Non Deterministic Finite Automata

THEOREM 1.26

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

Non Deterministic Finite Automata

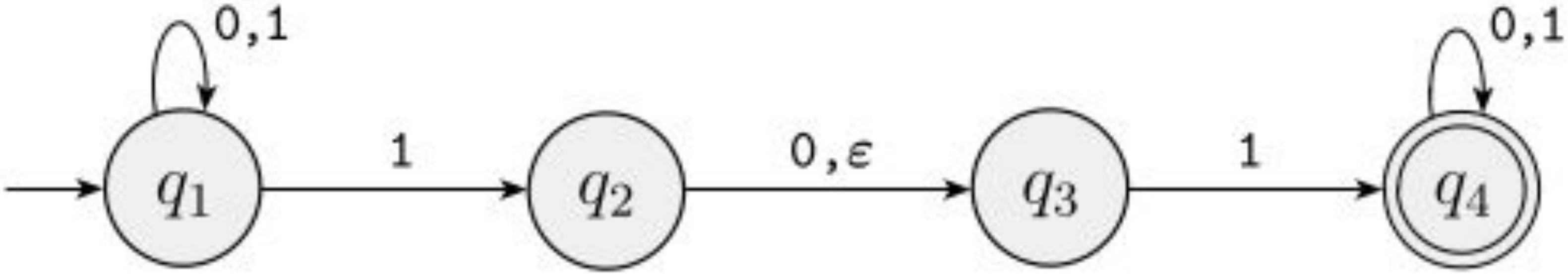
- ➔ ***Recap:*** *An automaton M to accept its input if either M1 or M2 accept for union operation.*
- ➔ *For concatenation, it(M) must accept a string, if its input can be broken into two pieces, where M1 accepts the first piece and M2 accepts the second piece.*
- ➔ *The problem is that M doesn't know where to break its input (i.e., where the first part ends and the second begins).*
- ➔ *Here comes the need of nondeterminism*

Non Deterministic Finite Automata

- *Nondeterminism is a generalization of determinism.*
- *Every nondeterministic finite automaton is equivalent to a deterministic finite automaton.*
- *Easy to understand and design any automaton*

Non Deterministic Finite Automata

➔ *Example of NFA, N_1*

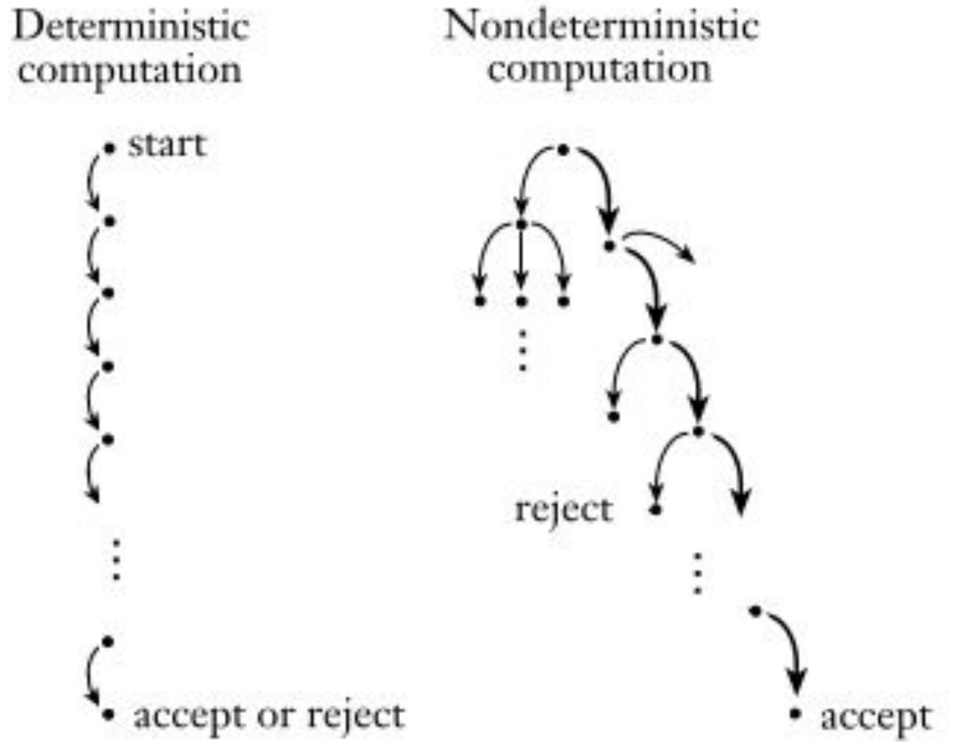


Non Deterministic Finite Automata

- ➔ *difference between a DFA and a NFA?*
  - ✓ *Outgoing arc*
  - ✓ *Extra symbol, ε*
- ➔ *How does an NFA compute?*
  - ✓ *possibilities in parallel thread*
  - ✓ *Split for ε transition*
  - ✓ *Thread dies*

Non Deterministic Finite Automata

➔ *How does an NFA compute?*

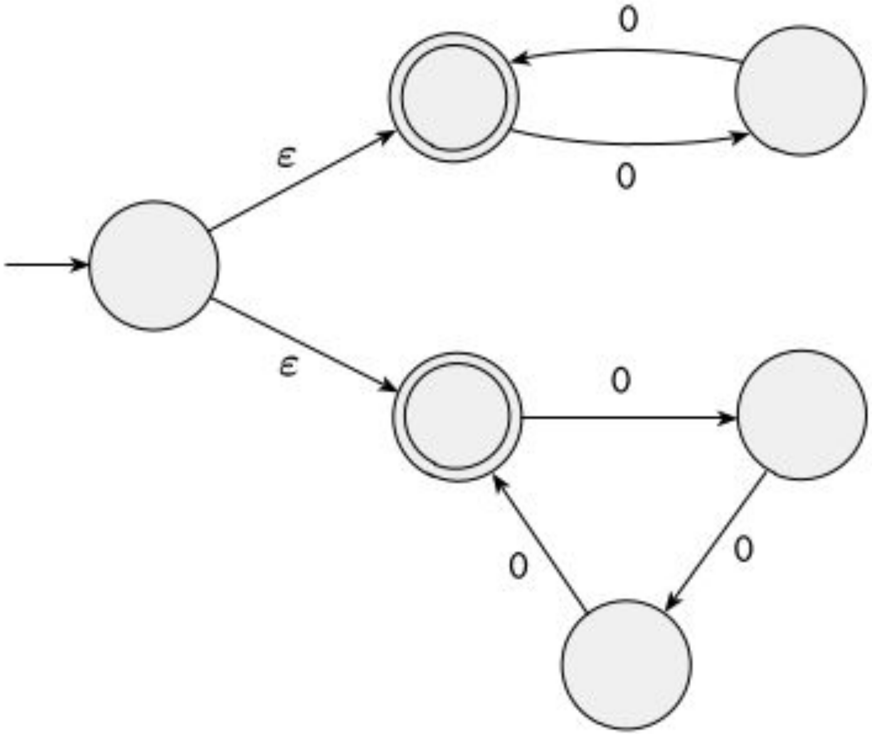


Non Deterministic Finite Automata

➔ *Example: Lets see Computation of* N_1 *on input 010110*

Non Deterministic Finite Automata

- ➔ *This NFA N_3 has an input alphabet {0} consisting of a single symbol. An alphabet containing only one symbol is called a* ***unary*** *alphabet.*



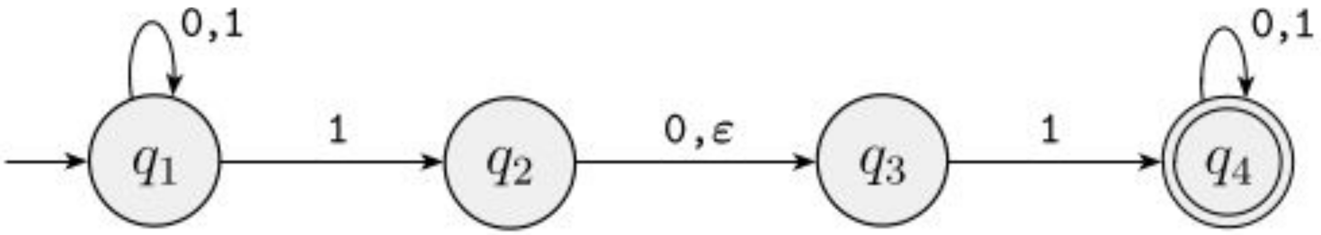
Formal Definition of NFA

DEFINITION 1.37

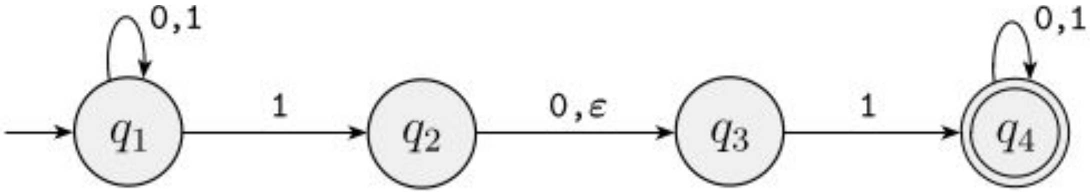
A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal Definition of N_1



Formal Definition of N_1



The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0,1\}$,
3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and
5. $F = \{q_4\}$.

Formal Definition of Computation

Let $N = (Q, \Sigma, \delta, q\ 0\ , F\ )$
N accepts w if we can write w as $w = y_1 y_2 \cdots y_m$,
where each y_i *is a member of* $\Sigma\varepsilon$ *and a sequence of states* $r_0, r_1, \ldots, r_m$
exists in Q with three conditions:

➔ ***Conditions:***

✓ $r_0 = q_0$,
✓ $r_{i+1} \in \delta(r_i, y_{i+1})$, *for* $i = 0, \ldots, m - 1$, *and*
✓ $r_m \in F$

Equivalence of NFAs and DFAs

THEOREM 1.39

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Equivalence of NFAs and DFAs

PROOF

Let $N = (Q, \Sigma, \delta, q_0, F)$ *be the NFA recognizing some language A.*

We want to construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ *recognizing A.*

- ➔ *Need to answer 5-tuples of DFA for the given NFA*
- ➔ *States of possibilities*
- ➔ *Transition for all possible subsets*
- ➔ *Determine start state and Final state*

Equivalence of NFAs and DFAs

PROOF

1. $Q' = P(Q)$
2. Σ
3. For $R \in Q'$ and $a \in \Sigma$,
   Let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

4. $q'_0 = \{q_0\}$
5. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$.

Equivalence of NFAs and DFAs

PROOF *[Considering* ε *]*

- ➔ *For* $R \subseteq Q$
- ➔ $E(R) = \{q \mid q$ *can be reached from R by traveling along 0 or more* ε *arrows*$\}$

3. *Let* $\delta'(R, a) = \{q \in Q \mid q \in \boldsymbol{E(\delta(r, a))}$ *for some* $r \in R\}$
4. $q'_0 = \boldsymbol{E(\{q_0\})}$

Example of Equivalence of NFAs and DFAs

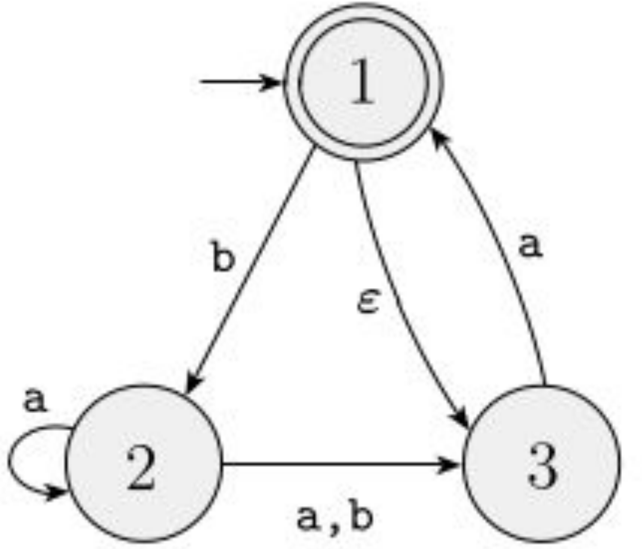


Figure: NFA N_4

Example of Equivalence of NFAs and DFAs

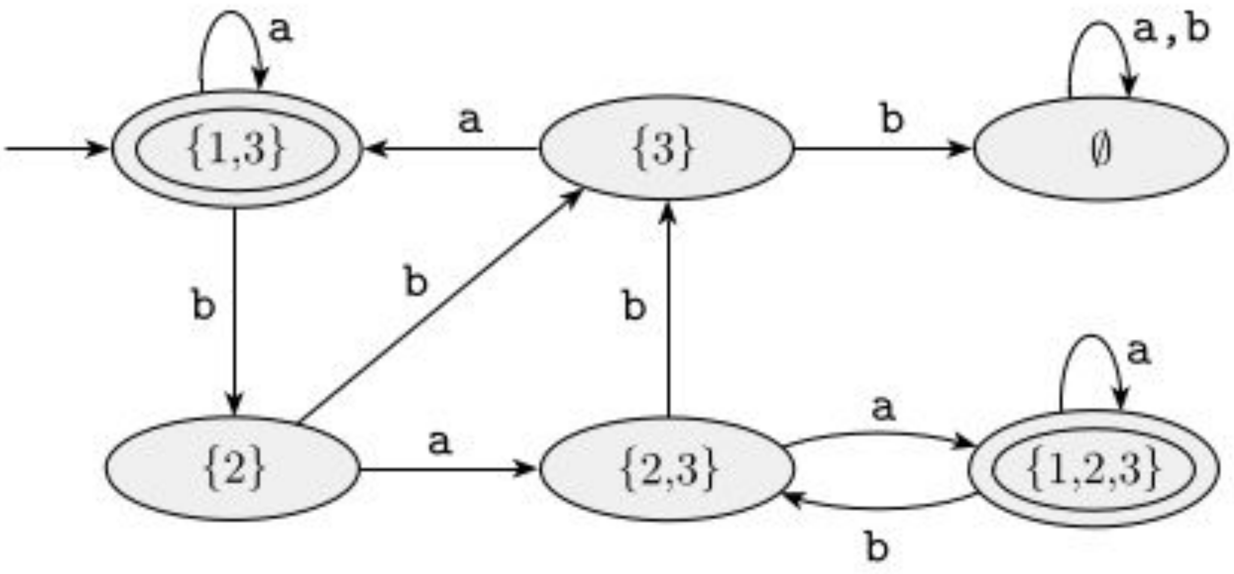


Figure: Equivalent DFA for N_4

Equivalence of NFAs and DFAs

COROLLARY 1.40

A language is regular if and only if some nondeterministic finite automaton recognizes it.

Equivalence of NFAs and DFAs

- ➔ *"If" =>*
  A language is regular if ***some NFA recognizes it.***

- ➔ *"Only if" =>*
  A language is regular *only if some NFA recognizes it. That is, if a language is regular, some NFA must be recognizing it.*

Closure Under The Regular Operations -Union

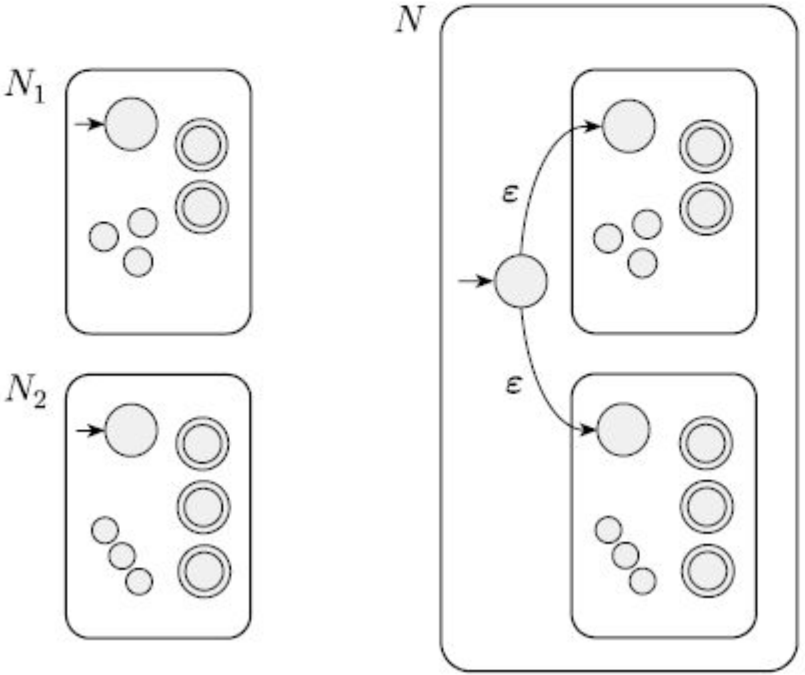
THEOREM 1.45

The class of regular languages is closed under the union operation.

Closure Under The Regular Operations -Union

Proof Idea:

Construction of an NFA N to recognize A1 ∪ A2



Closure Under The Regular Operations -Union

PROOF

Let $N1 = (Q1, \Sigma, \delta 1, q1, F1)$ *recognize* $A1$*, and*
$N2 = (Q2, \Sigma, \delta 2, q2, F2)$ *recognize* $A2$.

Let's Construct $N = (Q, \Sigma, \delta, q0, F)$ *to recognize* $A1 \cup A2$.

Closure Under The Regular Operations -Union

PROOF (.... continued)

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. Σ
3. **δ**

   for any $q \in Q$

   and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

4. *The state q_0 is the start state of N.*
5. *The set of accept states, $F = F_1 \cup F_2$.*

Closure Under The Regular Operations -Concatenation

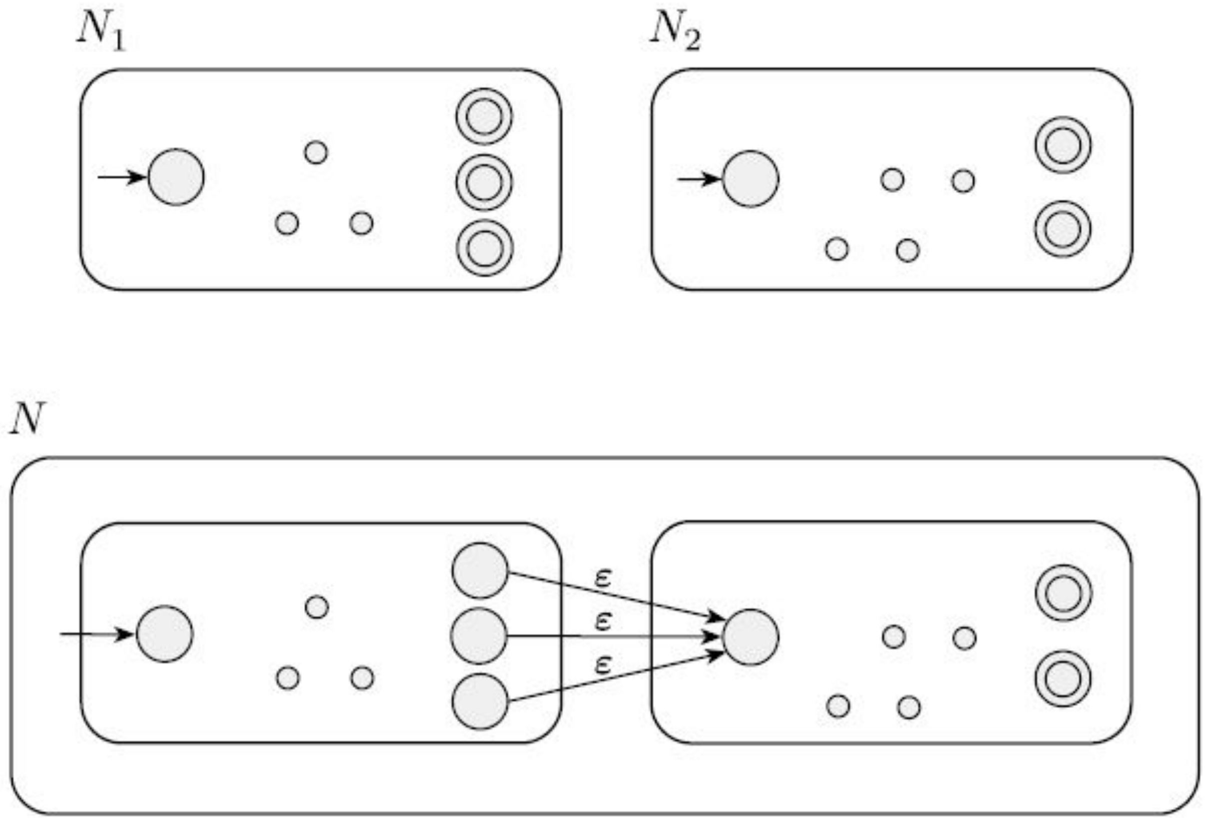
THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

Closure Under The Regular Operations -Concatenation

Proof Idea:

Construction of N to recognize $A1 \circ A2$



Closure Under The Regular Operations -Concatenation

Proof:

1. *Q = Q1 ∪ Q2.*
2. *The state q1 is the same as the start state of N1.*
3. *The accept states F2 are the same as the accept states of N2.*
4. ***δ** for any q ∈ Q and any a ∈ Σε,*

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

Closure Under The Regular Operations -Star

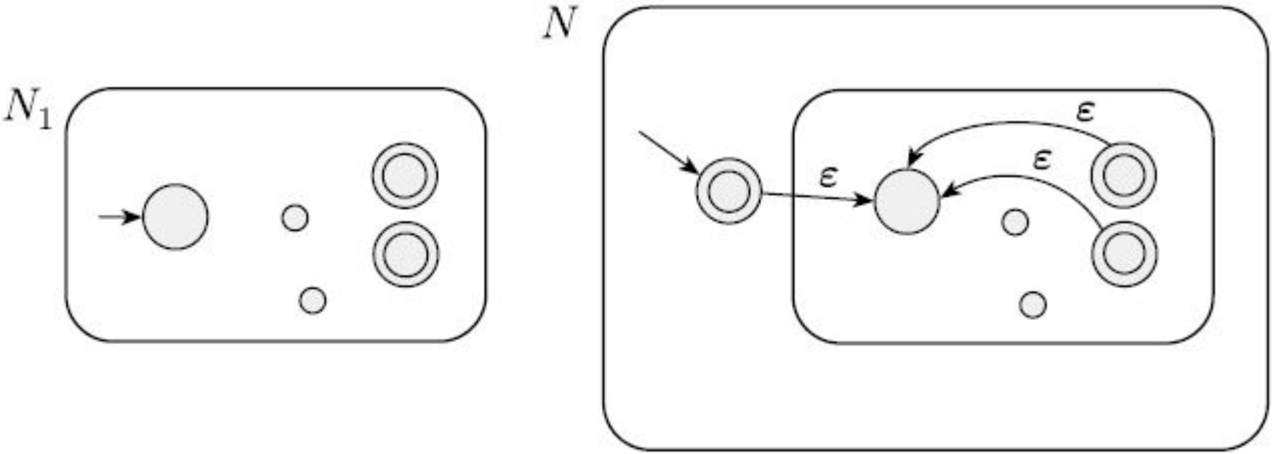
THEOREM 1.49

The class of regular languages is closed under the star operation.

Closure Under The Regular Operations -Star

Proof Idea:

Construction of N to recognize A^*



Closure Under The Regular Operations -Star

Proof:

1. $Q = \{q_0\} \cup Q_1$
2. *The state q_0 is the new start state.*
3. $F = \{q_0\} \cup F_1.$
4. ***δ*** *for any $q \in Q$ and any $a \in \Sigma\varepsilon$,*

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Example NFA

- ➔ $L = (ab \cup aba)^*$
- ➔ *As many as $(ab \cup aba)$'s you like.*
- ➔ $(ab \cup aba)^* = (ab \cup aba)(ab \cup aba)(ab \cup aba) \ldots\ldots\ldots(ab \cup aba)$

✓	*ab*	*belongs*
✓	*aba*	*belongs*
✓	*ababa*	*belongs*
✓	*abaab*	*belongs*
✓	*abab*	*belongs*
✗	*abababba*	*does not belong*

Non Deterministic Finite Automata

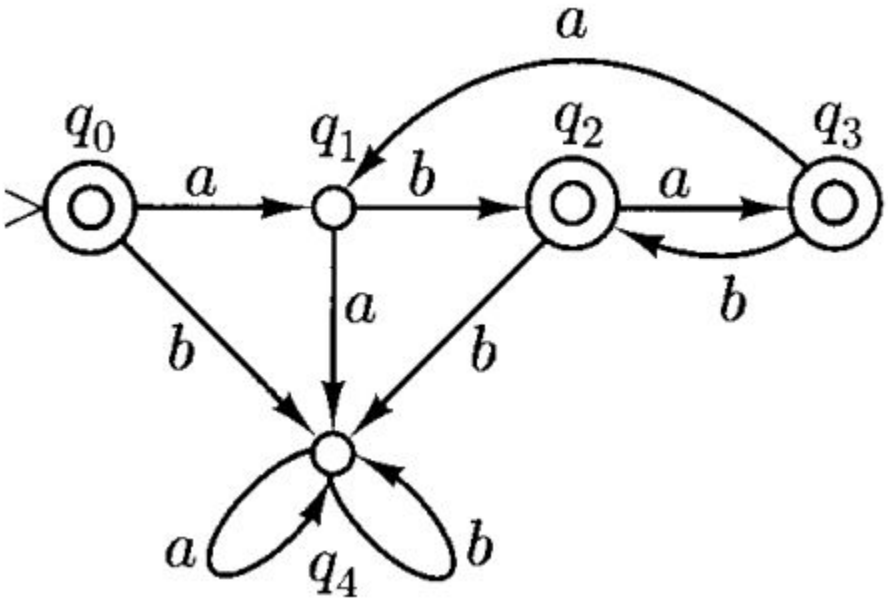


*Figure: DFA for L = (ab ∪ aba)**

Non Deterministic Finite Automata

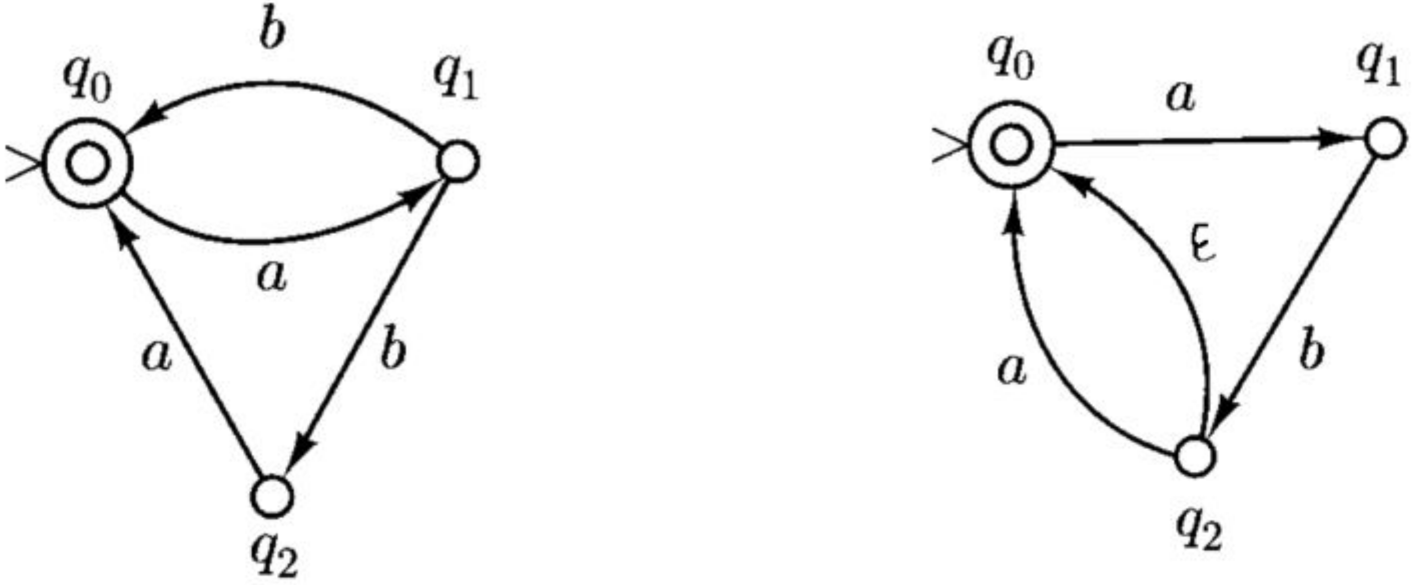


*Figure: NFA for L = (ab ∪ aba)**