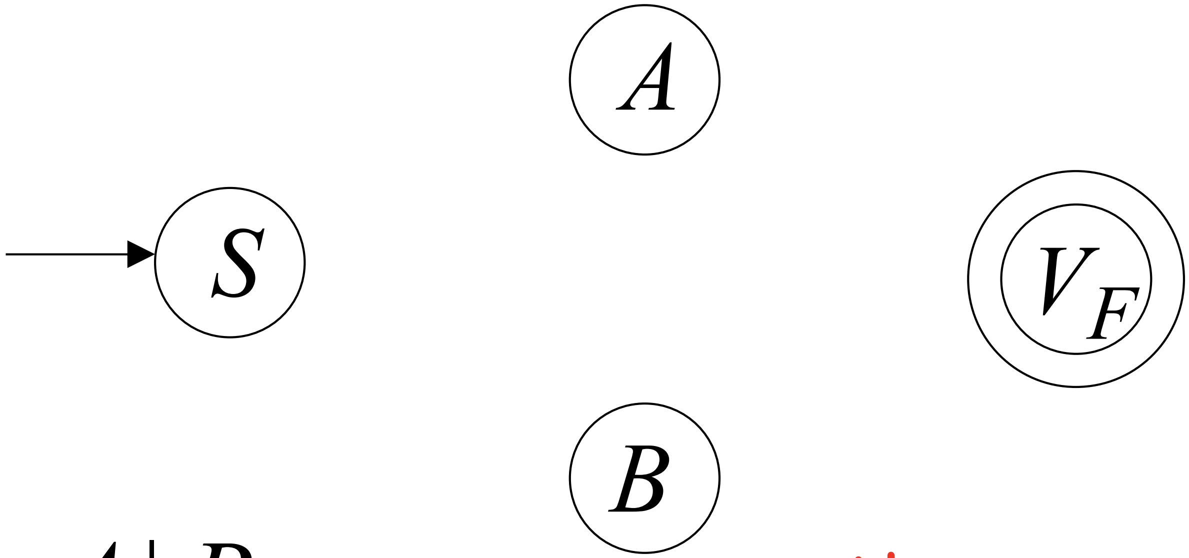
$$S \to aA \mid B$$

$$A \to aa\, B$$

$$B \to b\, B \mid a$$

right-linear

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



special final state

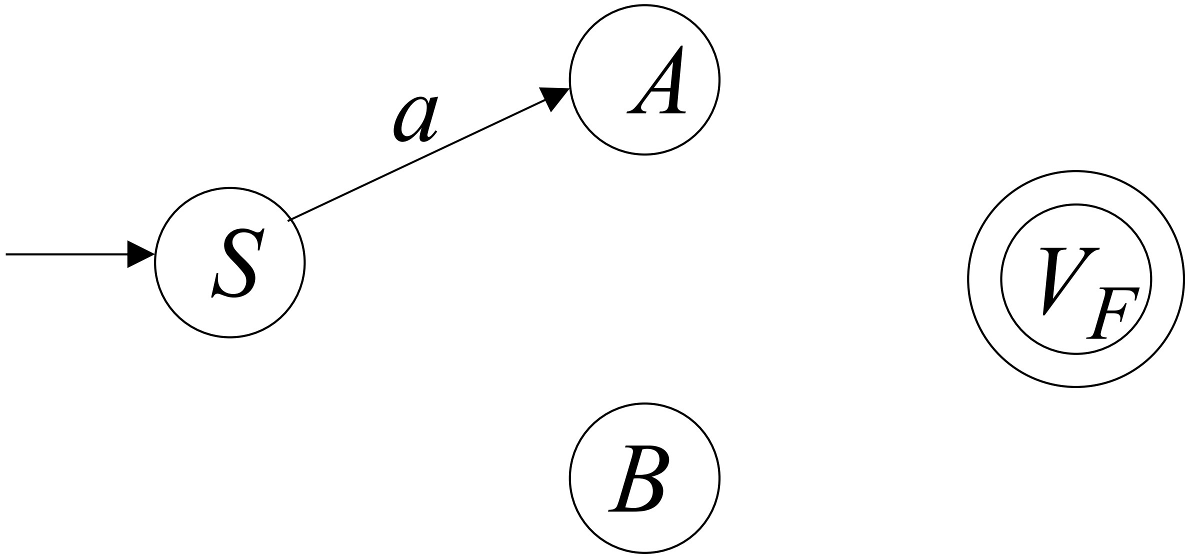
↳ variable

$$S \to aA \mid B$$

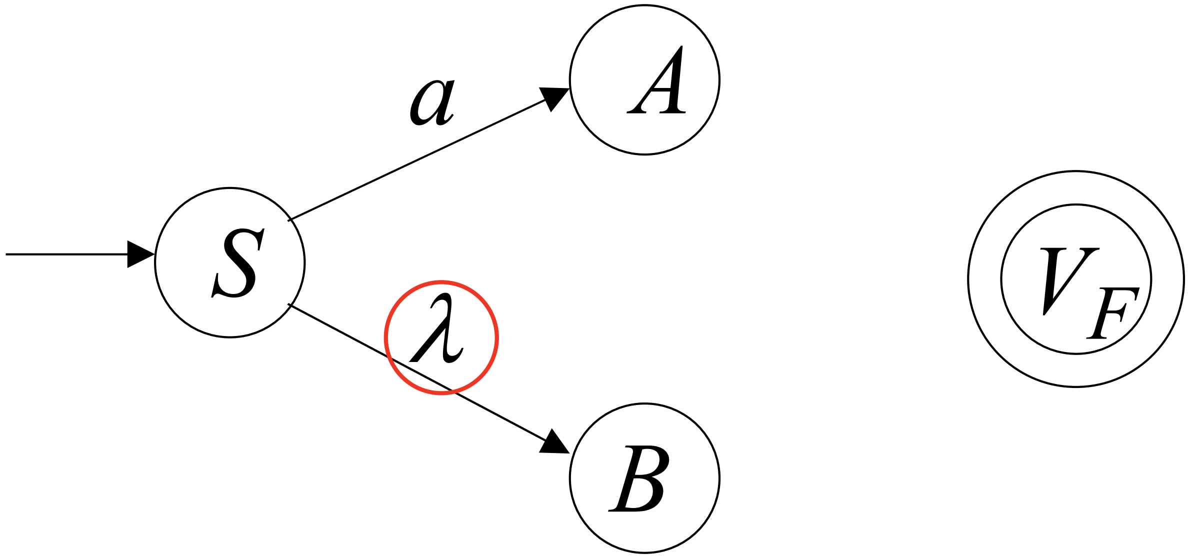
$$A \to aa\,B$$

$$B \to b\,B \mid a$$

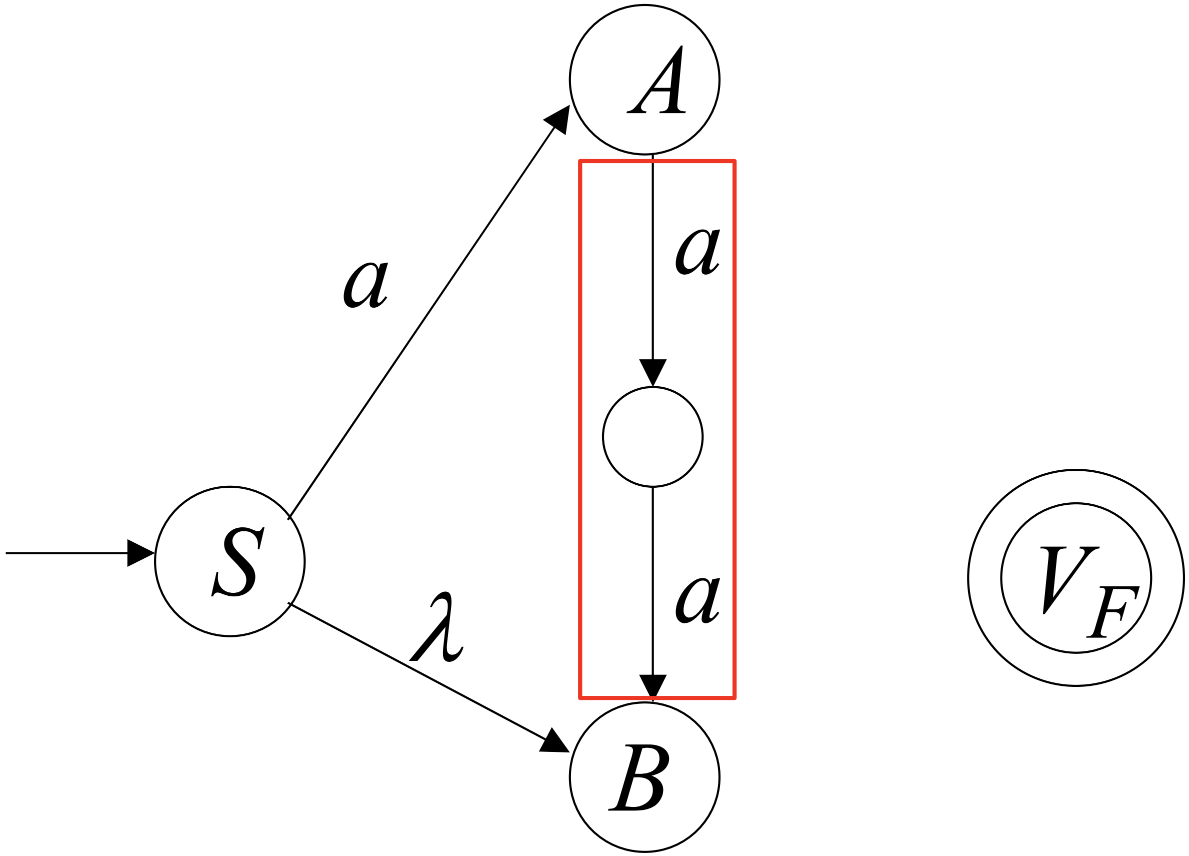
- Add edges for each production:



$$S \to aA$$

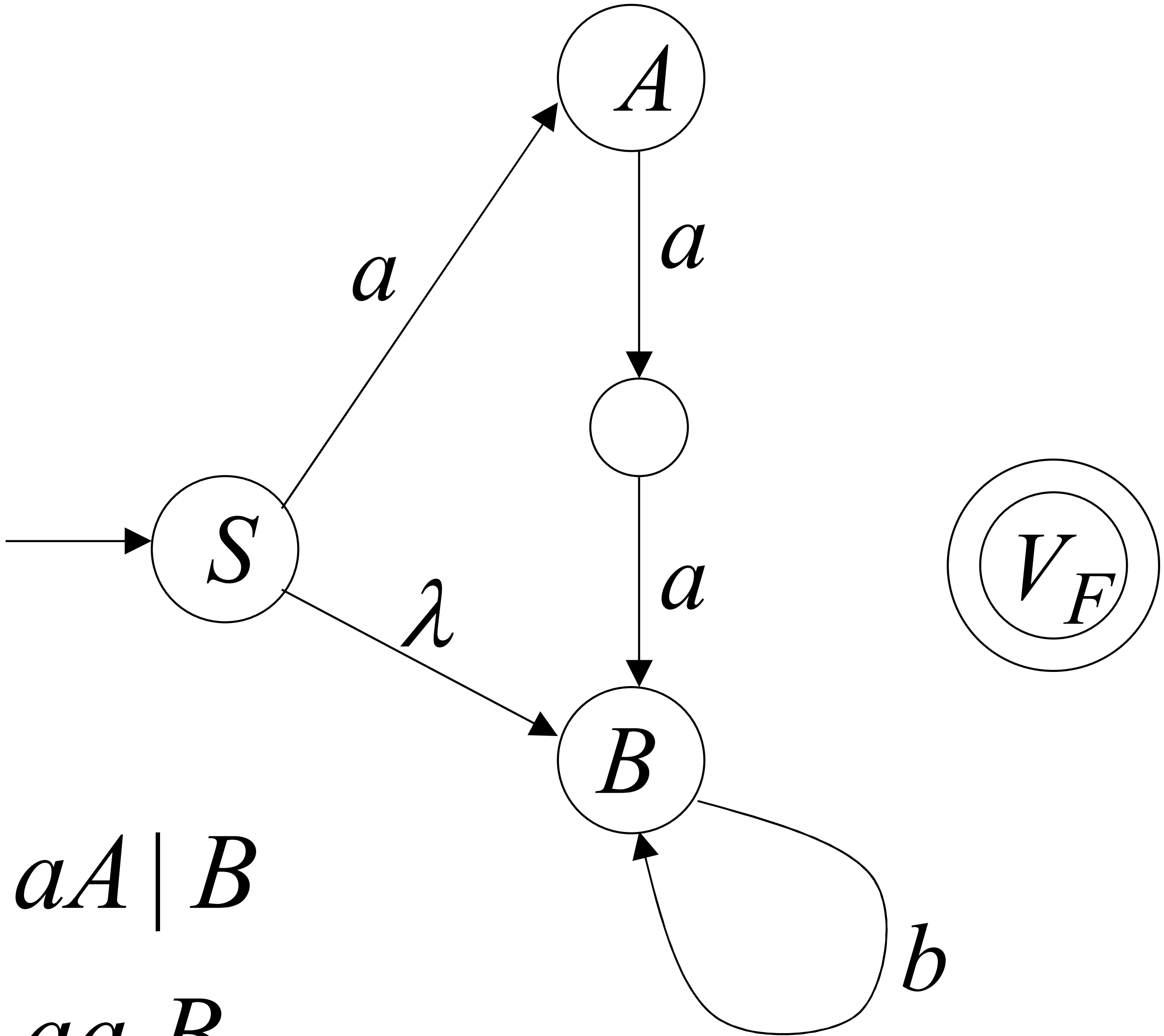


$$S \rightarrow aA \mid B$$



$$S \rightarrow aA \mid B$$

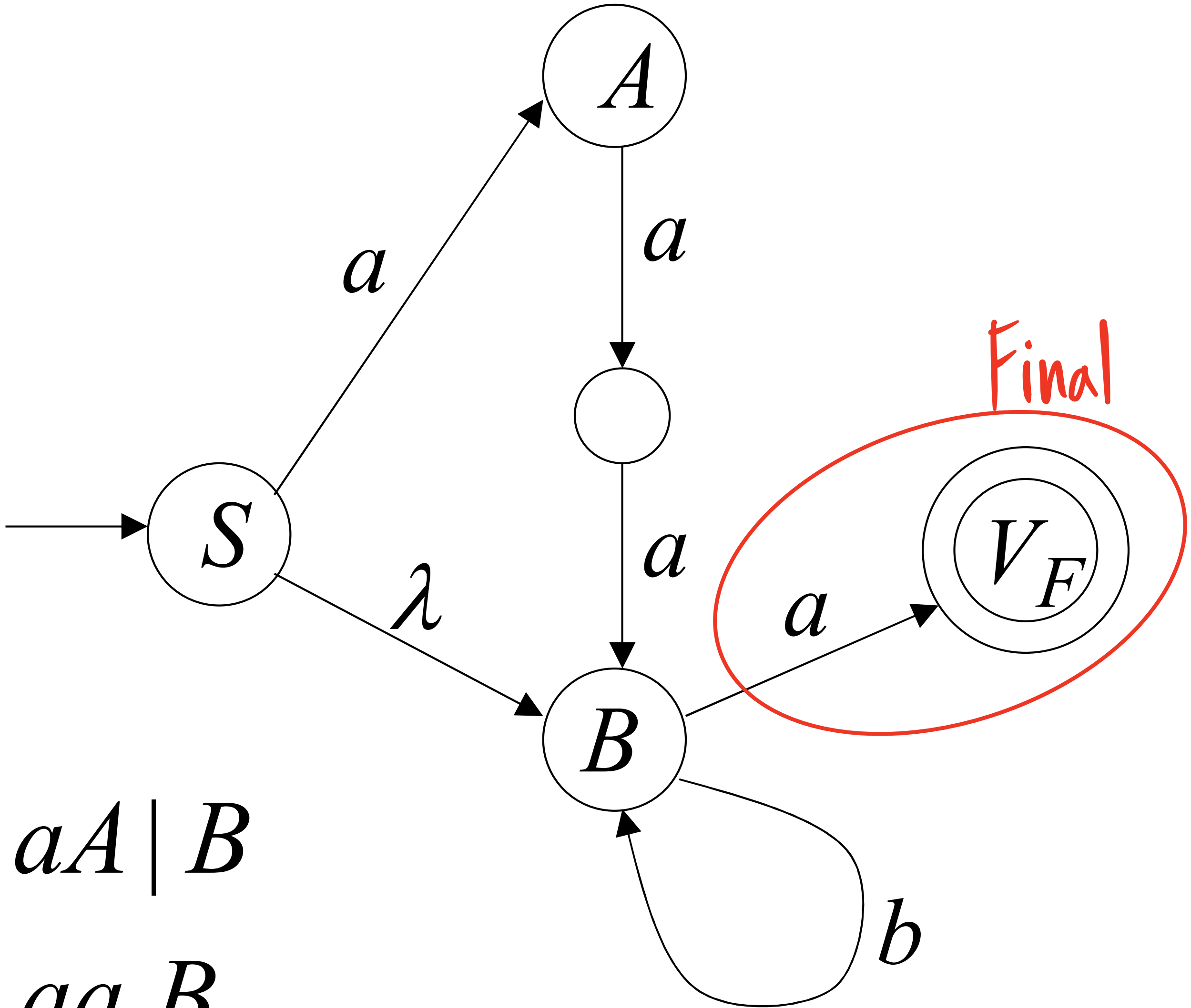
$$A \rightarrow \boxed{aa}\, B$$



$$S \to aA \mid B$$

$$A \to aa\ B$$

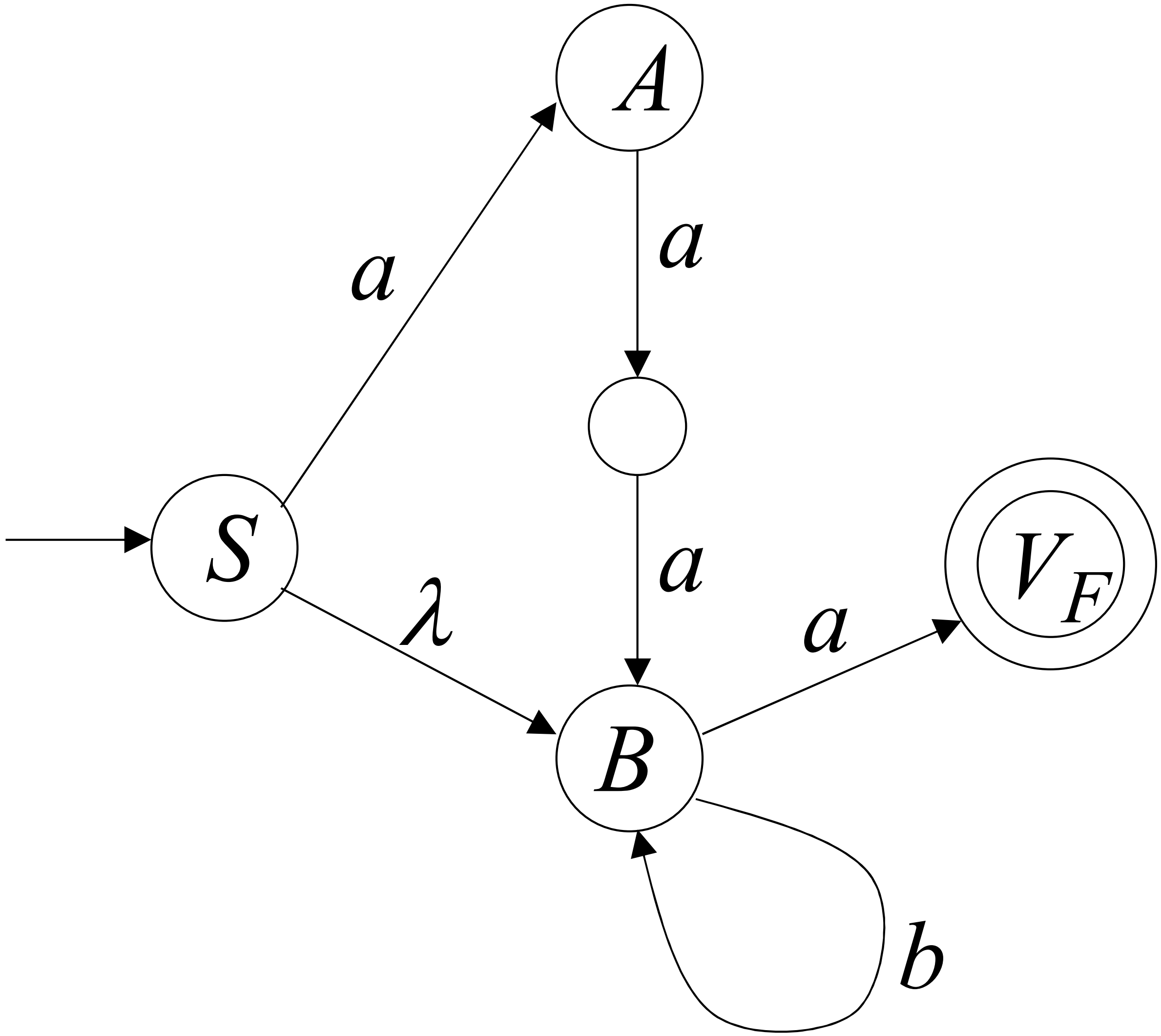
$$B \to bB$$



$$S \rightarrow aA \mid B$$

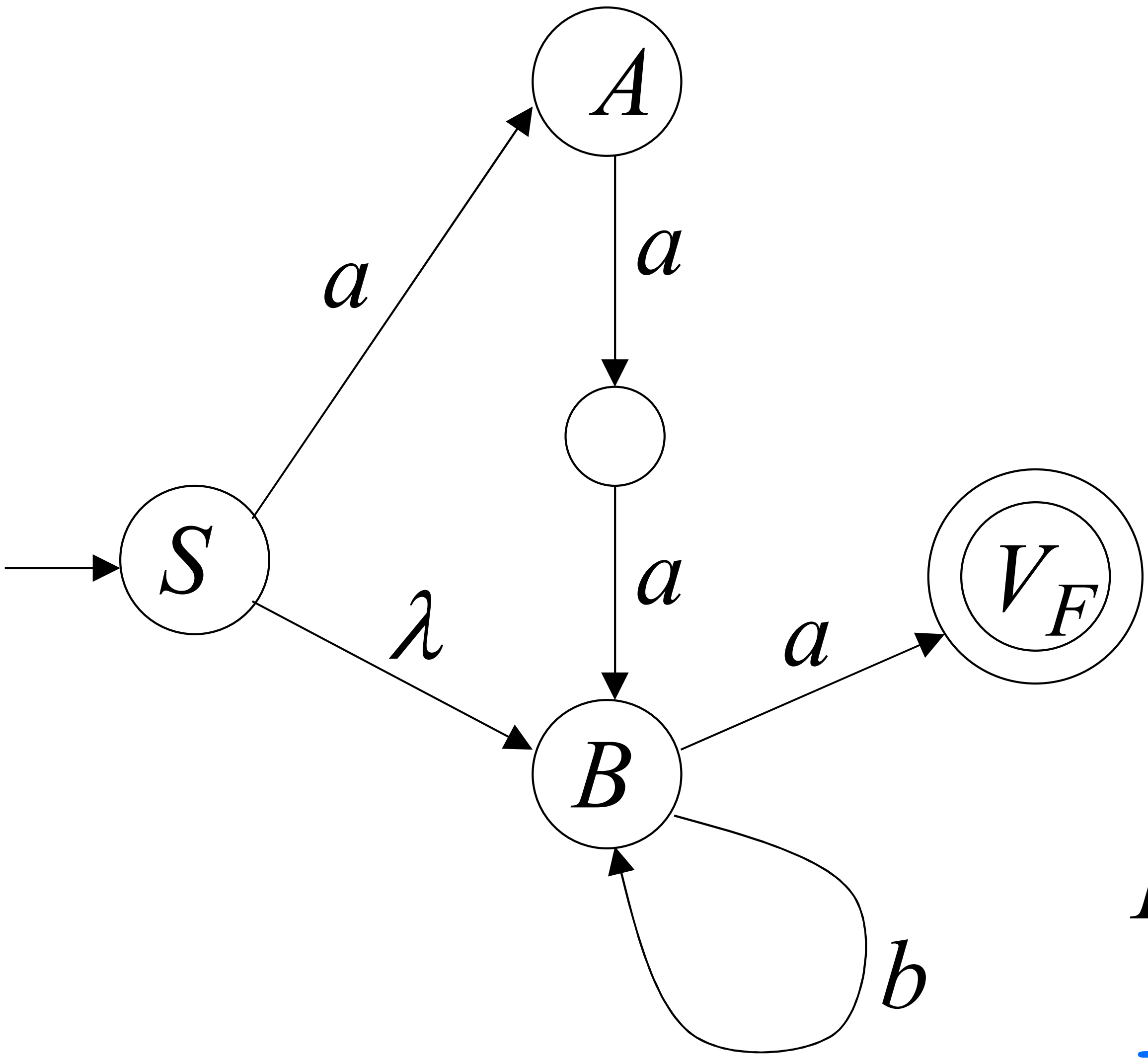
$$A \rightarrow aa\, B$$

$$B \rightarrow bB \mid a$$



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

NFA M



Grammar

G

$$S \to aA \mid B$$

$$A \to aa\,B$$

$$B \to bB \mid a$$

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

$= aaab^*a + b^*a$

In General

A right-linear grammar G

has variables: $V_0, V_1, V_2, \ldots$

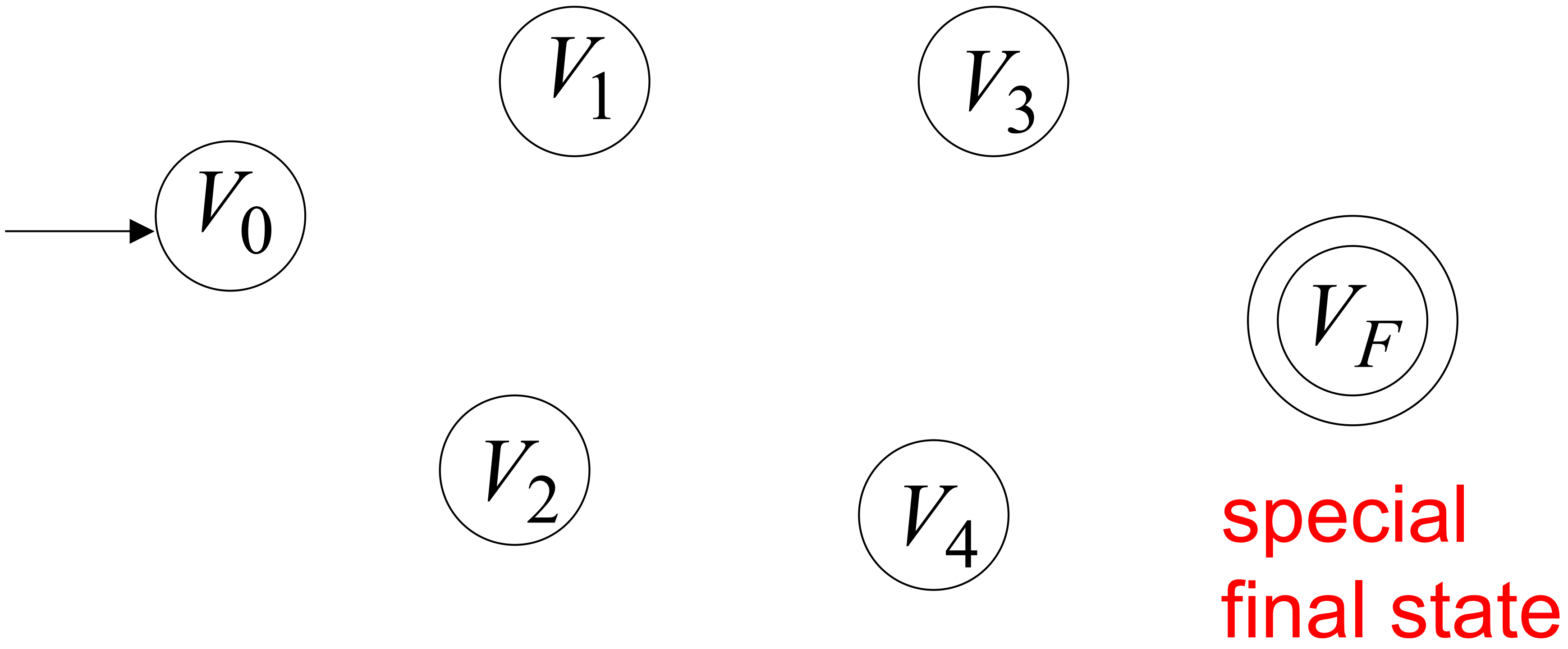
and productions: $V_i \rightarrow a_1 a_2 \cdots a_m V_j$

or

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

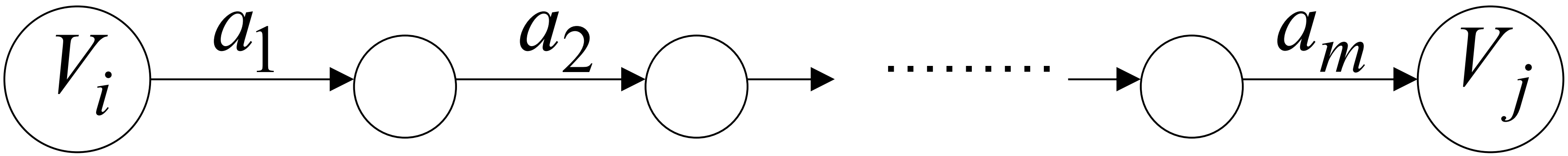
We construct the NFA M such that:

each variable V_i corresponds to a node:



For each production: $V_i \to \overbrace{a_1 a_2 \cdots a_m}^{\text{多個terminal}} V_j$

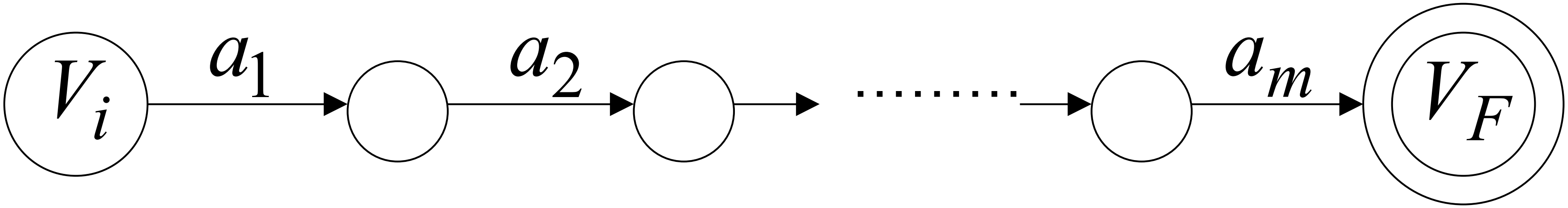
we add transitions and intermediate nodes



不具名的state

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

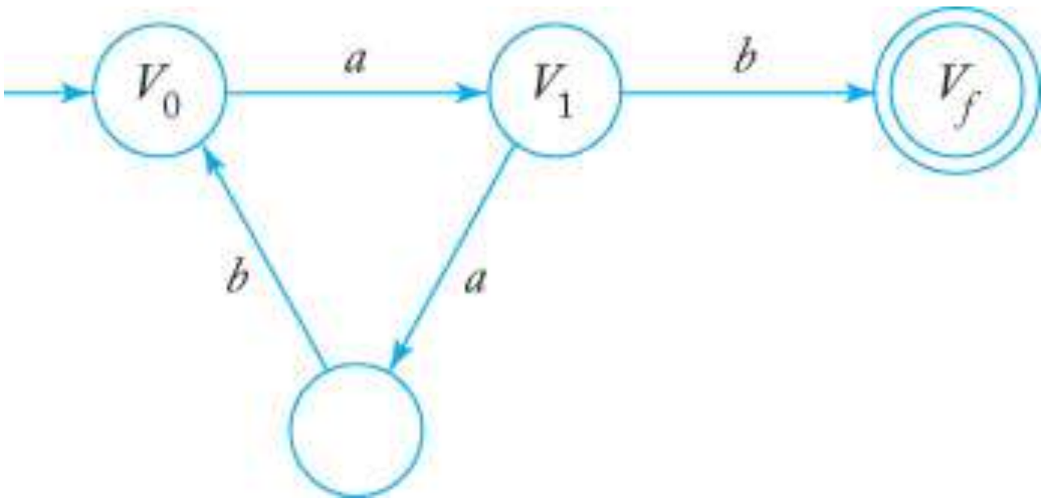
we add transitions and intermediate nodes



Example 3.15

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

$$V_0 \rightarrow aV_1,$$
$$V_1 \rightarrow abV_0 | b$$



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

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

$$A \to B a_1 a_2 \cdots a_k$$

$$A \to Bv$$

$$\Big\Downarrow \text{Reverse}$$

Right linear G'

$$A \to a_k \cdots a_2 a_1 B$$

$$A \to v^R B$$

- 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:

$L(G') \Rightarrow L(G')^R \Rightarrow L(G)$

Regular Language | Regular Language | Regular Language

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 $L = L(M)$.

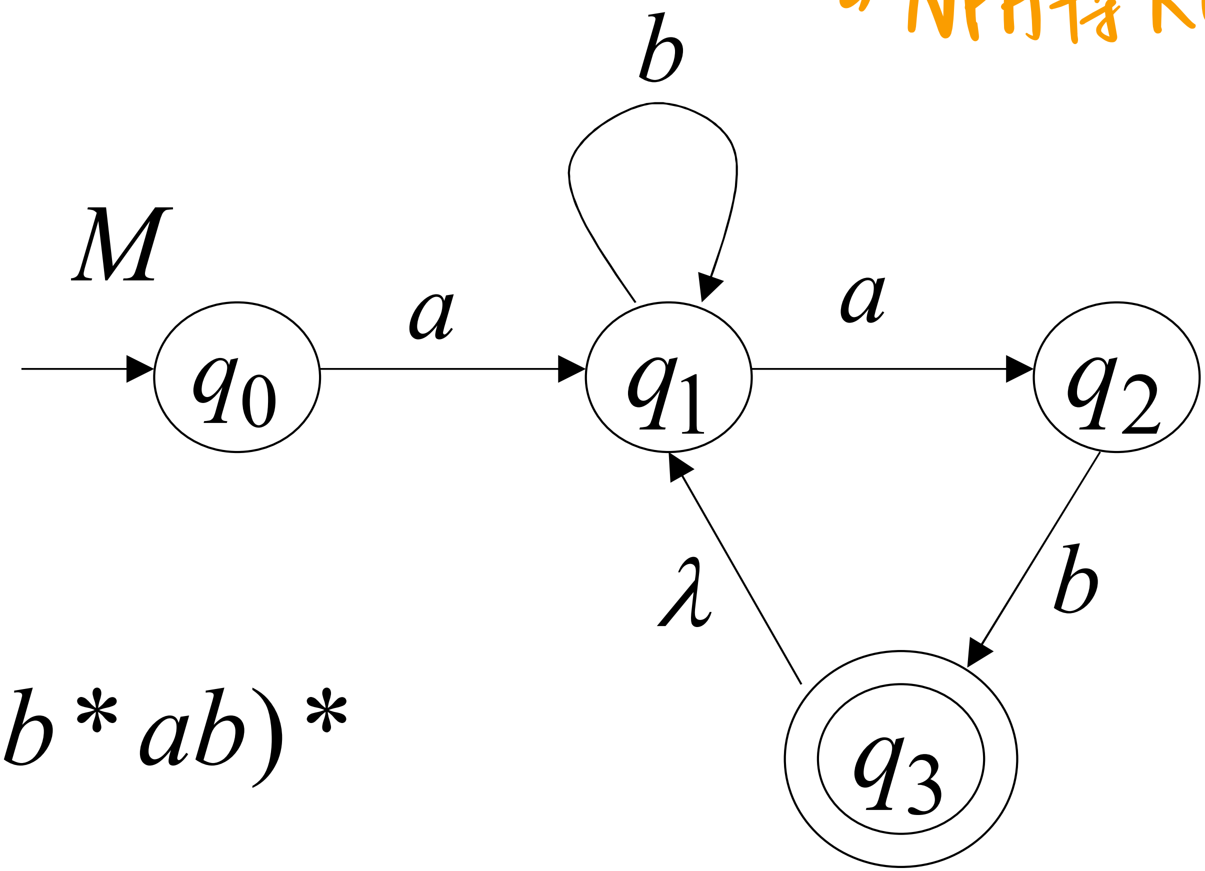
Construct from M to a regular grammar G such that $L(M) = L(G)$

Since L is regular

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

↳ NFA轉 RG

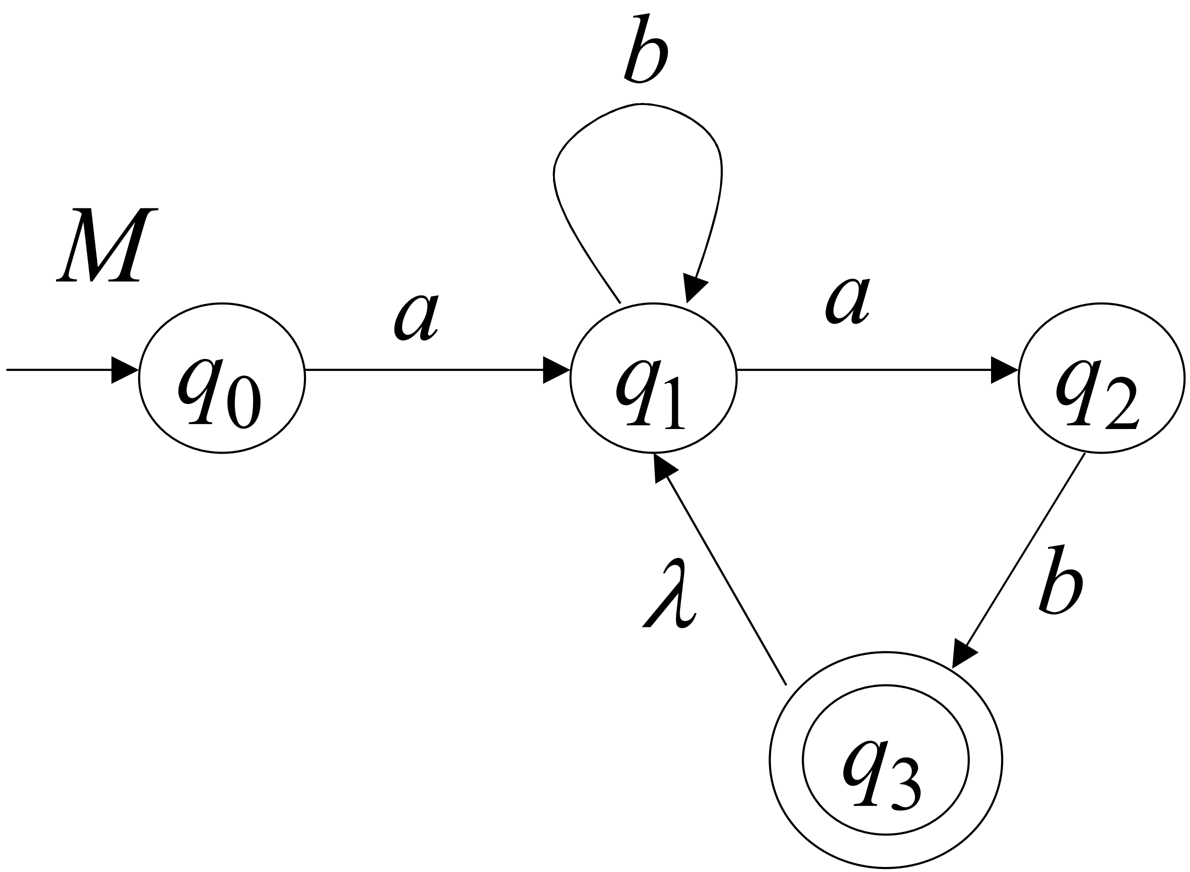
Example:



$L = ab^*ab(b^*ab)^*$

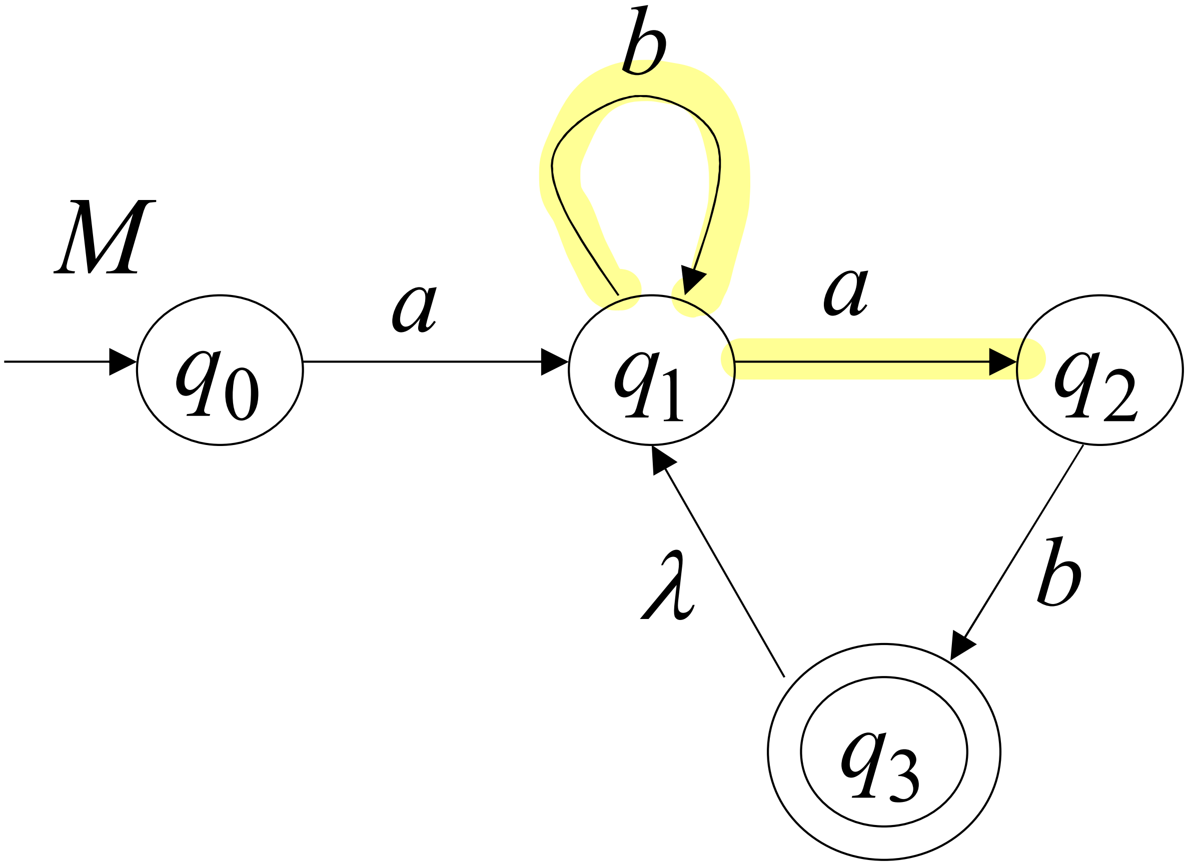
$L = L(M)$

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



$$q_0 \rightarrow aq_1$$

M



$$q_0 \rightarrow aq_1$$

$$q_1 \rightarrow bq_1$$

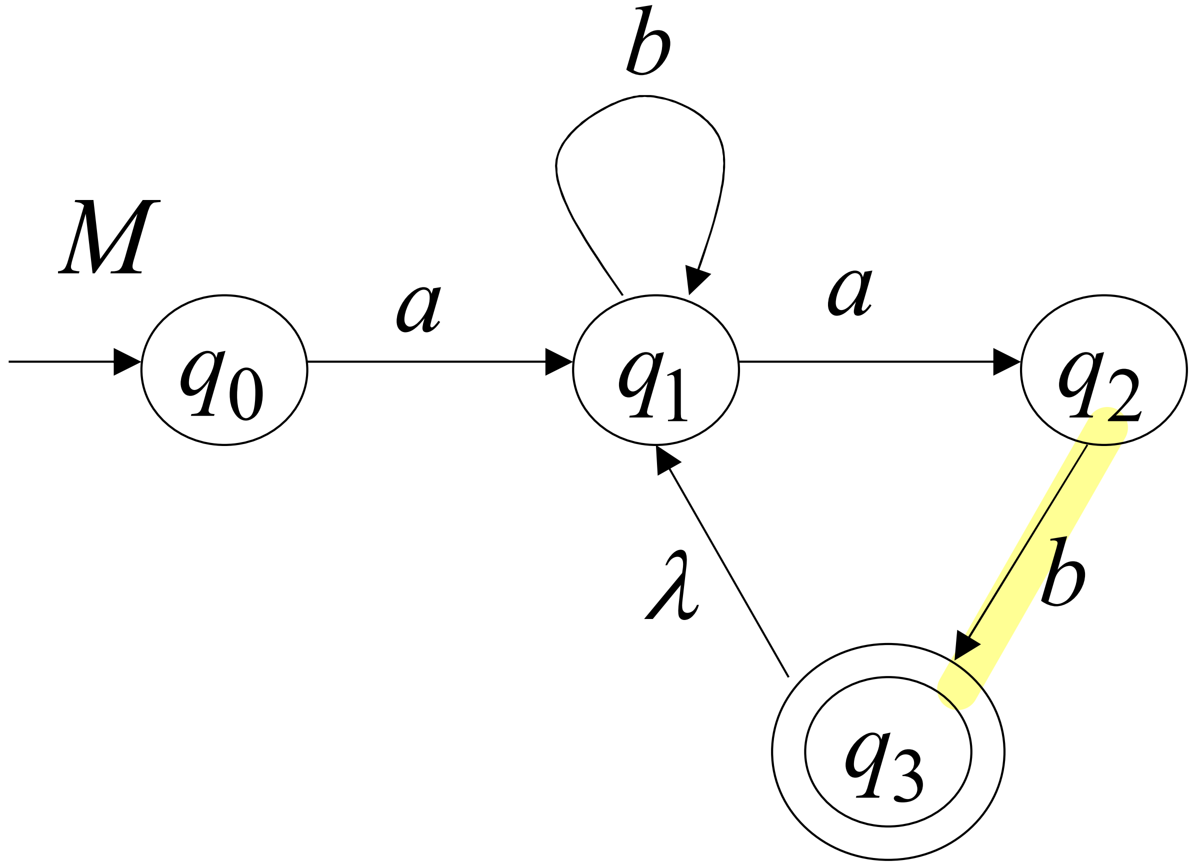
$$q_1 \rightarrow aq_2$$

$$q_0 \rightarrow aq_1$$

$$q_1 \rightarrow bq_1$$

$$q_1 \rightarrow aq_2$$

$$\boxed{q_2 \rightarrow bq_3}$$



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

G

$q_0 \to aq_1$

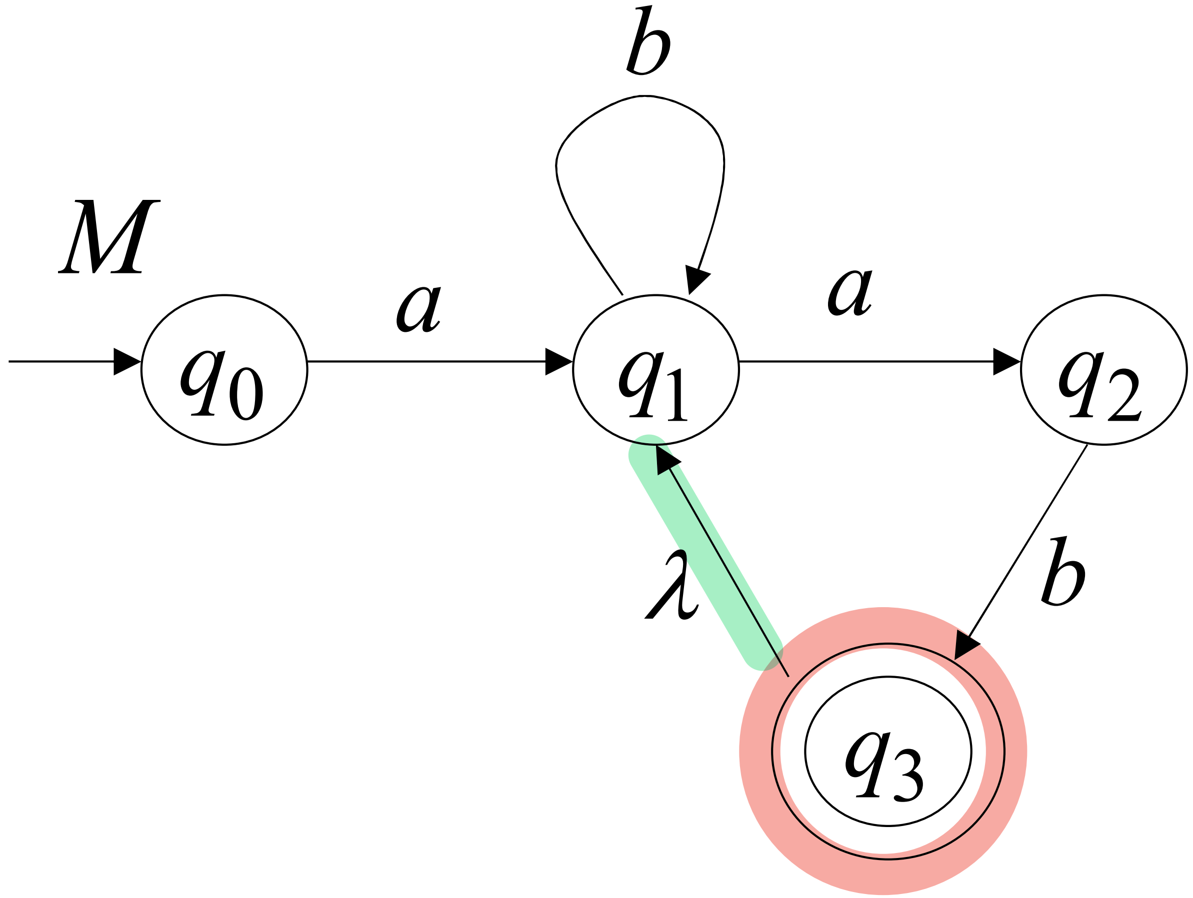
$q_1 \to bq_1$

$q_1 \to aq_2$

$q_2 \to bq_3$

$q_3 \to q_1$

$q_3 \to \lambda$ ↳ final state 需通向 λ



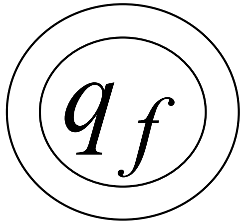
In General

For any transition: $q \xrightarrow{a} p$

Add production: $q \rightarrow ap$

variable terminal variable

For any final state:



Add production:

$$q_f \to \lambda$$

Since G is right-linear grammar

G is also a regular grammar

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

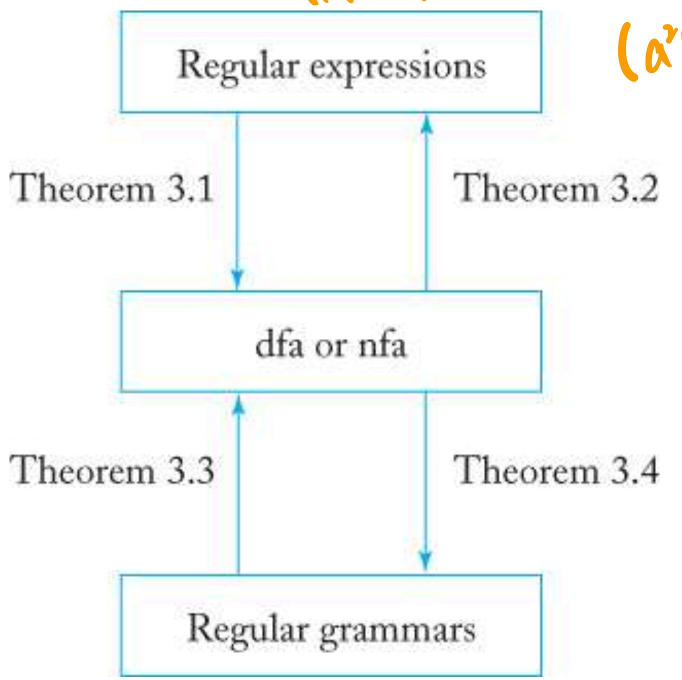
Summary

- We now have several ways of describing regular languages:

↳無記憶性，∴ $a^n b^n$ 找不到
($a^2 b^2$、$a^3 b^3$ … 可以)

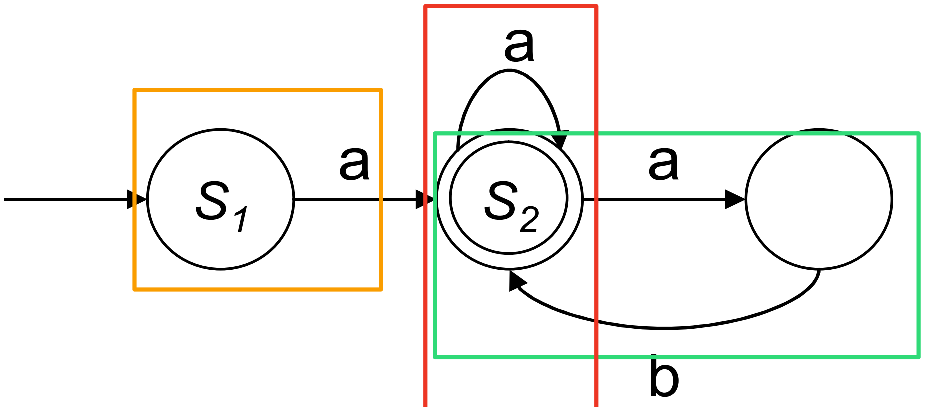
model
工具

  - DFA
  - NFA
  - RE
  - RG



Short Quiz

- Find a regular grammar that generates the language L(aa*(ab+a)*).



$$S_1 \to aS_2$$
$$S_2 \to aS_2 \mid abS_2 \mid \lambda$$

Questions?