Chapter 3

Regular Expressions, Regular Languages and

Regular Grammars

Regular Expression

- We have known that a regular language can be described by **some dfa**.
- Now we look at other ways of representing regular languages.

Definition

Let Σ be a given alphabet. Then

1. ϕ, λ and $a \in \Sigma$ are all regular expressions. These are called primitive regular expressons.
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 if and only if it can derived from the primitive regular expressions by a finite number of applications of the rules in (2).

Regular Expressions…

Regular Expression (RE):

- E is a regular expression over Σ if E is one of:
  - ε
  - a, where $a \in \Sigma$
  - If ***r*** and ***s*** are regular expressions (REs), then the following expressions also regular:
    - $r \mid s$ ➔ (r or s)
    - rs ➔ (r followed by s)
    - r^* ➔ (r repeated zero or more times)

- **Each RE has an equivalent regular language (RL).**

Regular Expressions...

- **Language Associated with Regular Expressions**

Definition

The language $L(r)$ denoted by any regular expression r is defined by the following rules.

1. ϕ is a regular expression denoting the empty set,
2. λ is a regular expression denoting $\{\lambda\}$,
3. for every $a \in \Sigma$, a is a regular expression denoting $\{a\}$.

Termination condition

If r_1 and r_2 are regular expressions, then

4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$,
5. $L(r_1 \cdot r_2) = L(r_1)L(r_2)$,
6. $L((r_1)) = L(r_1)$,
7. $L(r_1^*) = (L(r_1))^*$.

Reduce L(r) to simpler components.

Regular Expressions…

Regular Language (RL):

- L is a regular language over Σ if L is one of:
  - $\varnothing$ empty set
  - $\{\varepsilon\}$ a set that contains empty string
  - $\{a\}$ where $a \in \Sigma$

  - If $\boldsymbol{R}$ and $\boldsymbol{S}$ are regular languages (RL), then the following languages also regular:
    - $\boldsymbol{R} \cup \boldsymbol{S} = \{\boldsymbol{w} \mid \boldsymbol{w} \in \boldsymbol{R} \text{ or } \boldsymbol{w} \in \boldsymbol{S}\}$
    - $\boldsymbol{RS} = \{\boldsymbol{rs} \mid \boldsymbol{r} \in \boldsymbol{R} \text{ and } \boldsymbol{s} \in \boldsymbol{S}\}$
    - $\boldsymbol{R}^* = \boldsymbol{R}^0 \cup \boldsymbol{R}^1 \cup \boldsymbol{R}^2 \cup \boldsymbol{R}^3 \cup \ldots$

Regular Expressions…

Rules for Specifying Regular Expressions:

1. ε is a regular expression ➔ L = {ε}
2. If *a* is in Σ, *a* is a regular expression ➔ L = {*a*}, the set containing the string *a*.
3. Let *r* and *s* be regular expressions with languages L(*r*) and L(*s*). Then

precedence

a. $r \mid s$ is a RE $\Rightarrow L(r) \cup L(s)$

b. rs is a RE $\Rightarrow L(r)\, L(s)$

c. r^* is a RE $\Rightarrow (L(r))^*$

d. (r) is a RE $\Rightarrow L(r)$, extra parenthesis

Regular Expressions…

abc

- concatenation (“followed by”)

a | b | c

- alternation (“or”)

*****

- zero or more occurrences

+

- one or more occurrences

Regular Expressions…

- **<u>Examples:</u>**
  - $\{0, 1\}\{00, 11\} = \{000, 011, 100, 111\}$
  - $\{0\}^* = \{\varepsilon, 0, 00, 000, 0000, \ldots\}$
  - $\{\varepsilon\}^* = \{\varepsilon\}$
  - $\{10, 01\}^* = \{\varepsilon, 10, 1010, 101010, \ldots,$
    $01, 0101, 010101, \ldots,$
    $1001, 100101, 10010101, \ldots,$
    $0110, 011010, 01101010, \ldots\}$
- **<u>Notational shorthand:</u>**
  - $L^0 = \{\varepsilon\}$
  - $L^i = LL^{i-1}$
  - $L^+ = LL^*$

Regular Expressions…

- Let *L* be a language over {a, b}, each string in *L* contains the substring **bb**
  - **L = {a, b}*{bb}{a, b}***

- L is regular language (RL). Why?
  - {a} and {b} are RLs
  - {a, b} is RL
  - {a, b}* is RL
  - {b}{b} = {bb} is also RL
  - Then L = {a, b}*{bb}{a, b}* is RL

Regular Expressions…

- Let *L* be a language over {a, b}, each string in *L*
  - begins and ends with an **a**
  - contains at least one **b**
  - **L = $\{a\}\{a, b\}^*\{b\}\{a, b\}^*\{a\}$**

- L is regular language (RL). Why?
  - {a} and {b} are RLs
  - {a, b} is RL
  - $\{a, b\}^*$ is RL
  - Then L = $\{a\}\{a, b\}^*\{b\}\{a, b\}^*\{a\}$ is RL

Regular Expressions…

- L = $\{a, b\}^*\{bb\}\{a, b\}^*$
  - RE = $(a|b)^*bb(a|b)^*$
- L = $\{a\}\{a, b\}^*\{b\}\{a, b\}^*\{a\}$
  - RE = $a(a|b)^*b(a|b)^*a$
- This RE = **(a)|((b)*(c))** is equivalent to **a|b*c**
- We say REs r and s are equivalent (r=s), iff r and s represent **the same language.**
  - Example: r = ***a***|***b***, s = ***b***|***a*** ➔ r = s Why?
  - **Since L(r) = L(s) = {a, b}**

Regular Expressions…

- Let $\sum$= {a, b}

  - RE ***a|b*** ➔ L = {a, b}
  - RE (***a|b***)(***a|b***) ➔ L = {aa, ab, ba, bb}
  - RE ***aa|ab|ba|bb*** same as above
  - RE ***a****** ➔ L = {ε, a , aa, aaa, …}
  - RE (***a|b***)* ➔ L = set of all strings of a's and b's including ε
  - RE (***a****b****)* same as above
  - RE ***a|a*b*** ➔ L = {a,b,ab,aab,aaab, …}

Regular Expressions…

Algebraic Properties of regular Expressions
$r \mid s = s \mid r$
$r \mid (s \mid t) = (r \mid s) \mid t$
$(r\ s)\ t = r\ (s\ t)$
$r\ (s \mid t) = r\ s \mid r\ t$ $(s \mid t)\ r = s\ r \mid t\ r$
$\varepsilon\ r = r\ \varepsilon = r$
$r^* = (r^*)^* = (r \mid \varepsilon)^+ = r^+ \mid \varepsilon$
$r^{**} = r^*$
$r^+ = r\ r^*$

Regular Expression Examples

- All strings of 1s and 0s.

  $(0 \mid 1)^*$

- All strings of 1s and 0s beginning with a 1.

  $1\,(0 \mid 1)^*$

Regular Expression Examples…

- All strings containing **two** or more **0**s.
  $(1|0)^*\mathbf{0}(1|0)^*\mathbf{0}(1|0)^*$

- All strings containing an **even** number of **0**s.
  $(1^*\mathbf{0}1^*\mathbf{0}1^*)^* \mid 1^*$

- All strings of alternating **0**s and **1**s.
  $(\varepsilon \mid 1)(0\,1)^*(\varepsilon \mid 0)$

- Strings over the alphabet {0, 1} with **no consecutive 0's**
  - $(1 \mid 01)^*(0 \mid \varepsilon)$
  - $1^*(01^+)^*(0 \mid \varepsilon)$
  - $1^*(011^*)^*(0 \mid \varepsilon)$

Regular Expression Examples…

- Describe the following in **English**:
  - (0|1)*
    - all strings over {0, 1}
  - b*ab*ab*ab* - describe the language?

Regular Expression Examples…

- **More Examples of RE:**
  - 01^*
    - {0, 01, 011, 0111, …..}
  - $(01^*)(01)$
    - {001, 0101, 01101, 011101, …..}
  - $(0 \mid 1)^*$ = {0, 1, 00, 01, 10, 11, …..}
    - i.e., all strings of 0 and 1
  - $(0 \mid 1)^*\ 00\ (0 \mid 1)^*$ = {00, 1001, …..}
    - i.e., all 0 and 1 strings containing a “00”

Regular Expression Examples…

- **More Examples of RE:**
  - $(1 \mid 10)^*$
    - all strings starting with “1” and containing no “00”
  - $(0 \mid 1)^*011$
    - all strings ending with “011”
  - 0^*1^*
    - all strings with no “0” after “1”
  - 00^*11^*
    - all strings with at least one “0” and one “1”, and no “0” after “1”

Exercise

- What languages do the following RE represent?
  - $((0 \mid 1)(0 \mid 1))^* \mid ((0 \mid 1)(0 \mid 1)(0 \mid 1))^*$

Connection Between RE & RL

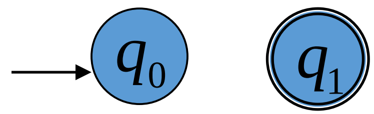
- A language *L* is called **regular** if and only if there exists some **DFA** M such that L = L(M).

- Since a DFA ***has an equivalent*** NFA, then
  - A language *L* is called **regular** if and only if there exists some NFA N such that L = L(N).

- If we have a RE *r*, we can construct an NFA that accept L(*r*).

Connection Between RE & RL…

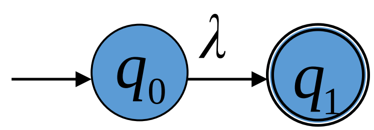
- For $\varnothing$ in the regular expression, construct NFA



$L = \{\ \} = \varnothing$

(1) nfa accepts ϕ.

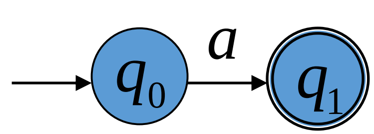
- For ε in the regular expression, construct NFA



$L = \{\varepsilon\}$

(2) nfa accepts $\{\lambda\}$.

- For $a \in \Sigma$ in the regular expression, construct NFA

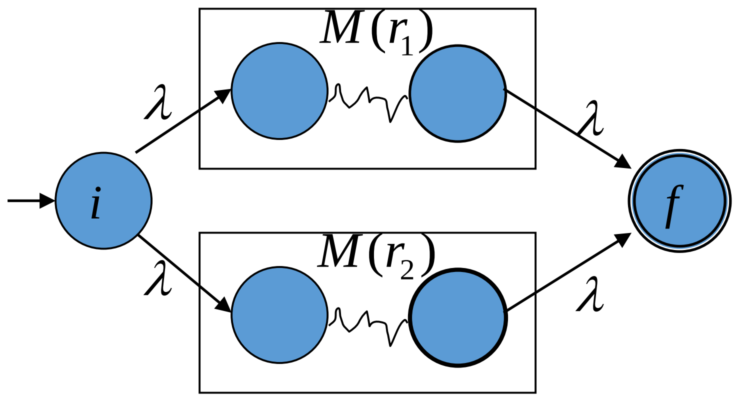


$L = \{a\}$

(3) nfa accepts a.

Connection Between RE & RL…

- If r1 and r2 are regular expressions, M_{r1} and M_{r2} are their NFAs. r1|r2   has NFA:



L = {L(M_{r1}) $\cup$ L(M_{r2})}

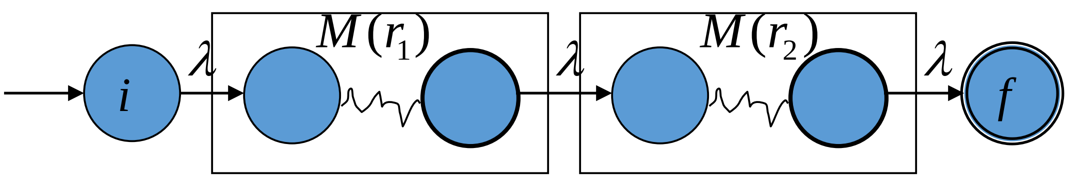
(4) nfa accepts $L(r_1 + \mathrm{r}_2)$.

where *i* and *f* are new start / final states, and ε-moves are introduced from *i* to the old start states of M_{r1} and M_{r2} as well as from all of their final states to *f*.

Connection Between RE & RL…

- If r1 and r2 are regular expressions, M_{r1}, M_{r2} their NFAs.
- r1r2 (concatenation) has NFA:

$L = \{L(M_{r1})L(M_{r2})\}$



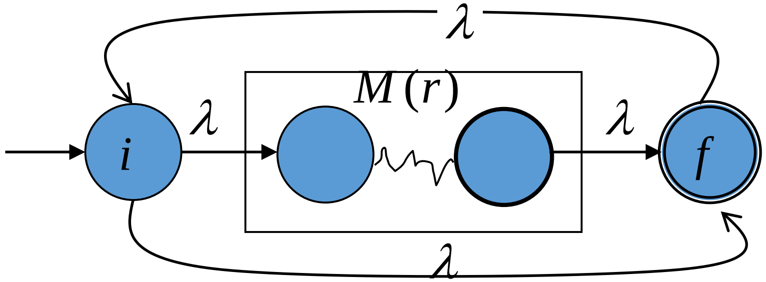
(5) nfa accepts $L(r_1 r_2)$.

where *i* is the start state of M_{r1} (or new under the alternative) and *f* is the final state of M_{r2} (or new). Overlap maps final states of M_{r1} to start state of M_{r2}

Connection Between RE & RL…

- If r is a regular expressions and M_r its NFA,
- r* (Kleene star) has NFA:

$L = \{L(M_r)^*\}$



(6) nfa accepts $L(r^*)$.

where : *i* is new start state and *f* is new final state

ε-move *i* to *f* (to accept null string)

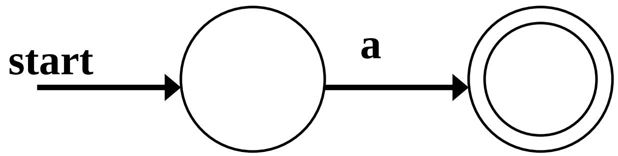
ε-moves *i* to old start, old final(s) to *f*

ε-move *f* to *i* (WHY?)

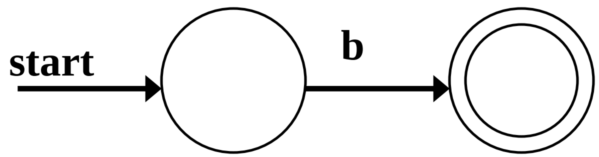
Connection Between RE & RL: Example

- Build an **NFA-ε** that accepts **(a|b)*ba**

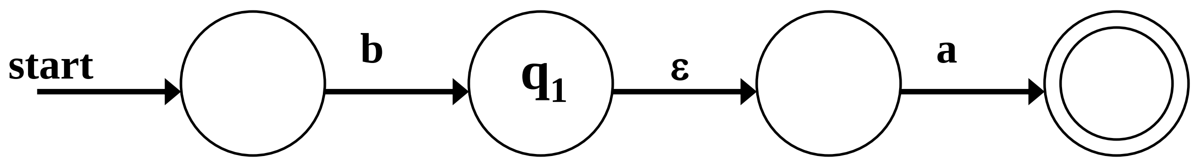
a



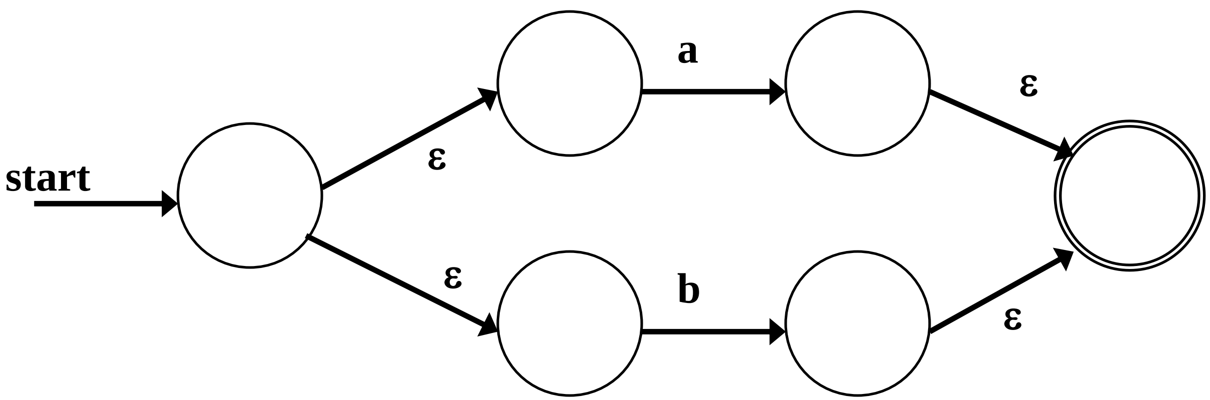
b



ba



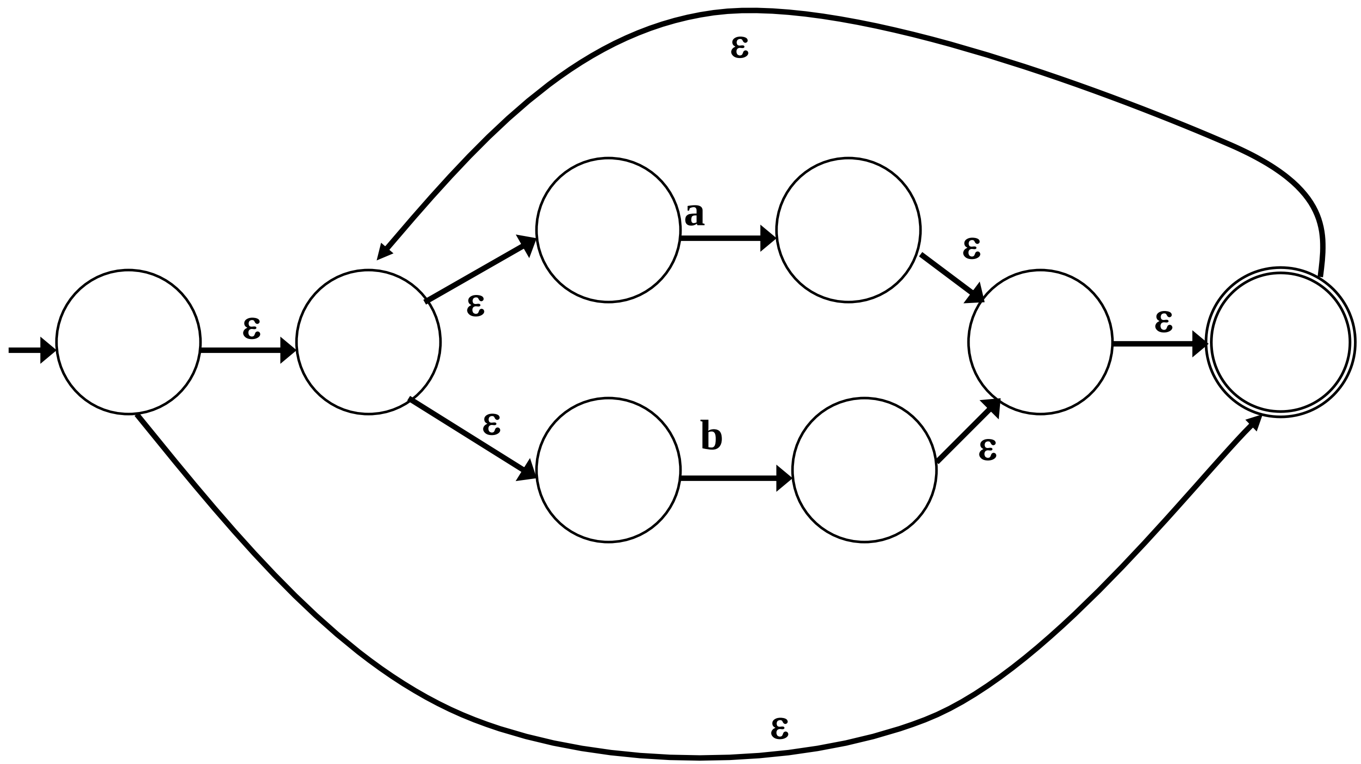
a|b



Example…

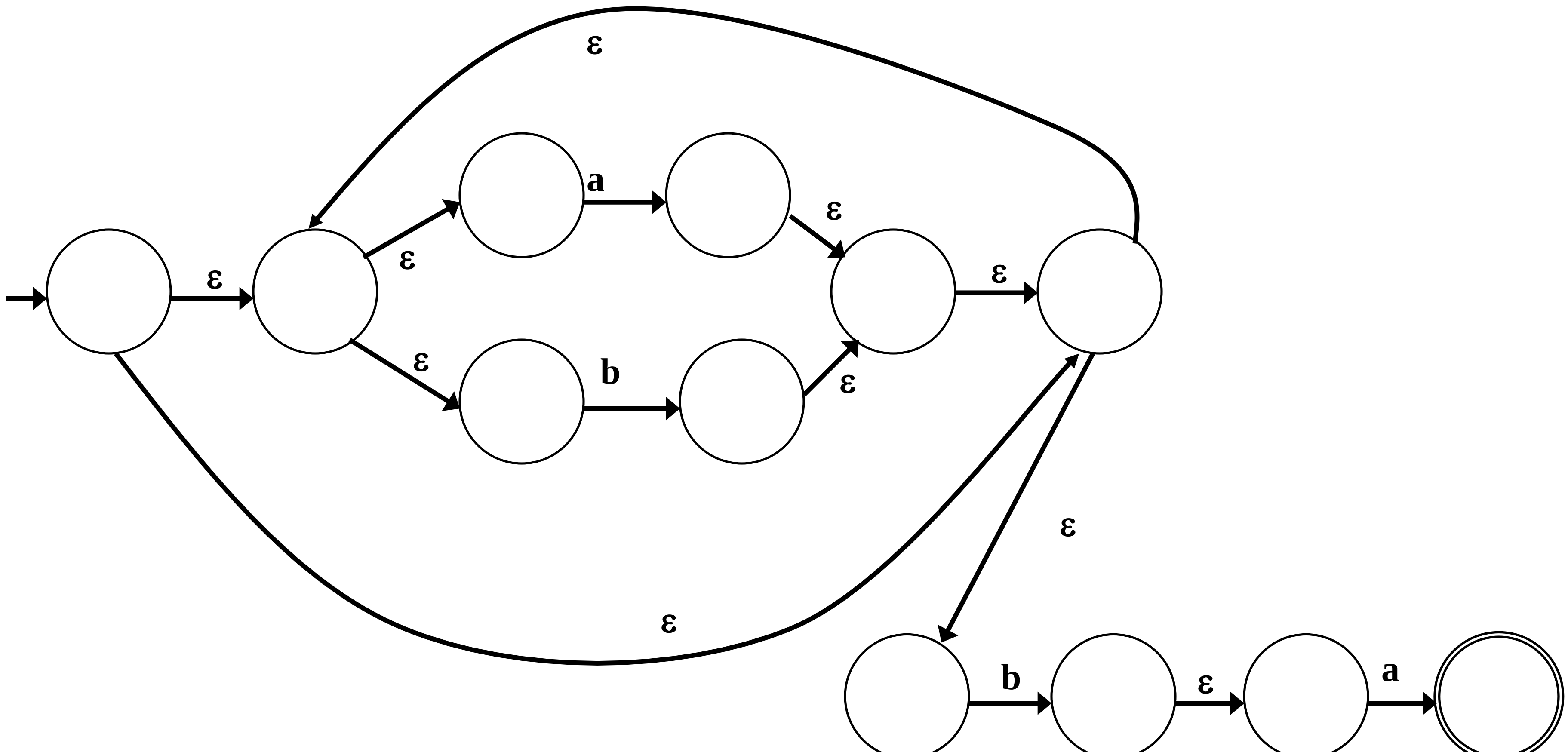
- Build an **NFA-ε** that accepts **(a|b)*ba**

(a|b)*



Example…

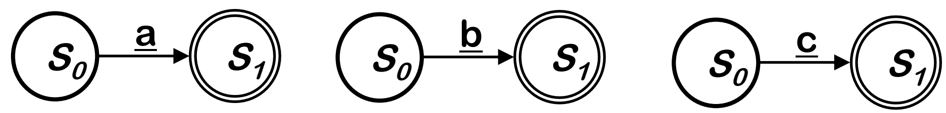
- Build an **NFA-ε** that accepts **(a|b)*ba**



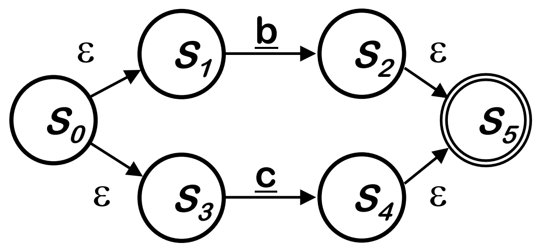
Example…

- **Let's try a (b | c)***

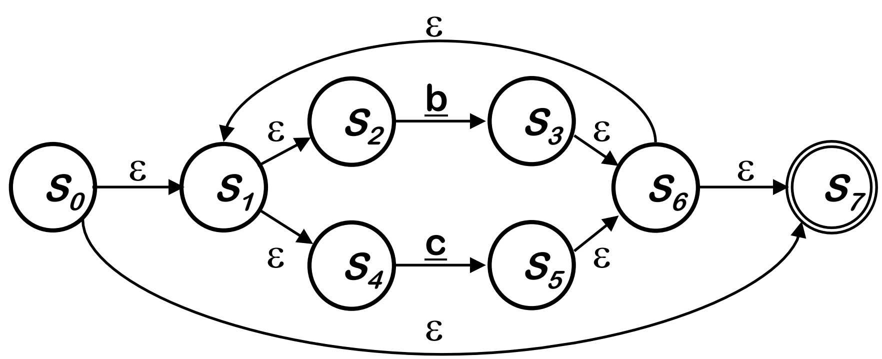
1. a, b, & c



2. b | c

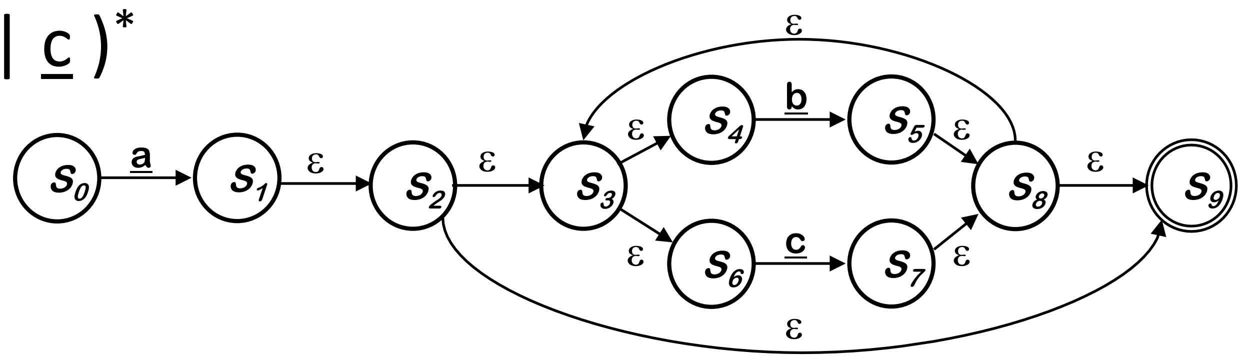


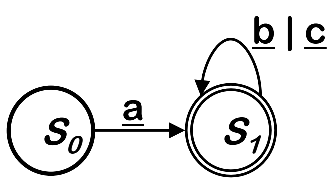
3. (b | c)*



Example…

4. $\underline{a}\ (\ \underline{b}\ |\ \underline{c}\ )^*$





Example…

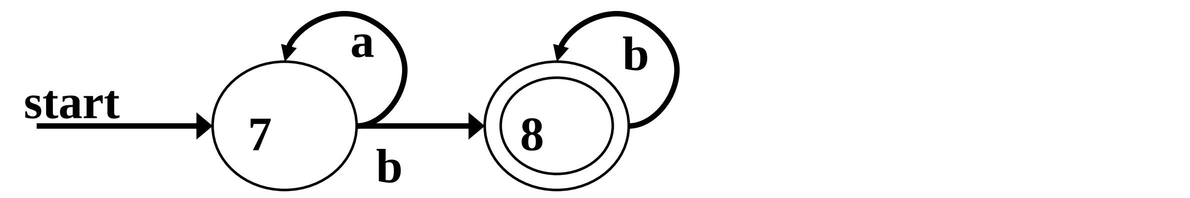
Let : **a**

abb

a^*b^+

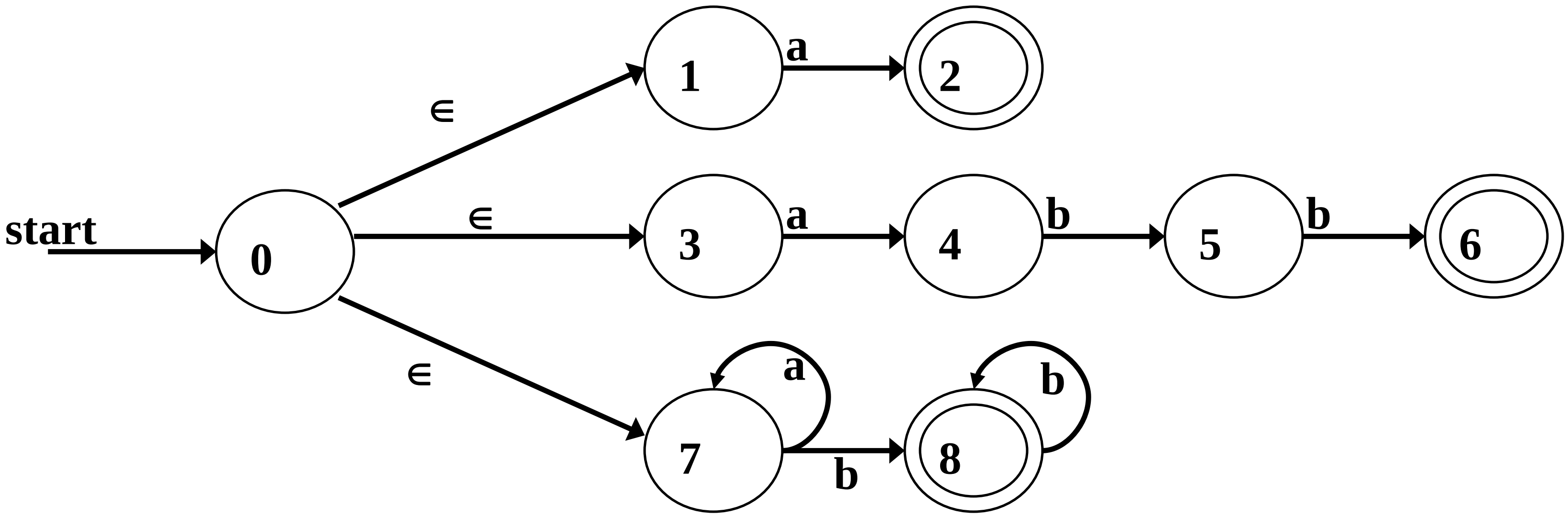
3 patterns

NFA's :



Example…

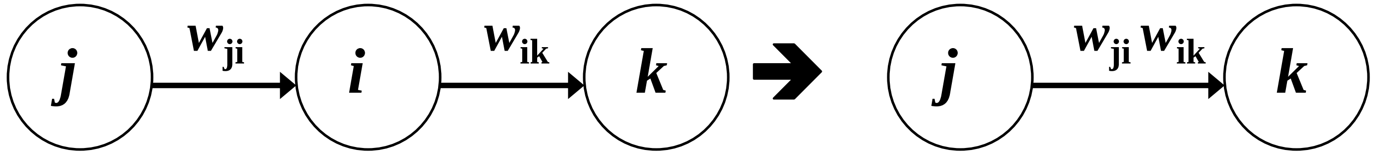
NFA for : a | abb | a^*b^+

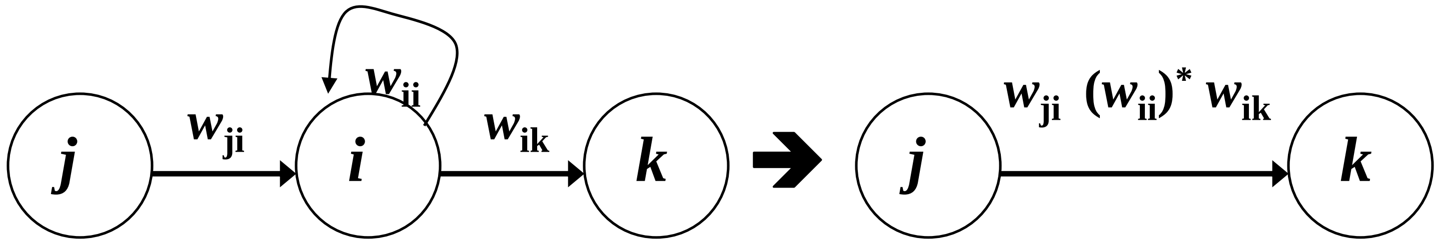


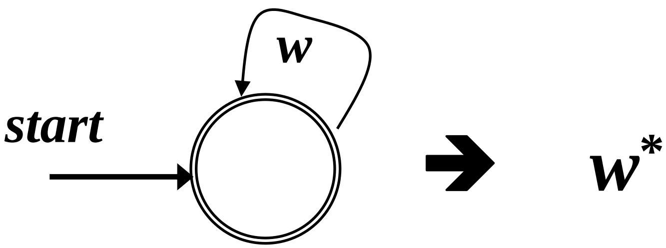
NFA to RE

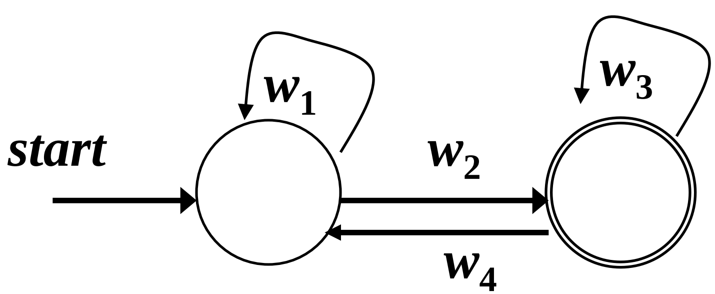
- If **L** is accepted by some **NFA-ε**, then **L** is represented by some ***regular expression.***
- An ***expression graph*** is like a state diagram but it can have *regular expressions* as labels on arcs.
- An NFA-ε is an ***expression graph***.
- An expression graph can be ***reduced to one*** with just ***two states***.
- If we reduce an NFA-ε in this way, the arc label then corresponds to the regular expression representing it.

NFA to RE…





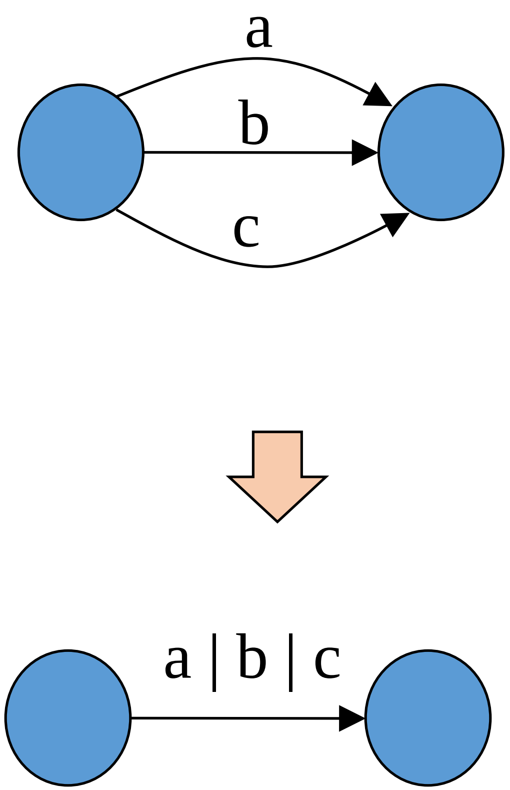




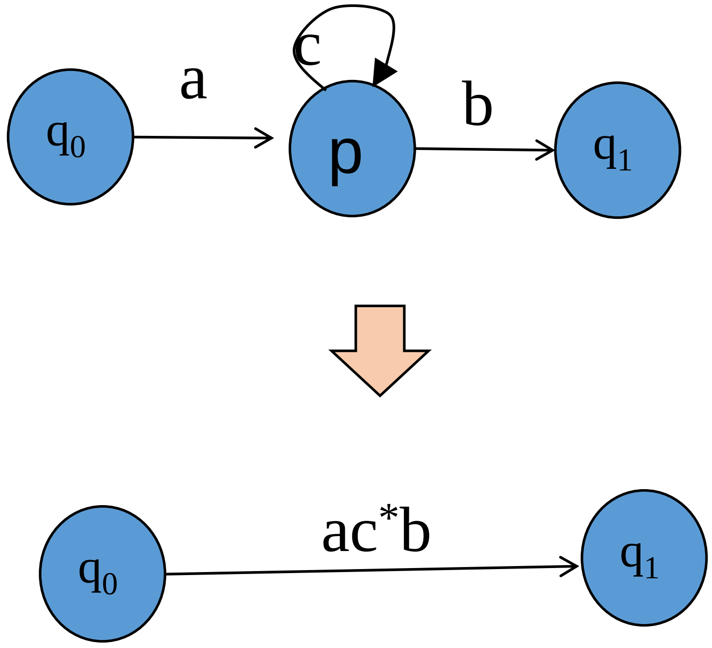
➔ $(w_1)^* w_2 (w_3 \mid w_4 (w_1)^* w_2)^*$

NFA to RE…

- Merge Edges :



Replace state by Edges

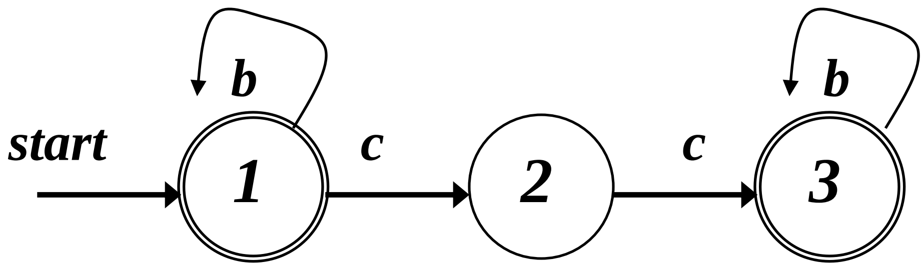


NFA to RE…

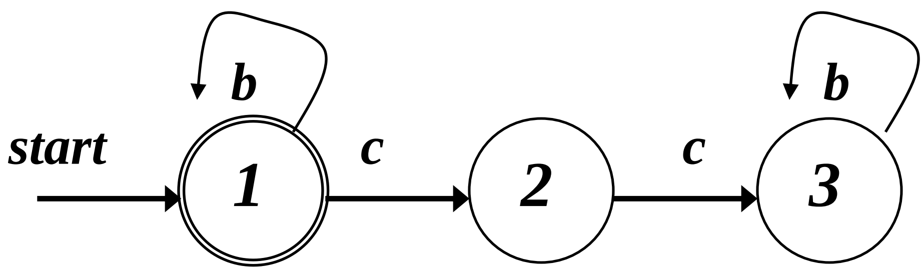
- Let **G** be the state diagram of a finite automata.
- Let ***m*** be the *number of final states* of **G.**
- Make ***m*** copies of **G**, each of which has ***one final state***.
- Call these graphs $\mathbf{G_1}$, $\mathbf{G_2}$, …, $\mathbf{G_m}$
- For each $\mathbf{G_t}$
  - Repeat
    - Do the steps in the previous slide
  - Until the only states in $\mathbf{G_t}$ are the start state and the single final state.
  - Determine the RE of $\mathbf{G_t}$
- The RE of **G** is obtained by joining RE's of each $\mathbf{G_t}$ by $\cup$ or $|$.

NFA to RE…

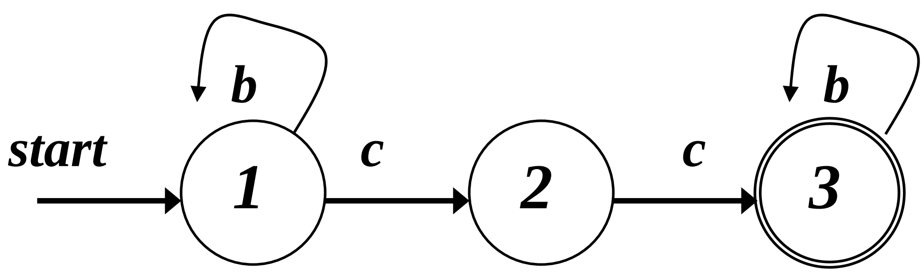
G:



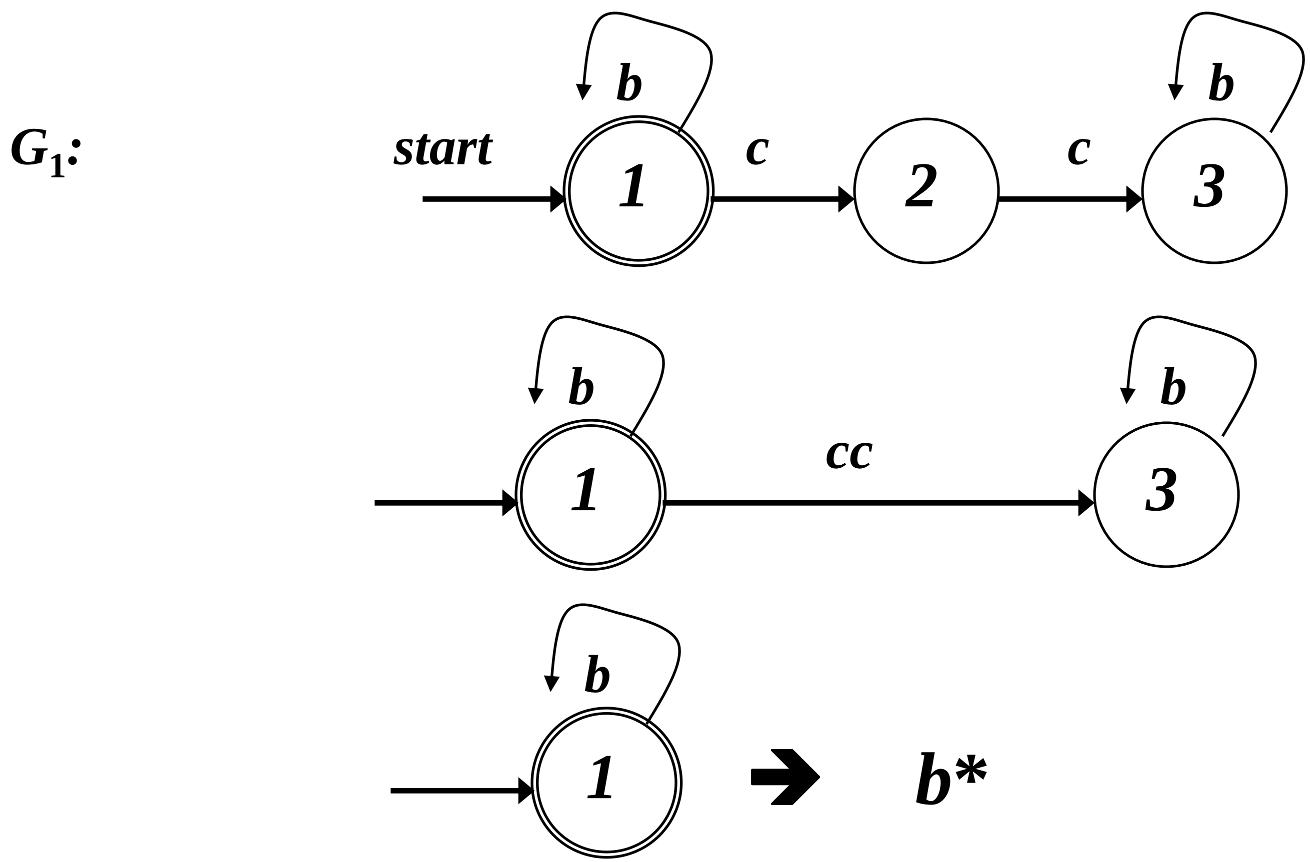
G_1:



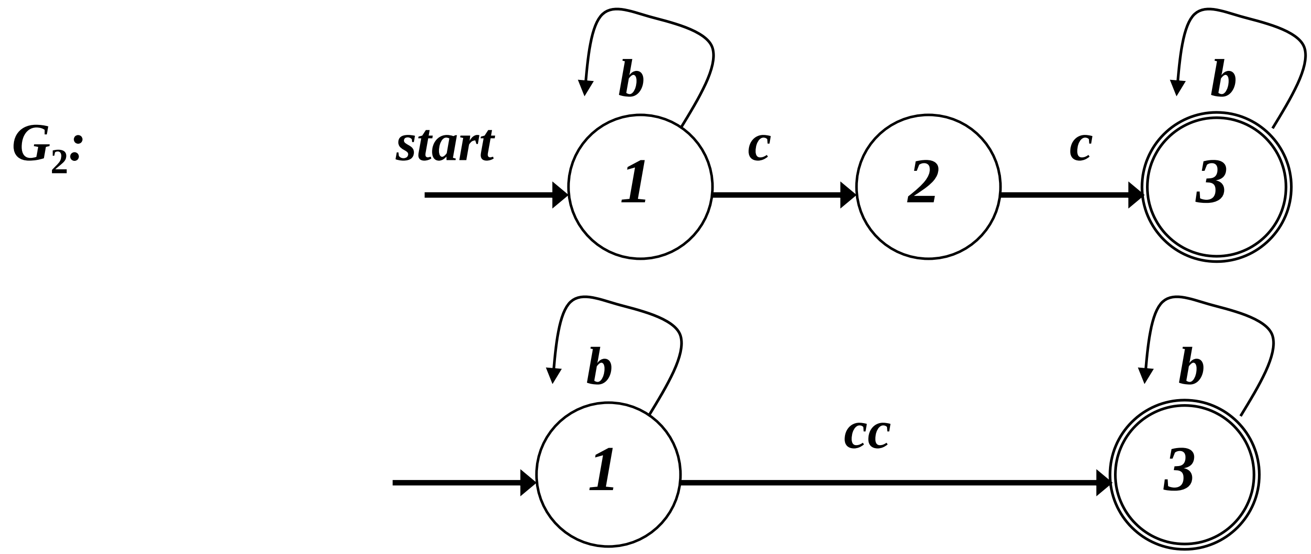
G_2:



NFA to RE…



NFA to RE…

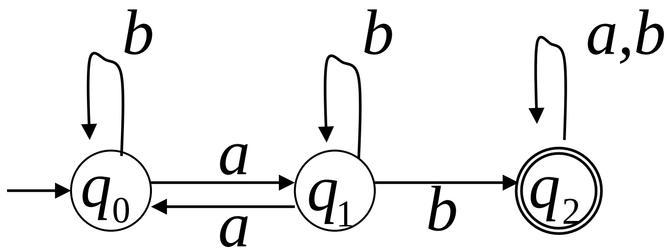


➔ b^*ccb^*

RE for **G** ➔ $b^* \mid b^*ccb^*$

NFA to RE: Exercise

- Given the following NFA, find the regular expression it accepts.



Applications of Regular Expressions

- In many programming languages the set of integer constants is defined by the regular expression ***sdd*,***
  where ***s*** stands for the sign, with possible values from *{+, -, λ}*, and ***d*** stands for the digits 0 to 9.

- An application of pattern matching occurs in ***text editing***.
- All text editors allow files to be scanned for the occurrence of a given string; most editors extend this to permit searching for patterns.
  - **For example**, the ***vi*** editor in the ***UNIX operating system*** recognizes the command /aba∗c/ as an instruction to search the file for the first occurrence of the string **ab**, followed by an arbitrary number of **a's**, followed by **ac**.

Regular Grammars

- The ***third way*** of describing regular languages is by means of certain simple grammars. (*what are the other two ways?)*

Right- and Left-Linear Grammars

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

$$A \rightarrow xB,$$

$$A \rightarrow x,$$

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

$$A \rightarrow Bx,$$

$$A \rightarrow x.$$

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

Regular Grammars…

Example

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

$S \to abS|a$ is right linear.

The grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 given as

$S \to S_1ab,\ S_1 \to S_1ab|S_2,\ S_2 \to a$ is left linear.

Both G_1 and G_2 are regular grammars.

$$\longrightarrow \quad \begin{aligned} L(G_1) &= L((ab)^* a) \\ L(G_2) &= L(aab(ab)^*) \end{aligned}$$

Example

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

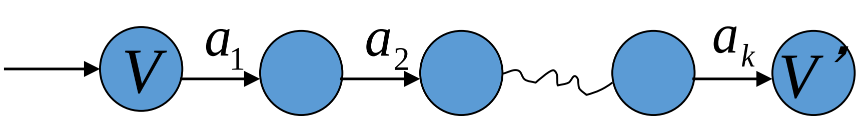
$S \to A,\ A \to aB|\lambda,\ B \to Ab$ is not regular.

Right-linear Grammars Generate Regular Languages

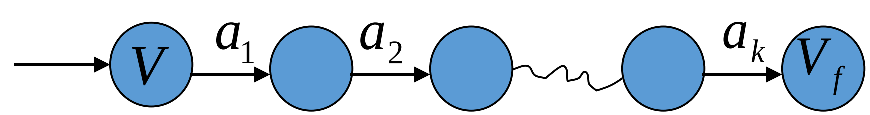
- Right-linear grammar $\longleftrightarrow$ Regular language
- Left-linear grammar $\longleftrightarrow$ Regular language

Theorem 3.3 Let *G=(V,T,S,P)* be a right-linear grammar. Then *L(G)* is a regular language.

Policy of Proof: Construct an *nfa* which accepts *L(G)*.



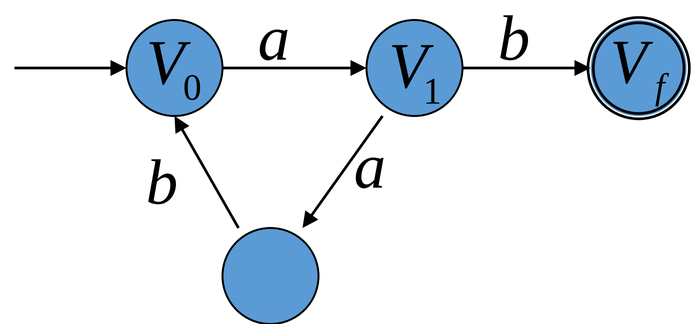
(1) nfa for $V \to a_1 a_2 \cdots a_k V'$



(2) nfa for $V \to a_1 a_2 \cdots a_k$

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

$$V_0 \to aV_1,$$
$$V_1 \to abV_0 | b.$$



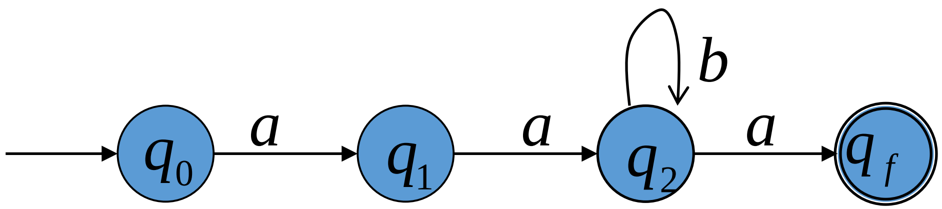
<u>Theorem 3.4</u> If L is a regular language on an alphabet Σ, then there exists a right-linear grammar *G=(V, Σ,S,P)* such that *L=L(G)*.

<u>Policy of Proof:</u> Let M be the dfa that accepts L. Construct the right-linear grammar G from M such that G generates L.

Example: Construct a right-linear grammar for $L(aab^*a)$.

<u>Solution</u>

A *dfa* which accepts $L(aab^*a)$.



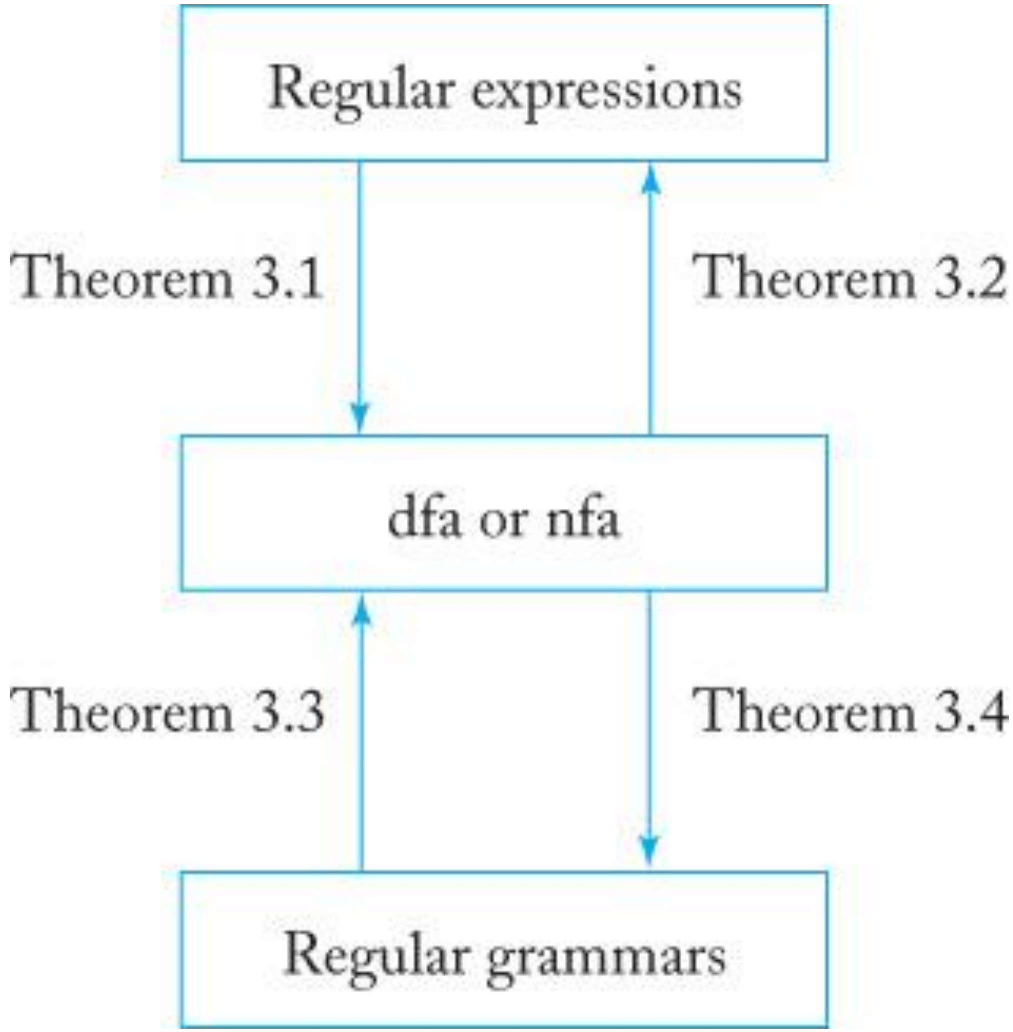
A *grammar* which generates $L(aab^*a)$:

$$G = (V, \Sigma, S, P)$$

where $V = \{q_0, q_1, q_2, q_f\}$, $\Sigma = \{a, b\}$, $S = q_0$ and P is as follows:

$$q_0 \to aq_1,\ q_1 \to aq_2,\ q_2 \to bq_2,\ q_2 \to aq_f,\ q_f \to \lambda .$$

Regular Grammars…



Reading (Self Study)

- Conversion of Right Linear Grammars into Left Linear Grammars and vice versa.