Chapter 1

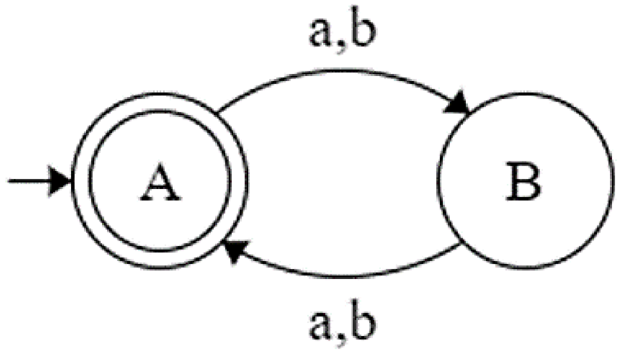
1.2 nondeterminism

Deterministic computation :

- When the machine is in a given state and reads the next input symbol, we know what the next state will be.
  - it is determined.
  - NO choices, no randomness, no guess, no errors,…
- deterministic finite automaton is abbreviated as DFA (FSM).

- **Rule 1: every state** of a **DFA** always has exactly one exiting transition arrow **for each symbol** in the alphabet.

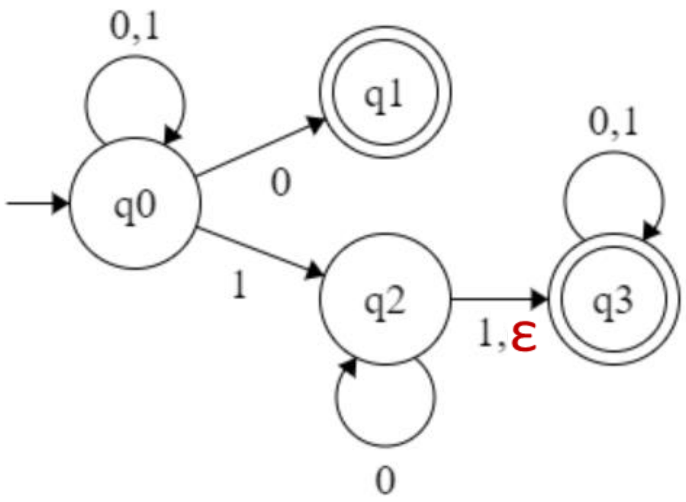
- **Symbols in DFA:** from alphabet.



Nondeterministic computation :

- **nondeterministic** finite automaton is abbreviated as NFA.
- several choices (or no choice) may exist for the next state at any point.
- **Edges in NFA:**
  - Multiple edges with the same labels out of a node.
  - ε edges.
    - Can take an state without scanning a symbol from alphabet

- **Symbols in NFA:** from alphabet or ε .



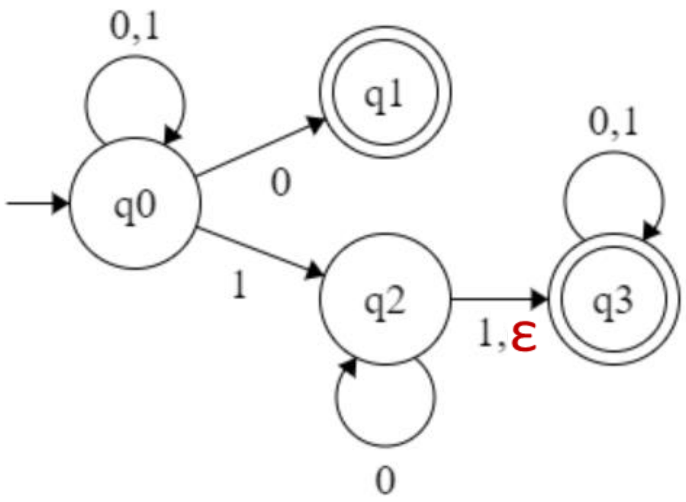
Chapter 1

Nondeterministic computation :

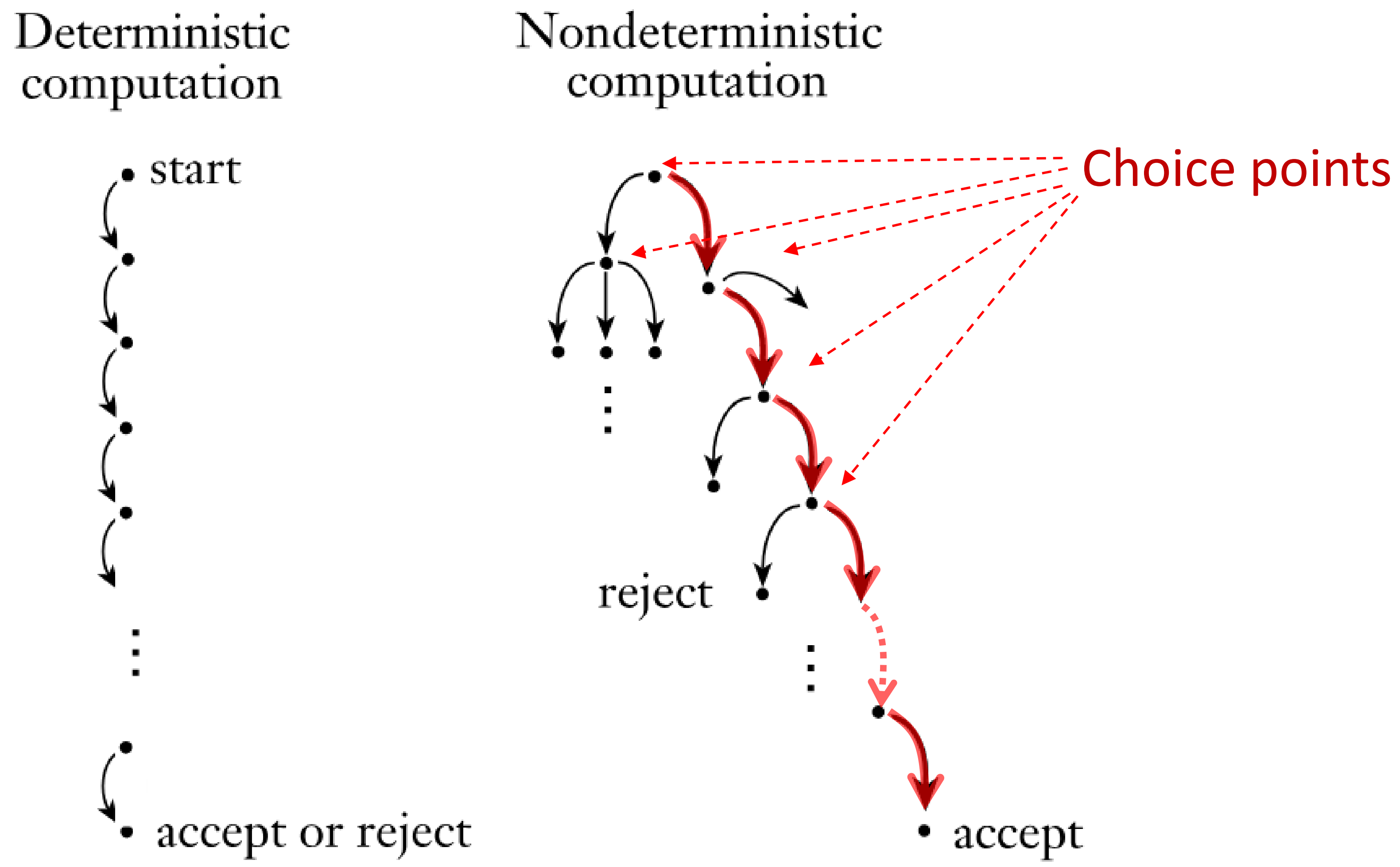
- **NFA** violates **Rule1**.(see the previous slide)
  - In the following example:
    - **State q0:** has two arrows for 0 and two arrows for 1.
    - **State q1:** has no arrow for 0 and 1.
    - Can take the **state q3** without scanning a symbol from alphabet
      - (q0 (**1**)→q2 and q0 (1, **ε**)→q3)

- IN NFA, all next states are chosen
  - In parallel
  - And examined simultaneously.

- Nondeterminism is a generalization of determinism

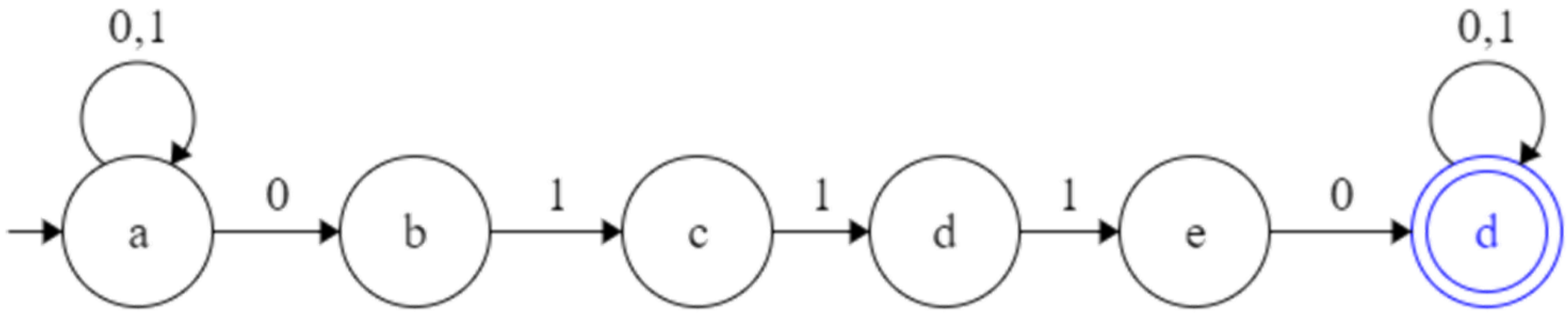


NONDETERMINISM vs DETERMINISM



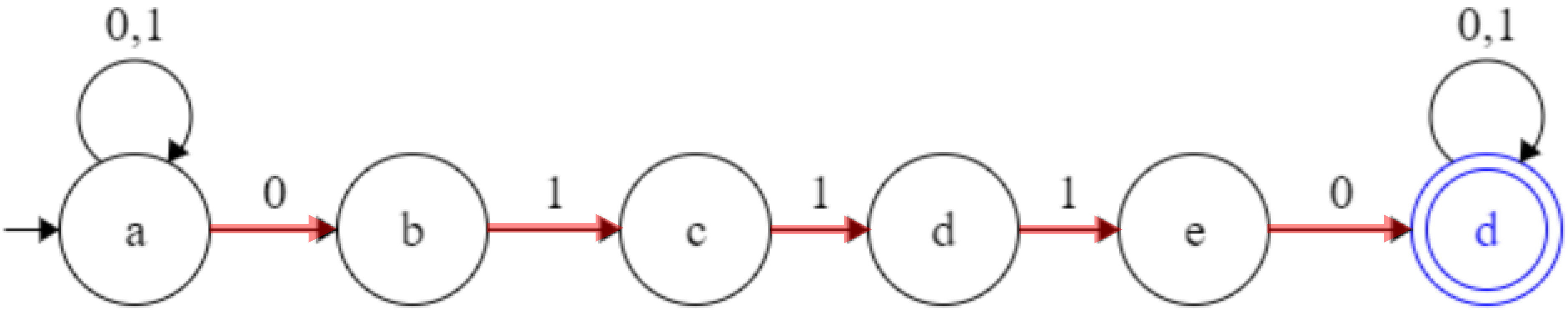
How does an NFA compute ?

- Does the following NFA accept all strings that contain 01110.

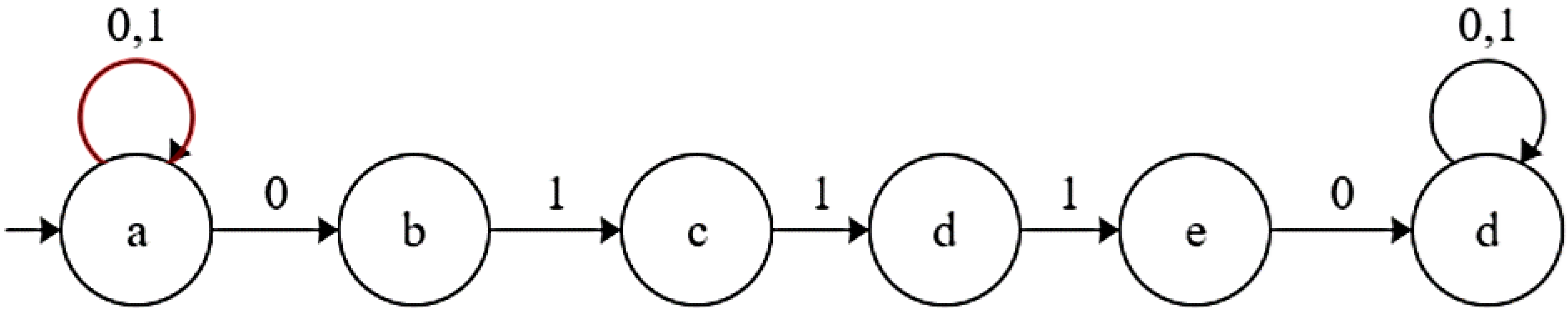


How does an NFA compute ?

- Does the following NFA accept all strings that contain 01110.



- This machine **accepts** 01110.
- There are **lots of bad choices** that they don't work.

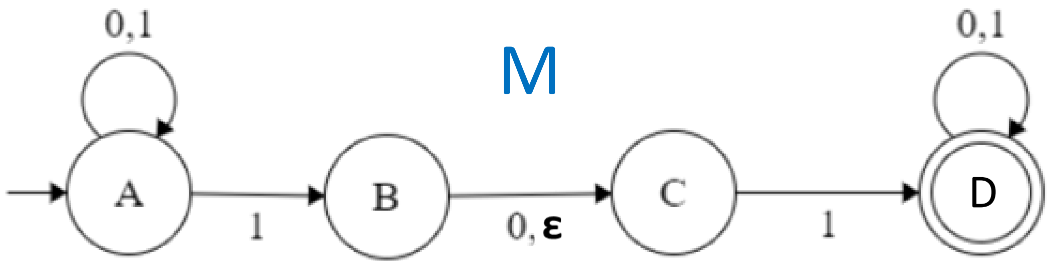


- If there is any way to run the machine that **ends with accept,**
  - Then the **NFA accepts** the string.

Chapter 1

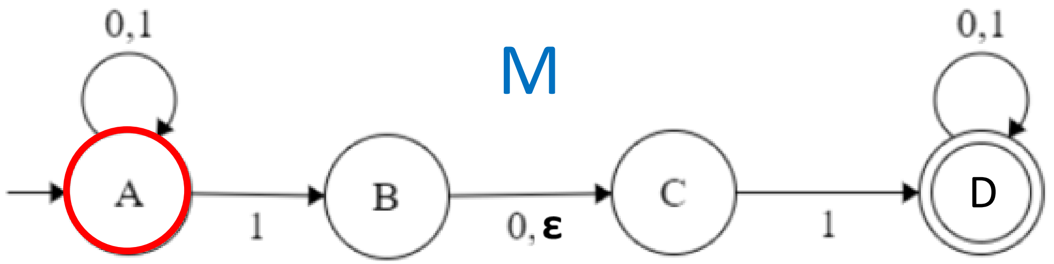
How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



How does an NFA compute ?

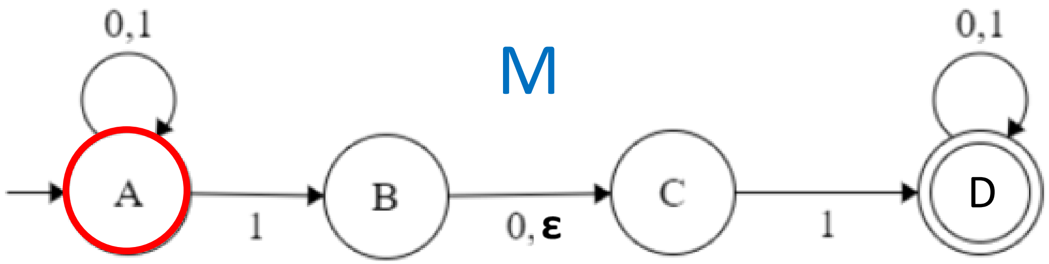
- Does the following NFA accept all strings that contain 010110.



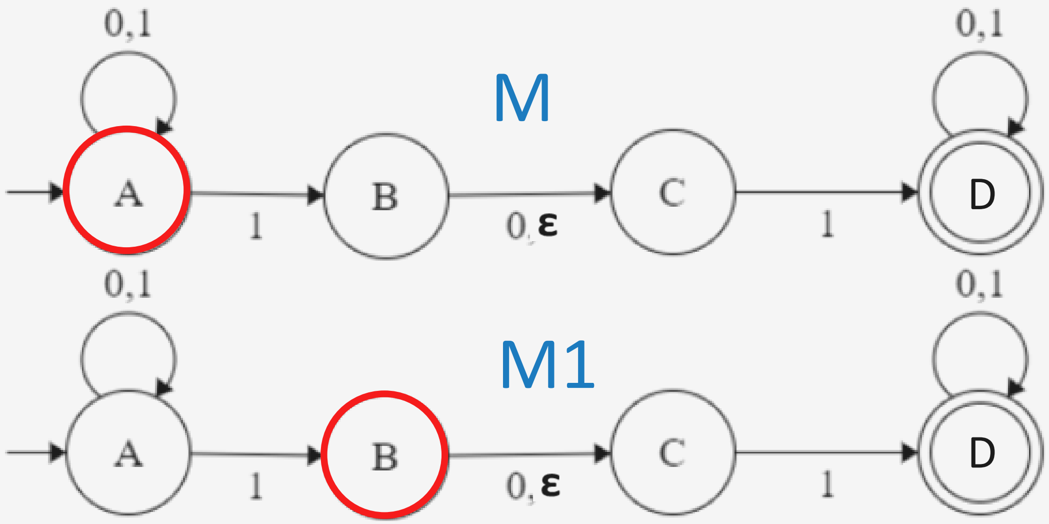
010110

How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



010110



010110

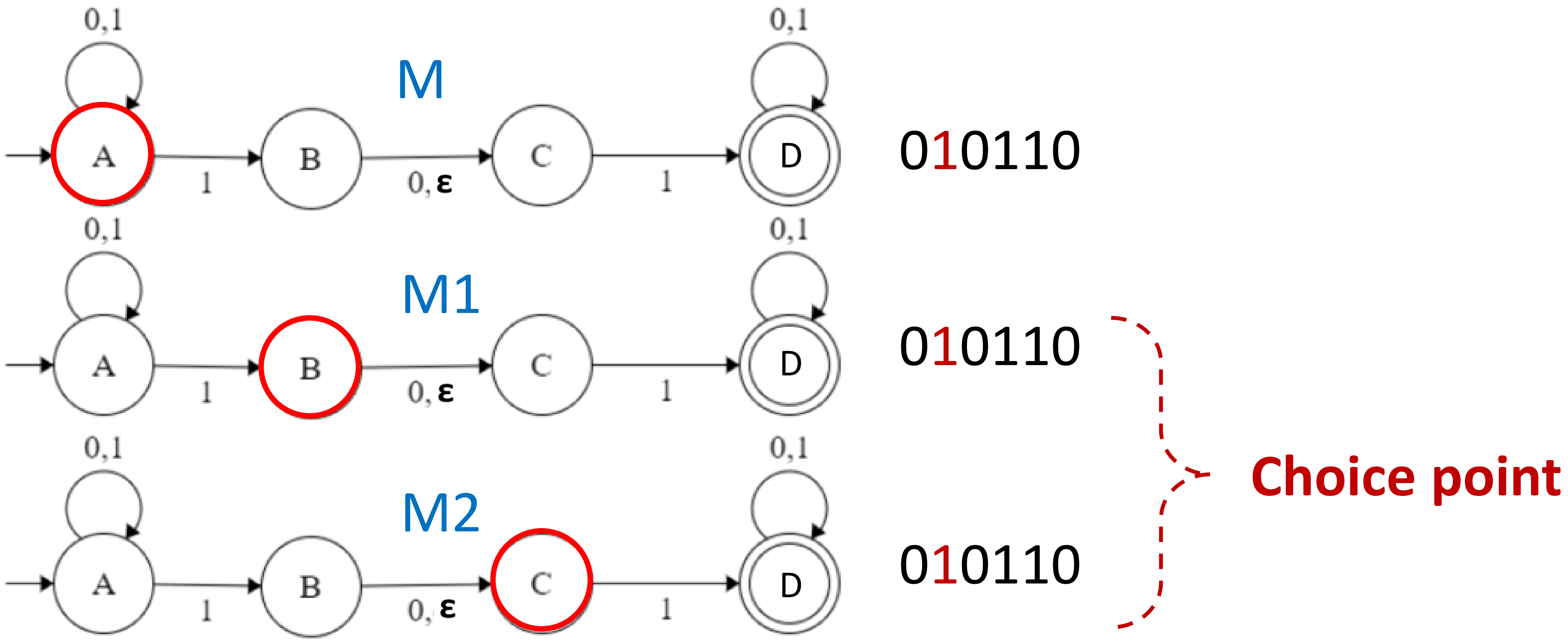
010110

Choice point

- When the machine sees the 2nd symbol of the inputs, a new copy of the machine, M1, is created.
- both machines will process the input in parallel (simultaneously).

How does an NFA compute ?

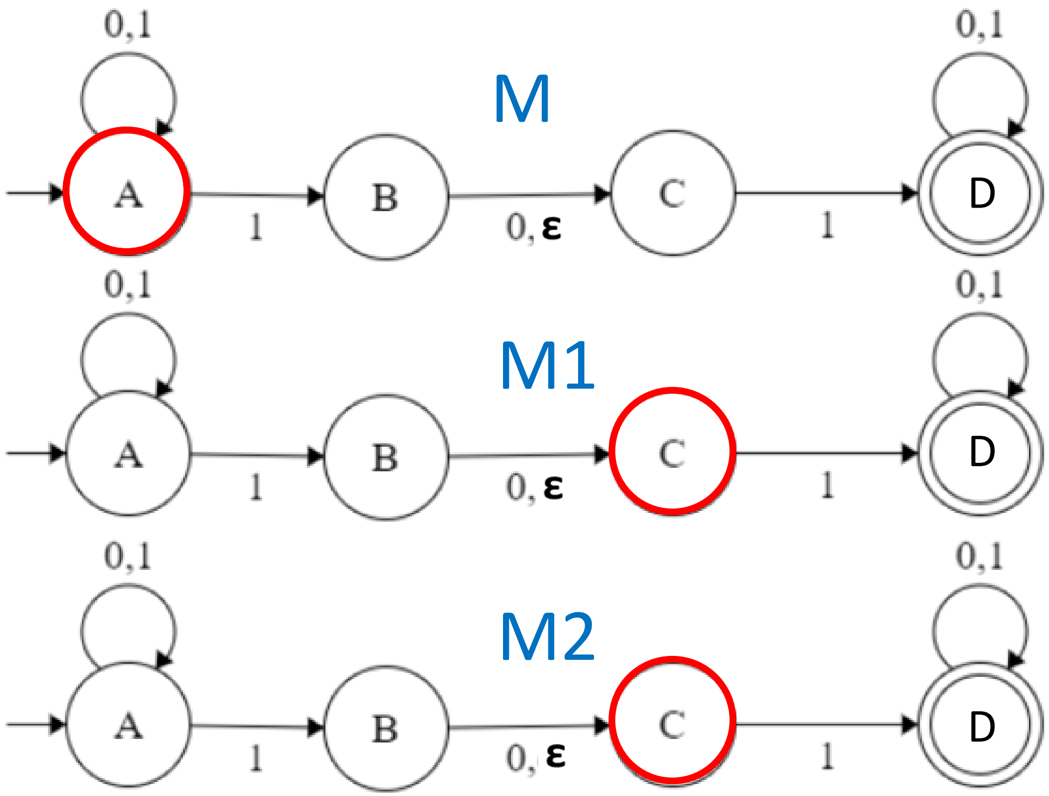
- Does the following NFA accept all strings that contain 010110.



- Since one of the edges out from B is ε, then a new copy of machine will be created
- and the current state of new machine will be changed to C.

How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



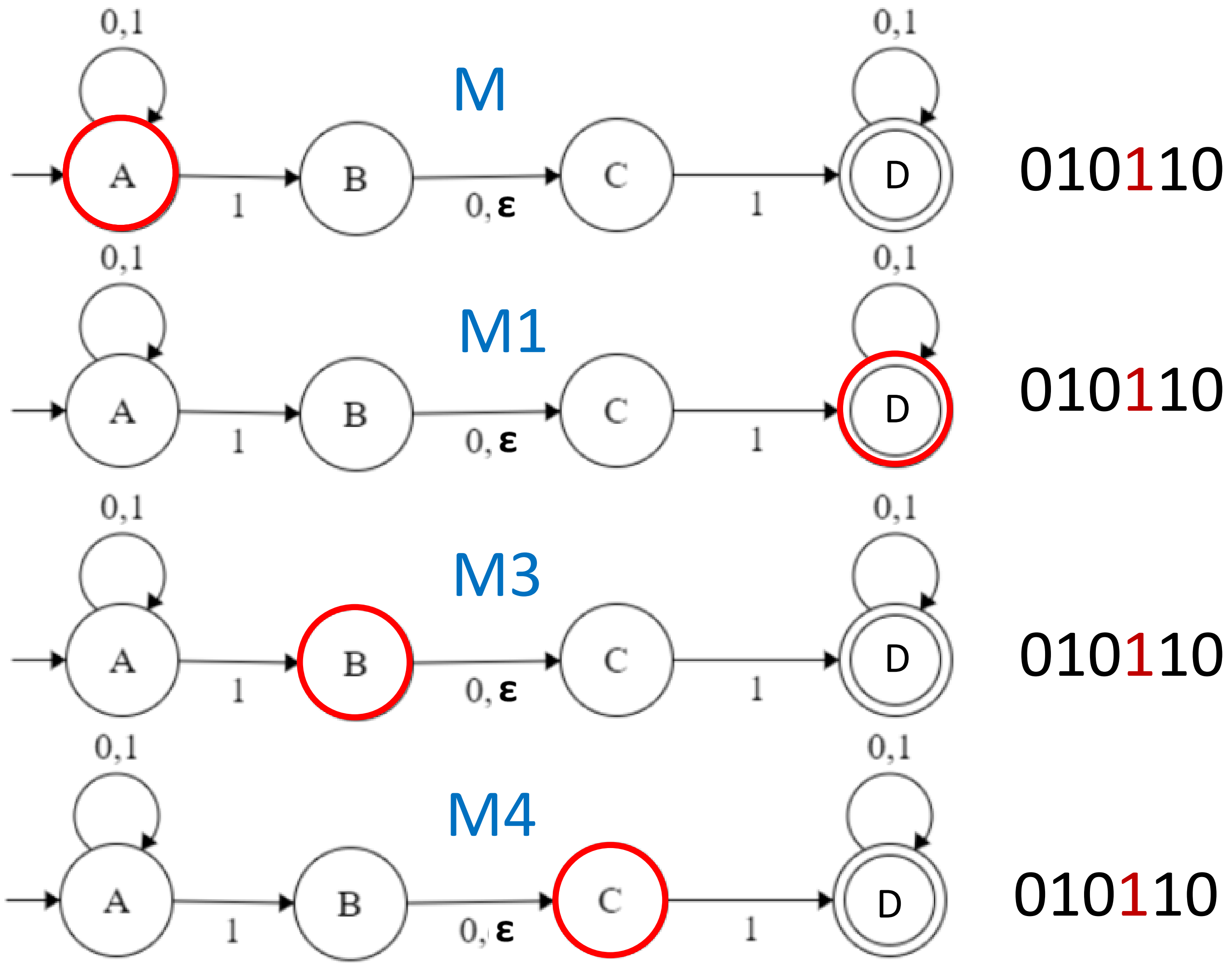
010110

010110

010110 → **Reject** (there is no path)

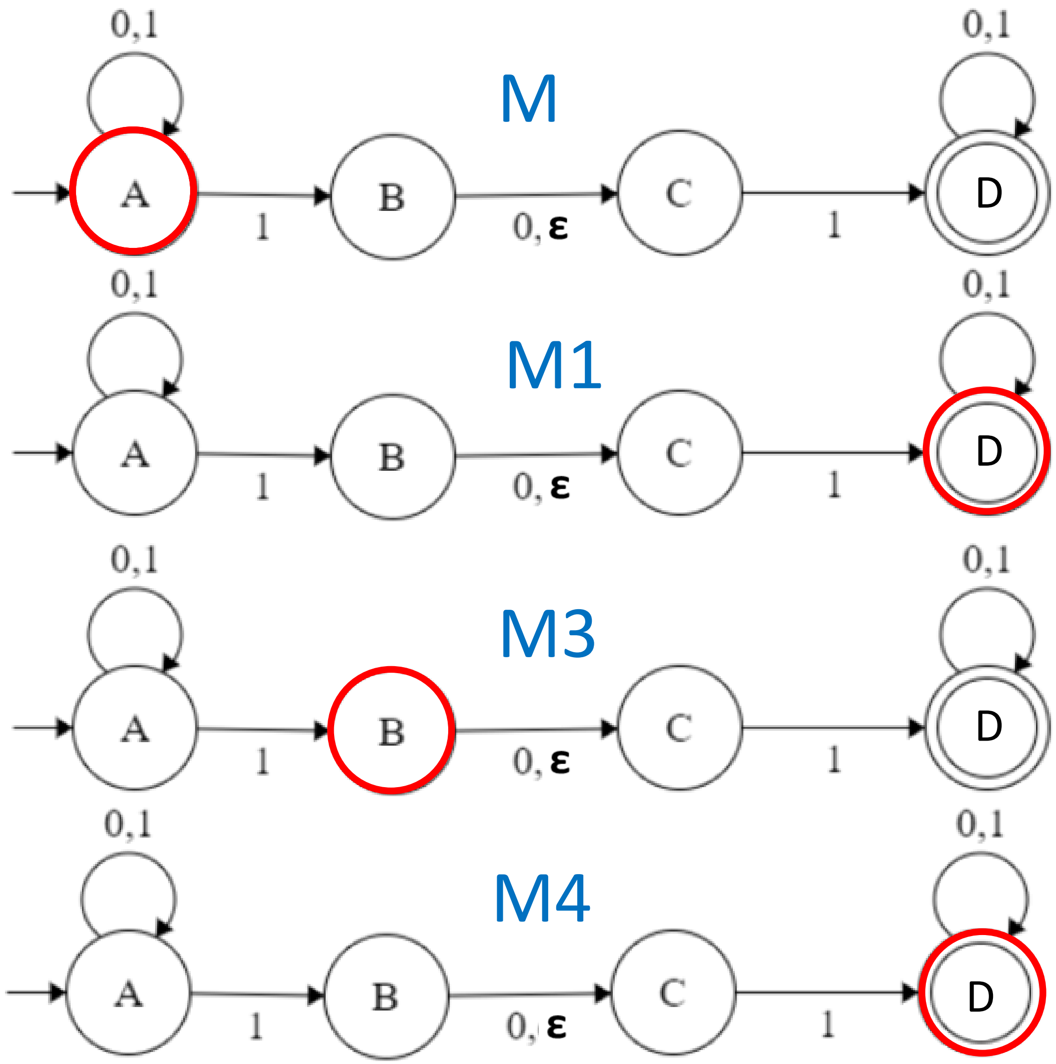
How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



010110

- this machine will be copied again.

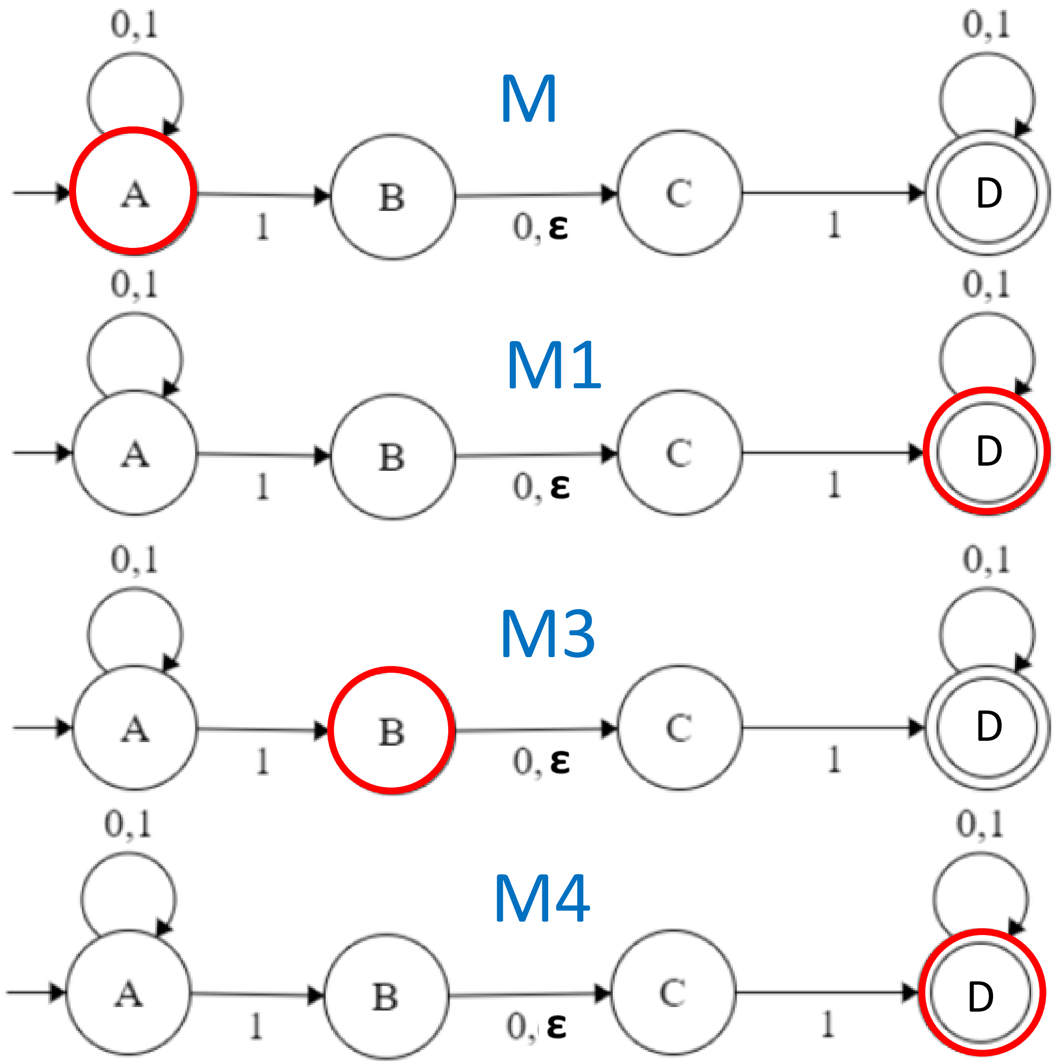
010110 → this copy will **accept** the string.

010110 (Reject)

010110 → this copy will **accept** the string.

How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.



010110

- this machine will be copied again.

accept

Reject

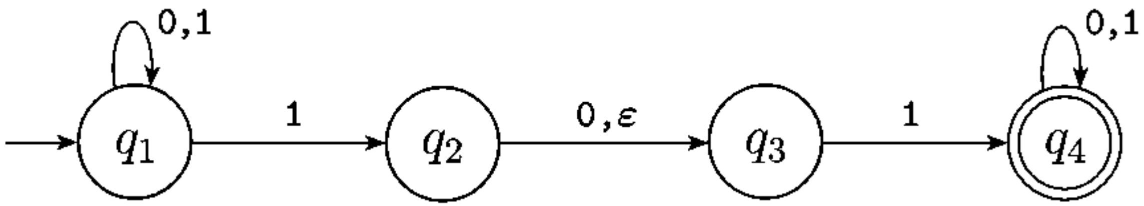
accept

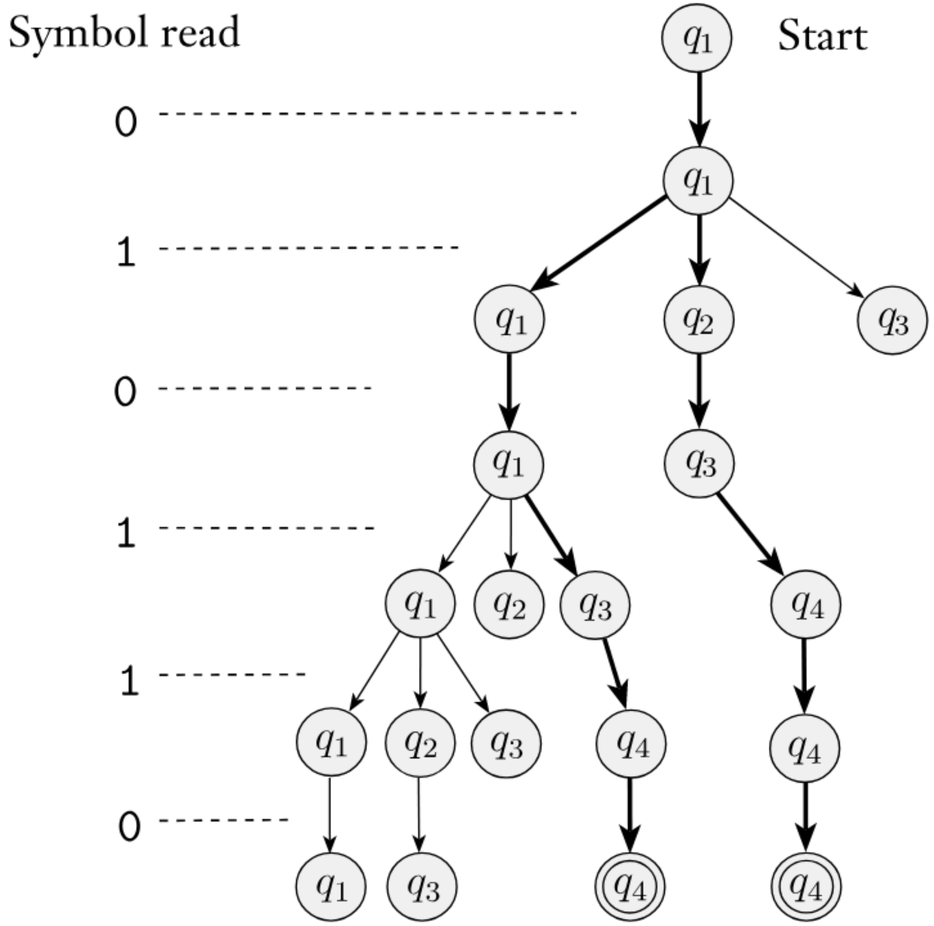
- If one of theses parallel machines accept the string, it is enough. We say that NFA accepts the string.

NONDETERMINISM

How does an NFA compute ?

- Does the following NFA accept all strings that contain 010110.
  - Using "Computation Tree"



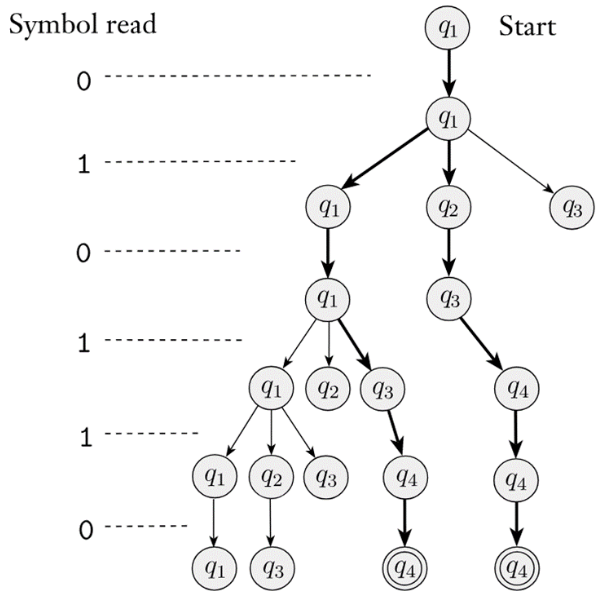


Chapter 1

Formal definition of a Nondeterministic finite automaton

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

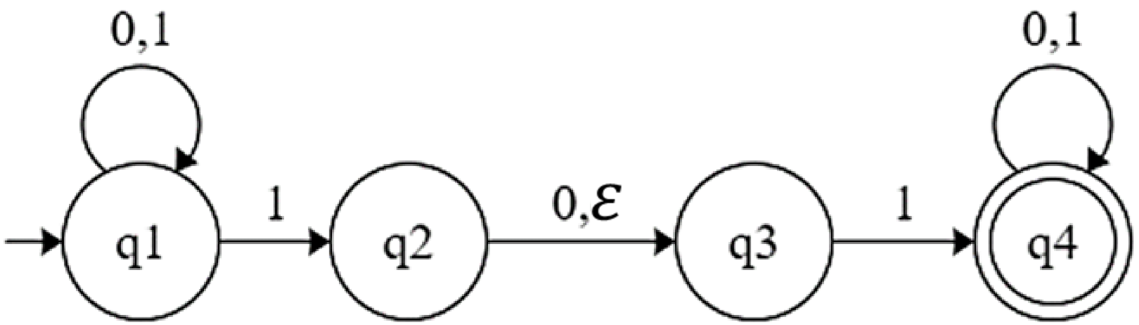
1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function, Where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept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



Formal definition of a Nondeterministic finite automaton

Example:

- What is the formal description of the following NFA ?



- Q = {q1,q2,q3,q4},
- $\Sigma = \{0,1\}$
- 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	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

- q1 is the start state
- F = {q4}

The formal definition of computation for an NFA

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w=$y_1 y_2 \dots y_m$ be a string over the alphabet Σ.
- Then N accepts w if a sequence of states $r_0, r_1, \dots, r_m$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \dots, m-1$, and
3. $r_m \in F$.

w	y_1	y_2	…	y_n	
State	r_0	r_1	…	r_{m-1}	r_m
					accept

NONDETERMINISM

- **Theorem:**
  - Every nondeterministic finite automaton, **NFA**, has an equivalent deterministic finite automaton, **DFA**.

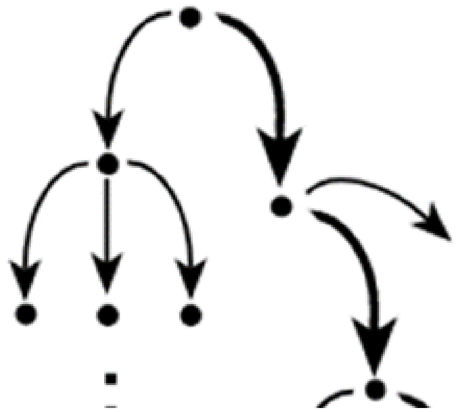
DFA $\leftrightarrow$ NFA

DFA ⟷ NFA

- **Proof Idea: (proof by construction)**
  - **DFA → NFA :** We know that A DFA is special case of NFA. ()
  - **NFA → DFA :** We should show that we can **convert** an NFA into an equivalent DFA that simulates the NFA.

DFA $\leftrightarrow$ NFA

- **Proof Idea: (cont.)**
  - If **k** is the number of states of the NFA, it has **2^k subsets of states**.
  - **Each subset** corresponds to **one of the possibilities** that the DFA must remember, so the DFA simulating the NFA will have 2^k states.



- **Next step:** start state ? accept states?
- **Next step:** transition function?

DFA $\leftrightarrow$ NFA

- **Proof:**
  - Let $\mathrm{N} = (Q, \Sigma, \delta, q_0, F)$ be the **NFA** recognizing some language A.
  - We construct a **DFA** $\mathrm{M} = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

  - **Case 1:** assume N has no ε arrows.
  - Q′=P(Q). (Every state of M is a set of states of N)
  - q_0'=$\{q_0\}$.
  - F′={R∈Q′|R contains an accept state of N}.
  - For R∈Q′ and a∈Σ, let
    - δ′(R, a) = {q∈Q|q∈δ(r, a) for some r∈R}. Or
    - δ′(R, a) = $\bigcup_{r\in R}$δ(r, a).

DFA $\leftrightarrow$ NFA

- **Proof: (cont.)**
  - Let $N = (Q, \Sigma, \delta, q_0, F)$ be the **NFA** recognizing some language A.
  - We construct a **DFA** $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

  - **Case 1:** assume N has no ε arrows.
  - Q′=P(Q).
  - q_0'={q_0}.
  - F′={R∈Q′|R contains an accept state of N}.
  - For R∈Q′ and a∈Σ, let
    - δ′(R, a) = {q∈Q|q∈δ(r, a) for some r∈R}. Or
    - δ′(R, a) = $\cup_{r \in R}$δ(r, a).

$Q = \{1,2,3\}$, q_0 =1 , F ={2}

$$Q' = \begin{Bmatrix} \emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\} \\ \{2,3\}, \{1,2,3\} \end{Bmatrix}$$

q_0 = {1}

F = {{2},{1,2},{2,3},{1,2,3}}

DFA $\leftrightarrow$ NFA

- **Proof: (Cont.)**
  - Let $\mathrm{N} = (Q, \Sigma, \delta, q_0, F)$ be the **NFA** recognizing some language A.
  - We construct a **DFA** $\mathrm{M} = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

  - **Case 2** (General case)**:** assume N has ε arrows.
  - Q′=P(Q). (Every state of M is a set of states of N)
  - q_0'=E($\{q_0\}$).
  - F′={R∈Q′|R contains an accept state of N}.
  - For R∈Q′ and a∈Σ, let
    - δ′(R, a) = {q∈Q|q ∈ E(δ(r, a)) for some r∈R}. Or

  - Where E(R)={q|q can be reached from R by traveling along 0 or more ε arrows}.

NONDETERMINISM

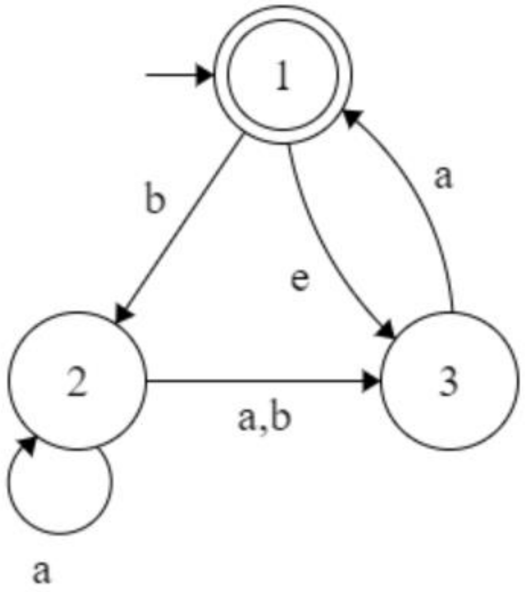
- **Corollary**
    - A language is regular if and only if some nondeterministic finite automaton recognizes it.

Regular Language ↔ DFA ↔ NFA

Chapter 1

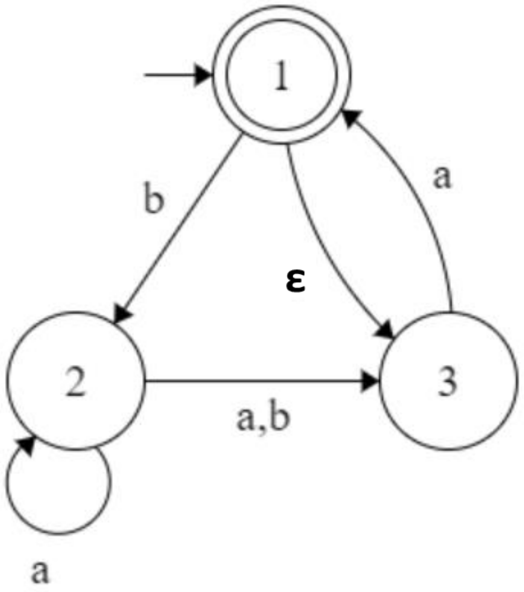
NONDETERMINISM

- **Example:** Find the equivalent DFA of the following NFA.
  - $\mathrm{M} = (\mathrm{Q}, \Sigma, \delta, 1, \{1\})\ where\ \mathrm{Q}= \{1,2,3\}, \Sigma = \{a, b\}$
  -
  - $\mathrm{N} \ = (Q', \Sigma, \delta', q_0', F')$
  - $\Sigma \ = \{a, b\}$
  - $Q' =$ {Ø,{1},{2},{3},{1,2},{1,3},{2,3),{1,2,3}}
  - $q_0' =$ E({1)} = {{1,3}}
  - $F' =$ {{1}, {1,2},{1,3}, {1,2,3}}



NONDETERMINISM

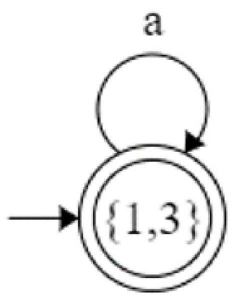
- **Example:** Find the equivalent DFA of the following NFA.
  - Transition δ'?
  - For R∈Q′ and a∈Σ, let
    - δ′(R, a) = {q∈Q | q ∈ E(δ(r, a)) for some r∈R}.

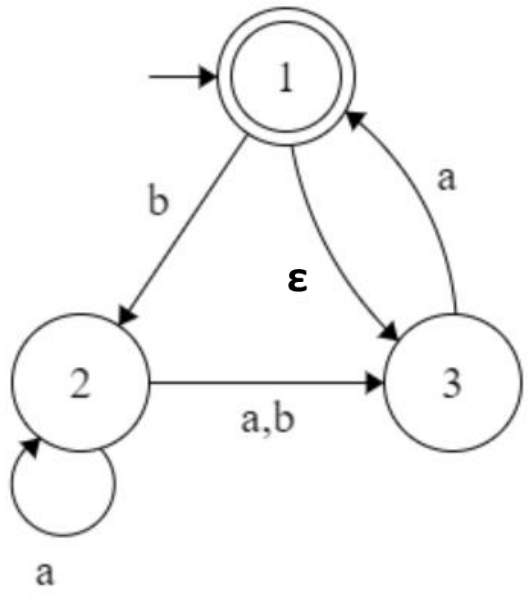


q	δ (q,a)	δ (q,b)	δ (q,ε)	$E(\delta$ (q,a))	$E(\delta$ (q,b))
1	∅	{2}	{3}	∅	{2}
2	{2,3}	{3}	∅	{2,3}	{3}
3	{1}	∅	∅	{1,3}	∅

NONDETERMINISM

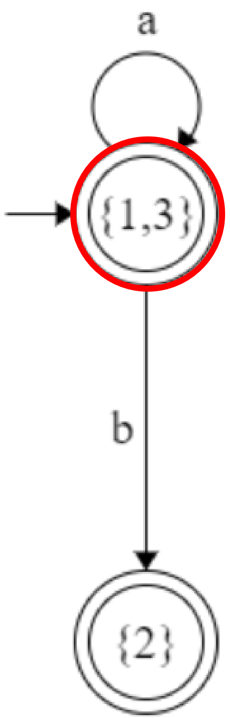
- **Example:** Find the equivalent DFA of the following NFA.
  - We start with the **start state** of DFA {1,3}.

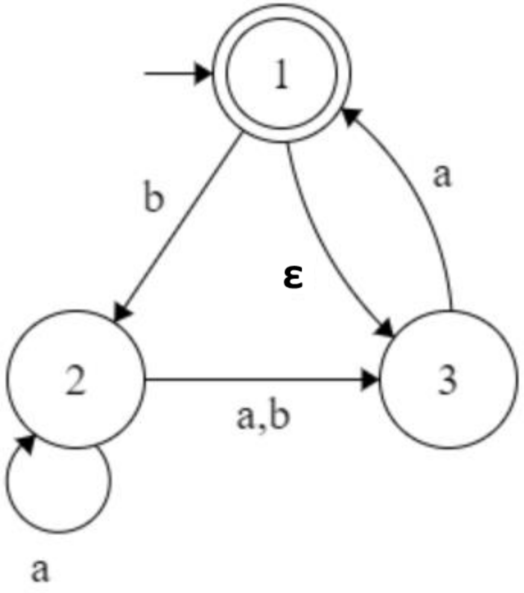




NONDETERMINISM

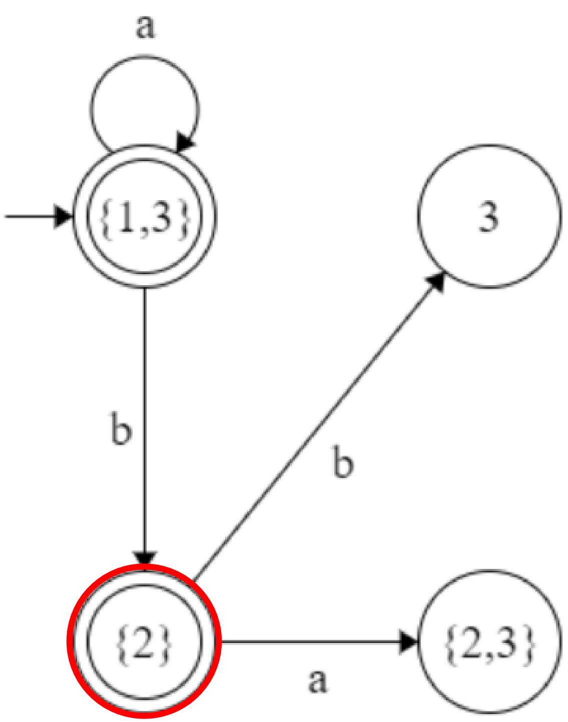
- **Example:** Find the equivalent DFA of the following NFA.
    - DFA after adding transitions from start state {1,3}.

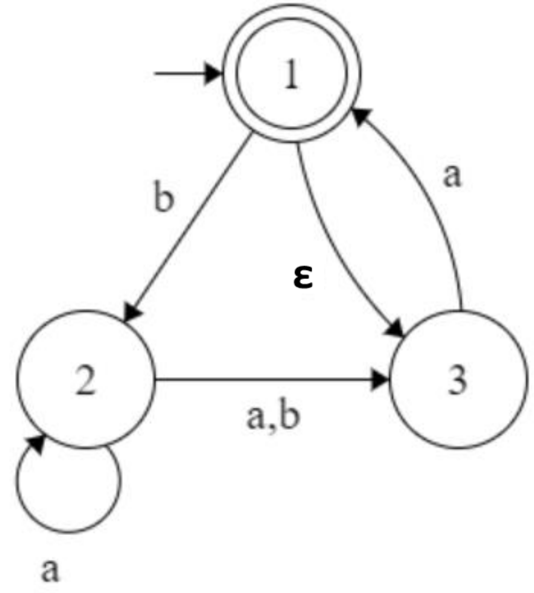




NONDETERMINISM

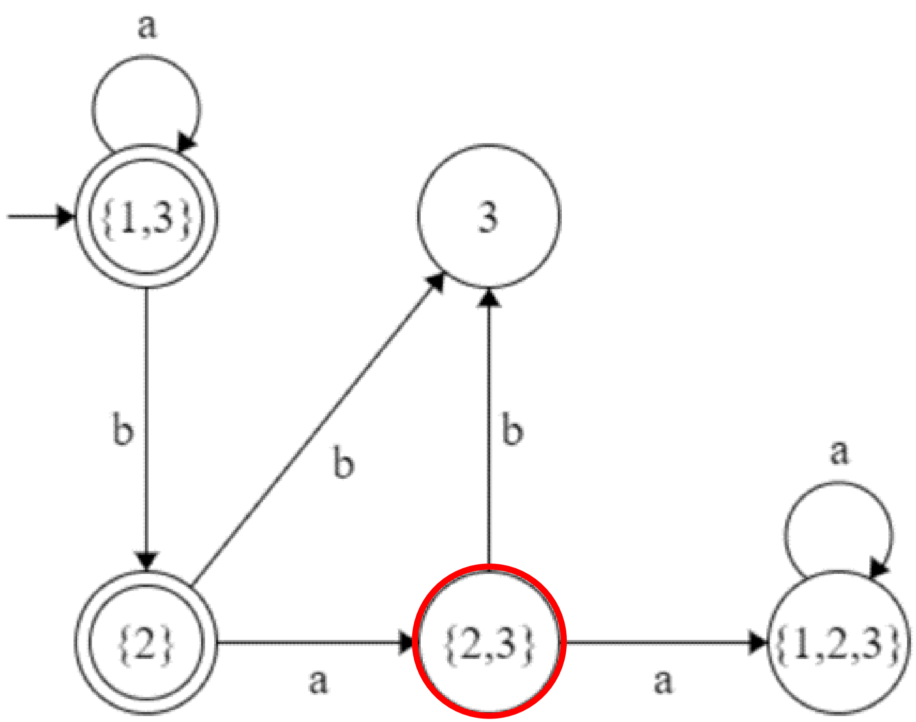
- **Example:** Find the equivalent DFA of the following NFA.
  - **Next step : repeat all the previous steps** for each newly created state, in our case {2}.

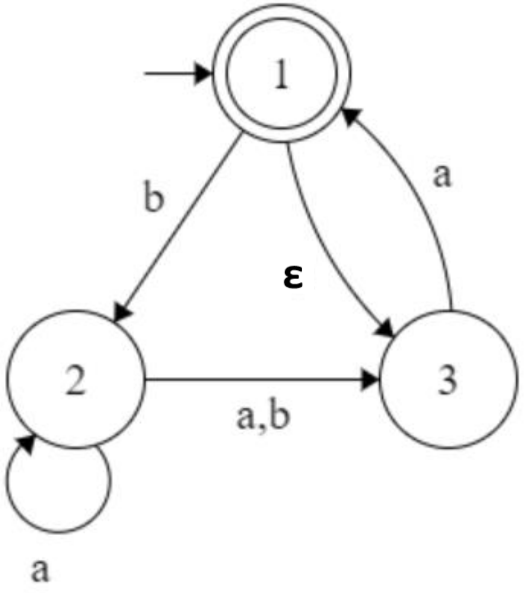




NONDETERMINISM

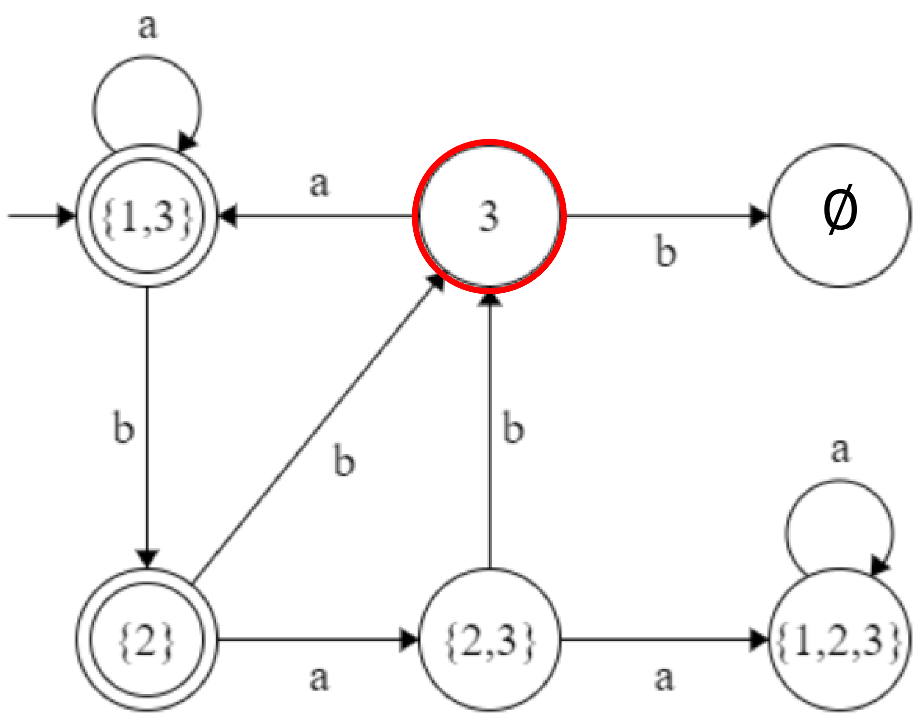
- **Example:** Find the equivalent DFA of the following NFA.
  - **Next step : repeat all the previous steps** for each newly created state, in our case {2}.

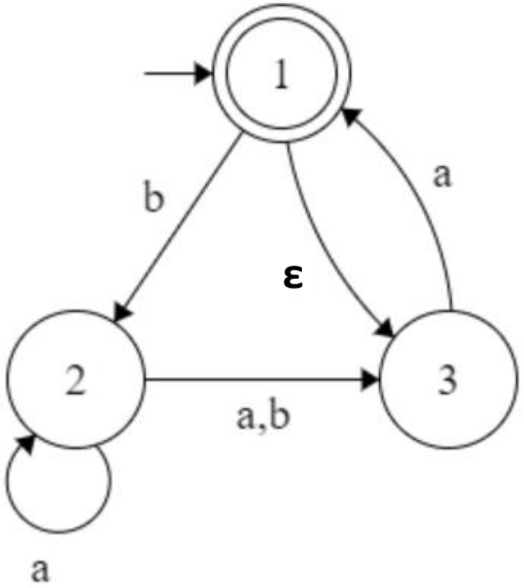




NONDETERMINISM

- **Example:** Find the equivalent DFA of the following NFA.
  - **Next step : repeat all the previous steps** for each newly created state, in our case {2}.

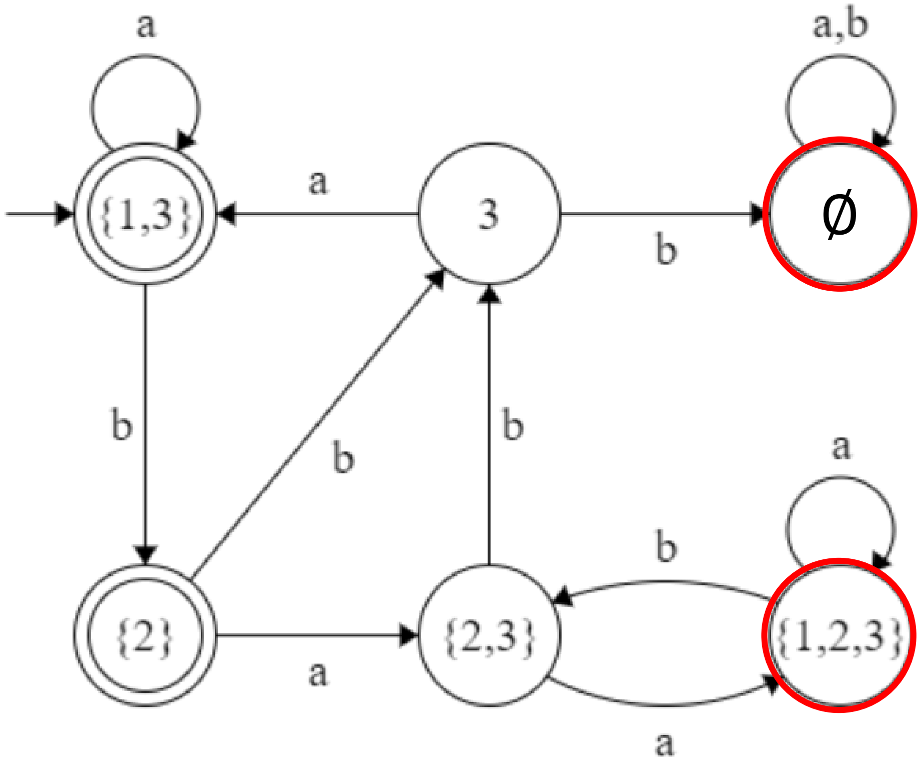


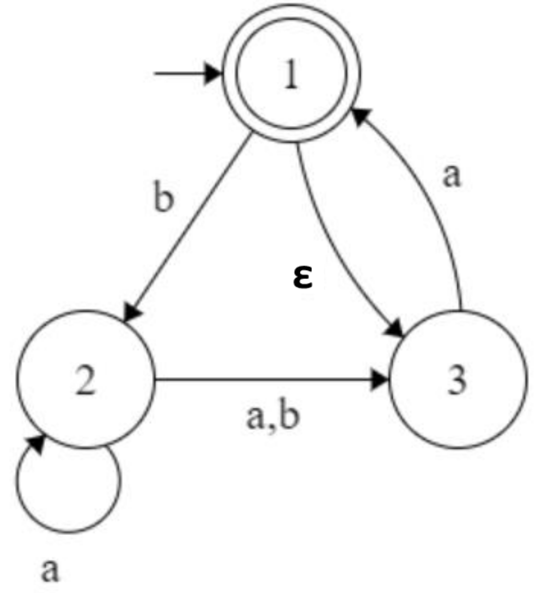


q	δ (q,a)	δ (q,b)	δ (q,ε)	$E(\delta$ (q,a))	$E(\delta$ (q,b))
1	∅	{2}	{3}	∅	{2}
2	{2,3}	{3}	∅	{1,2,3}	{3}
3	{1}	∅	∅	{1,3}	∅

NONDETERMINISM

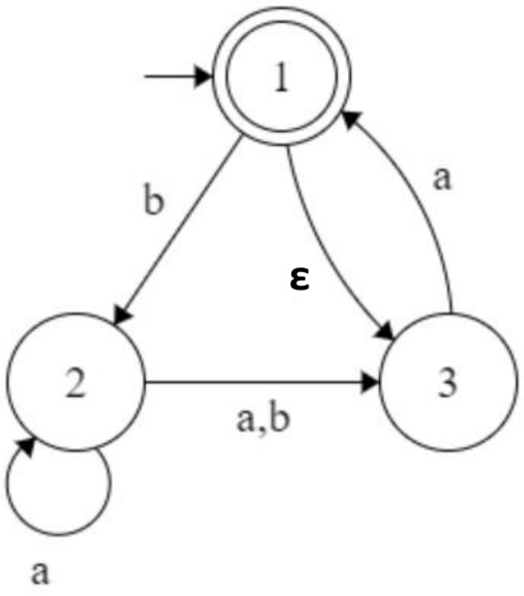
- **Example:** Find the equivalent DFA of the following NFA.
  - **Next step : repeat all the previous steps** for each newly created state, in our case {2}.

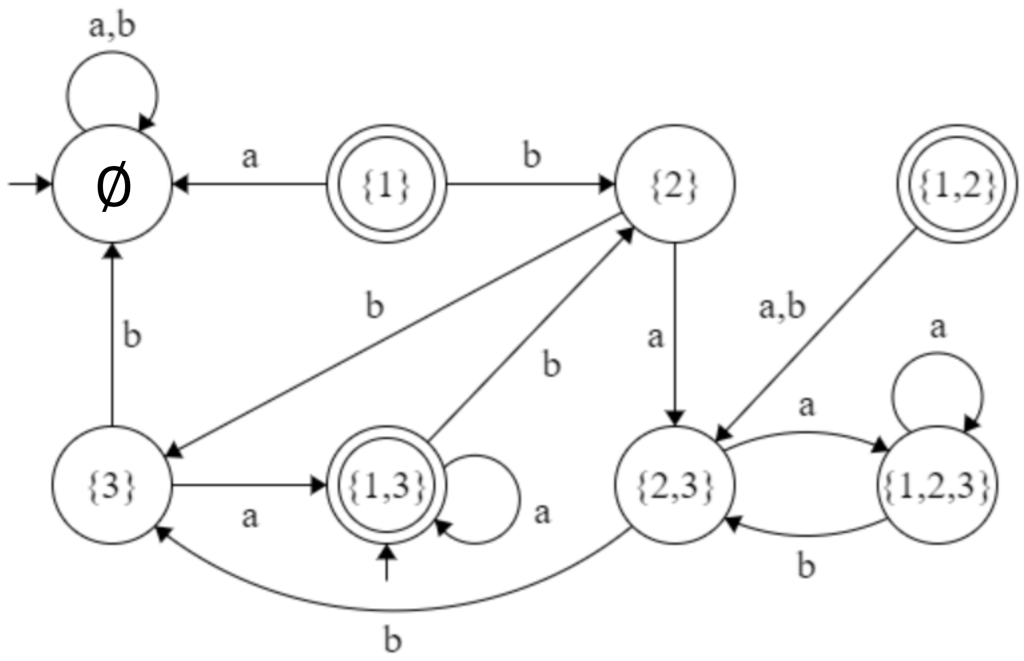




NONDETERMINISM

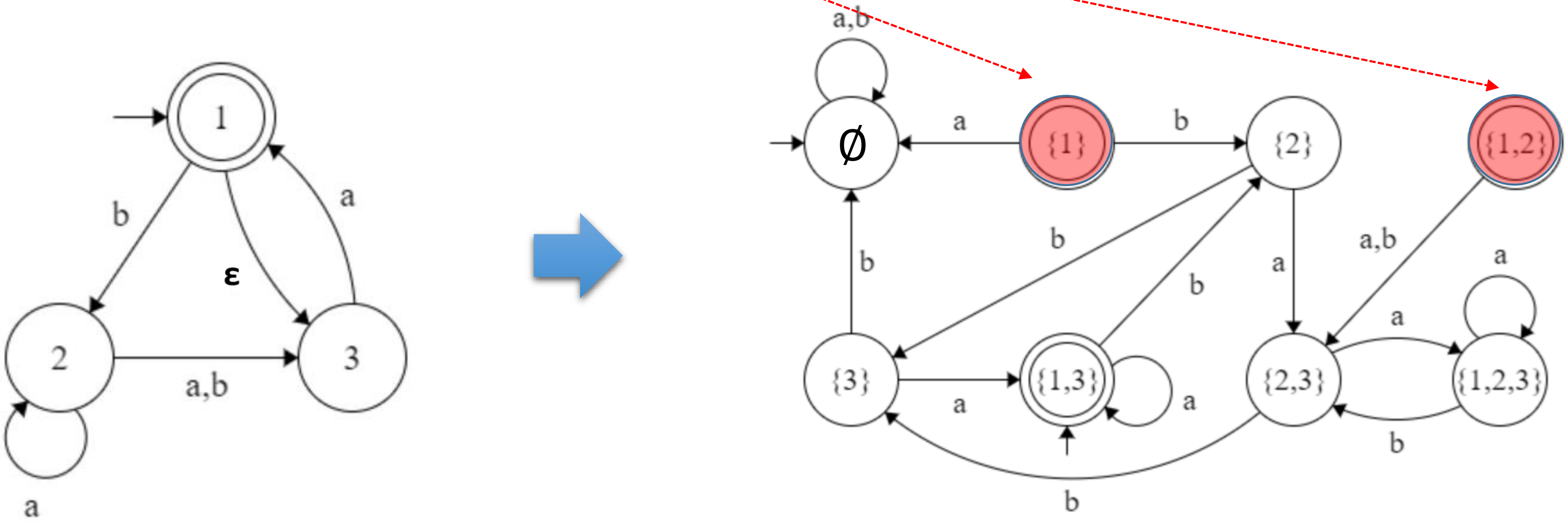
- **Example:** Find the equivalent DFA of the following NFA. (**Method 2**)
  - **Step1** : Considering all possible states (power set)
  - **Step2** : Showing all of the transitions for all of states
  - **Step 3**: Removing unreachable states





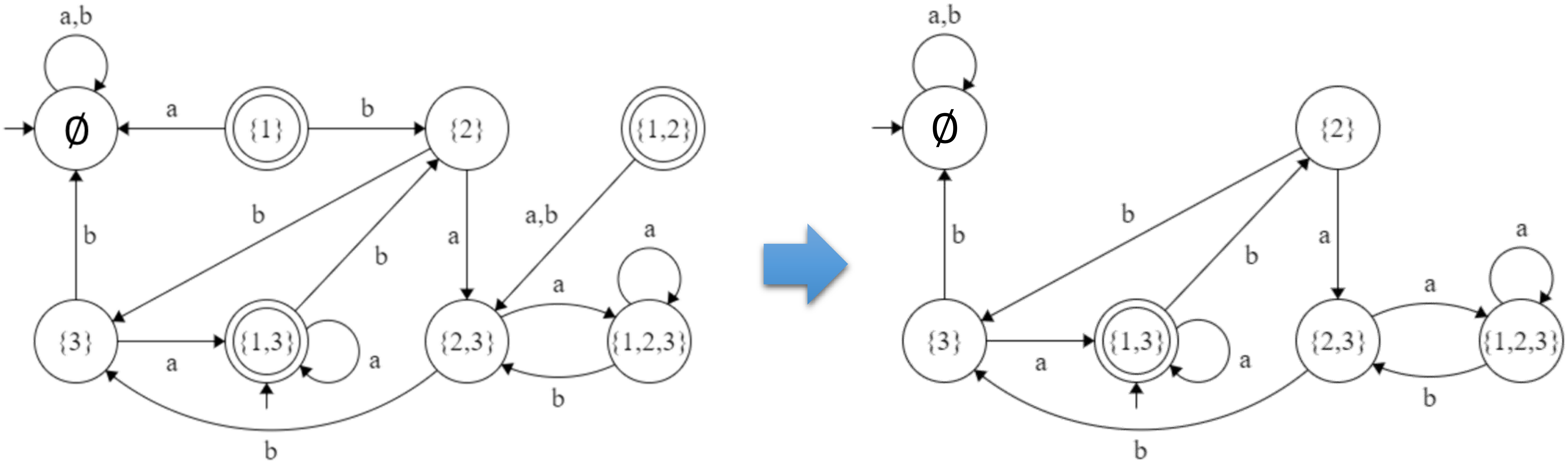
NONDETERMINISM

- **Example:** Find the equivalent DFA of the following NFA. (**Method 2**)
  - **Step1** : Considering all possible states (power set)
  - **Step2** : Showing all of the transitions for all of states
  - **Step 3**: **Removing unreachable** states



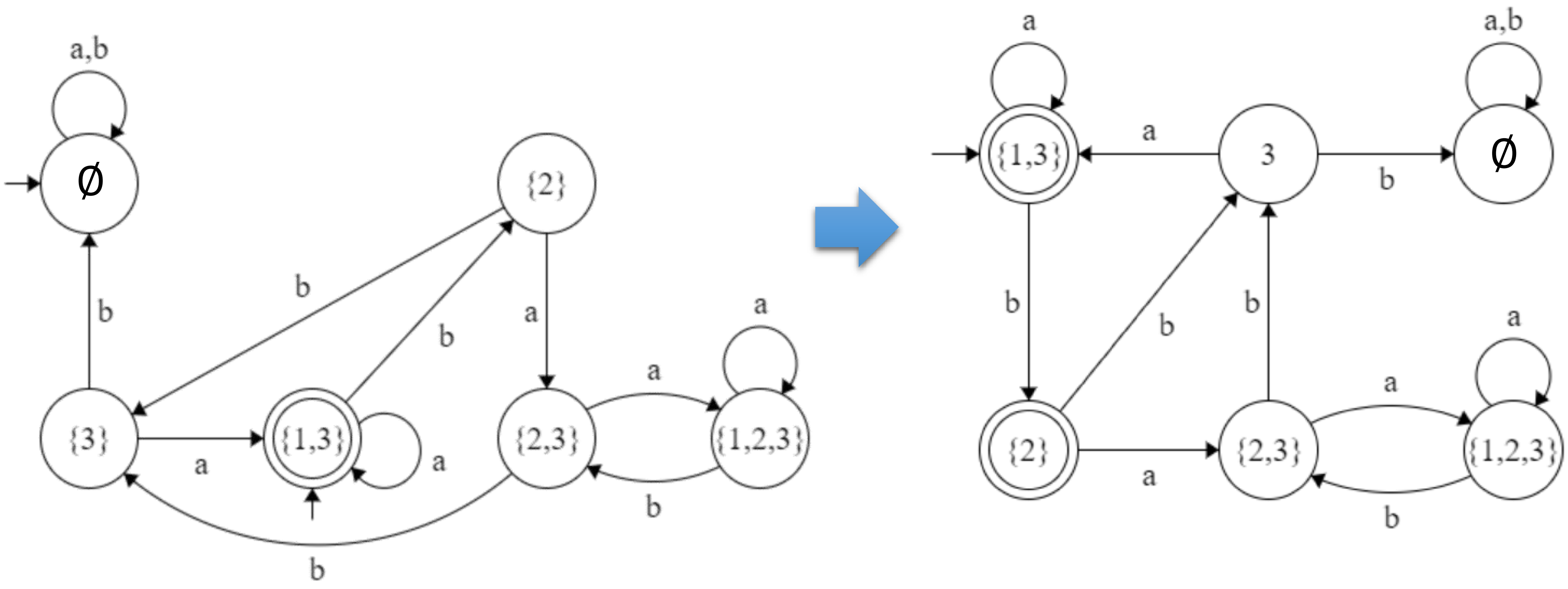
NONDETERMINISM

- **Example (cont.)**
  - Step 3: Removing **unreachable** states



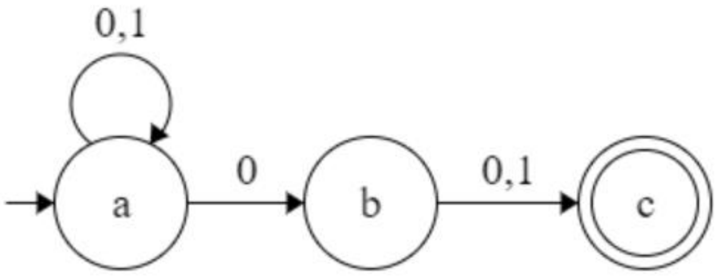
NONDETERMINISM

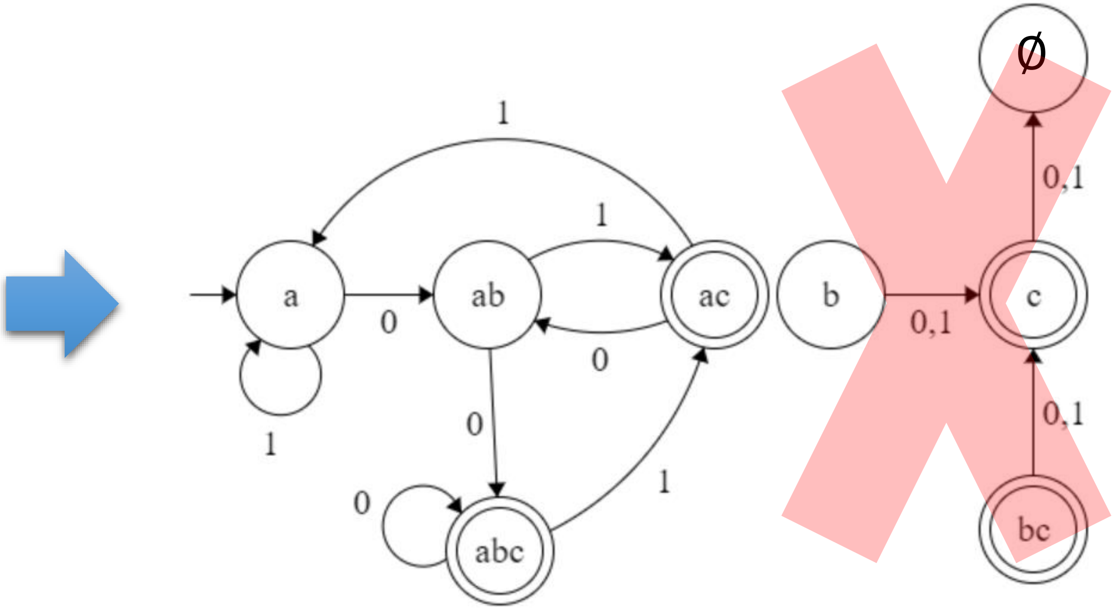
- **Example (cont.)**
    - Rearranging the position of the states



NONDETERMINISM

- **Example:**
  - All strings over {0,1}* that have a “0” in the second to the last position.

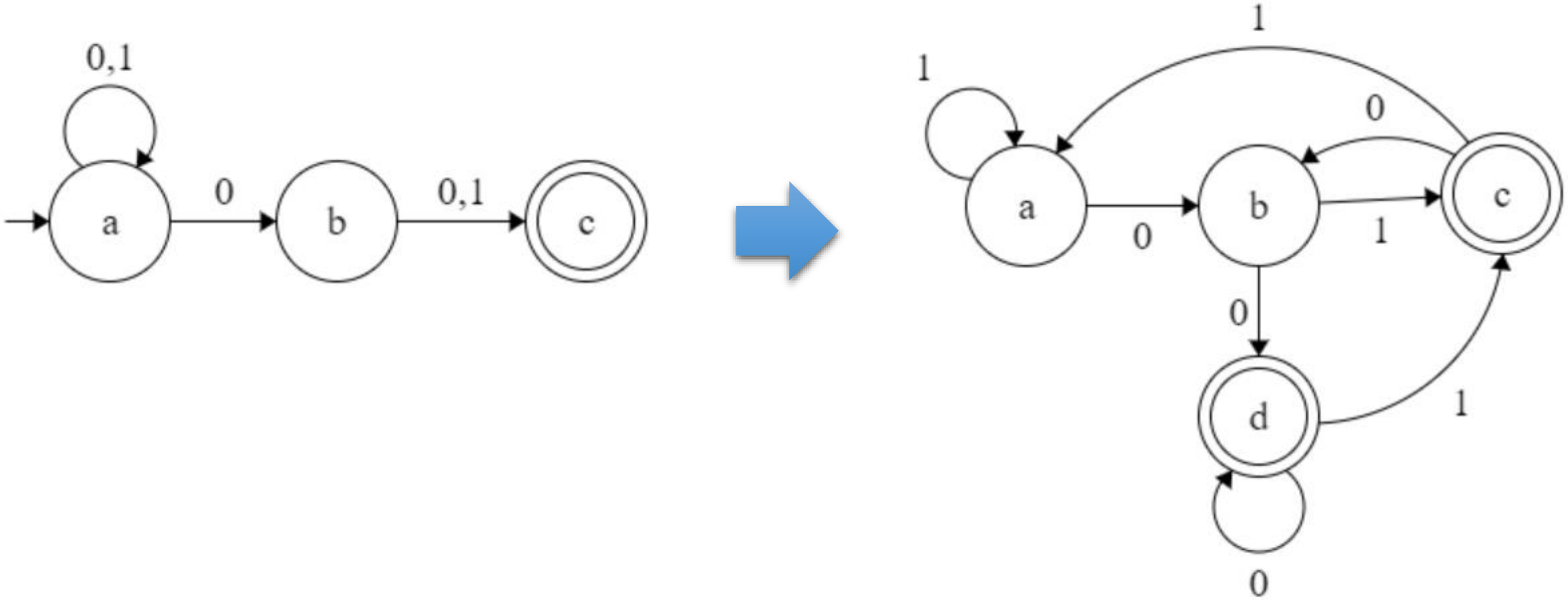




NONDETERMINISM

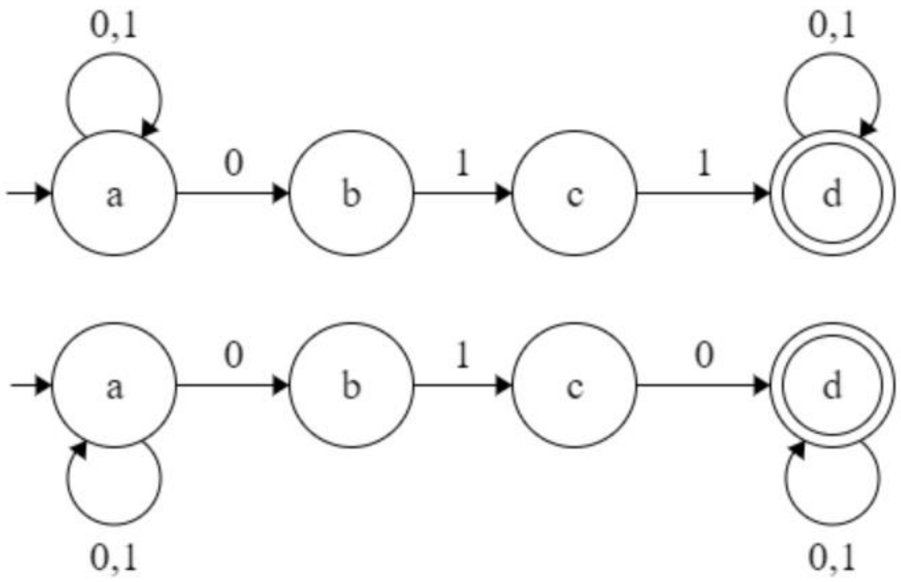
- **Example:**
  - All strings over {0,1}* that have a “0” in the second to the last position.

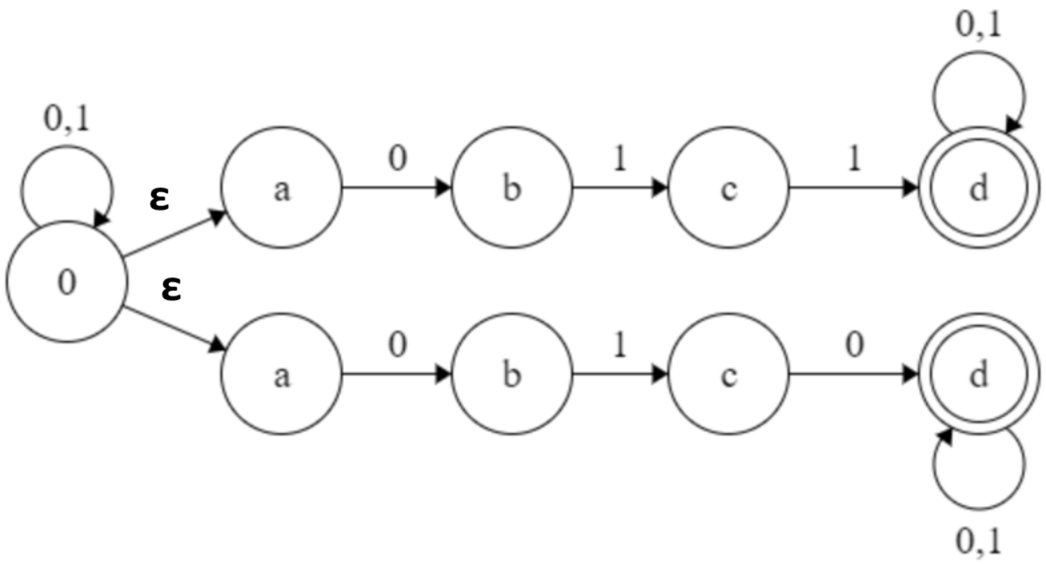
After removing the unreachable states and renaming the states.



NONDETERMINISM

- **Example:**
  - All strings over {0,1}* that contain either 010 or 011.
    - **Questions:**
      - When to start looking for?
      - Which string to look for?
        - Nondeterminism is a useful tools to design this machine.





- **Find DFA for the machine above.**

CLOSURE UNDER THE REGULAR OPERATIONS

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

- Proof Idea: **proof by construction(a new proof)**

  - To prove that $A_1 \cup A_2$ is regular, we demonstrate an **NFA**, call it $\mathrm{N} = (Q, \Sigma, \delta, q_0, F)$, that recognizes $A_1 \cup A_2$.

  - Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

  - We construct N from N_1 and N_2.

- If one of them accepts the input, N will accept it, too.

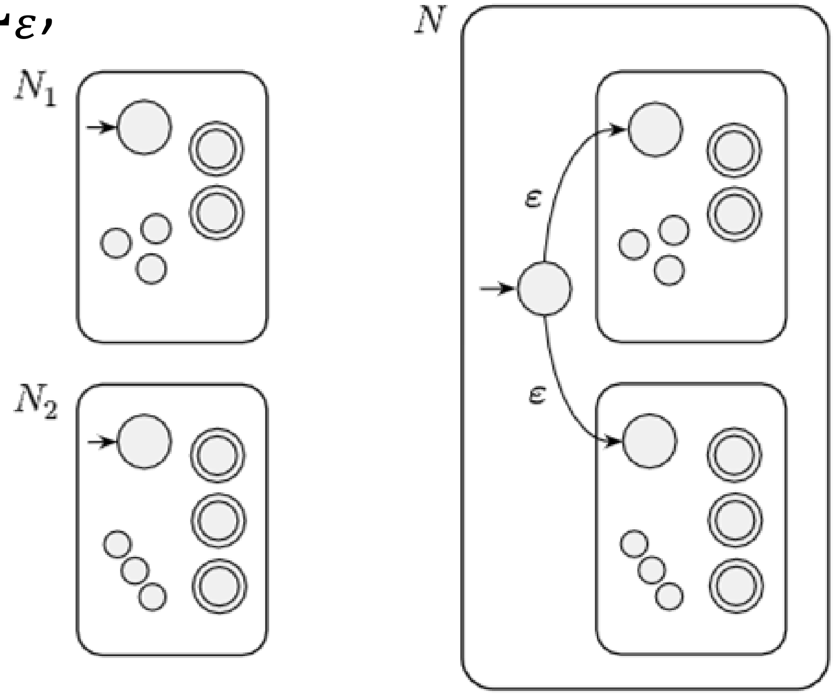
NONDETERMINISM

Theorem

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

- Proof
    1. $Q = \{q_0\} \cup Q_1 \cup Q_2$. (a new start state q0 is added)
    2. q_0 the start state of N
    3. $F = F_1 \cup F_2$.
    4. Define δ so that for any q ∈ Q and any a ∈ Σ_ε,

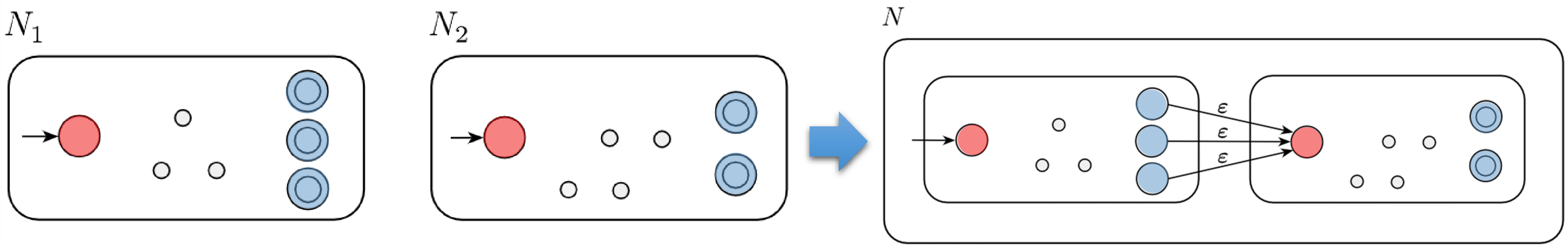
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$



NONDETERMINISM

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

- Proof Idea : (proof by construction)
  - Let two **NFAs** $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_1 and A_2 in order.
  - We construct $\text{N} = (Q, \Sigma, \delta, q_0, F)$ from N_1 and N_2 that recognizes $A_1 \circ A_2$.
  - combine them into a new **NFA** N as we did for the case of union
    - but in a different way.
  - N accepts when the **input** can be **split into two parts**, the first accepted by N_1 and the second by N_2

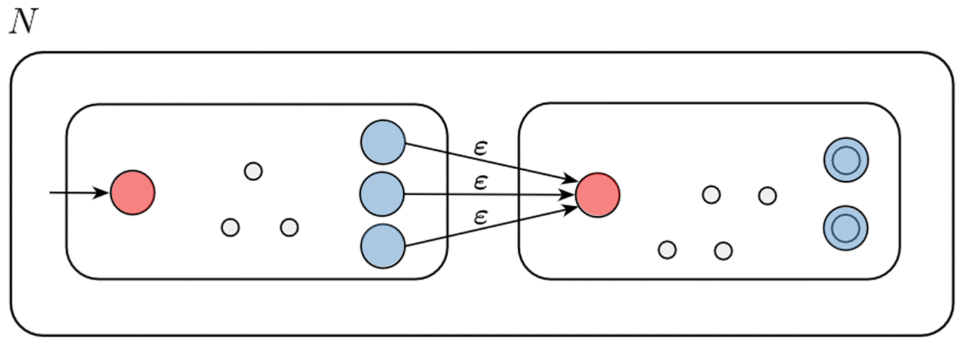


NONDETERMINISM

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

- Proof
  1. $Q = Q_1 \cup Q_2$
  2. *The state q_1 is the same as the start state of N_1.*
  3. The accept states F are the same as the accept states of N_2.
  4. Define δ so that for any q ∈ Q and any a ∈ Σ_ε,

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

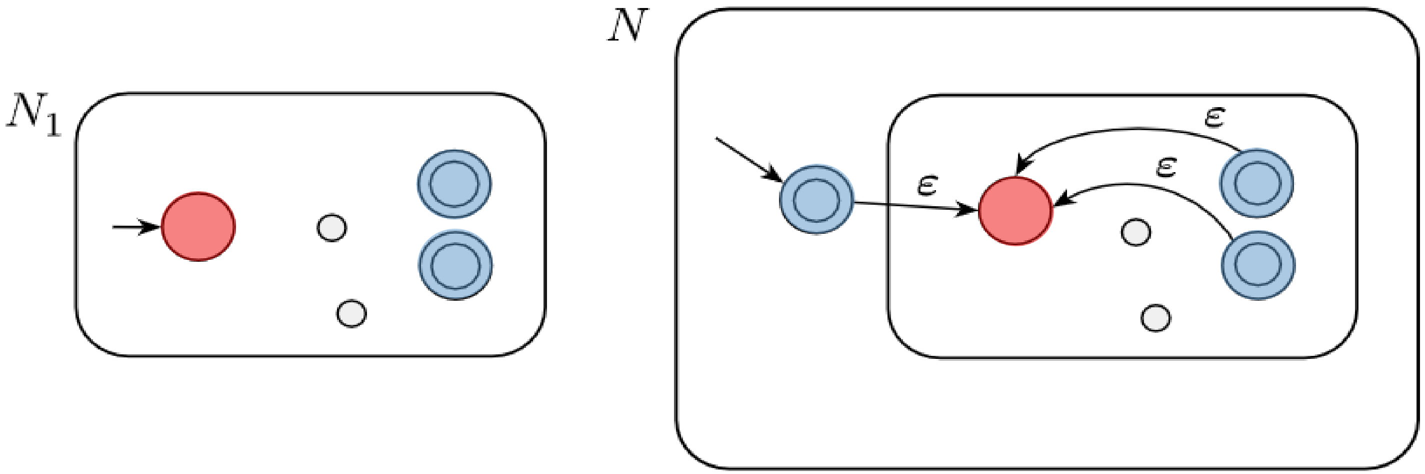


Chapter 1

NONDETERMINISM

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

- Proof Idea : (proof by construction)
  - Modifying an **NFA** $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes A_1 to $N = (Q, \Sigma, \delta, q_0, F)$ that recognize A_1^*.
  - The resulting **NFA** N will **accept** its input whenever it can be **broken into several pieces** and $\boldsymbol{N_1}$ accepts each piece.



Chapter 1

NONDETERMINISM

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

- **Proof :** (proof by construction)
  1. $Q = Q_1 \cup \{q_0\}$
  2. The state q_0 is the new **start state**.
  3. $F = \{q_0\} \cup F_1$.
  4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$

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