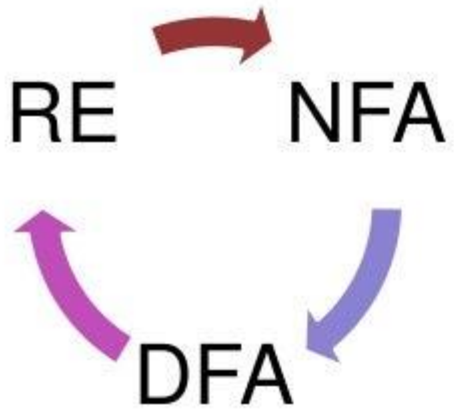
Regular Expressions

CSE 2233

Regular Languages

Regular Language:

- A language is called a regular language if some finite automaton(DFA) recognizes it.
- A language is regular if and only if some nondeterministic finite automaton(NFA) recognizes it.
- If a language is described by a regular expression(RE), then it is also regular.



Regular Operations

- 3 operations are defined on languages called *Regular Operations*.
- Here, the name *Regular Operations* is just a name that has been chosen for these 3 operations.
- Regular Operations are used to study properties of the Regular Languages.

Definition:

Let Σ be an alphabet and $A,B \subseteq \Sigma^*$ be languages. Then the regular operations are as follows:

- Union: The language $A \cup B \subseteq \Sigma^*$ is defined as, $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$
- Concatenation: The language $AB \subseteq \Sigma^*$ is defined as, $AB = \{ wx \mid w \in A \text{ and } x \in B \}$
- Kleene star/ Star: The language A^* is defined as, $A^* = \{ x_1 x_2 \dots \dots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$

Regular Operations - Example

Let,

Alphabet $\Sigma = \{ a, b, c, \dots \dots \dots, z \}$

Language A = { good, bad } $\subseteq \Sigma^*$

Language B = { boy, girl } $\subseteq \Sigma^*$

Then,

$A \cup B$ = { good, bad, boy, girl }

AB = { goodboy, goodgirl, badboy, badgirl }

A^* = { ε, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, … … }

Regular Languages – Closure Property

- The regular languages are closed with respect to the regular operations:

  if $A, B \subseteq \Sigma^*$ are regular languages, then the languages $A \cup B$, AB, and A^* are also regular.

- There are many other operations on languages aside from the regular operations under which the regular languages are also closed.

  - Subtraction:

    The language $A \setminus B \subseteq \Sigma^*$ is defined as, $A \setminus B = \{ w \mid w \in A \text{ and } w \notin B \} = A \cap \overline{B}$
  - Complement:

    The language $\overline{A} \subseteq \Sigma^*$ is defined as, $\overline{A} = \{ w \mid w \in \Sigma^* \text{ and } w \notin A \} = \Sigma^* \setminus A$
  - Intersection:

    The language $A \cap B \subseteq \Sigma^*$ is defined as, $A \cap B = \{ w \mid w \in A \text{ and } w \in B \} = \overline{\overline{A} \cup \overline{B}}$
  - Reverse:

    The language $A^R \subseteq \Sigma^*$ is defined as, $A^R = \{ w_k w_{k-1} \dots \dots w_2 w_1 \mid w = w_1 w_2 \dots \dots w_k \in A \}$

Regular Expressions

Arithmetic Expressions:

- We use the operations + and x to build up expressions such as (5+3)x4
- The value of arithmetic expression is the number 32.

Regular Expression:

- We use regular operations to build up expressions describing languages called Regular Expressions.
- The value of regular expression is a language.
- For example, $(0 \cup 1)0^*$

Note:

REs have an important role in computer science applications. It provides powerful methods to describe different string patterns in UNIX commands, modern programming languages, text editors etc.

Regular Expressions – Definition

Regular Expression:

R is a regular expression if R is

- a for some a in the alphabet Σ,
- ε,
- ϕ,
- $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
- $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
- (R_1^*), where R_1 is a regular expression.

Regular Expressions – Precedence Order

Precedence of Regular Operations:

1. Parentheses
2. Star
3. Concatenation
4. Union

So according to the precedence sequence: (ab U a)* = (((a) (b)) U (a)) *

RE – Practices

Language, L(R)	Regular Expression
{ w | w contains a single 1 }	$0^* 1 0^*$
{ w | w consists of exactly three 1's }	$0^* 1 0^* 1 0^* 1 0^*$
{ w | w has at least a single 1 }	$(0 \mid 1)^* 1 (0 \mid 1)^*$
{ w | w contains at least three 1's }	$(0 \mid 1)^* 1 (0 \mid 1)^* 1 (0 \mid 1)^* 1 (0 \mid 1)^*$
{ w | w has at most one 1 }	$0^* \mid 0^* 1 0^*$
{ w | w contains an even number of 1's }	$(0^* 1 0^* 1 0^*)^* 0^*$
{ w | the number of 1's withing w can be evenly divided by 3 }	$(0^* 1 0^* 1 0^* 1 0^*)^* 0^*$
{ w | w is a string of even length }	$(\Sigma\Sigma)^*$, here $\Sigma = (0 \mid 1)$
{ w | the length of w is a multiple of 3 }	$(\Sigma\Sigma\Sigma)^*$, here $\Sigma = (0 \mid 1)$

RE – Practices

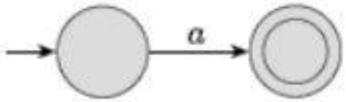
Language, L(R)	Regular Expression
{ w | w starts with 101 }	$101\ (0 \mid 1)^*$
{ w | w contains the string 001 as a substring }	$(0 \mid 1)^*\ 001\ (0 \mid 1)^*$
{ w | w ends with 3 consecutive 1's }	$(0 \mid 1)^*\ 111$
{ w | w doesn't end with 11 }	$\varepsilon \mid 0 \mid 1 \mid (0 \mid 1)^*\ (00 \mid 01 \mid 10)$
{ w | w ends with an even nonzero number of 0's }	$(\ (0 \mid 1)^*\ 1 \mid \varepsilon\ )\ 00\ (00)^*$
{ w | w starts and ends with the same symbol }	$0 \mid 1 \mid 0\Sigma^*0 \mid 1\Sigma^*1$, here $\Sigma = (0 \mid 1)$
{ w | w has at least 3 characters and the 3rd character is 0 }	$(0 \mid 1)\ (0 \mid 1)\ 0\ (0 \mid 1)^*$
{ w | every zero in w is followed by at least one 1 }	$1^*\ (01^+)^*$
{ w | w consists of alternating zeros and ones }	$(1 \mid \varepsilon)\ (01)^*\ (0 \mid \varepsilon)$

RE – Practices

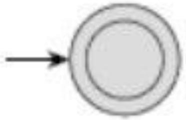
Language, L(R)	Regular Expression
{ 01, 10 }	01 | 10
$01^* \mid 1^*$	$(0 \mid \varepsilon)\ 1^*$
$\{ \varepsilon, 0, 1, 01 \}$	$(0 \mid \varepsilon)\ (1 \mid \varepsilon)$
ϕ	$R\phi$
$\{ \varepsilon \}$	ϕ^*
{ w | w starts with 0 and has odd length or, starts with 1 and has even length }	$0\ (\ (0 \mid 1)\ (0 \mid 1)\ )^* \mid 1\ (0 \mid 1)\ (\ (0 \mid 1)\ (0 \mid 1)\ )^*$

RE → NFA

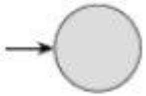
- $R = a$ for some $a \in \Sigma$. Then $L(R) = \{ a \}$, and the following NFA recognizes $L(R)$



- $R = \varepsilon$. Then $L(R) = \{ \varepsilon \}$ and the following NFA recognizes $L(R)$



- $R = \phi$. Then $L(R) = \phi$, and the following NFA recognizes $L(R)$

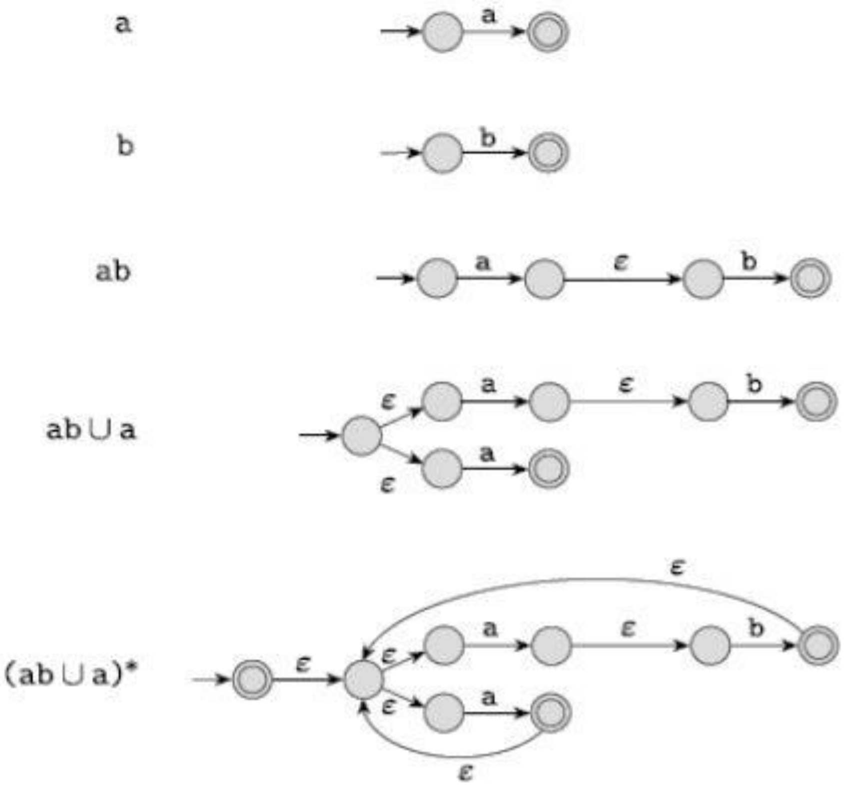


- $R = R_1 \cup R_2$, so $L(R) = L(R_1) \cup L(R_2)$ = NFA_1 recognizes $L(R_1)$ U NFA_2 recognizes $L(R_2)$
- $R = R_1 \circ R_2 = R_1R_{2=}$ $= L(R_1) \circ L(R_2)$ = NFA_1 recognizes $L(R_1)$ o NFA_2 recognizes $L(R_2)$
- $R = R_1^*$ $= L(R_1)^*$ = (NFA_1 recognizes $L(R_1)$)*

RE → NFA : Example 1

Convert the following RE to equivalent NFA:

(ab U a)*

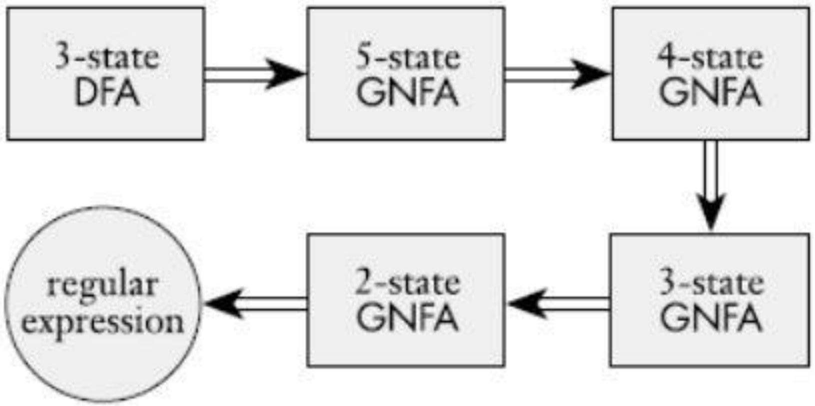


RE → NFA : Practices

- $(a \cup b)^* a b a$
- $(0 \cup 1)^* 0 0 0 (0 \cup 1)^* = \Sigma^* 0 0 0 \Sigma^*$ where $\Sigma = \{0, 1\} = 0 \cup 1$
- $(((0 0)^* (1 1)) \cup 0 1)^*$
- $(0 1 \cup 0 0 1 \cup 0 1 0)^*$
- $a (a b b)^* \cup b$
- $a^+ \cup (a b)^+$
- $(a \cup b^+) a^+ b^+$ [Hint: replace a^+ with $(a a^*)$]
- $1^* (0 1^+)^*$
- $(\Sigma \Sigma \Sigma)^*$
- $0 \Sigma^* 0 \cup 1 \Sigma^* 1 \cup 0 \cup 1$
- ϕ^*
- $(0 \cup \varepsilon) 1^*$

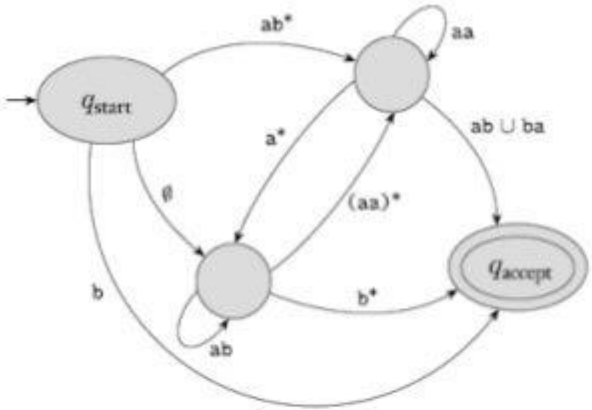
DFA → GNFA → RE

Steps to follow:



GNFA

- A GNFA is a nondeterministic finite automaton(NFA) in which each transition is labeled with a regular expression over the alphabet set instead of only members of the alphabet or ε.
- A single *initial state* with all possible outgoing transitions and no incoming transitions.
- A single *final state* without outgoing transitions but all possible incoming transitions. The accept state is not the same as the start state.
- A single transition between every two states, including self loops.



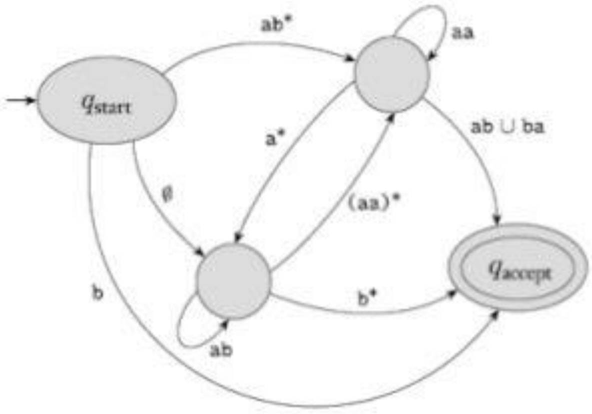
GNFA – Formal Definition

A generalized nondeterministic finite automaton is a 5 tuple, $(Q, \Sigma, \delta, q_{start}, q_{accept})$, where

- Q is the finite set of states
- Σ is the input alphabet
- $\delta: (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow \mathcal{R}$ is the transition function

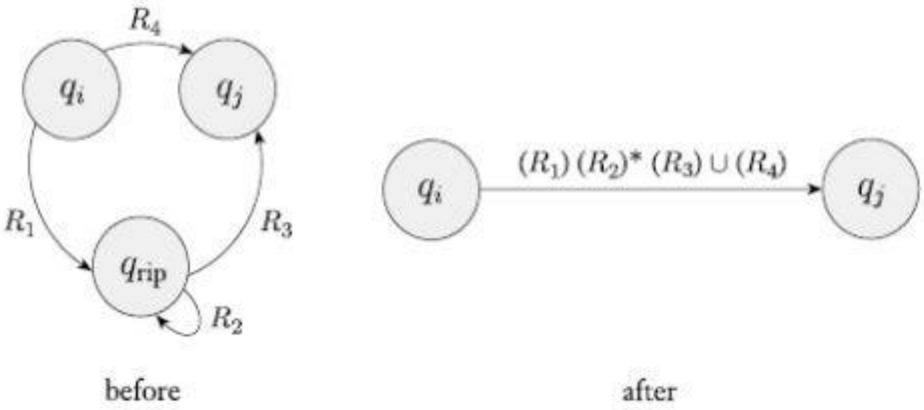
[if you travel from state a ϵ $(Q - \{q_{accept}\})$ to state b ϵ $(Q - \{q_{start}\})$ then this relation will generate $\mathcal{R}$ regular expression]

- q_{start} is the start state and
- q_{accept} is the accept state



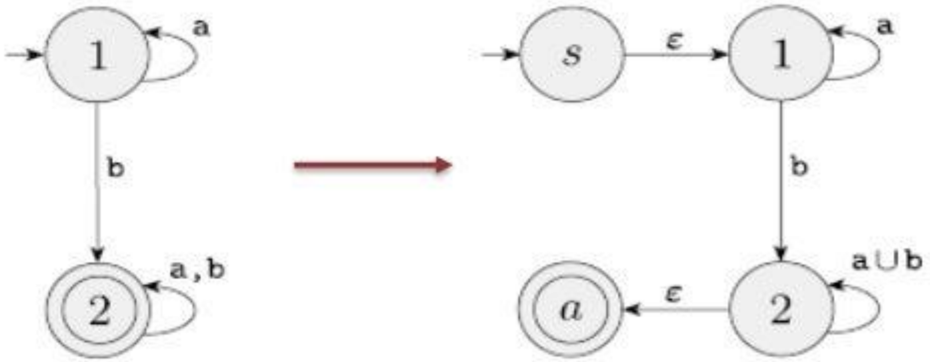
GNFA – State Minimization Rule

Removing q_{rip} state,



DFA → GNFA : Steps to Follow

- Add a new start state with an ε arrow to the old start state
- Add a new accept state with ε arrows from the old accept states to this new accept state
- If any arrows have multiple labels, replace each with a single arrow whose label is the union of the previous labels
- Add arrows labeled ϕ between states that had no arrows. [better no to show this in the diagram, for simplicity]



GNFA → RE : Algorithm

CONVERT(G):

1. Let k be the number of states of G.
2. If k=2, then G must consist of a start state, an accept state, and a single arrow connecting them and labeled with a regular expression R

   – Return the expression R.
3. If k>2, we select any state $q_{rip} \in Q$ different from q_{start} and q_{accept} and let G' be the GNFA $(Q', \Sigma, \delta', q_{start}, q_{accept})$, where

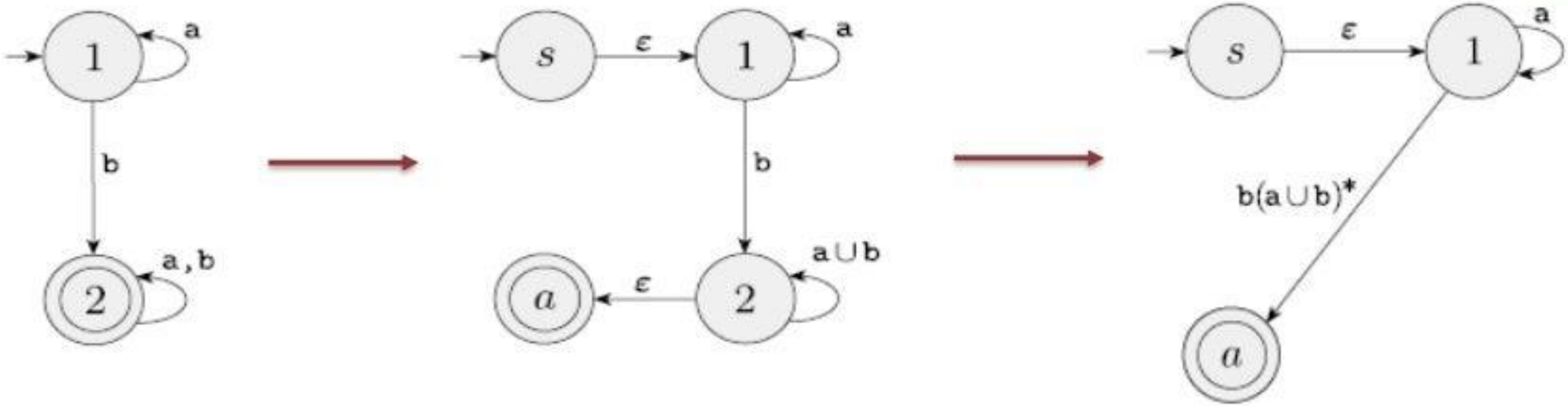
$$Q' = Q - \{q_{rip}\}$$

   and for any $q_i \in Q' - \{q_{accept}\}$ and any $q_j \in Q' - \{q_{start}\}$, let

$$\delta'(q_i, q_j) = (R_1)\,(R_2)^*(R_3) \cup (R_4)$$

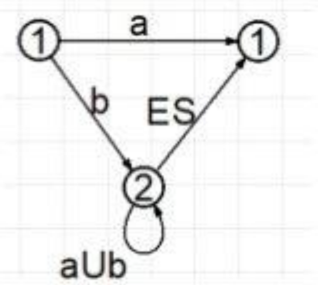
   for $R_1 = \delta(q_i, q_{rip})$, $R_2 = \delta(q_{rip}, q_{rip})$, $R_3 = \delta(q_{rip}, q_j)$, and $R_4 = \delta(q_i, q_j)$
4. Compute *CONVERT(G')* and return this value.

GNFA → RE : Removing State 2

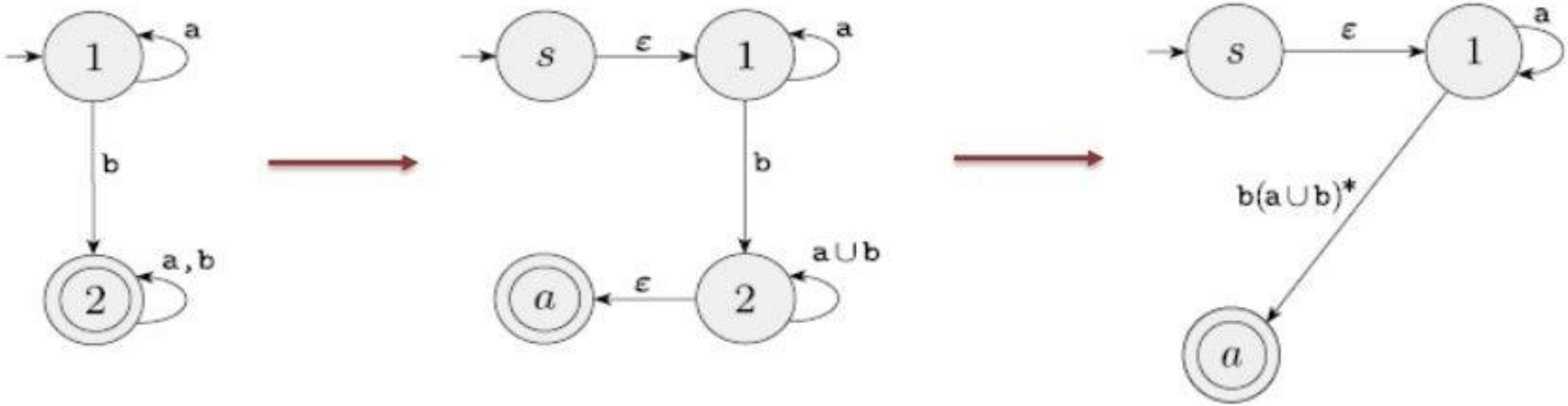


Here, 2 is associated with states *1* and *a*. All possible combinations are (1,1), (1,a), (a,a), (a,1). But there will be no outgoing from ***a***, so no need to consider (a,a) and (a,1) states.

For (1,1), R = a U (b (aUb)* ϕ)) = a

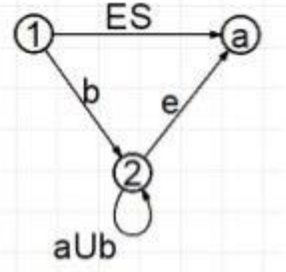


GNFA → RE : Removing State 2

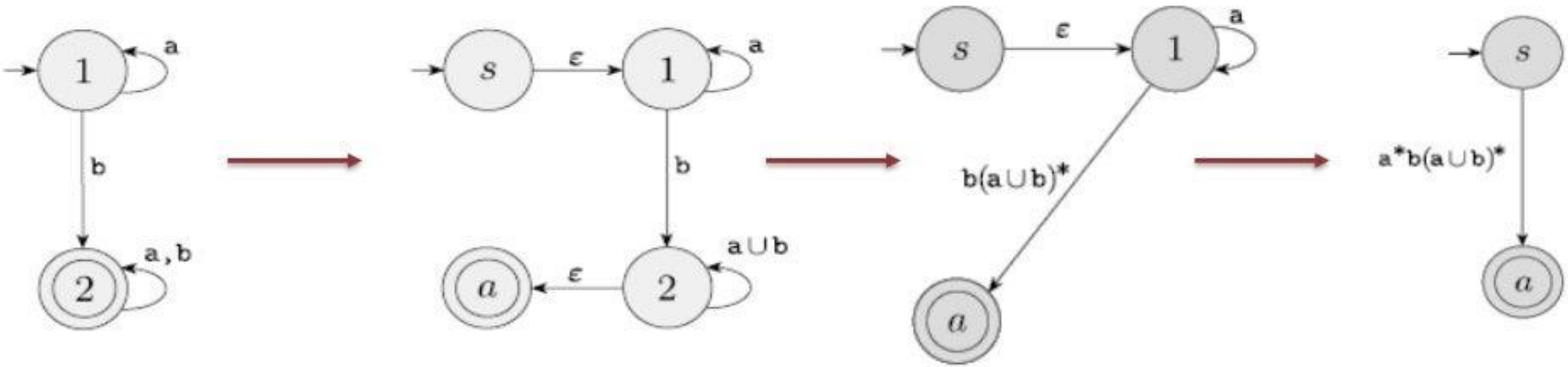


Here, 2 is associated with states *1* and *a*. All possible combinations are (1,1), (1,a), (a,a), (a,1). But there will be no outgoing from ***a***, so no need to consider (a,a) and (a,1) states.

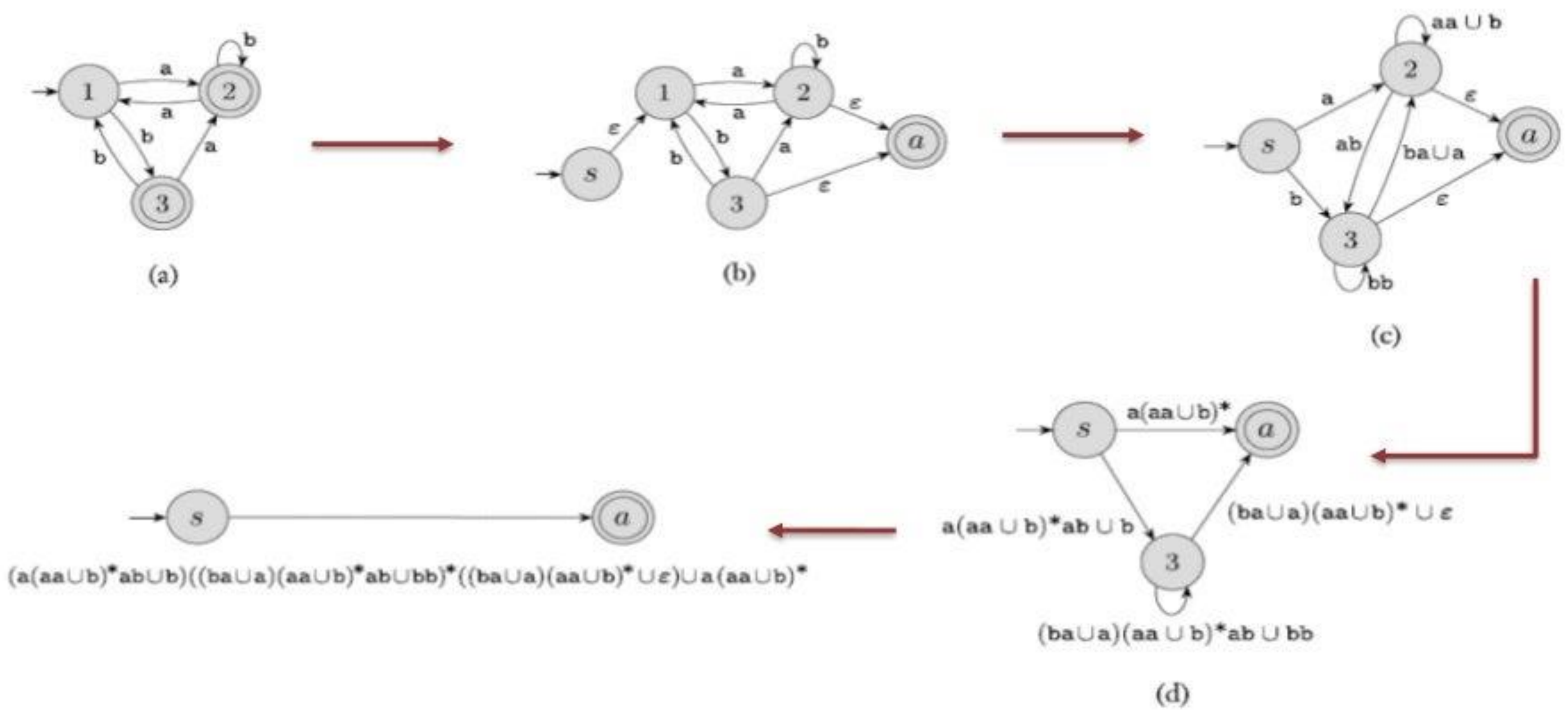
For (1,a), R = ϕ U (b (aUb)* ε) = b(aUb)*



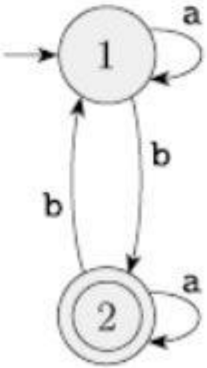
GNFA → RE : Removing State 1



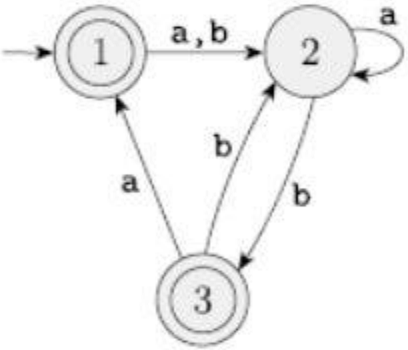
DFA → GNFA → RE : Example 2



DFA → GNFA → RE : Practices



Practice 1



Practice 2