Theory of Automata

Regular Expressions

Dr. Sabina Akhtar

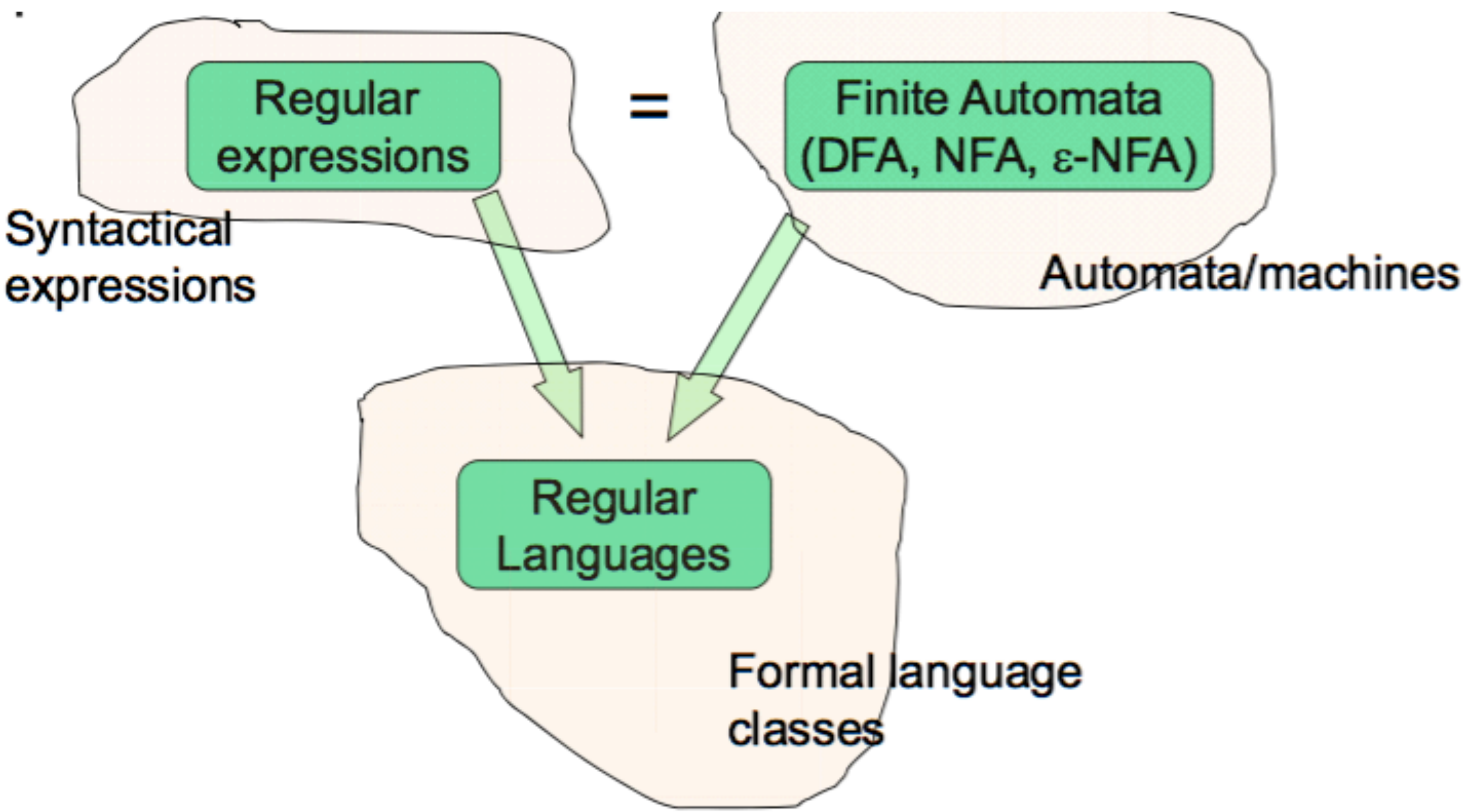
Revision

REGULAR EXPRESSIONS

Introduction

- *Regular expressions* are an algebraic way to describe languages.
- Similar to programming language
- Important applications
  - Text-search application
  - Compiler components
- User-friendly alternative to NFA

Regular expressions



Regular Expressions vs. Finite Automata

- Offers a declarative way to express the pattern of any string we want to accept
  - E.g., 01*+ 10*

- Automata => more machine-like

  < input: string , output: [accept/reject] >
- Regular expressions => more program syntax-like

- Unix environments heavily use regular expressions
  - E.g., bash shell, grep, vi & other editors, sed
- Perl scripting – good for string processing
- Lexical analyzers such as Lex, Javacc

Definition

- Basis 1: If *a* is any symbol, then **a** is a RE, and L(**a**) = {a}.
  - Note: {a} is the language containing one string, and that string is of length 1.
- Basis 2: ϵ is a RE, and L(ϵ) = {ϵ}.
- Basis 3: ∅ is a RE, and L(∅) = ∅.

Inductive Construction

Let R_1 and R_2 be two regular expressions representing languages L_1 and L_2 , resp.

- The string $\left(R_1 \cup R_2\right)$ is a regular expression representing the set $L_1 \cup L_2$.
- The string $\left(R_1 R_2\right)$ is a regular expression representing the set $L_1 \circ L_2$.
- The string $\left(R_1\right)^*$ is a regular expression representing the set L_1^* .

Building Regular Expressions

- Let E be a regular expression and the language represented by E is L(E)
- Then:
  - (E) = E
  - L(E + F) = L(E) U L(F)
  - L(E F) = L(E) L(F)
  - L(E*) = (L(E))*

Examples

Precedence of Operators

- Highest to lowest
  - * operator (star)
  - . (concatenation)
  - + operator
- Example:

  $01^*+1 = (0.((1)^*))+ 1$

Examples: RE's

- L(**01**) = {01}.
- L(**01**+**0**) = {01, 0}.
- L(**0**(**1**+**0**)) = {01, 00}.
  - Note order of precedence of operators.
- L(**0***) = {ϵ, 0, 00, 000,... }.
- L((**0**+**10**)*(ϵ+**1**)) = all strings of 0's and 1's without two consecutive 1's.

Example

- Example 3.2: Consider the language consisting of strings of a’s and b’s containing aab.

- Regular expression?

Class Activity

- ***L = { w | w is a binary string which does not contain two consecutive 0s or two consecutive 1s anywhere)***
  - E.g., w = 01010101 is in L, while w = 10010 is not in L

- Write the regular expression.

Solution

- Goal: Build a regular expression for L
- Four cases for w:
  - Case A: w starts with 0 and |w| is even
  - Case B: w starts with 1 and |w| is even
  - Case C: w starts with 0 and |w| is odd
  - Case D: w starts with 1 and |w| is odd
- Regular expression for the four cases:
  - Case A: $(01)^*$
  - Case B: $(10)^*$
  - Case C: $0(10)^*$
  - Case D: $1(01)^*$
- Since L is the union of all 4 cases:
  - Reg Exp for L = $(01)^* + (10)^* + 0(10)^* + 1(01)^*$
- If we introduce ε then the regular expression can be simplified to:
  - Reg Exp for L = $(\varepsilon + 1)(01)^*(\varepsilon + 0)$

Algebraic Laws of Regular Expressions

- <u>Commutative:</u>
  - $E+F = F+E$
- <u>Associative:</u>
  - $(E+F)+G = E+(F+G)$
  - $(EF)G = E(FG)$
- <u>Identity:</u>
  - $E+\Phi = E$
  - $\varepsilon E = E \varepsilon = E$
- <u>Annihilator:</u>
  - $\Phi E = E\Phi = \Phi$

Algebraic Laws...

- Distributive:
  - $E(F+G) = EF + EG$
  - $(F+G)E = FE+GE$
- Idempotent: $E + E = E$
- Involving Kleene closures:
  - $(E^*)^* = E^*$
  - $\Phi^* = \varepsilon$
  - $\varepsilon^* = \varepsilon$
  - $E^+ = EE^*$
  - $E? = \varepsilon + E$

Class Activity

- ***L = { w | w is a binary string that contains 111 as a substring anywhere in the string)***
  - E.g., w = 0101011101 is in L, while w = 10010 is not in L

111

101110001

- Write the regular expression.

RE : $(0+1)^* \, 111 \, (0+1)^*$

Class Activity

$(11)^*$

$\epsilon \quad 111$

- ***L = { w | w is a binary string that contains odd number of 1s in the string)***
  - E.g., w = 0100100001 is in L, while w = 10010 is not in L

- Write the regular expression.

$(11)^*$

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

Class Activity

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

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

1.20 For each of the following languages, give two strings that are members and two strings that are *not* members—a total of four strings for each part. Assume the alphabet $\Sigma = \{\mathtt{a},\mathtt{b}\}$ in all parts.

a. $\mathtt{a}^*\mathtt{b}^*$

b. $\mathtt{a}(\mathtt{ba})^*\mathtt{b}$

e. $\Sigma^*\mathtt{a}\Sigma^*\mathtt{b}\Sigma^*\mathtt{a}\Sigma^*$

f. $\mathtt{aba} \cup \mathtt{bab}$

bbb bb aba

Class Activity

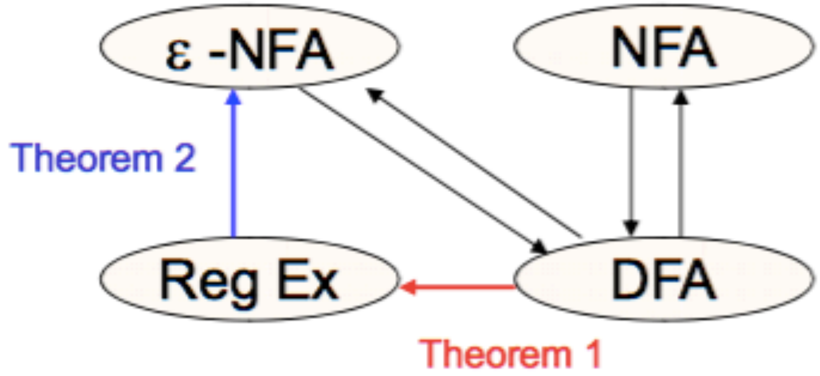
- Give regular expressions for the following languages.
  - The set of strings over $\Sigma = \{a, b, c\}$ containing at least one a and at least one b.
  - The set of strings of 0's and 1's whose tenth symbol from the right end is 1.
  - The set of strings of 0's and 1's with at most one pair of consecutive 1's.

FINITE AUTOMATA AND REGULAR EXPRESSION

Finite Automata (FA) & Regular Expressions (Reg Ex)

- To show that they are interchangeable, consider the following theorems:
  - *Theorem 1: For every DFA A there exists a regular expression R such that L(R)=L(A)*
  - *Theorem 2: For every regular expression R there exists an ε-NFA E such that L(E)=L(R)*

Proofs in the book

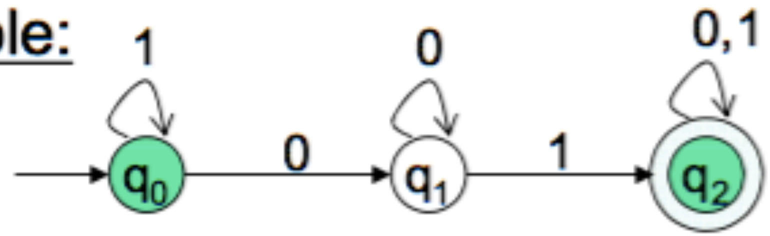


Kleene Theorem

DFA to RE construction
(using state elimination technique)

Informally, trace all distinct paths (traversing cycles only once)
from the start state to *each of the* final states
and enumerate all the expressions along the way

Example:

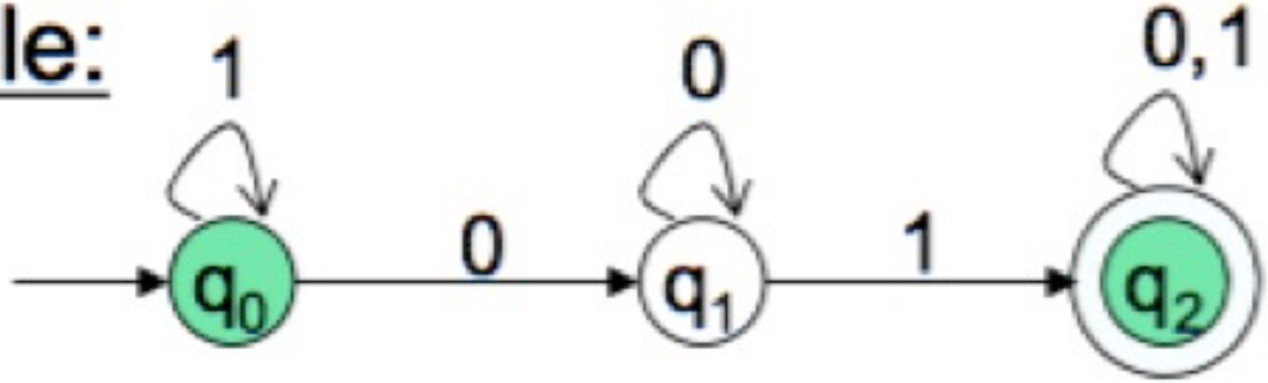


$$\underbrace{(1^*)}_{1^*}\ \underbrace{0\ \ (0^*)}_{00^*}\ \underbrace{1}_{1}\ \underbrace{(0+1)^*}_{(0+1)^*}$$

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

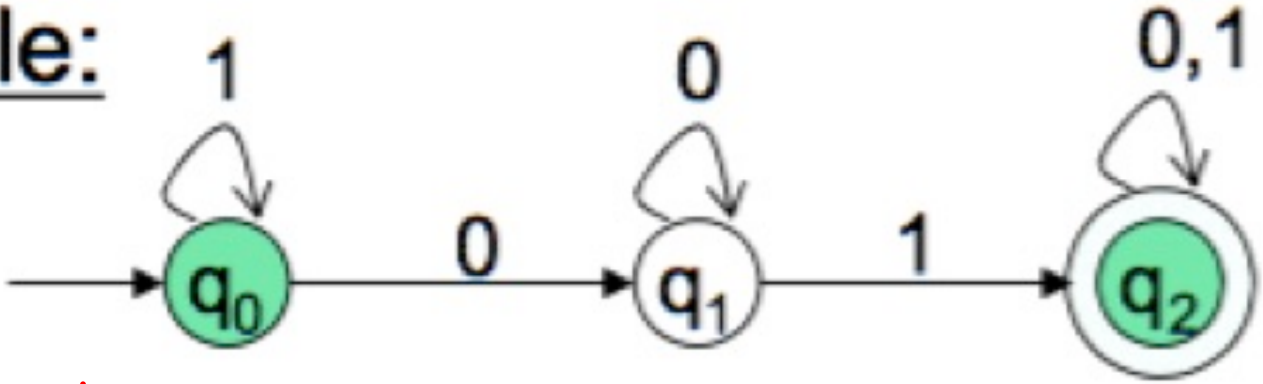
Q) What is the language?

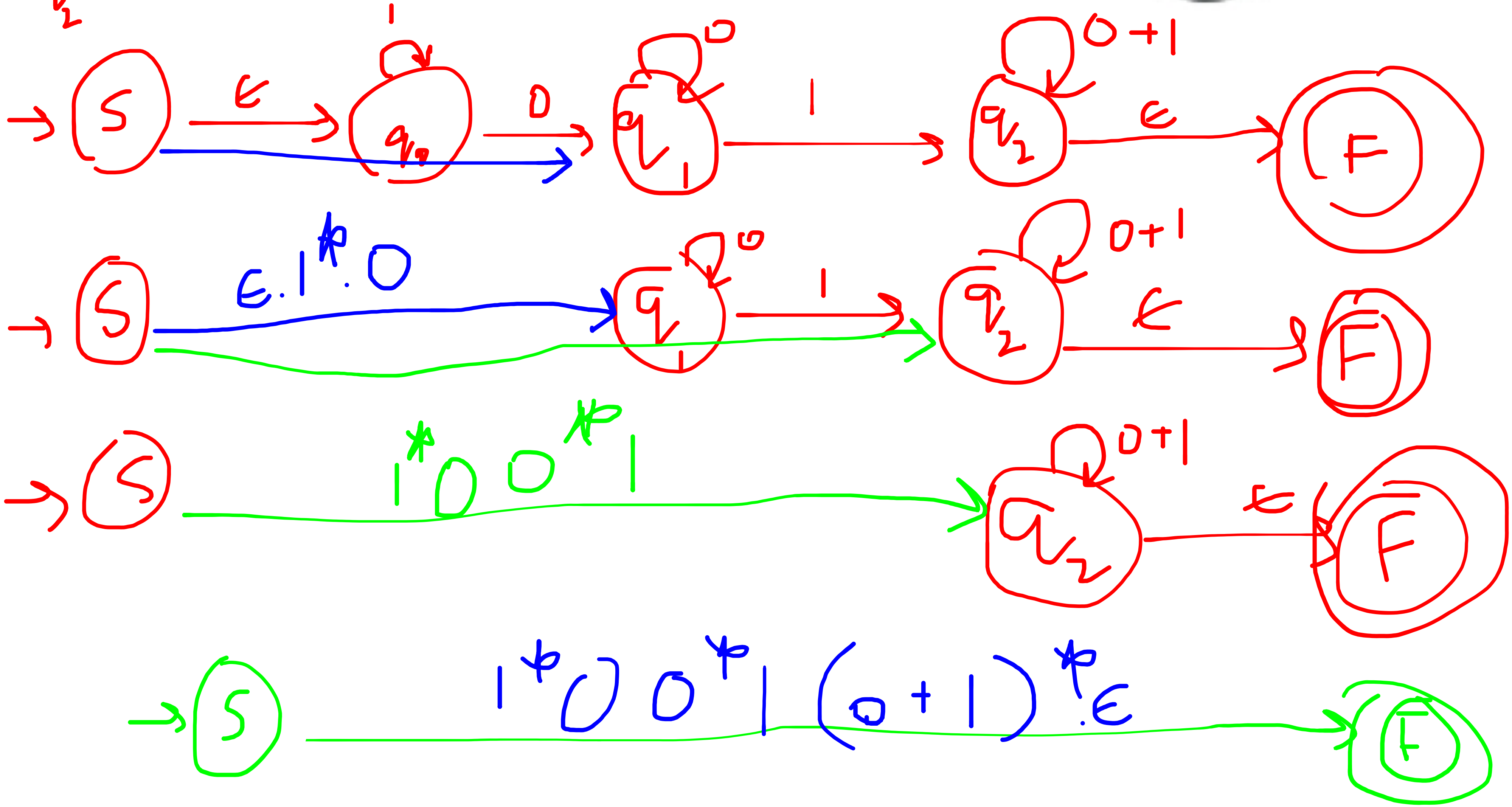
Example:



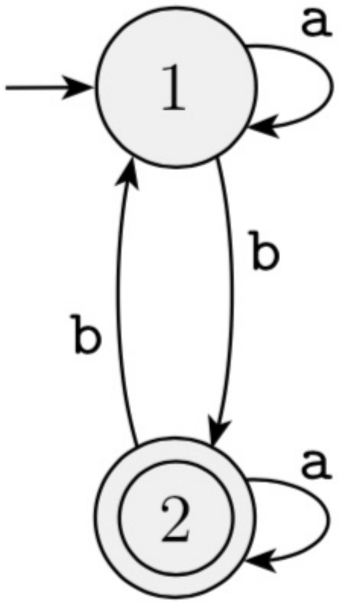
Example:

q_0
q_1
q_2

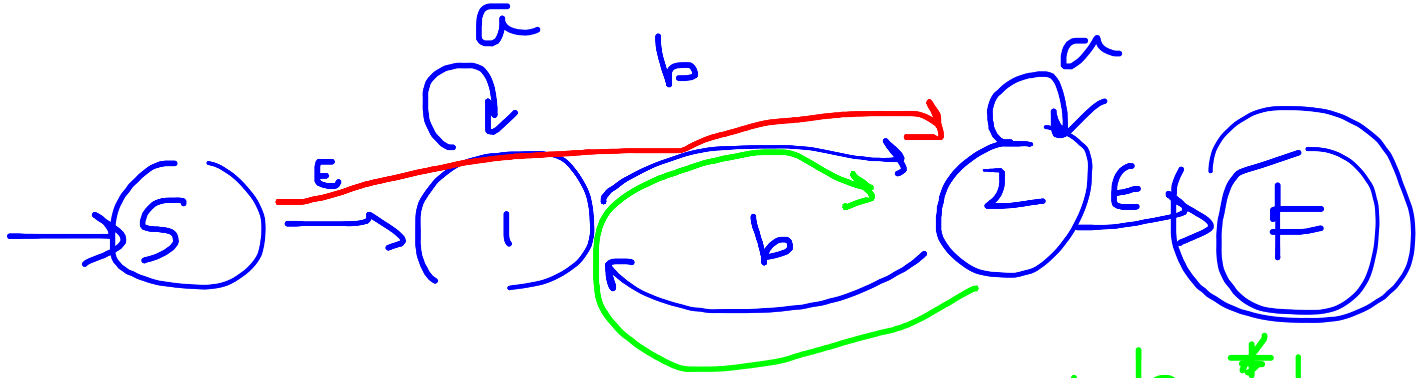


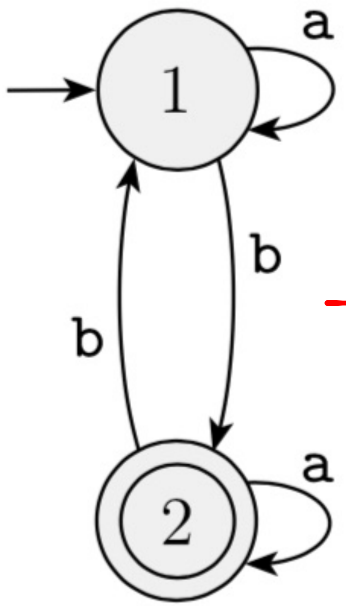


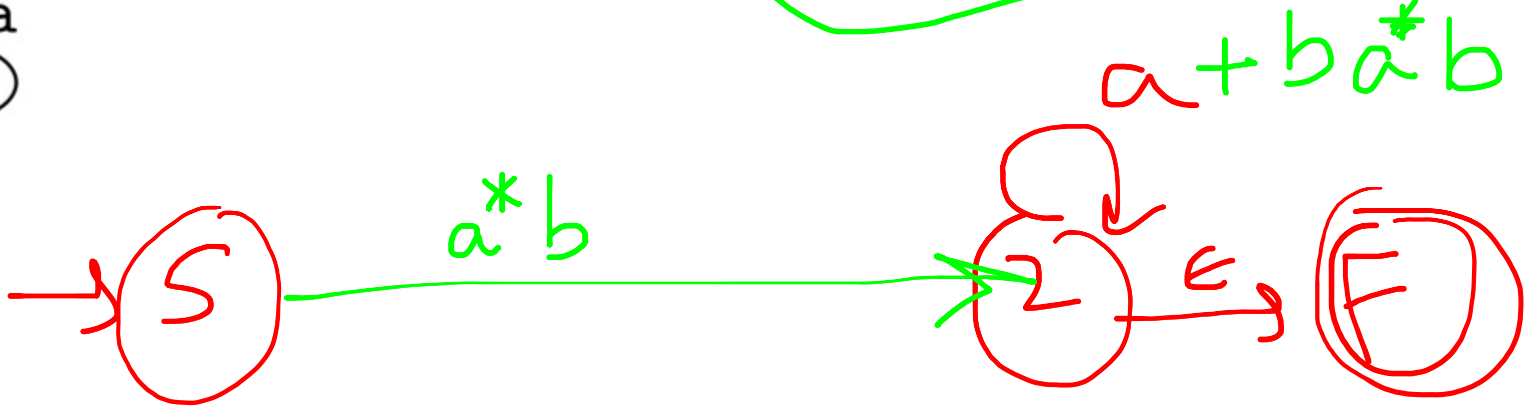
Example

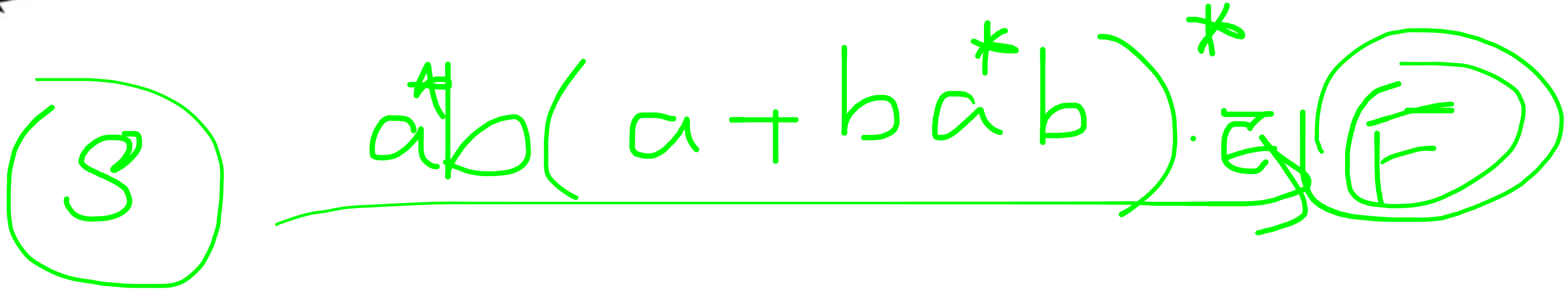


Example









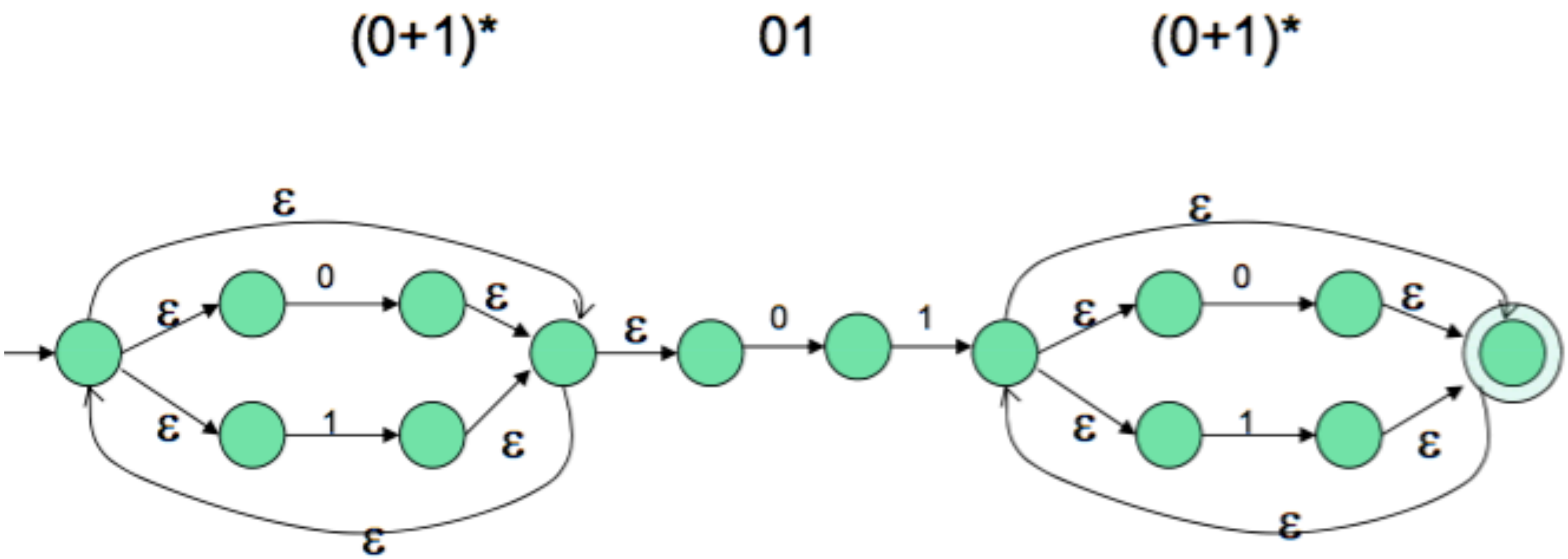
1

2

REGULAR EXPRESSIONS DENOTE FA-RECOGNIZABLE LANGUAGES

RE to ε-NFA construction

<u>Example:</u> (0+1)*01(0+1)*

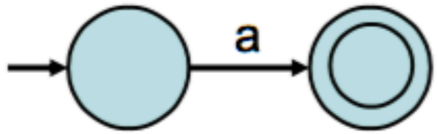


Languages denoted by regular expressions

- The languages denoted by regular expressions are exactly the regular (FA-recognizable) languages.
- Theorem 1: If R is a regular expression, then L(R) is a regular language (recognized by a FA).
  - Proof: Easy.
- Theorem 2: If L is a regular language, then there is a regular expression R with L = L(R).
  - Proof: Harder, more technical.

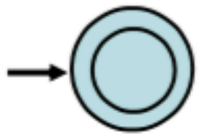
Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language (recognized by a FA).
- Proof:
  - For each R, define an NFA M with L(M) = L(R).
  - Proceed by induction on the structure of R:
    - Show for the three base cases.
    - Show how to construct NFAs for more complex expressions from NFAs for their subexpressions.
  - Case 1: R = a
    - L(R) = { a }



Accepts only a.

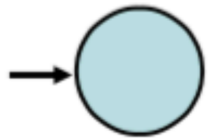
  - Case 2: R = ε
    - L(R) = { ε }



Accepts only ε.

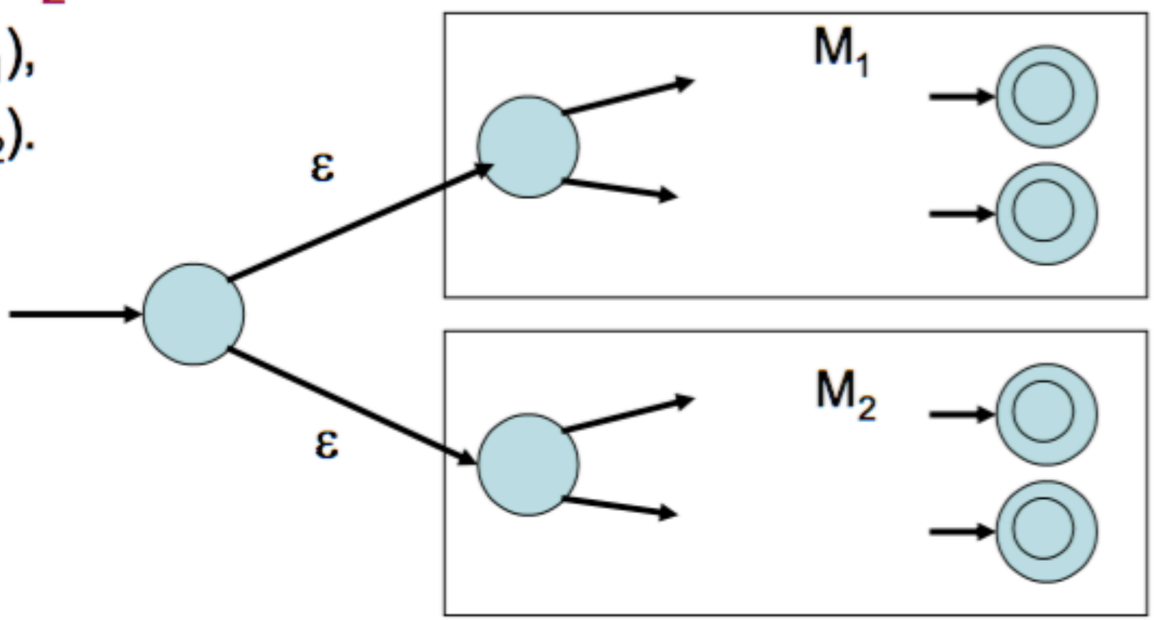
Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language (recognized by a FA).
- Proof:
  - Case 3: $R = \varnothing$
    - $L(R) = \varnothing$



Accepts nothing.

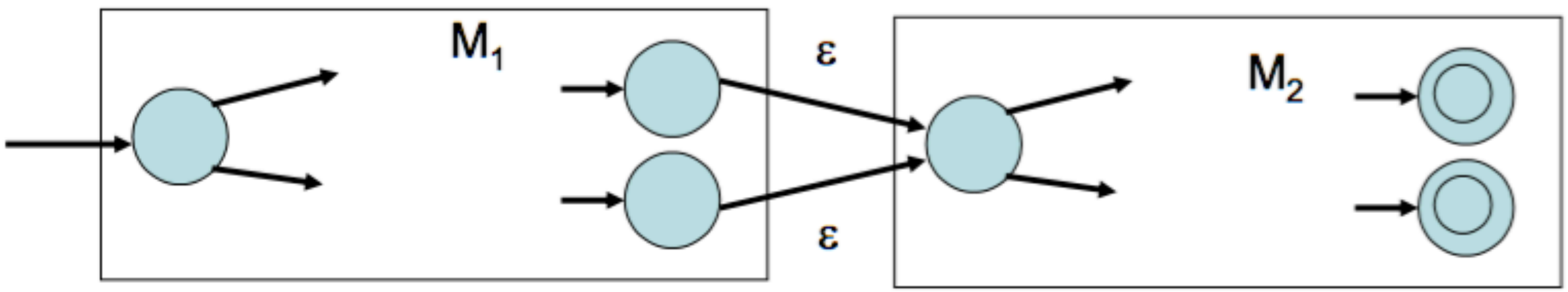
  - Case 4: $R = R_1 \cup R_2$
    - M_1 recognizes $L(R_1)$,
    - M_2 recognizes $L(R_2)$.
    - Same construction we used to show regular languages are closed under union.



Theorem 1

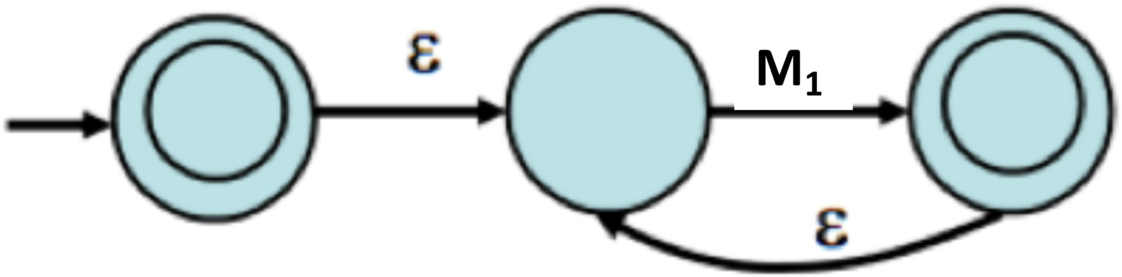
- Theorem 1: If R is a regular expression, then L(R) is a regular language (recognized by a FA).
- Proof:
  - Case 5: $R = R_1 \circ R_2$
    - M_1 recognizes $L(R_1)$,
    - M_2 recognizes $L(R_2)$.

    - Same construction we used to show regular languages are closed under concatenation.



Theorem 1

- Theorem 1: If R is a regular expression, then L(R) is a regular language (recognized by a FA).
- Proof:
  - Case 6: $R = (R_1)^*$
    - M_1 recognizes $L(R_1)$,
    - Same construction we used to show regular languages are closed under star.

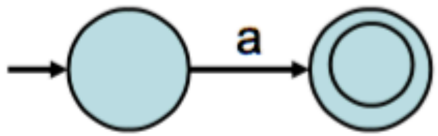


Example for Theorem 1

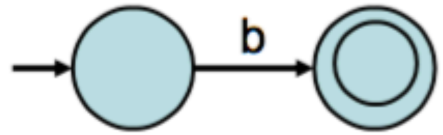
- $L = ab \cup a^*$
- Construct machines recursively:

Example for Theorem 1

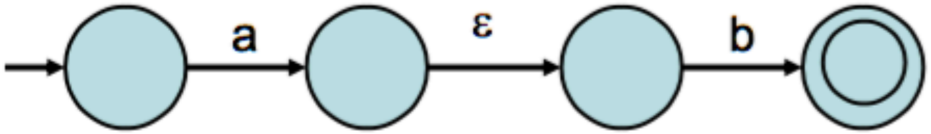
- $L = ab \cup a^*$
- Construct machines recursively:
- a:



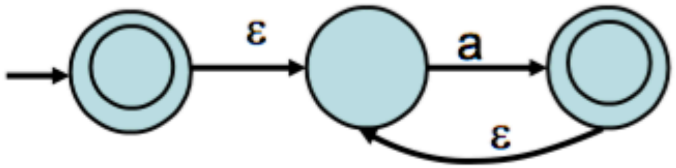
b:



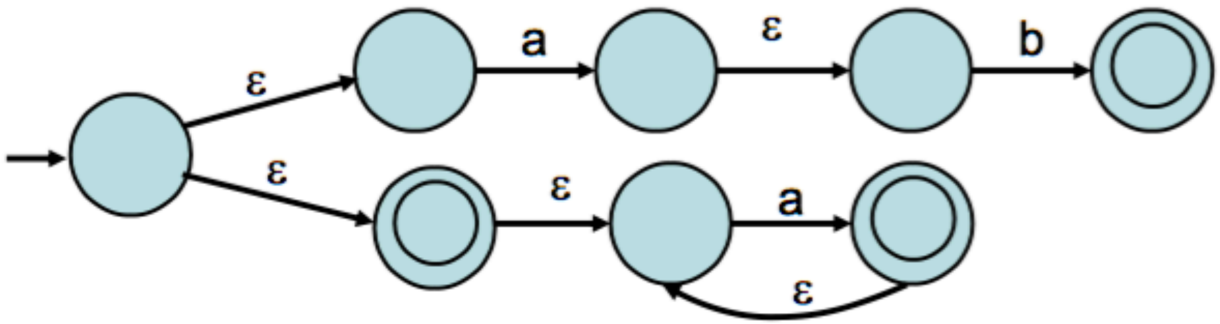
- ab:



- a*:



- ab $\cup$ a*:



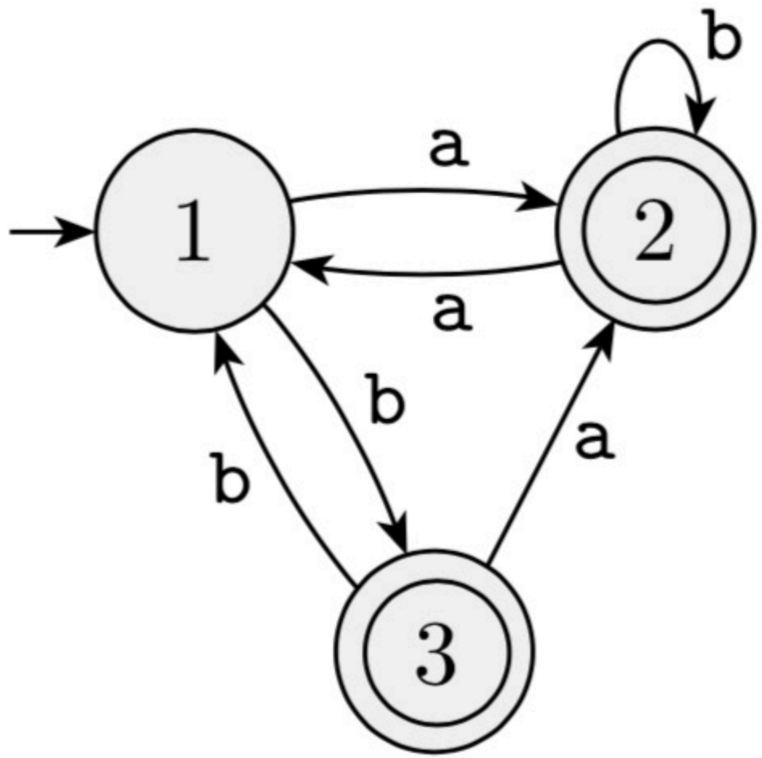
Class Activity

- Convert RE to ε-NFA
  - (0+1)*1(0+1)
  - 01*
  - (0+1)01

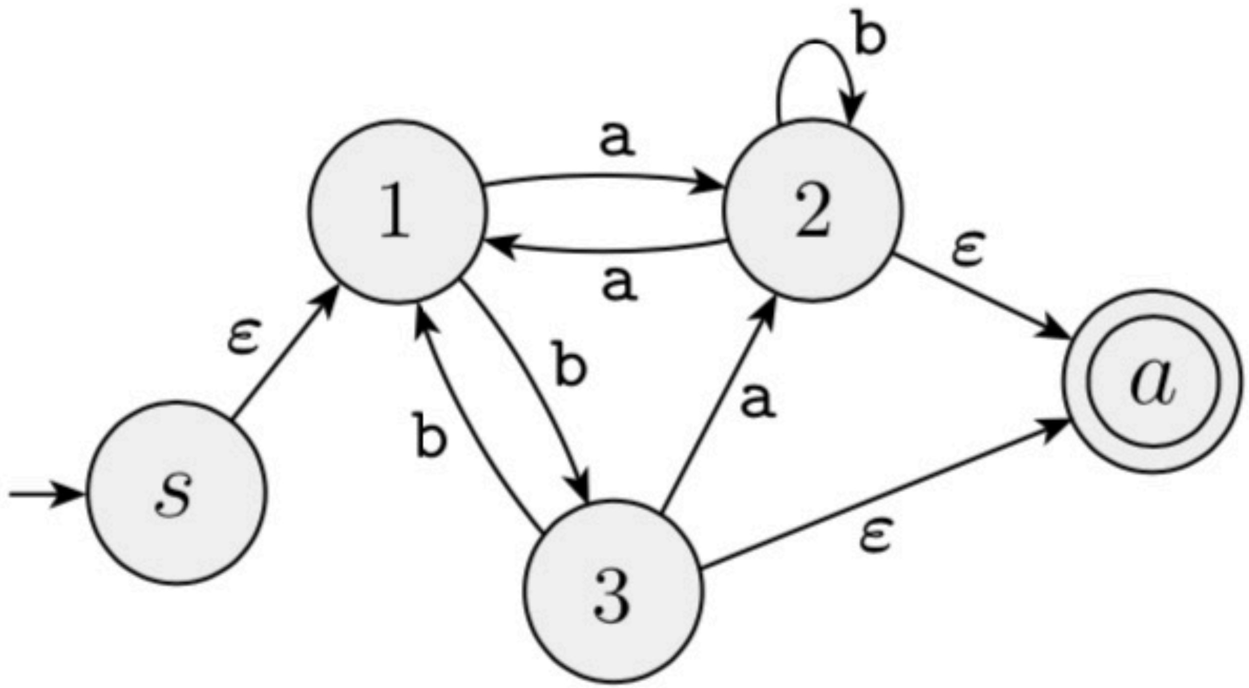
$(0+1)^*1(0+1)$

Class Activity

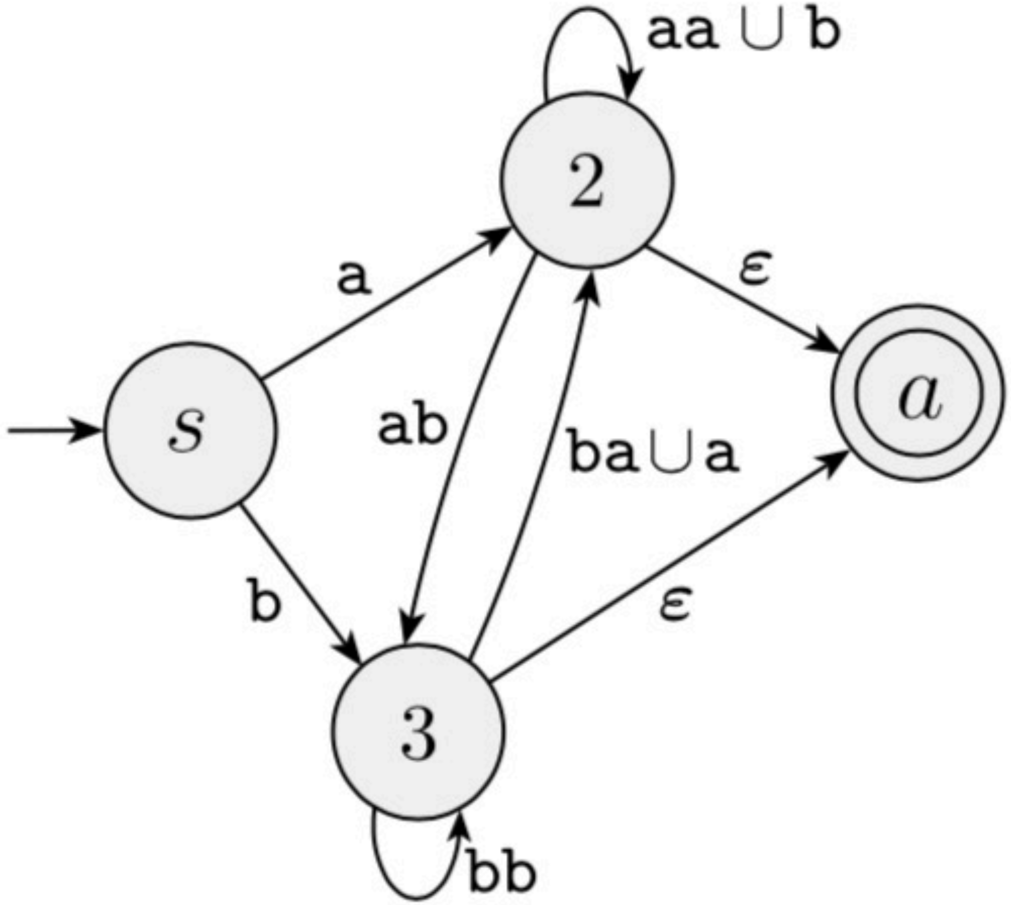
- Convert to RE eliminating the states in the following order
  - 1,2,3



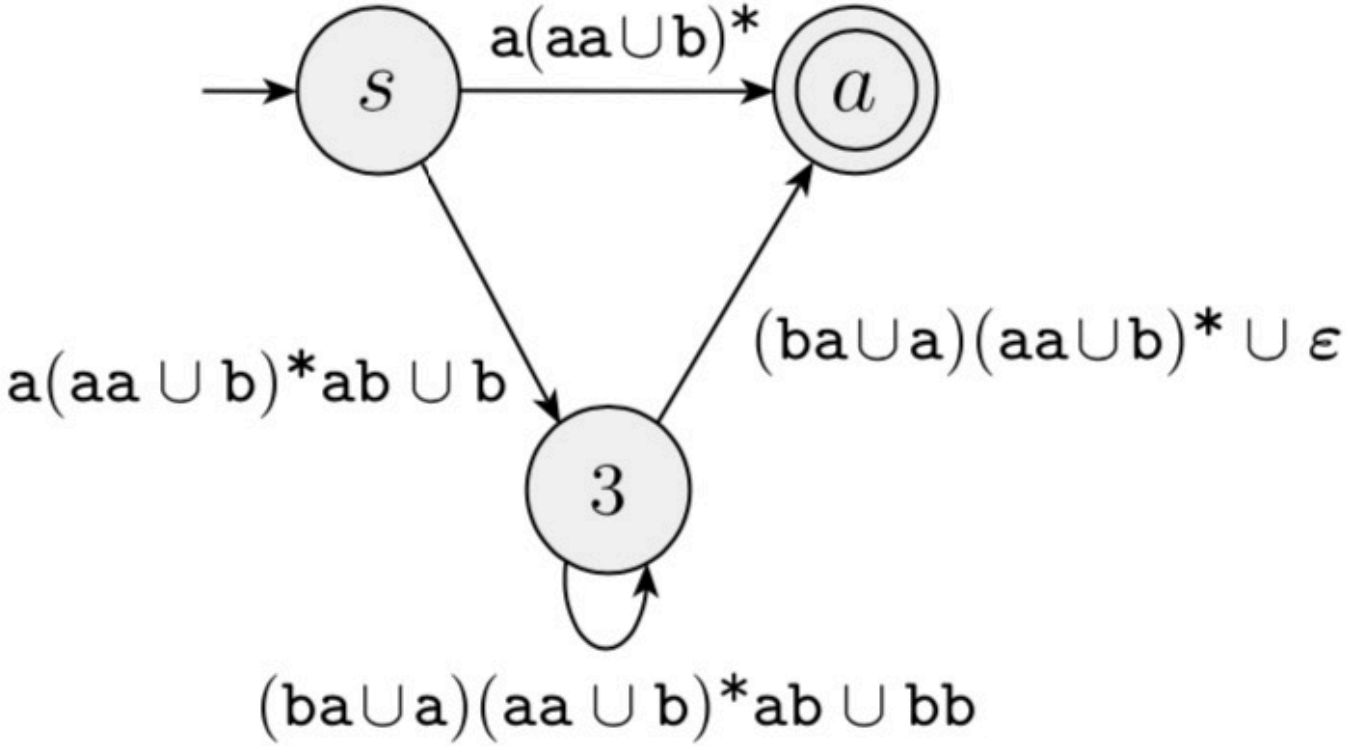
Solution



Solution



Solution



Solution

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

References

- Book Chapter 3
- Lectures from Stanford University
  - http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES
- Lecture by Prof. Nancy Lynch from MIT
- Lectures from Washington State University
  - http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/