REGULAR EXPRESSIONS

Shekh. Md. Saifur Rahman

Lecturer, CSE Department

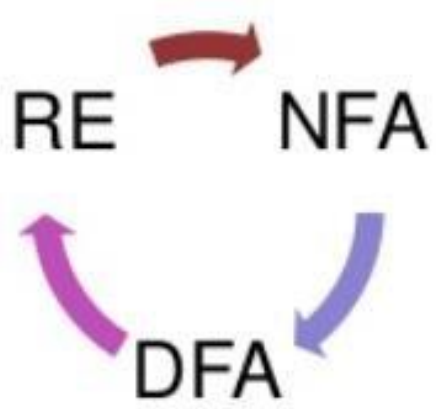
UIU

Introduction

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.



Introduction

- In arithmetic, we can use the operations + and × to build up expressions such as (5 + 3) × 4.
- Similarly, we can use the regular operations to build up expressions describing languages.
- These are called regular expressions.
- An example is:

  (0 U 1)0*

Importance

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.

REs is heavily used in pattern matching.

Patterns are very flexible and provide us with a way to make our own pattern to validate the input.

Regular expressions are useful in search and replace operations. The typical use case is to look for a sub-string that matches a pattern and replace it with something else.

Before going further, a small recap

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 ⊆ Σ* 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, $A^* = \{ x_1 x_2 \ldots \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$

Example

- Let, Alphabet Σ = { a, b, c, … … … , z }
- Language A = { good, bad } ⊆ Σ*
- Language B = { boy, girl } ⊆ Σ*

Then,

A∪B = { good, bad, boy, girl }

AB = { goodboy, goodgirl, badboy, badgirl }

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

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.

Regular Expressions - Definition

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

Regular Expression – Precedence Order

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

So according to the precedence sequence:

(ab U a)* = (((a) (b)) U (a)) *

Let’s get our hands dirty

In the following instances, we assume that the alphabet Σ is *{ 0 , 1 }*

1)0* 1 0*

- { w | w contains a single 1 }

2)0* 1 0* 1 0* 1 0*

- { w | w consists of exactly three 1’s }

3) Σ* 1 Σ*

- { w | w has at least a single 1 }

4) Σ* 001 Σ*

- { w | w has the string 001 as a substring }

Example - Continued

5) { w | w contains at least three 1's }

- (0 | 1)* 1 (0 | 1)* 1 (0 | 1)* 1 (0 | 1)*

6) { w | w has at most one 1 }

- 0* | 0* 1 0*

7) { w | w contains an even number of 1's }

- (0* 1 0* 1 0*)* 0*

8) { w | the number of 1's within w can be evenly divided by 3 }

- (0* 1 0* 1 0* 1 0*)* 0*

Example - Continued

9) { w | w is a string of even length }

- $(\Sigma\Sigma)^*$

10) { w | the length of w is a multiple of 3 }

- $(\Sigma\Sigma\Sigma)^*$

Practices

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

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 – Equivalence with Finite Automata

Regular expressions and finite automata are equivalent in their descriptive power.

This fact is surprising because finite automata and regular expressions superficially appear to be rather different.

However, any regular expression can be converted into a finite automaton that recognizes the language it describes, and vice versa.

Recall that a regular language is one that is recognized by some finite automaton.

RE – Equivalence with Finite Automata

THEOREM 1.54

A language is regular if and only if some regular expression describes it.

This theorem has two directions.

We state and prove each direction as separate lemma

RE → NFA

LEMMA 1.55

If a language is described by a regular expression, then it is regular.

Say that we have a regular expression *R* describing some language *A*.

We show how to convert *R* into an NFA recognizing *A*.

We have previously learned that, if an NFA recognizes *A* then *A* is regular.

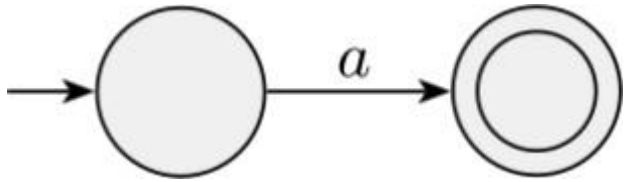
Let's convert *R* into an NFA *N*.

We consider the six cases in the formal definition of regular expressions.

RE → NFA

1. R = a for some a ∈ Σ. Then L(R) = { a }

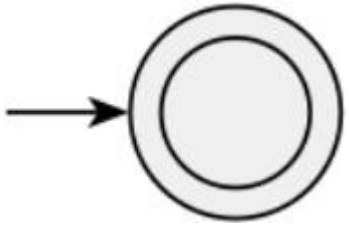
   the following NFA recognizes L(R)



RE → NFA

2. R = ε . Then L(R) = { ε },

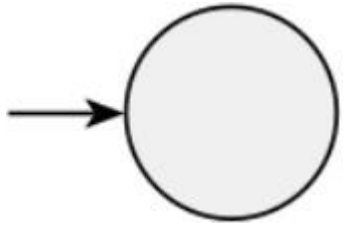
   the following NFA recognizes L(R)



RE → NFA

3. R = ϕ .

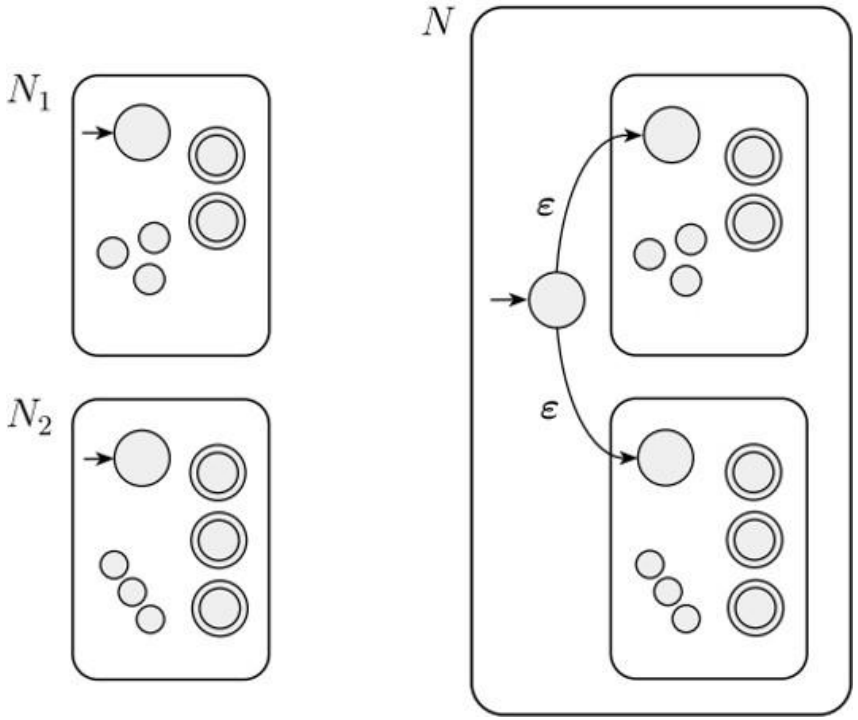
   Then L(R) = ϕ



RE → NFA

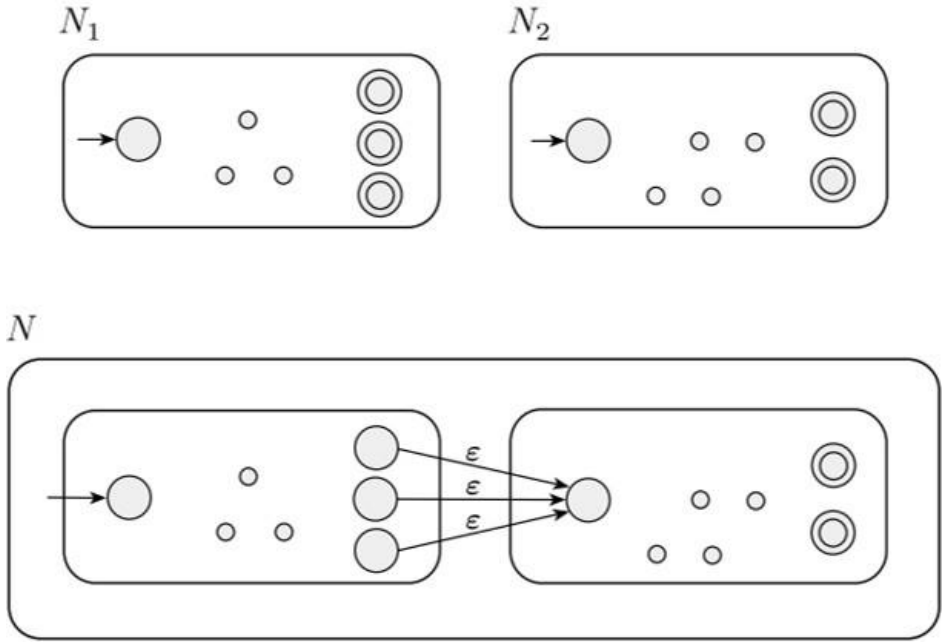
4. R = R_1 U R_2, so L(R) = L(R_1) U L(R_2) =

   NFA_1 recognizes L(R_1) U NFA_2 recognizes L(R_2)



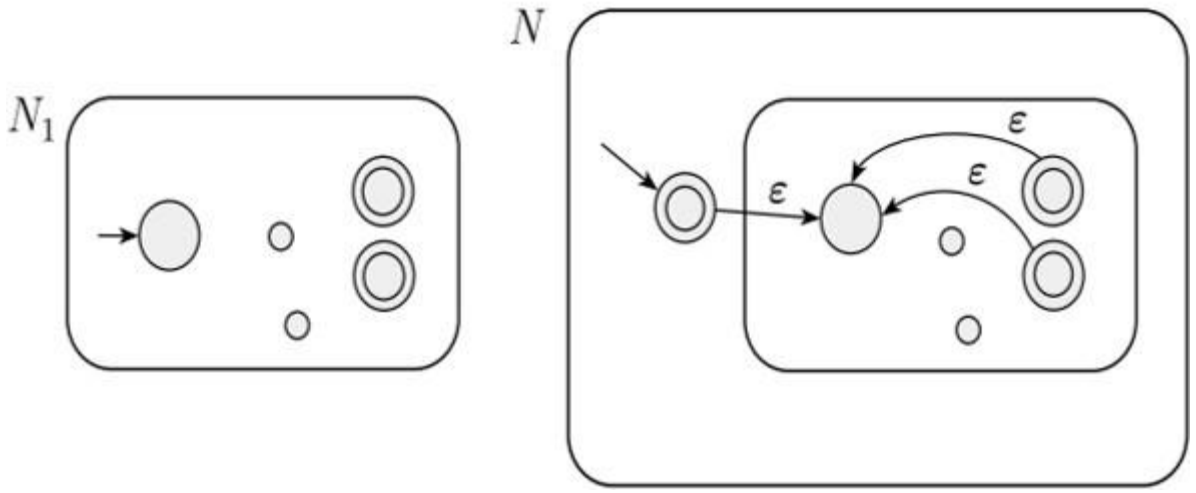
RE → NFA

5. R = R_1 o R_2 = R_1R_2 = L(R_1) o L(R_2) = NFA_1 recognizes L(R_1) o NFA_2 recognizes L(R_2)



RE → NFA

6. R = R_1 * = L(R_1)* = (NFA_1 recognizes L(R_1))*



RE → NFA

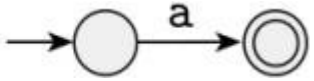
We use the constructions given in the proofs that the class of regular languages is closed under the regular operations.

In other words, we construct the NFA for *R* from the NFA's for R_1 and R_2 (or just R_1 in case 6) and the appropriate closure construction.

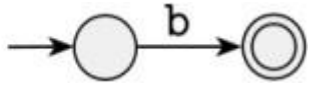
Convert the following RE to equivalent NFA

ab U a)*

a



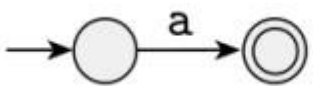
b



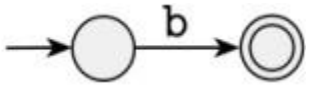
Convert the following RE to equivalent NFA

(ab U a)*

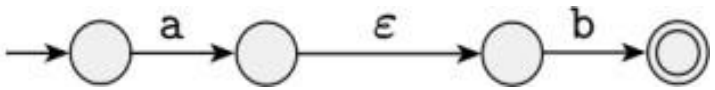
a



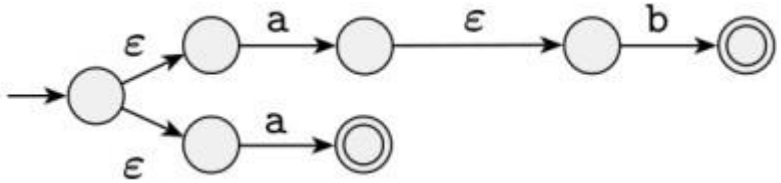
b



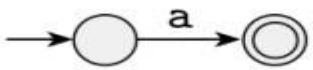
ab



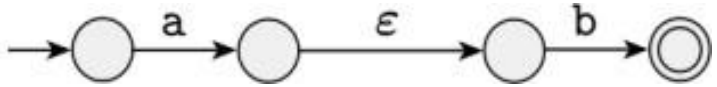
ab ∪ a

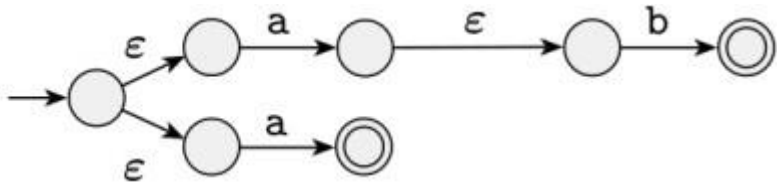


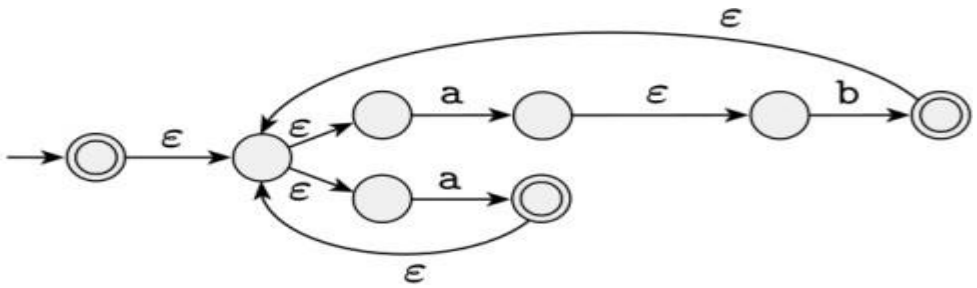
(ab U a)*

a 

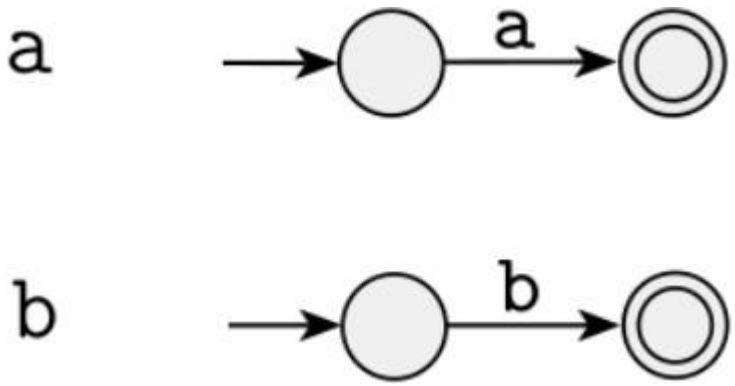
b 

ab 

ab ∪ a 

(ab ∪ a)* 

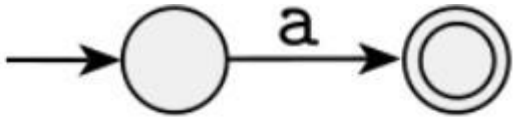
(a U b)*aba



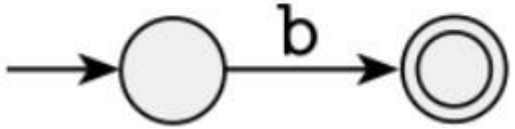
(a U b)*aba

---

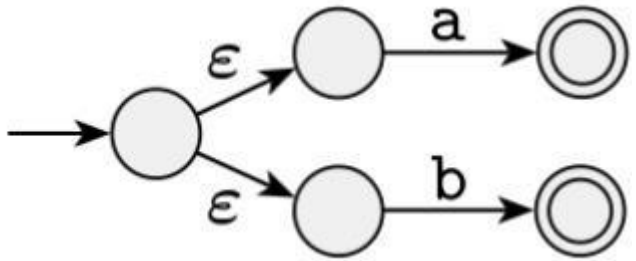
a



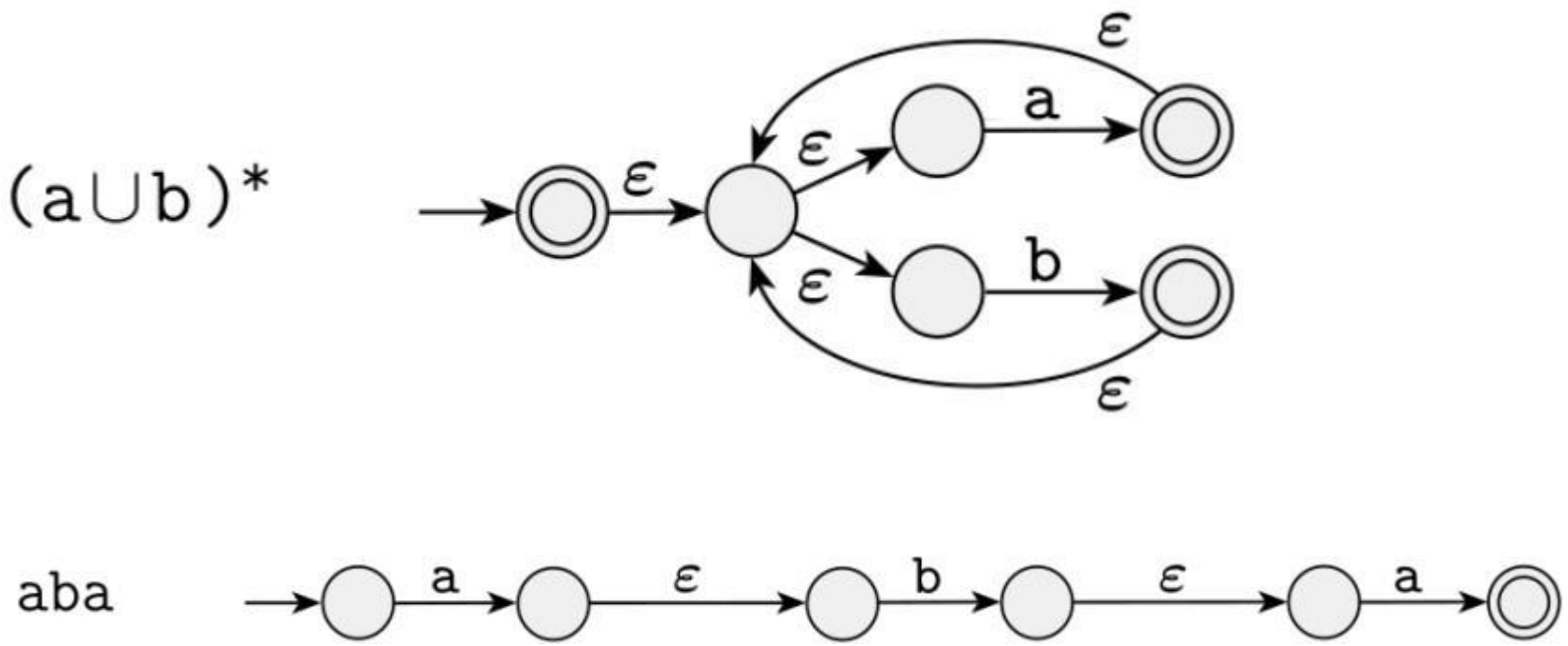
b



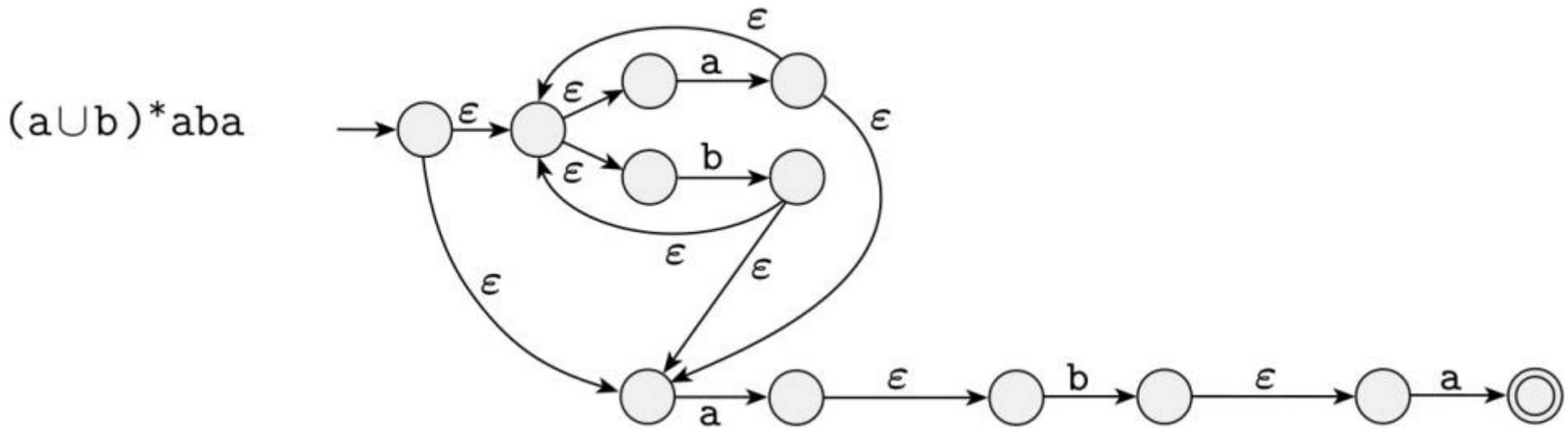
a∪b



(a U b)*aba



(a U b)*aba



RE → NFA : Practices

(a U b) * a b a

(0 U 1) * 0 0 0 (0 U 1)* = Σ* 0 0 0 Σ* where Σ = 0, 1 = 0 U 1

(((0 0)* (1 1)) U 0 1) *

(0 1 U 0 0 1 U 0 1 0) *

a (a b b) * U b

a+ U (a b) +

(a U b+) a+ b+ [Hint: replace a+ with (a a*)]

1* (0 1+)*

(Σ Σ Σ) *

0 Σ* 0 U 1 Σ* 1 U 0 U 1

ϕ*

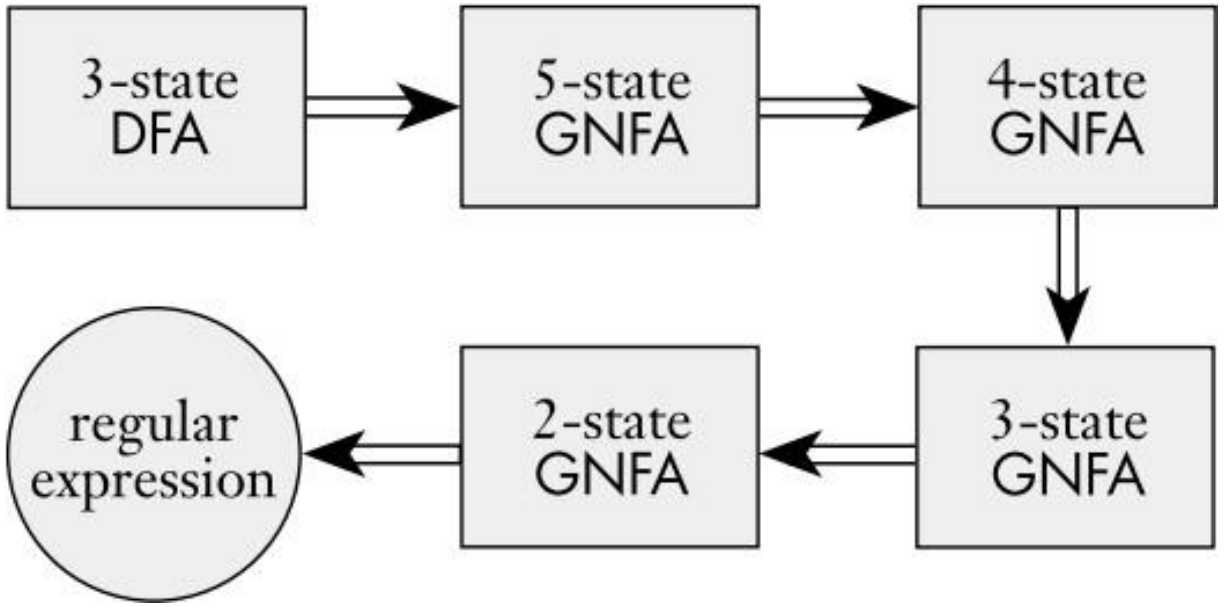
(0 U ε) 1*

FA → RE ?

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

Steps to follow



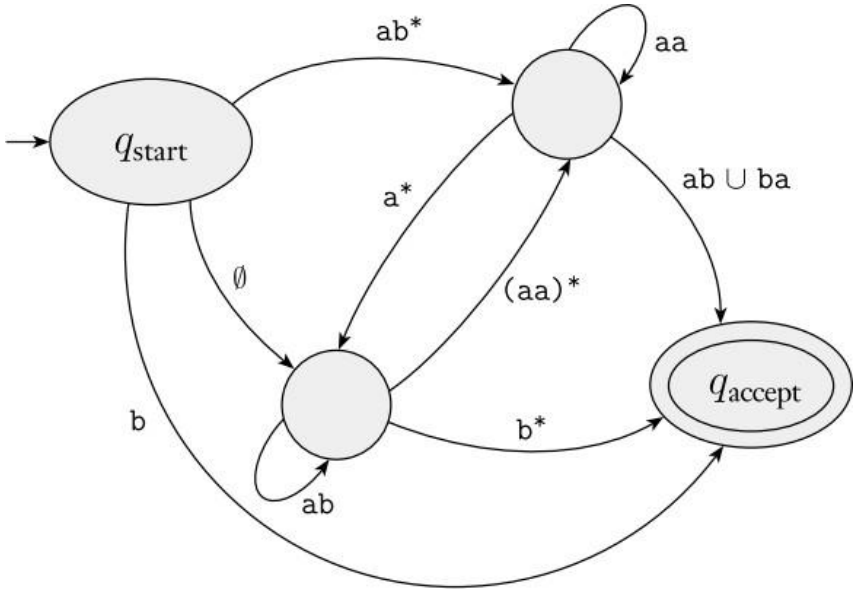
GNFA

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.



Formal Definition

DEFINITION 1.64

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

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta\colon (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathcal{R}$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

GNFA – State Minimization Rule

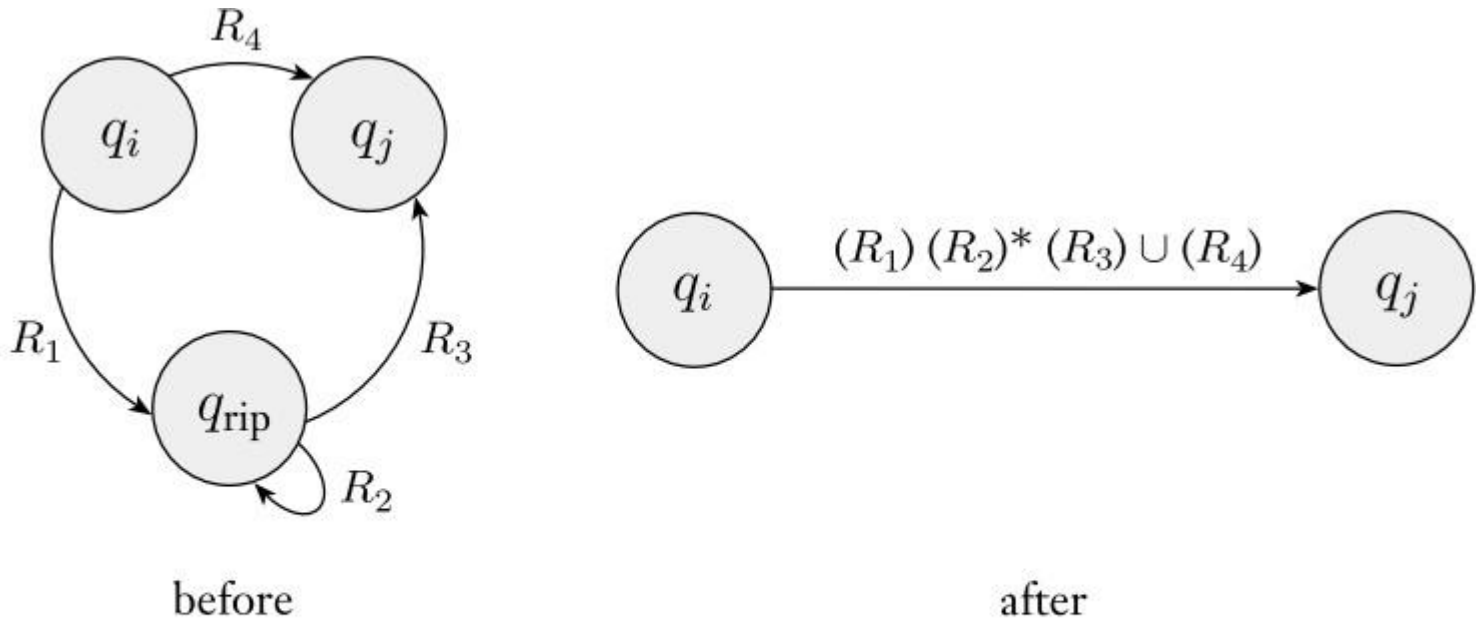


FIGURE 1.63
Constructing an equivalent GNFA with one fewer state

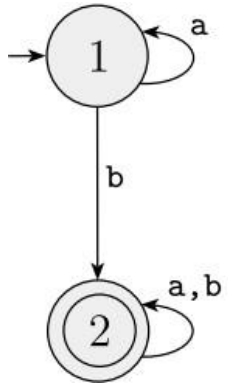
DFA → GNFA – Steps to follow

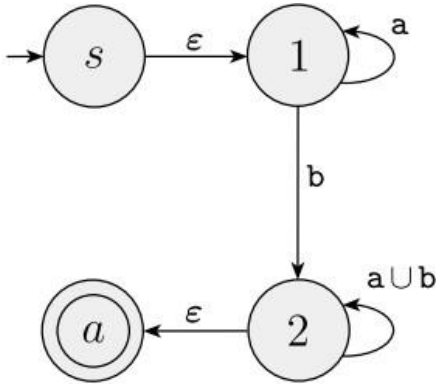
Add a new start state with an ε arrow to the old start state

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

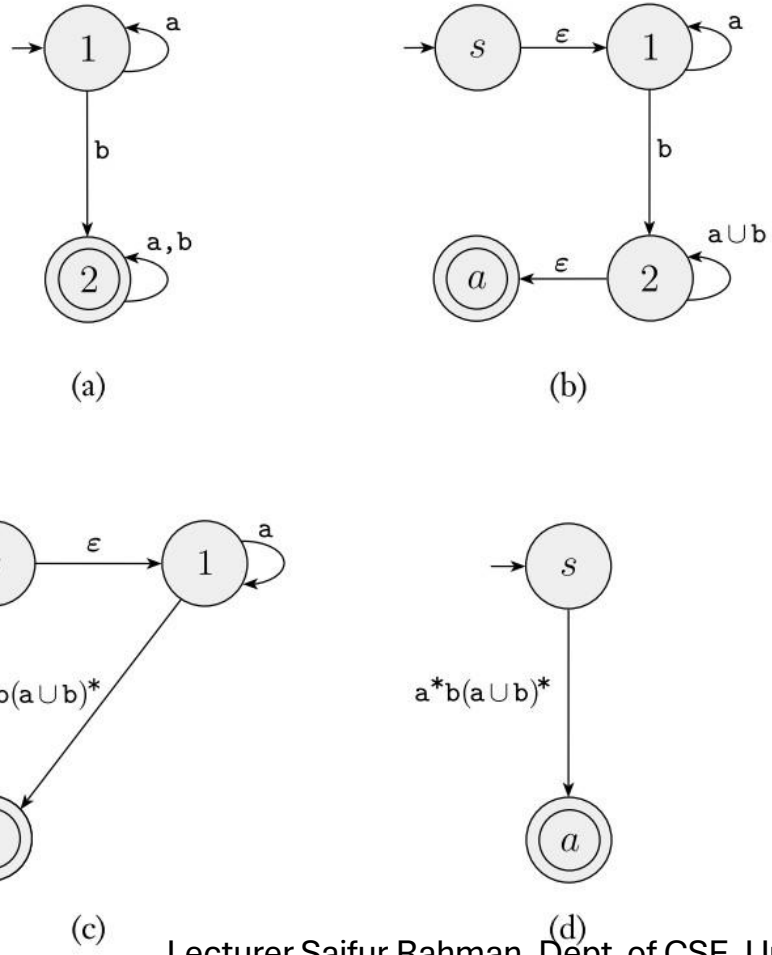
   $$Q' = Q - \{q_{rip}\}$$

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

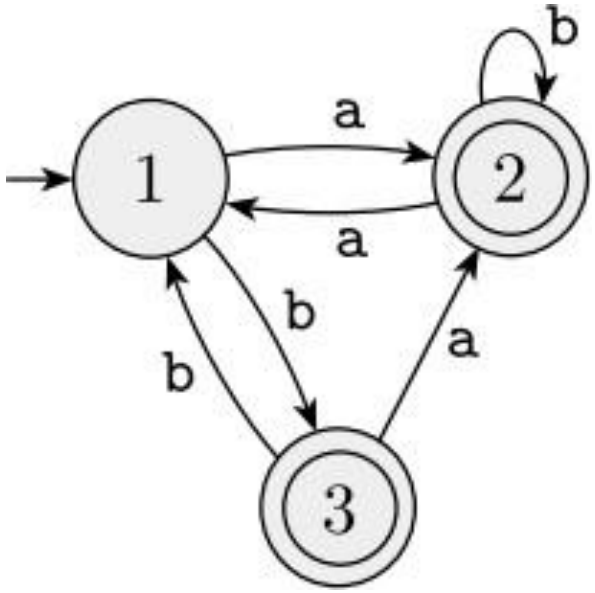
   $$\delta'(q_i, q_j) = (R_1)\,(R_2)^*(R_3) \cup (R_4)$$

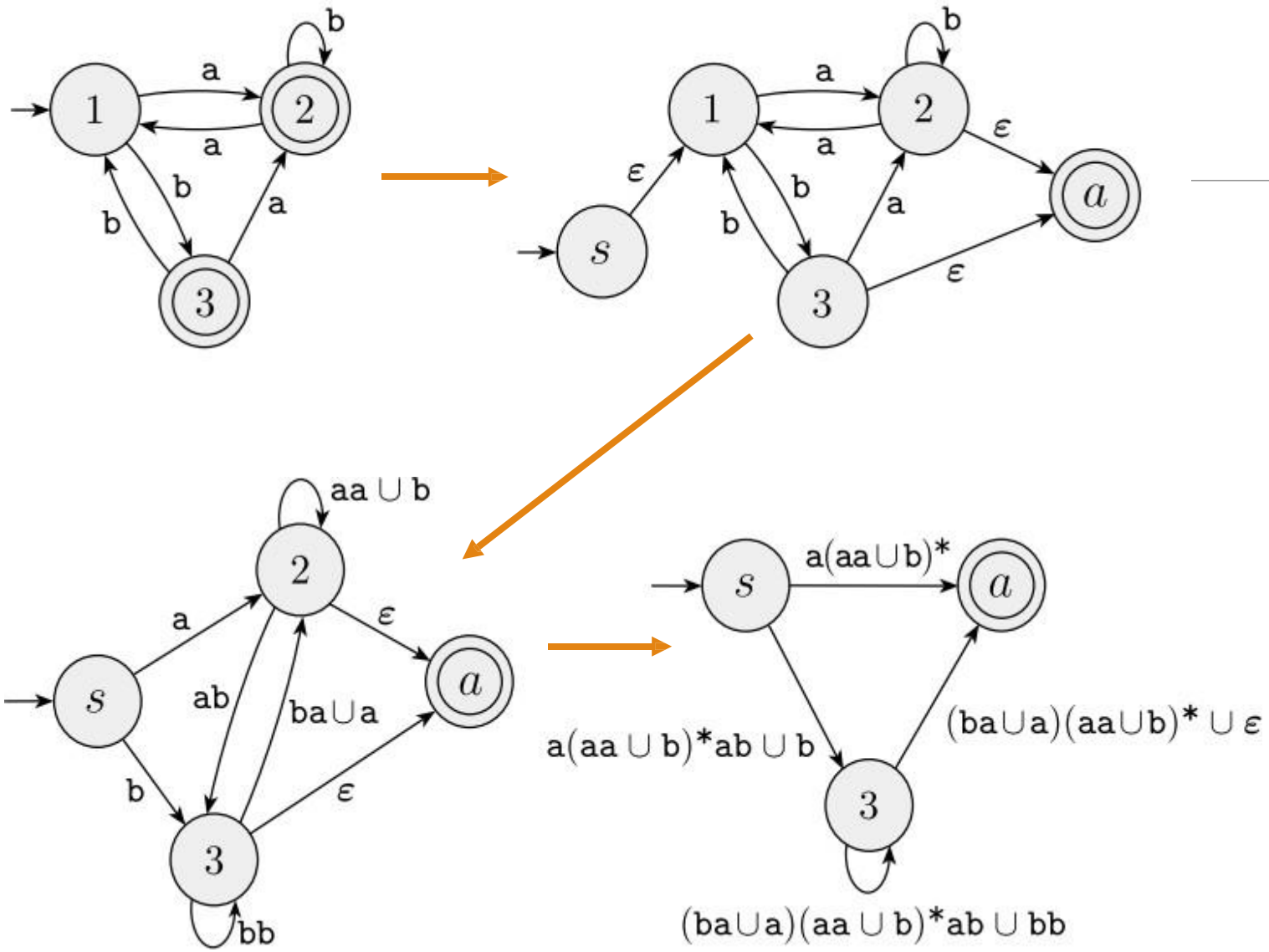
   for $R_1 = \delta(q_i, q_{rip})$, $R_2 = \delta(q_{rip}, q_{rip})$, $R_3 = \delta(q_{rip}, q_j)$, and $R_4 = \delta(q_i, q_j)$
4. Compute *CONVERT(G')* and return this value.

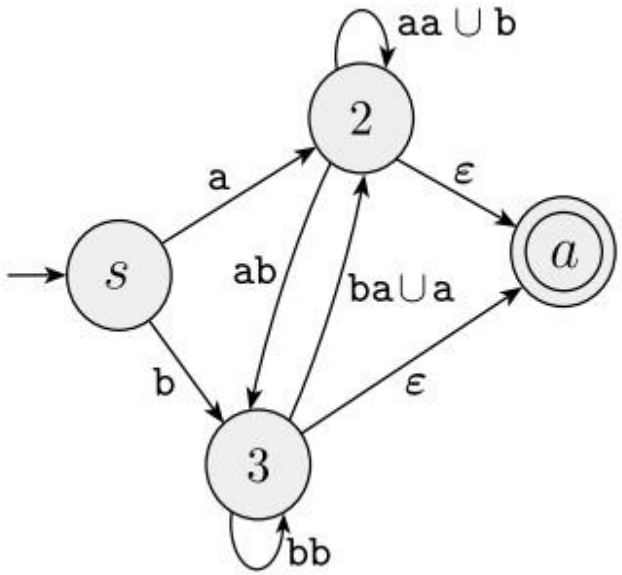
GNFA → RE

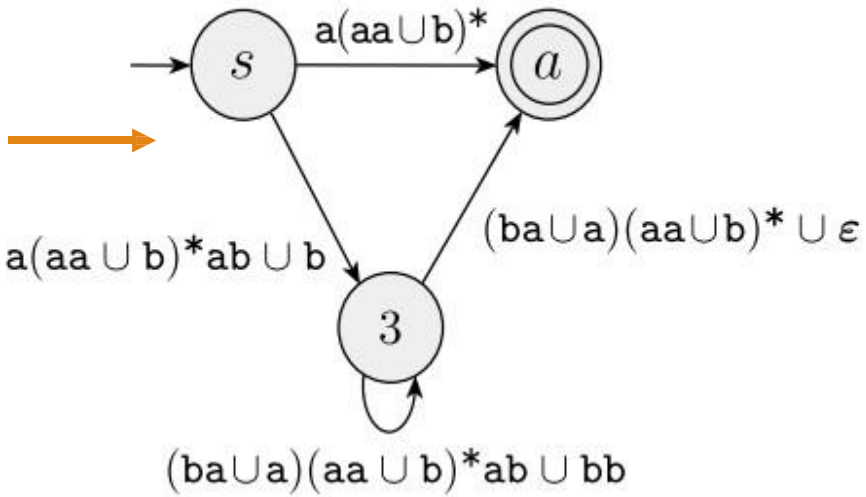


Example



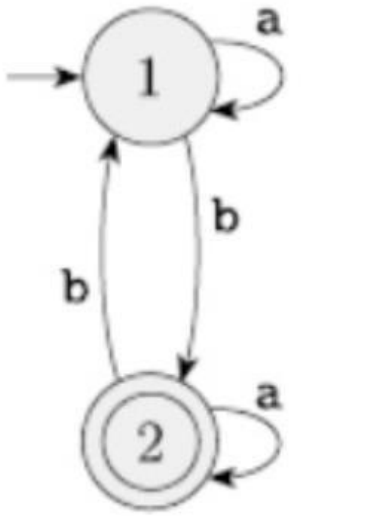

1
2
3
a
b
s
ε
aa ∪ b
ab
ba∪a
bb
a(aa∪b)*
a(aa ∪ b)*ab ∪ b
(ba∪a)(aa∪b)* ∪ ε
(ba∪a)(aa ∪ b)*ab ∪ bb



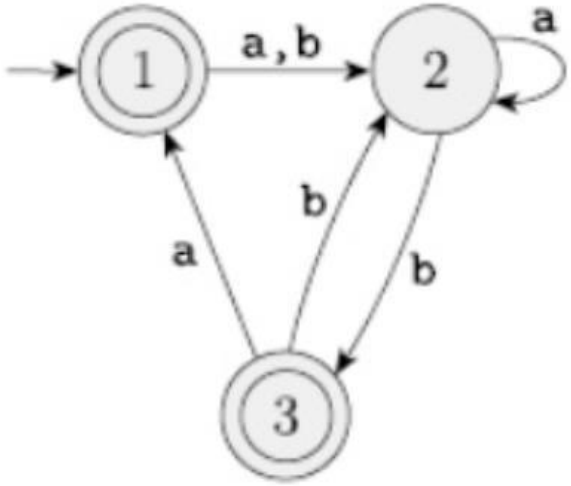


$$(a(aa \cup b)^*ab \cup b)((ba \cup a)(aa \cup b)^*ab \cup bb)^*((ba \cup a)(aa \cup b)^* \cup \varepsilon) \cup a(aa \cup b)^*$$

DFA → GNFA → RE : Practices



Practice 1



Practice 2