5- Equivalence of Finite Automata and Regular Expressions

5.1 Converting Regular Expression to Finite Automata

Kleene's Theorem:

For any Regular Expression R that represents Language L(R), there is a Finite Automata that accepts same language.

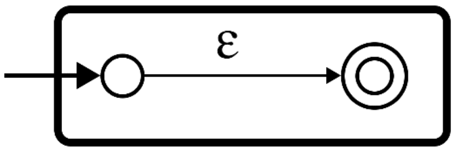
PROOF: Suppose L = L(R) for some regular expression R, we show that L = L (E) for some ε - NFA E with:

1. Exactly one accepting state.
2. No arcs (transitions) into the initial or start state.
3. No arcs (transitions) out of the final or accepting state.

The proof is by structural induction on R.

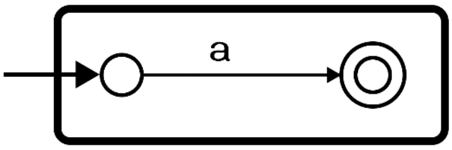
Basis:

1. ε Is a regular expression. **ε - NFA** for ε is



2. $\emptyset$ Is a regular expression. **ε - NFA** for $\emptyset$ is

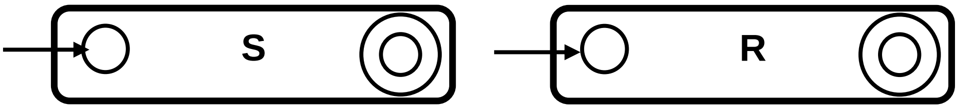
3. **a** is a regular expression. ***ε*** **- NFA** for **a** is



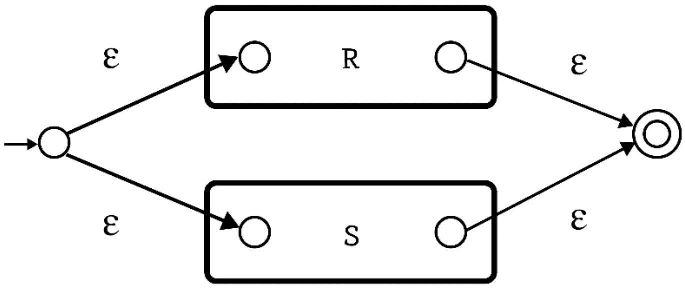
Induction:

R and S are regular expressions.

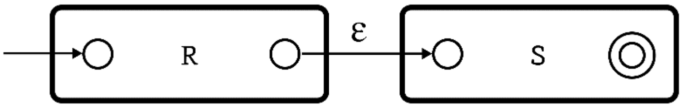
The automata for R and S are



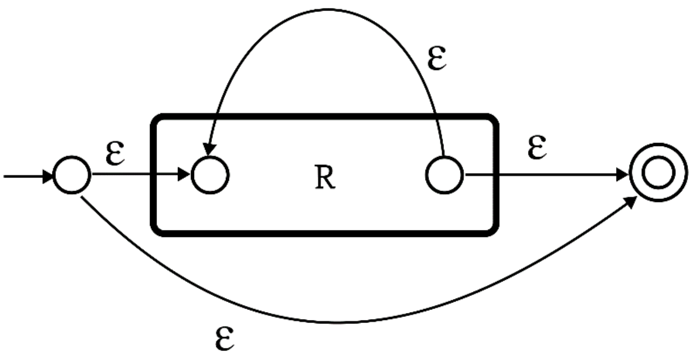
1. The automata for **R + S** is



2. The automata for **RS** is



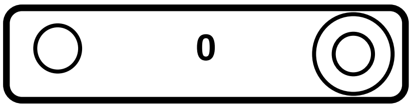
3. The automata for **R*** is



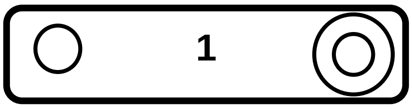
4. The automata for **R** and **(R)** are same.

Ex: Convert the Regular Expression **(0 + 1)*011** to its equivalent Finite Automaton.

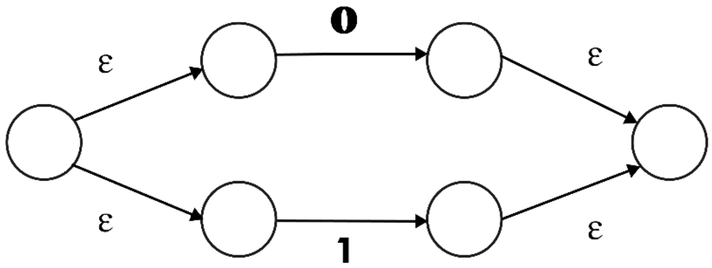
a. The automaton for **0** is:



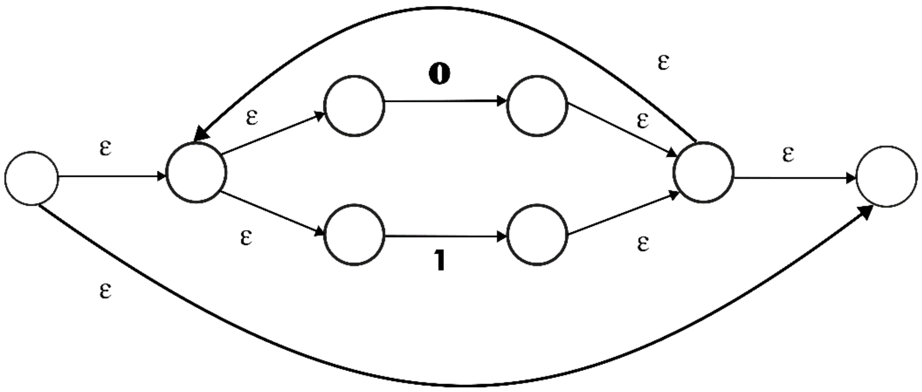
b. The automaton for **1** is:



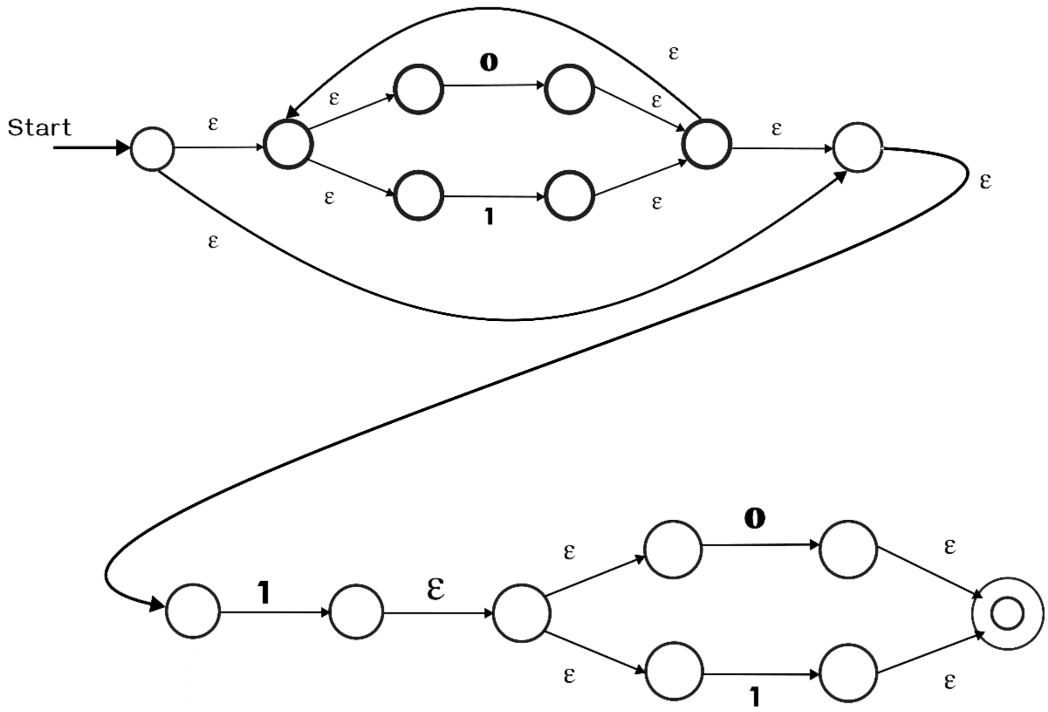
c. The automaton for **(0 + 1)** is:



d. The automaton for **(0 + 1)*** is:



e. The automaton for **(0 + 1)*011** is:



5.2 DFA to Regular Expression

5.2.1 To build expressions that label paths in the DFA.

General Description

- Start with elementary expressions i.e., expressions for paths not passing through any state.
- Build expressions for the paths going through larger sets of states (Incrementally).
- Finally paths may go through any state.

The Method:

- Let the states of the DFA be numbered from 1 to n (n is the number of states).
- Let R^k_{ij} = Regular Expression for the path from state i to state j. Intermediate states on this path must be <= k.

Inductive Definition:

Basis: k = 0 i.e., no intermediate states on the path.

Case 1: if i ≠ j

- If there is no transition from state i to state j then $R_{ij}^0 = \emptyset$.
- If there is a transition from state i to state j on a, then R_{ij}^0 = a.
- If there are transitions from state i to state j on $a_1, a_2, \ldots, a_k$ then $R_{ij}^0 = a_1 + a_2 + \ldots a_k$.

Case 2: if i = j

- If there is no transition from state i to state j then $R_{ij}^0 = \varepsilon$.
- If there is a transition from state i to state j on a, then $R_{ij}^0 = \varepsilon$ + a.
- If there are transitions from state i to state j on $a_1, a_2, \ldots, a_k$ then $R_{ij}^0 = \varepsilon + a_1 + a_2 + \ldots a_k$.

Induction: **k > 0**

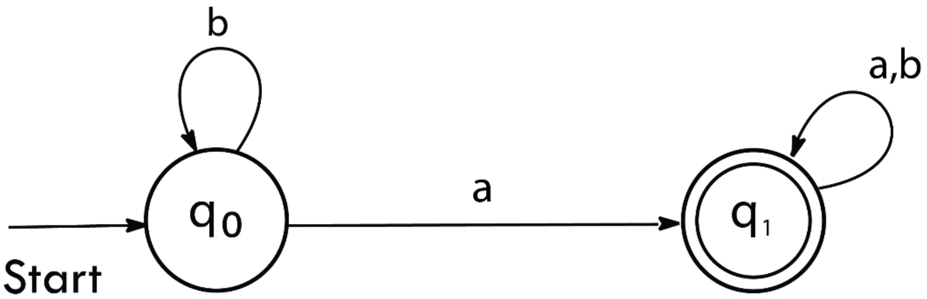
Case 1: The path from state i to state j does not go through k, then $R_{ij}^{k} = R_{ij}^{k-1}$.

Case 2: The path from state i to state j goes through k, then

$$\boldsymbol{R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1}}$$

1. We need to construct these expressions in the increasing order of k.
2. Finally we get R_{ij}^{n} for all i and j.
3. The Regular Expression equivalent to the DFA is union of $\boldsymbol{R_{1j}^{n}}$ for all **j** ∈ **F**.

Ex: Convert the below DFA to its equivalent Regular Expression.



For k = 0

R_{11}^{0}	ε + 1
R_{12}^{0}	0
R_{21}^{0}	ø
R_{22}^{0}	ε + 1

For k = 1

	By Direct Substitution	Simplified
R_{11}^{1}	ε + 1 + (ε + 1) (ε + 1)* (ε + 1)	1*
R_{12}^{1}	0 + (ε + 1) (ε + 1)*0	1*0
R_{21}^{1}	Ø + Ø (ε + 1)* (ε + 1)	Ø
R_{22}^{0}	ε + 0 + 1 + Ø(ε + 1)*0	ε + 0 + 1

For k = 2

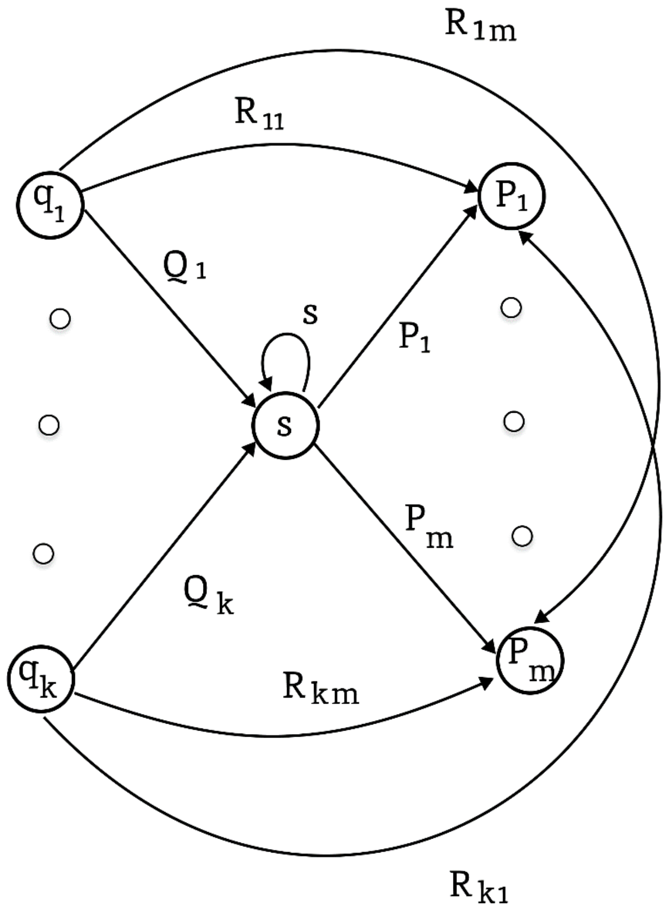
	By Direct Substitution	Simplified
R_{11}^2	1* + 1*0 + (ε + 0 + 1) Ø	1*
R_{12}^2	1* + 1*0 + (ε + 0 + 1)*(ε + 0 + 1)	1*0(0+1)*
R_{21}^2	Ø + (ε + 0 + 1) (ε + 0 + 1)* Ø	Ø
R_{22}^2	ε + 0 + 1 + (ε + 0 + 1)*(ε + 0 + 1)(ε + 0 + 1)	ε + 0 + 1

In this DFA 1 is the start state and 2 is the final state and the DFA has only two states.

Therefore the Regular Expression for the DFA is R_{12}^2 = 1*0(0+1)*

5.2.2 DFA to Regular Expression (State Elimination Method)

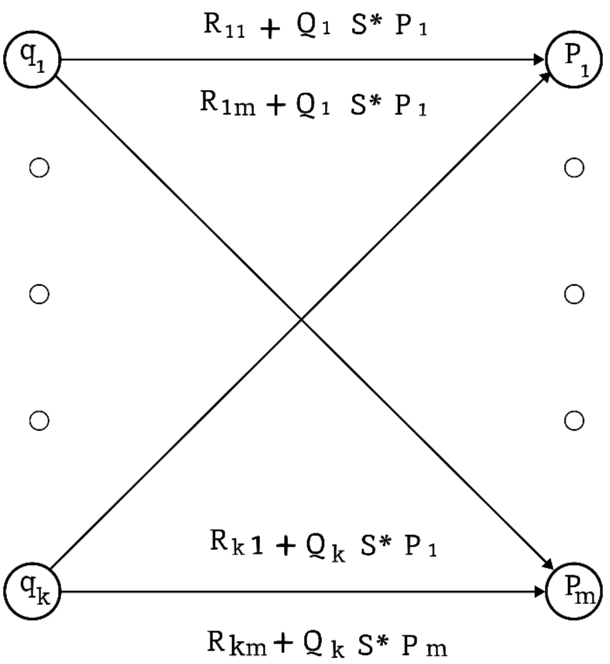
- This method involves eliminating intermediate states on the paths.
- Let **s** be an intermediate state on the path from state **q** to state **p**.
- To eliminate states **s** the transition from state **q** to state **p** must include the regular expression for the path from state **q** to state **s** and state **s** to state **p.**



State **s** to be eliminated.

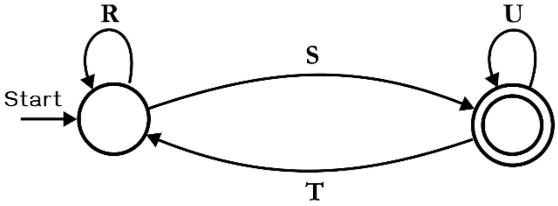
- **q_1, q_2, . . . , q_k** predecessors of **s**.
- **p_1, p_2, . . . , p_m** successors of **s**.
- **Q_i** is the RE for the path from **q_i** to **s**.
- **P_j** is the RE for the path from **s** to **p_j**.
- **R_{ij}** is the RE for the path from **q_i** to **p_j**

After Eliminating state **s**



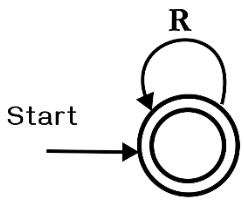
The Method:

- If a state p is not on the path from the start state to final state ignore it.
- For each q ∈ F, eliminate all the states s on the path from the start state to q.
- Label the transitions with Regular Expressions.
- The resulting DFA has only the start state q_0 and the accepting state q.
- If $q \neq q_0$ then the DFA looks like



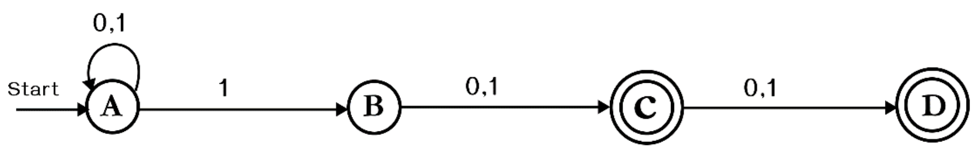
- Regular Expression for the above two state DFA is (R + SU*T)*SU* .

- If $q = q_0$ then the DFA looks like

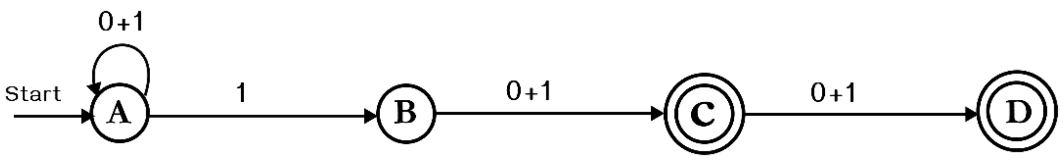


- Regular Expression for the above DFA is R*.
- The Regular Expression for the DFA is the union of Regular expressions obtained for each final state q.

Ex: Convert the below DFA to its equivalent Regular Expression.

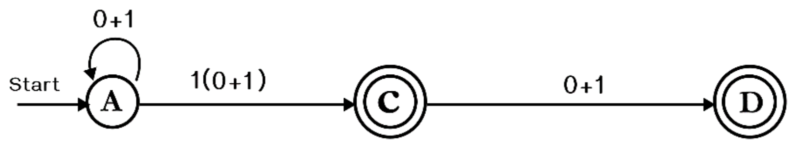


Relabelling the transitions with Regular Expressions

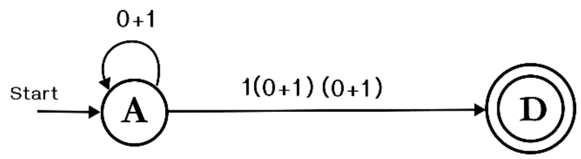


Eliminating the intermediate states on the path from A to D.

Eliminate B, the machine looks like



Eliminate C, the machine looks like



Only two states are left, the start and the final states.

$R = 0 + 1 \qquad S = 1(0+1)(0+1) \qquad U = \varepsilon \quad T = \emptyset$

The Regular Expression (R + SU*T)*SU* after substitution is

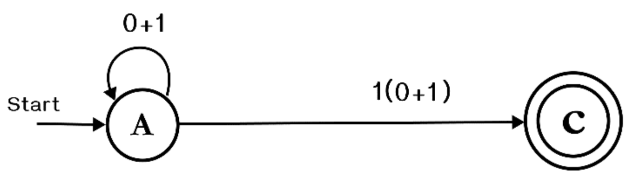
$((0+1) + 1(0+1)(0+1))^* \, 1(0+1)(0+1) \, \varepsilon^*$

After simplification

$\mathbf{R_1 = (0 + 1)^*1(0 + 1)(0 + 1)}$

Eliminating the intermediate states on the path from A to C.

Ignore D and Eliminate B, the machine looks like



Only two states are left, the start and the final states.

$R = 0 + 1 \qquad S = 1(0+1) \quad U = \varepsilon \quad T = \emptyset$

The Regular Expression (R + SU*T)*SU* after substitution is

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

After simplification

$$\mathbf{R_2 = (0 + 1)^*1(0 + 1)}$$