Unit3
Regular Expression

By Prashant Gautam

Introduction

- Regular Expression are those algebraic expressions used for representing regular languages, the languages accepted by finite automaton.
- Regular expressions offer a declarative way to express the strings we want to accept. This is what the regular expression offer that the automata do not.
- A regular expression is built up out of simpler regular expression using a set of defining rules. Each regular expression ‘r’ denotes a language L(r).
- The defining rules specify how L(r) is formed by combining in various ways the languages denoted by the sub-expressions of ‘r’.

Let Σ be an alphabet, the regular expression over the alphabet Σ are defined inductively as follows;

-Φ is a regular expression representing empty language.

-Є is a regular expression representing the language of empty strings.

- if ‘a’ is a symbol in Σ, then ‘a’ is a regular expression representing the language {a}.

if ‘r’ and‘s’ are the regular expressions representing the language L(r) and L(s) then

- r U s is a regular expression denoting the language L(r)UL(s).
- r.s is a regular expression denoting the language L(r).L(s).
- r* is a regular expression denoting the language(L(r))*.
- (r) is a regular expression denoting the language (L(r))[L(r)] [defines parenthesis may be placed around regular expressions if we desire].

Regular operator:

Union (U / |):

- If L1 and L2 are any two regular languages then
- L1UL2 ={s | s ε L1, or s ε L2 }

e.g.

- L1 = {00, 11}, L2 = (Є, 10}
- L1UL2 = {Є, 00, 11, 10}

Concatenation (.):

- If L1 and L2 are any two regular languages then, L1.L2 = {l1.l2|l1 ε L1 and l2 ε L2}
- e.g. L1 = {00, 11} and L2 = {Є, 10}

Kleen Closure (*):

- If L is any regular Language then,

$L^* = L_i = \bigcup_{i=0}^{\infty} .L0\ UL1\ UL2\ U\ldots\ldots\ldots\ldots$

Positive closure (+):

- L+= $\bigcup_{i=0}^{\infty}$ Li= L*-L0

Precedence of regular operators

- Closure (*) has highest precedence
- Concatenation (.) has next highest precedence.
- Union (U / | / +) has lowest precedence.

Regular language

- Let Σ be an alphabet, the class of regular language over Σ is defined inductively as;
  - Φ is a regular language representing empty language
  - {Є} is a regular language representing language of empty strings.
  - For each a ε Σ, {a} is a regular language.
  - If L1, L2…… Ln is regular languages, then so is L1U L2U……..ULn.
  - If LI,L2,L3,……..Ln are regular languages, then so is L1.L2.L3……Ln
  - If L is a regular language, then so is L*

Application of regular languages

- **Validation**: Determining that a string complies with a set of formatting
- **constraints**. Like email address validation, password validation etc.
- **Search and Selection**: Identifying a subset of items from a larger set on the basis of a pattern match.
- **Tokenization**: Converting a sequence of characters into words, tokens (like keywords, identifiers) for later interpretation.

Algebraic Rules/laws for regular expression

- **Commutativity**: The union of regular expression is commutative but concatenation of regular expression is not commutative. i.e. if r and l are regular expressions representing like languages L(r) and L(l) then,

$$r+l = l+r \text{ i.e. } r \cup l = l \cup r$$

$$\text{but } r.l \neq l.r$$

- **Associativity**: The unions as well as concatenation of regular expressions are associative.
- i.e. if l, r, s are regular expressions representing regular languages L(l),L(r) and L(s) then,

$$l+(r+s) = (l+r)+s$$

$$\text{And } l.(r.s) = (l.r).s$$

- **Distributive law:** if l, r, s are regular expressions representing regular languages L(l),L(r) and L(s) then,

l(m+n) = lm+ln ------ left distribution

(m+n)l = ml+nl ------ right distribution.

Identity law:

- Φ is identity for union. i.e. for any regular expression r representing regular expression L(r).

r + Φ = Φ + r = r i.e. ΦUr=r

Є is identity for concatenation.

i.e. Є.r = r = r. Є

Annihilator: An annihilator for an operator is a value such that when the operator is applied to the annihilator and some other value, the result is annihilator.

- Φ is annihilator for concatenation.

  i.e. $\Phi.r = r.\Phi = \Phi$

Idempotent law of union:

For any regular expression r representing the regular language

L(r), r + r = r.

This is the idempotent law of union.

Law of closure:

for any regular expression r, representing the regular language L(r),

- (r*)*=r*
- Closure of Φ = Φ* = Є
- Closure of Є = Є* = Є
- Positive closure of r, r+ = rr*.

Practice Session

- **Given the alphabet A = {0, 1}**

1. RE denotes the language of all string that begins with a ‘1’ and ends with a ‘ 0 ’ . ❼

1(1+0)*0

2. RE denotes the language of all strings that ends with 00 (binary number multiple of 4) ❼

(1+0)*00

3. RE denotes the set of all stings that describe alternating 1s and 0s ❼

(01)*+ (10)*

4. RE denotes the string having exactly three 1’s. ❼

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

5. RE denotes the string having at most two 0’s ❼

1*(0+ ε)1*(0+ ε) 1*

6. RE denotes the regular expression to specify the identifier like in C ❼

(A | B | C |……| Z | a | b | c |……| z | _). ((A | B | C |……| Z | a | b | c |……| z | _) (0|1 | 2 |…….| 9))*

7. RE denotes string having substring 0 0 1 ❼

(1+0)* 001 (1+0)*

Finite Automata and Regular Expression:

- The regular expression approach for describing language is fundamentally different from the finite automaton approach. However, these two notations turn out to represent exactly the same set of languages, which we call regular languages.

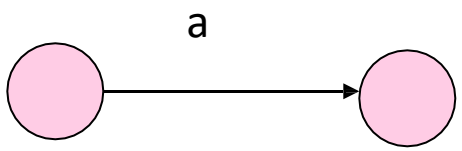
Regular expression to finite automata

Theorem: 2

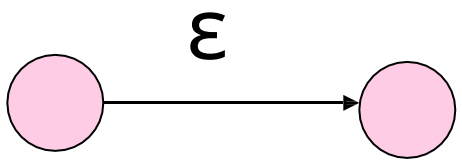
- Every language defined by a regular expression is also defined by a finite automaton.
- [For any regular expression r, there is an Є-NFA that accepts the same language represented by r].

RE to ε-NFA: Basis

- Symbol **a**:

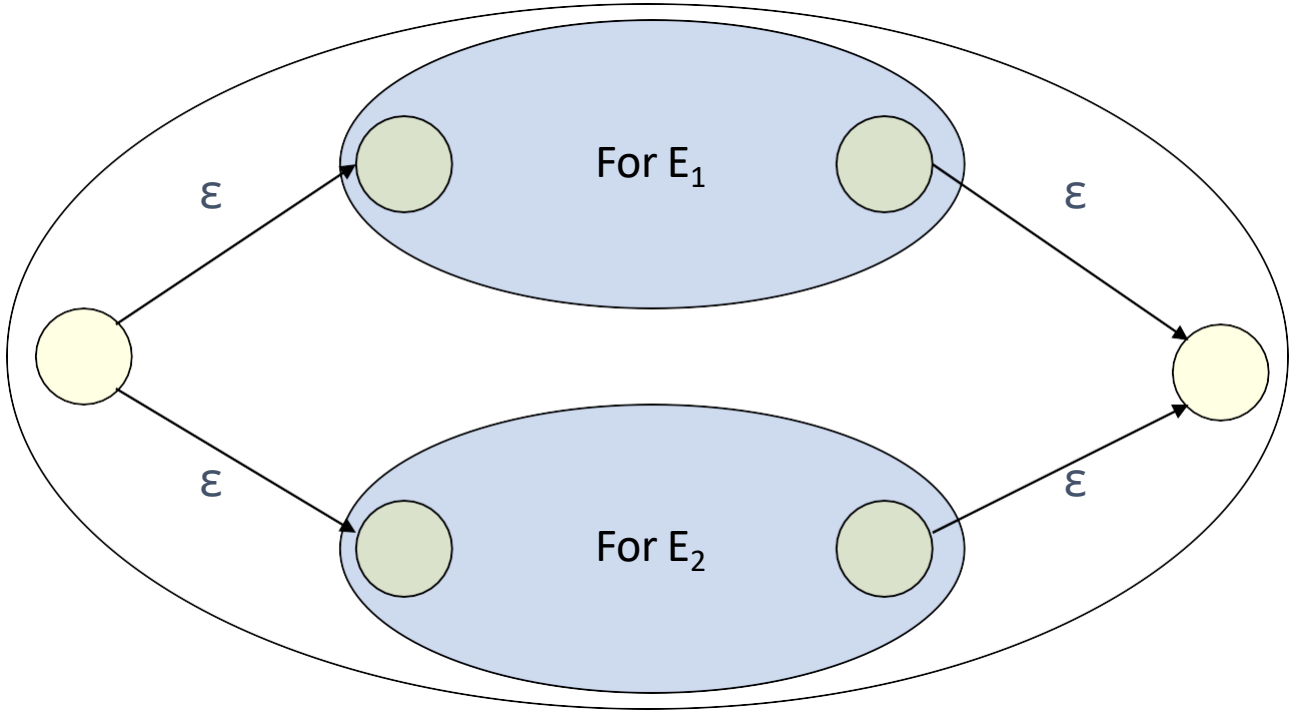


- ε:



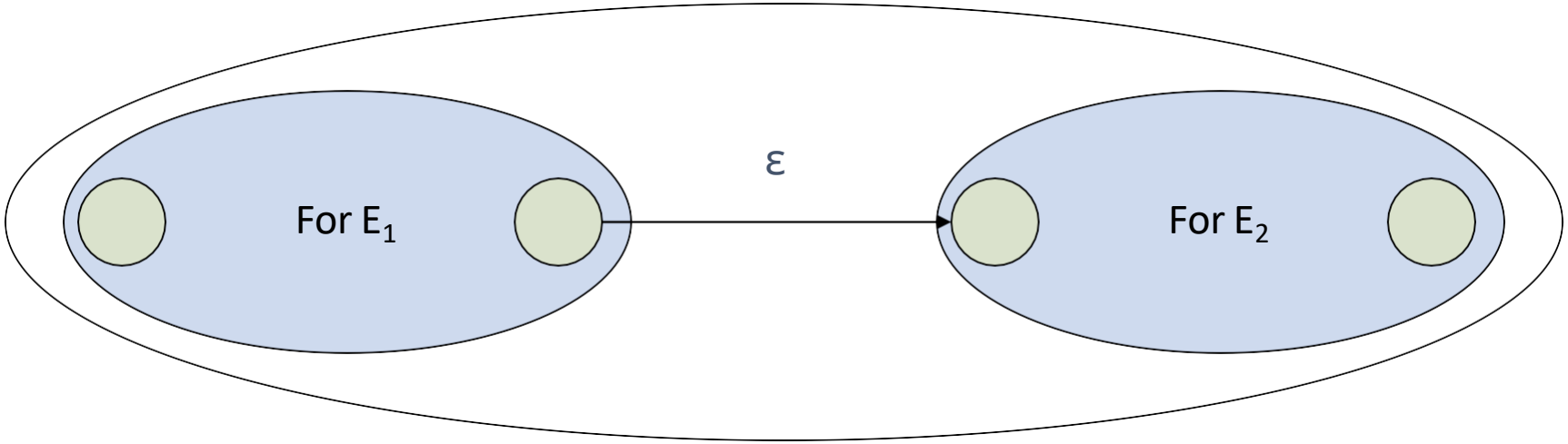
- ∅:

REto ε-NFA: Induction 1 – Union



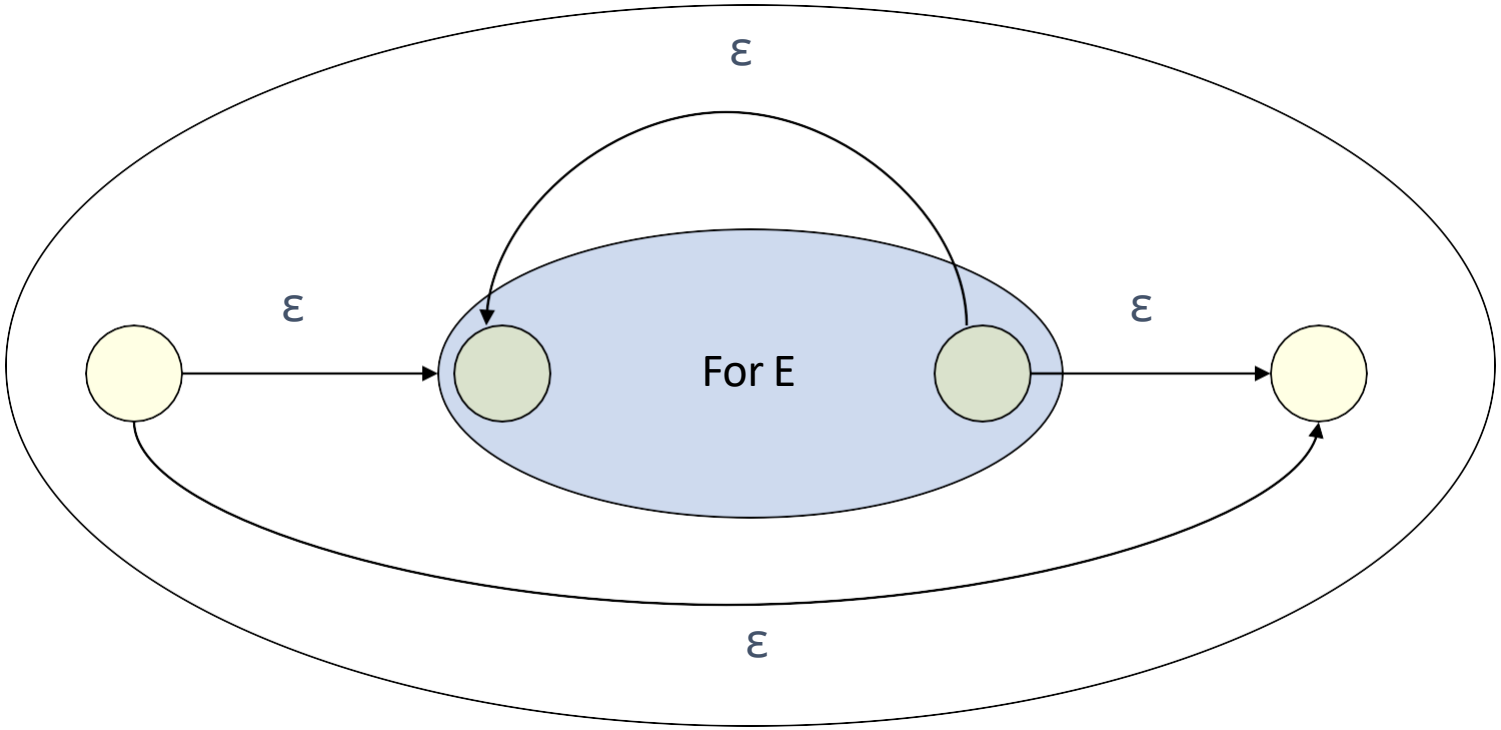
For $E_1 \cup E_2$

RE to ε-NFA: Induction 2 – Concatenation



For E_1E_2

RE to ε-NFA: Induction 3 – Closure



For E*

Conversion: Regular Expression to e-NFA

- Thomson's Construction

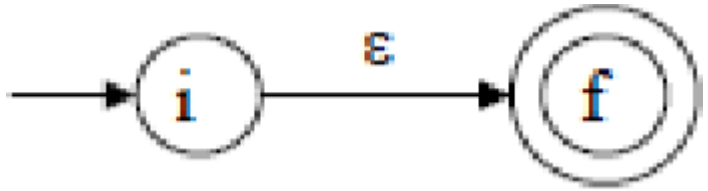
Thomson's Construction is simple and systematic method. It guarantees that the resulting NFA will have exactly one final state, and one start state.

Method:

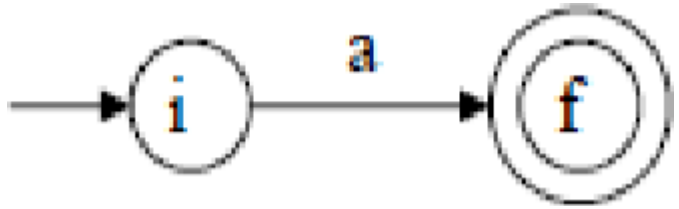
- First parse the regular expression into sub-expressions
- Construct NFA's for each of the basic symbols in regular expression
- Finally combine all NFA's of sub-expressions and we get required NFA of given regular expression.

Rules

1. To recognize an empty string ε

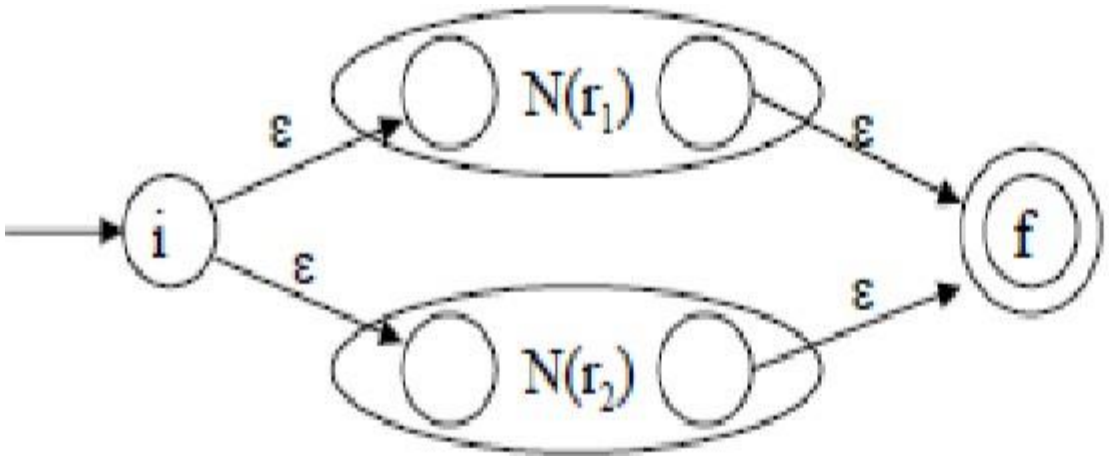


2. To recognize a symbol a in the alphabet Σ



3. If N (r1) and N (r2) are NFAs for regular expressions r1 and r2

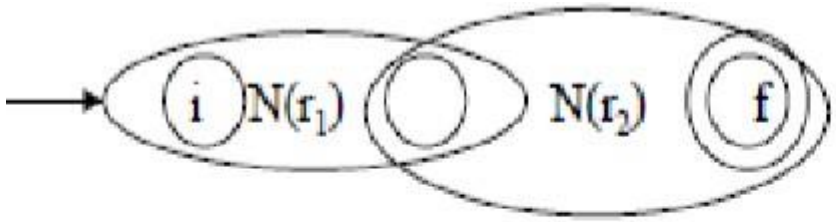
a) For regular expression r1 + r2



$N(r_1 + r_2)$

3.

b) For regular expression r1 r2

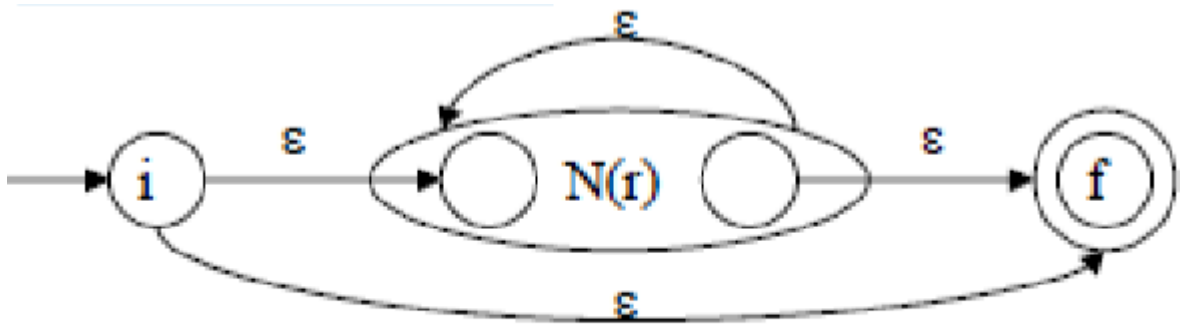


$N(r_1\ r_2)$

The start state of $N(r_1)$ becomes the start state of $N(r_1r_2)$ and final state of $N(r_2)$ become final state of $N(r_1r_2)$

3.

c) For regular expression r*

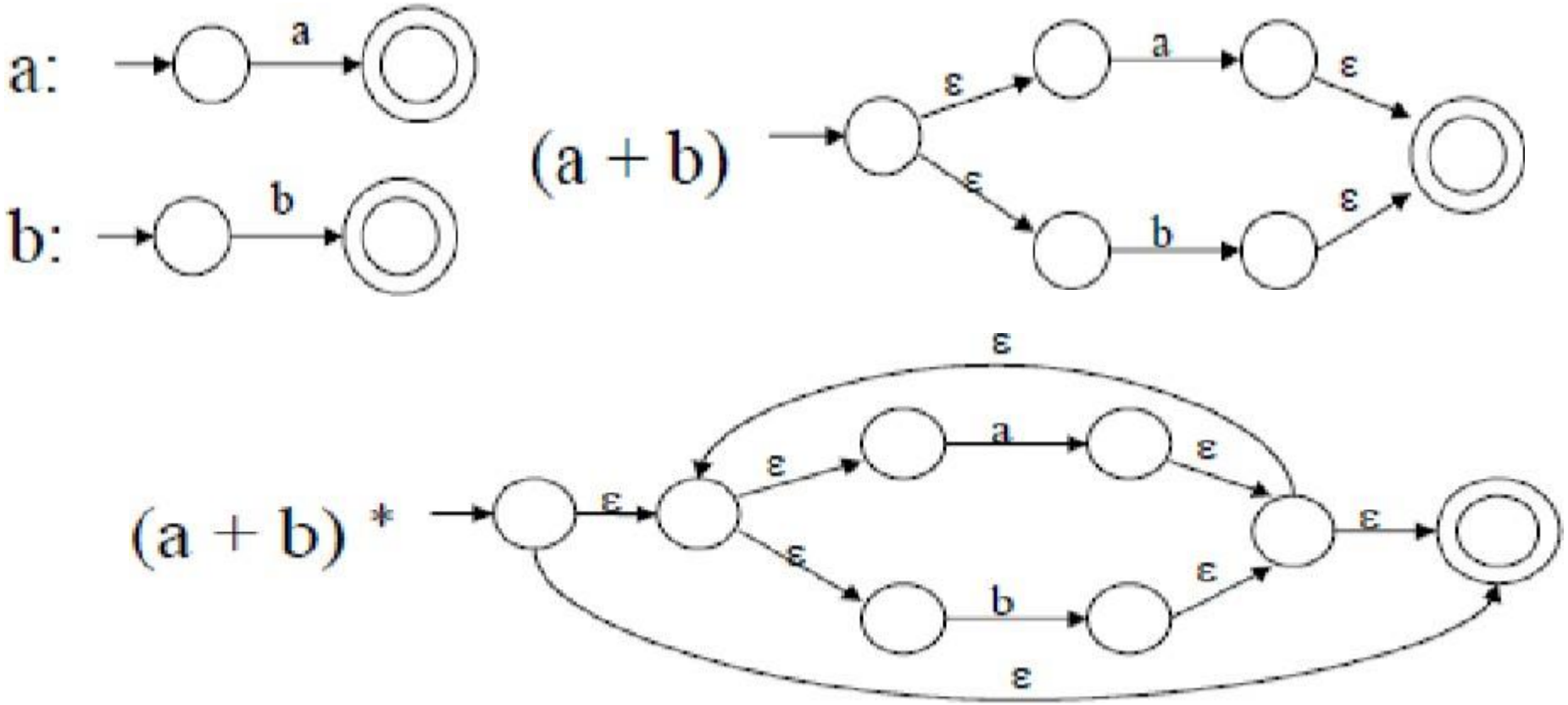


N (r*)

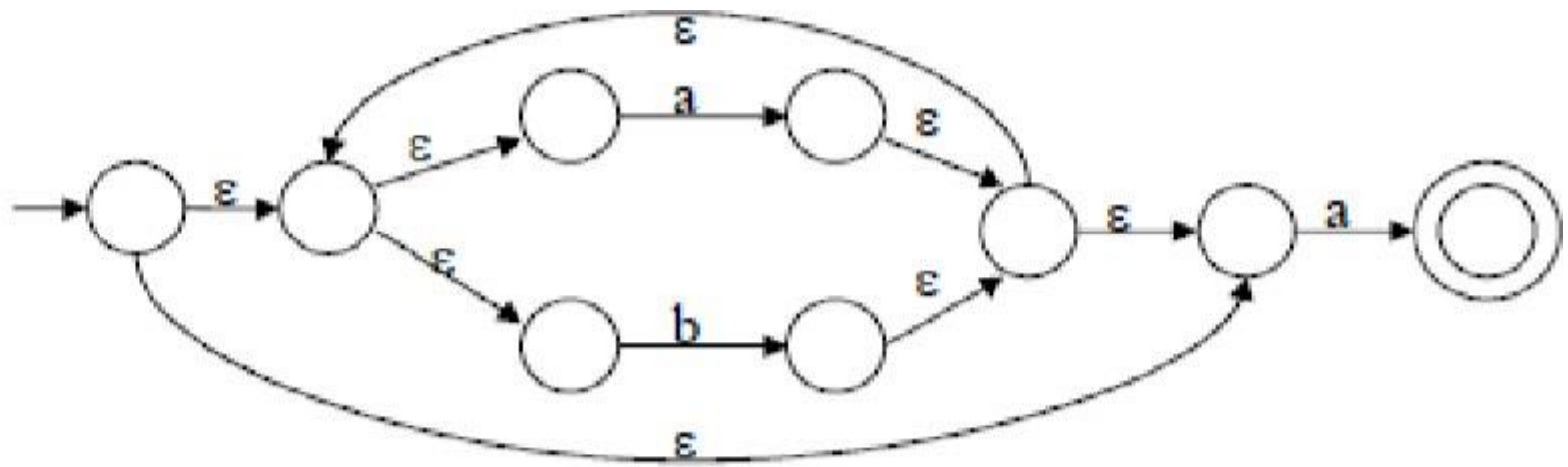
Using rule 1 and 2 we construct NFA's for each basic symbol in the expression, we combine these basic NFA using rule 3 to obtain the NFA for entire expression.

Example

- NFA construction of RE (a + b) * a



(a+b) * a



TRY Yourselves

(ab +a) *

Assignment: RE to e-NFA

- (0+1)*
- (0+1)*1(0+1)
- (00+1)*10
- (00+1)*10
- (00+1)*10
- (00+1)*10

DFA to Regular Expression: Arden's Theorem

- Let's assume that **P** and **Q** be two regular expressions.
- In case if **P** does not include a **ϵ** (null string), then

  R = **Q** + **RP** has a unique solution that is

  R = **QP***

Proof

- $R = Q + (Q + RP)P$

  [After putting the value $R = Q + RP$]

- $R = Q + QP + RPP$
- If you include the value of R recursively repetitively then you will get the following equation –

$$R = Q + QP + QP^2 + QP^3.....$$

$$R = Q\ (\varepsilon + P + P^2 + P^3 + ...\ )$$

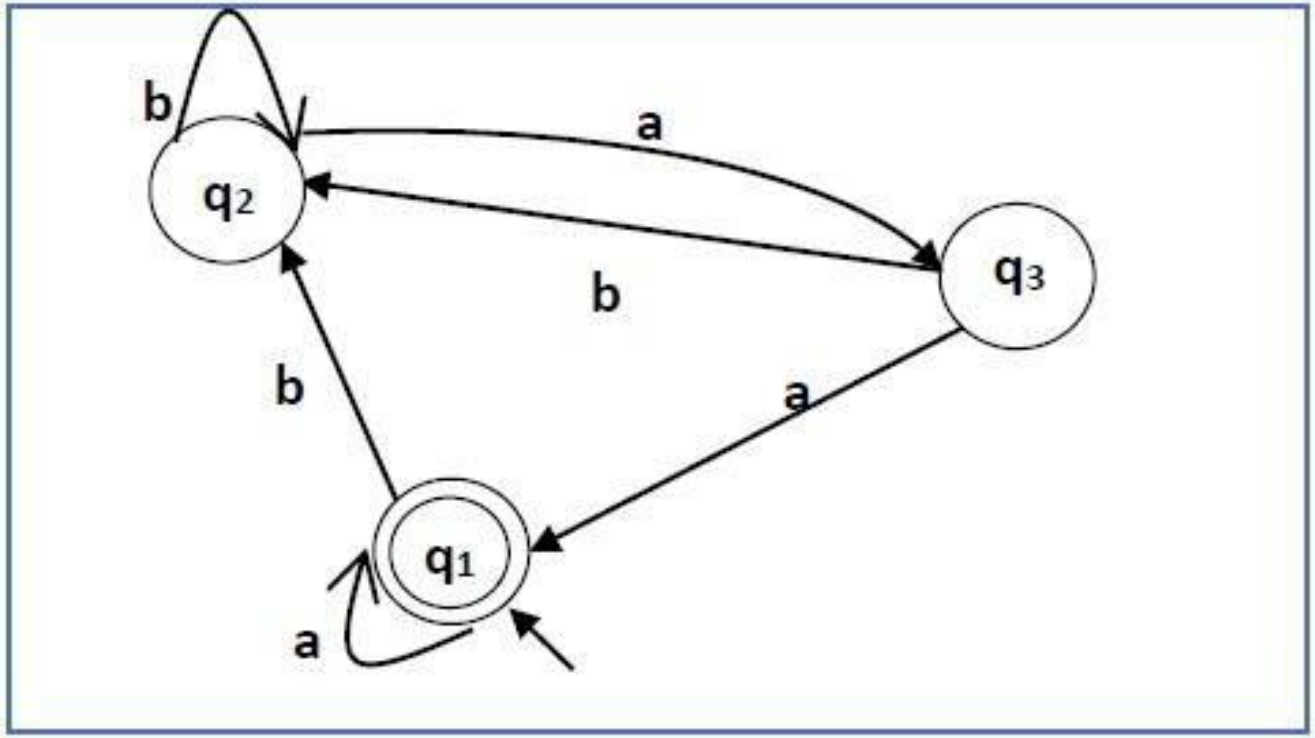
$R = QP^*$ [As P^* represents $(\varepsilon + P + P2 + P3 + ...)$]

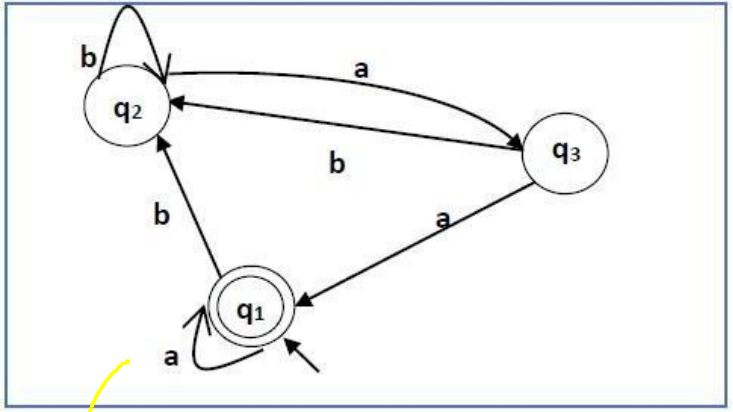
Assumptions for Applying Arden's Theorem

- The transition diagram must not have ϵ transitions

Problem

Construct a regular expression related to the automata given below.





$R = Q + RP$

$R = QP^*$

Solving.

$q_2 = q_1 b + q_2 b + q_3 b$

$q_2 = q_1 b + q_2 b + q_2 ab$ [from (3)]

$\frac{q_2}{R} = \frac{q_1 b}{Q} + \frac{q_2 (b + ab)}{R \quad P}$

$q_2 = q_1 b (b + ab)^*$ — (4)

Now substituting value of q_2 & q_3 in eq (1)

$q_1 = q_1 a + q_2 aa + \epsilon$

$q_1 = q_1 a + q_1 b (b + ab)^* . aa + \epsilon$

$R \quad q_1 = \underset{Q}{\epsilon} + \underset{R}{q_1} [a + b (b + ab)^* . aa]$

$q_1 = \epsilon (a + b (b + ab)^* aa)^*$

$= (a + b (b + ab)^* . aa)^*$

Solution

- Here the initial state is **q2** and the final state is **q2**.
- The equations for the three states q1, q2, and q3 are as follows −

$q_1 = q_1a + q_3a + \varepsilon$ (ε move is because q1 is the initial state.

$q_2 = q_1b + q_2b + q_3b$

$q_3 = q_2a$

Now, we will solve these three equations –

$$q_2 = q_1b + q_2b + q_3b$$

$$= q_1b + q_2b + (q_2a)b \qquad \text{(Substituting value of } q_3\text{)}$$

$$= q_1b + q_2(b + ab)$$

$$= q_1b\,(b + ab)^* \qquad \text{(Applying Arden's Theorem)}$$

$$q_1 = q_1a + q_3a + \varepsilon$$

$$= q_1a + q_2aa + \varepsilon \qquad \text{(Substituting value of } q_3\text{)}$$

$$= q_1a + q_1b(b + ab^*)aa + \varepsilon \qquad \text{(Substituting value of } q_2\text{)}$$

$$= q_1(a + b(b + ab)^*aa) + \varepsilon$$

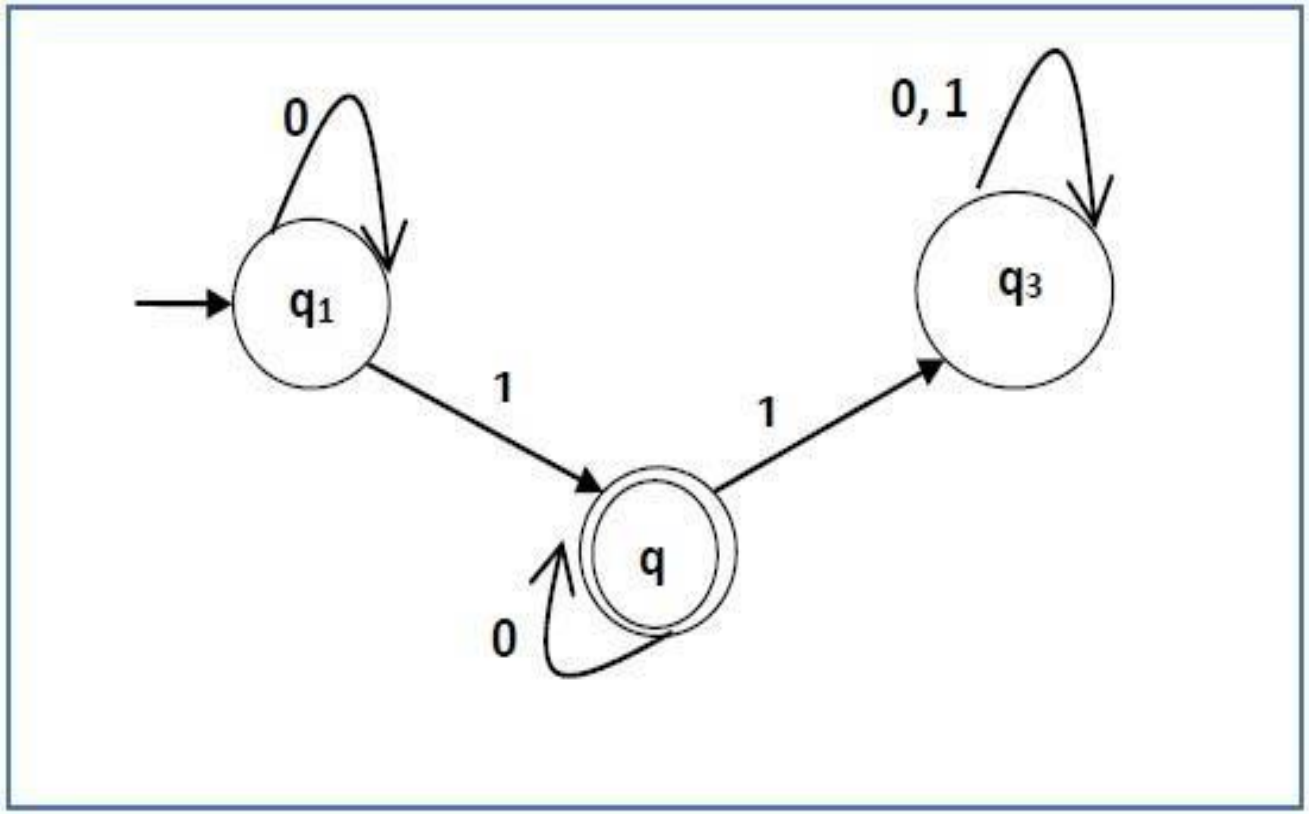
$$= \varepsilon\,(a + b(b + ab)^*aa)^*$$

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

Hence, the regular expression is $(a + b(b + ab)^*aa)^*$.

Problem 2 : Assignment

construct a regular expression related to the automata given below –



Solution

- Here the initial state is q1 and the final state is q2

- Let’s write the below equations –

$$q_1 = q_1 0 + \varepsilon$$

$$q_2 = q_1 1 + q_2 0$$

$$q_3 = q_2 1 + q_3 0 + q_3 1$$

Now, we will solve these three equations –

$$q_1 = \varepsilon 0^* \text{ [As, } \varepsilon R = R]$$

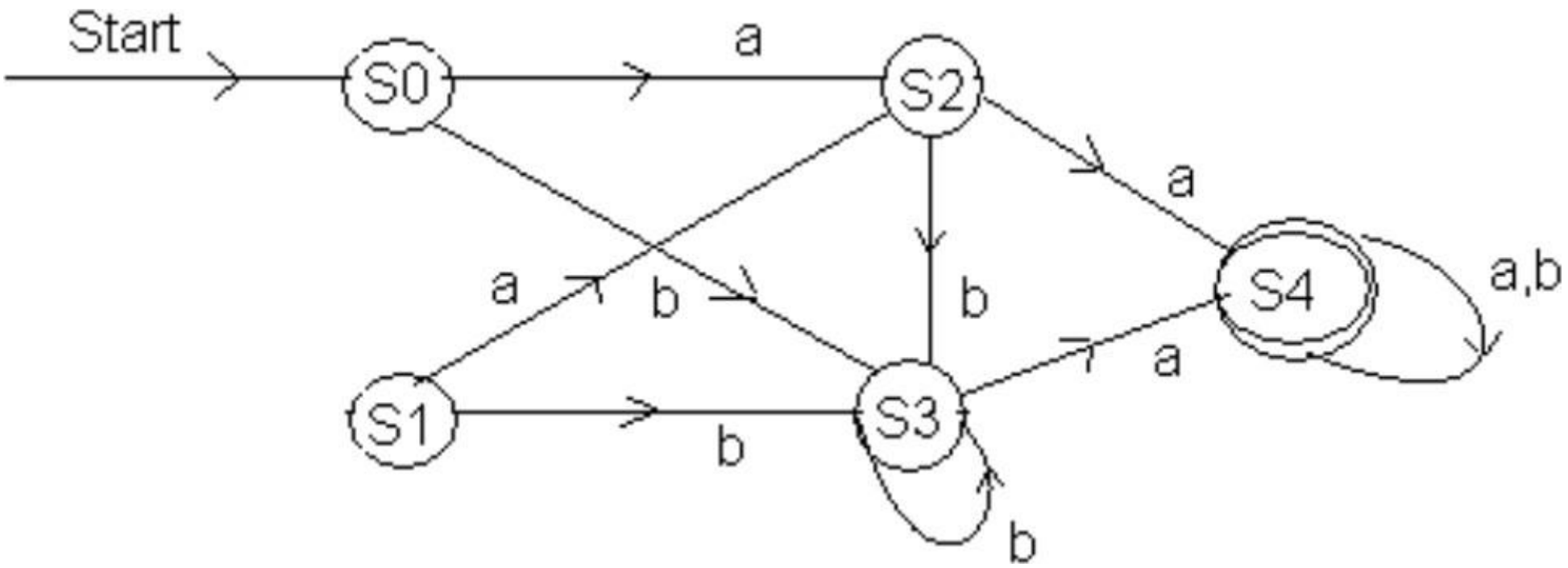
So, $q_1 = 0^*$

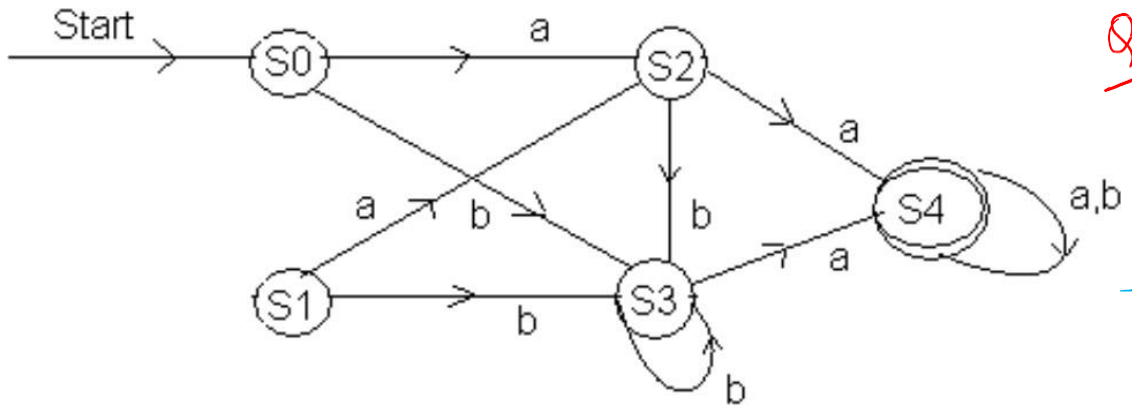
$$q_2 = 0^*1 + q_2 0$$

So, $q_2 = 0^*1(0)^*$ [By Arden's theorem]

Hence, the regular expression is **0*10***

1. DFA State Minimization





Q. Minimize

(reduce the no. of states as far as possible)

1) Any one Acc. state so no need minimize.

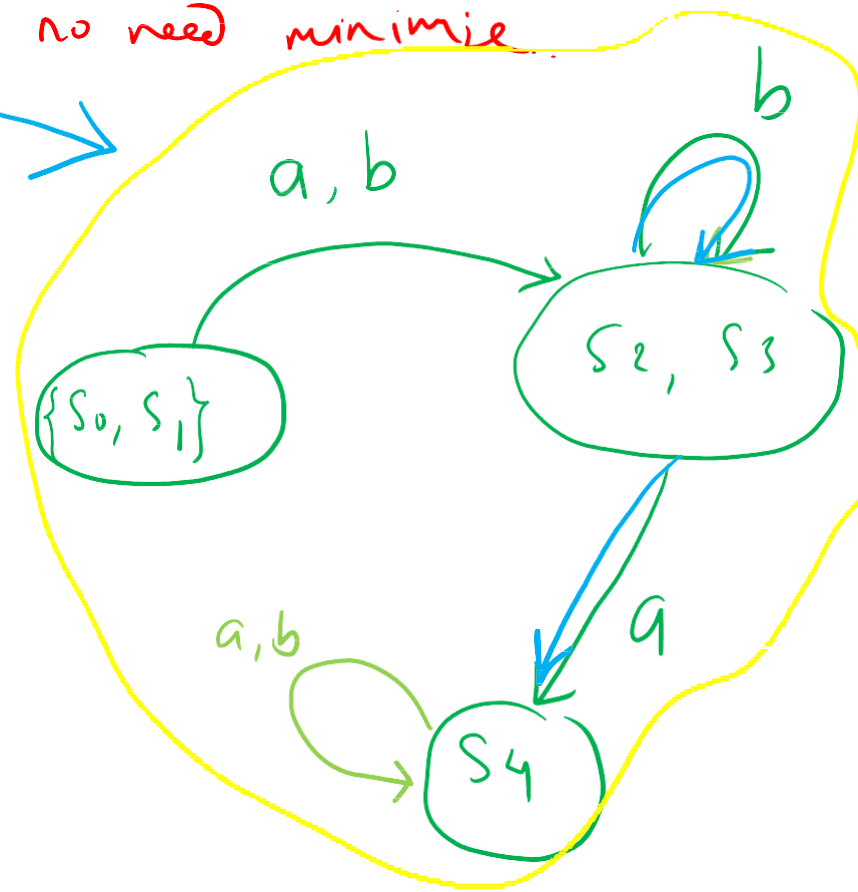
Soln:

Acc. state.

	a	b
S_1	S_2	S_3
S_1	S_2	S_3
S_2	S_4	S_3
S_3	S_4	S_3

	a	b
S_0, S_1	$S_2, 3$	$S_3, 2$
S_2, S_3	S_4	$S_3, 2$

	a	b
S_4	S_4	S_4



2. DFA State Minimization (Assignment)