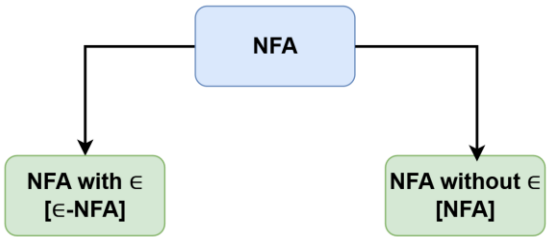
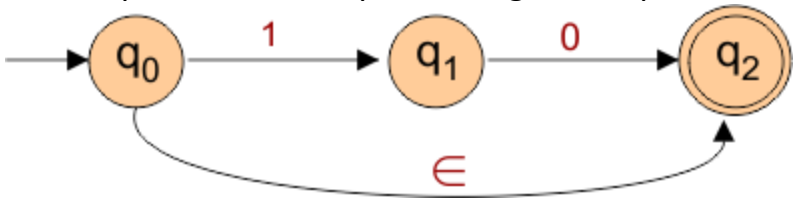
Introduction to NFA (Non-deterministic Finite Automata)

Table of content:

❑ Non-Deterministic Finite Automata (NFA) Representation



1. Example of NFA with Epsilon: The following is an example of Non-Deterministic Finite Automata with epsilon that accepts a string 10 or epsilon.



The following is the explanation of 5 tuples of the above-given NFA

- Q : {q0, q1, q2} are set of states
- ∑ : {0, 1} are the set of input alphabets
- δ refers to the transition function
- S: q0 refers to the initial state
- F: {q2} refers to the set of final state

Transition function δ is defined as

- δ (q0, 1) = q1
- δ (q0, ∈) = q2
- δ (q1, 0) = q2

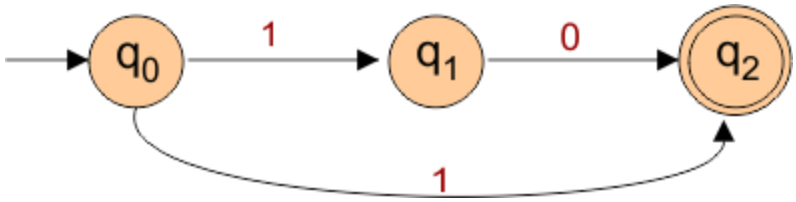
The following table shows the Transition table:

States / Alphabets	0	1	∈
q0	–	q1	q2
q1	q2	–	–
q2	–	–	–

Introduction to NFA (Non-deterministic Finite Automata)

2. Example of NFA without Epsilon

The following is an example of Non-Deterministic Finite Automata without epsilon that accepts a string 10 or 1.



The following is the explanation of 5 tuples of the above-given NFA

{ Q{ q0,q1,q2}, ∑{0,1}, δ, S{q0}, F{q2} }

where

- {q0,q1,q2} are set of states
- {0,1} are set of input alphabets
- δ refers to the transition function
- q0 is the initial state
- {q2} represent the final state

Transition function δ can also defined as

- δ (q0, 1) = q1
- δ (q0, 1) = q2
- δ (q1, 0) = q2
-

The transition Table for the above NFA is

States / Alphabets	0	1
q0	–	q1, q2
q1	q2	–
q2	–	–

Important

- Every NFA is converted into its equivalent DFA.
- Every DFA is an NFA, but not every NFA is a DFA.

Introduction to NFA (Non-deterministic Finite Automata)

❑ Formal definition of NFA

❑ Practice problems in NFA

1. Design an NFA with Σ = {0, 1} for all binary strings where the second last bit is 1.
2. Please design an NFA with input alphabet Σ = {0, 1} that accepts all the strings that end with 01.
3. Construct an NFA with Σ = {0, 1} in which each string must contain “double ‘1’ is followed by single ‘0’.
4. Design an NFA with Σ = {0, 1} such that every string includes the substring 1101.
5. Draw an NFA with Σ = {0, 1} such that the third symbol from the right is “1”.
6. Construct an NFA with Σ = {0, 1} in which each string must contain “11” followed by “00”.
7. Construct an NFA with Σ = {0, 1}, where each string must contain either “01” or “10”.
8. Construct an NFA with Σ = {0, 1} that accepts all strings that begin with 1 and end with 0.
9. Construct an NFA with Σ = {0, 1} for the language L = $\{0^m1^n \mid m \geq 0$ and $n \geq 1\}$.
10. Construct an NFA with Σ = {0, 1} for the language L = {(01)n | n≥1 }.
11. Construct an NFA with Σ = {0, 1} for the language L = {(01)n | n≥0 }.
12. Construct an NFA with Σ = {a, b, c} where strings contain some “a’s” followed by some “b’s” followed by some c’s.

Introduction to NFA (Non-deterministic Finite Automata)

❑ Formal definition of NFA:

The following is the explanation of 5 tuples (Q, ∑, δ, q_0, F) of NFA where

- **Q** finite set of all states ($q_0, q_1, q_2, \ldots q_n$). Where n is a finite number
- **∑** represents a finite set of symbols called the alphabet. i.e. {0, 1}.
- δ: Q x ∑ → 2^Q represents a transition function
- **q_0** represents the initial state from where any input is processed ($q_0 \in Q$).
- **F** represents a set of final states/states where F will be a subset (⊆) of Q.

❑ Practice problems of NFA:

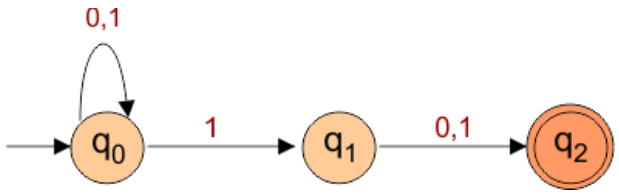
NFA Example 01

Design an NFA with ∑ = {0, 1} for all binary strings where the **second last bit is 1.**

Solution: The language generated by this example will include all strings in which the **second-last bit is 1.**

L= {10, 010, 000010, 11, 101011……..}

The following NFA automaton machine accepts all strings where the **second last bit is 1.**



Where,

- {q0, q1, q2} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

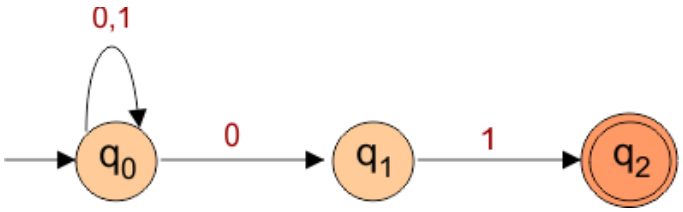
States	0	1
q0	q0	q0,q1
q1	q2	q2
q2	–	–

Introduction to NFA (Non-deterministic Finite Automata)

NFA Example 2

Please design an NFA with input alphabet ∑ = {0, 1} that accepts all the **strings that end with 01.**
Solution: The language generated by this example will include all strings **that end with 01.**

L= {01, 0101, 0000101, 101, 101001……..}
The following NFA automaton machine accepts all strings that end with 01.



Where,

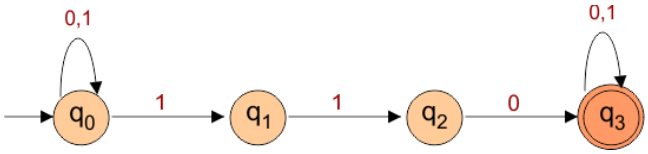
- {q0, q1, q2} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	0	1
q0	q0,q1	q0
q1	–	q2
q2	–	–

NFA Example 3

Construct an NFA with ∑ = {0, 1} in which each string must contain **"double '1' is followed by single '0'.**
Solution: The language generated by this example will include all strings **that must contain "double '1' is followed by single '0'.**
L= {110, 0110, 1110, 10100110……..}



NFA for the above language is Where,

- {q0, q1, q2,q3} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q3} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	0	1
q0	q0	q0,q2
q1	–	q2
q2	q3	–
q3	q3	q3

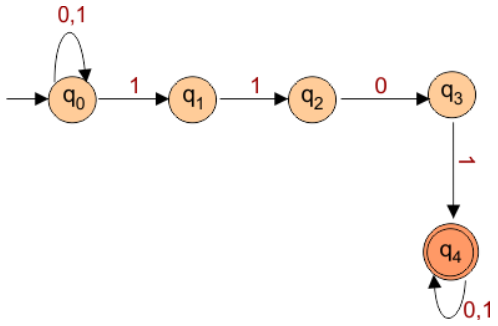
Introduction to NFA (Non-deterministic Finite Automata)

NFA Example 4

Design an NFA with ∑ = {0, 1} such that **every string includes the substring 1101.**

Solution: The language generated by this example will include all strings **that must contain substring 1101.**

L= {1101, 110101, 1101000101, 01101, 1011011……..}



NFA for the above language is Where,

- {q0, q1, q2,q3,q4} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q4} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

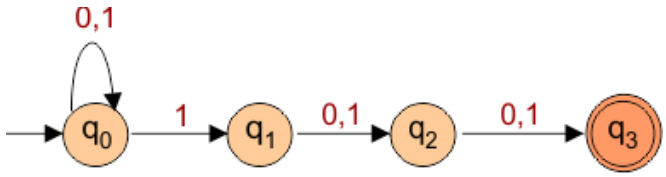
States	0	1
q0	q0	q0,q1
q1	–	q2
q2	q3	–
q3	–	q4
q4	q4	q4

NFA Example 5

Draw an NFA with ∑ = {0, 1} **such that the third symbol from the right is "1".**

Solution: The language generated by this example will include all strings **where the third symbol from the right is "1".**

L= {100, 111, 101, 110, 0100, 1100, 00100, 100101……..}



NFA for the above language is Where,

- {q0, q1, q2} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q3} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is –

States	0	1
q0	q0	q0,q1
q1	q2	q2
q2	-q3	q3
q3	–	–

Introduction to NFA (Non-deterministic Finite Automata)

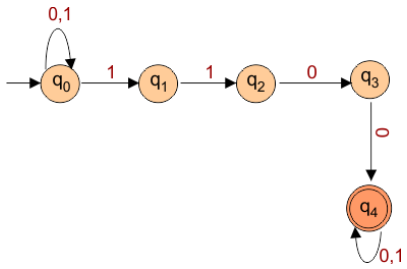
NFA Example 6

Construct an NFA with ∑ = {0, 1} in which **each string must contain "11" followed by "00".**

Solution

The language generated by this example will include **each string must contain "11" followed by "00".**

L= {1100, 01100, 11100, 011001, 00110001 ……..}



NFA for the above language is Where,

- {q0, q1, q2, q3, q4} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q4} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is

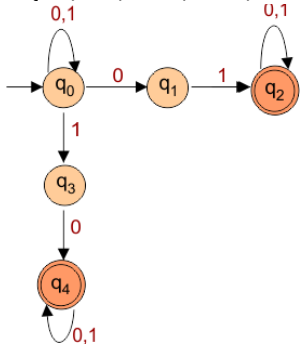
States	0	1
q0	q0	q0,q1
q1	–	q2
q2	q3	–
q3	q4	–
q4	q4	q4

NFA Example 7

Construct an NFA with ∑ = {0, 1}, where **each string must contain either "01" or "10".**

Solution: The language generated by this example will include **each string must contain either "01" or "10".**

L= {01, 10, 001, 110, 1110, 0001……..}



NFA for the above language is Where,

- {q0, q1, q2, q3, q4} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2,q4} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is

States	0	1
q0	q0,q1	q0,q3
q1	–	q2
q2	q2	q2
q3	q4	–
q4	q4	q4

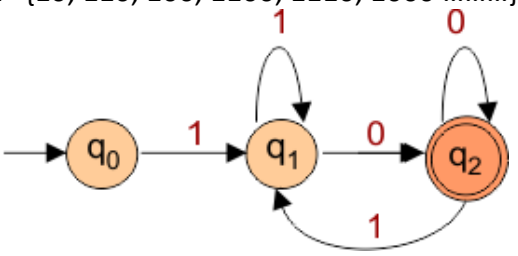
Introduction to NFA (Non-deterministic Finite Automata)

NFA Example 8

Construct an NFA with ∑ = {0, 1} that accepts all strings that **begin with 1 and end with 0.**

Solution: The language generated by this example will include all strings that **begin with 1 and end with 0.**

L= {10, 110, 100, 1100, 1110, 1000 ……..}



NFA for the above language is Where,

- {q0, q1, q2} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	0	1
q0	–	q1
q1	q2	q1
q2	q2	q1

NFA Example 9

Construct an NFA with ∑ = {0, 1} for the language L = {0^m1^n | m **≥0 and n≥1** }.

Solution: The language-generated strings will be like as

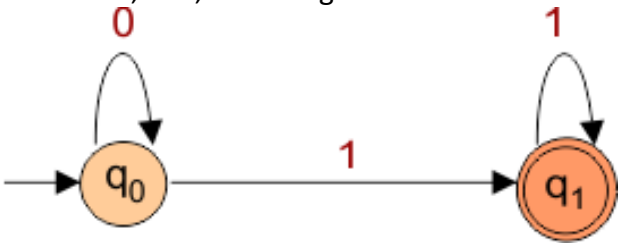
When m=0 and n=1, the string will be 1 because 0^01^1=1.

When m=1 and n=2, then the string will be 1 because 0^11^2=011.

When m=2 and n=3, then the string will be 1 because 0^21^3=00111.

By choosing a random value, i.e.,

When m=3, n=2, the string will be 1 because 0^31^2=00011.And so on.



So, the NFA transition diagram for the above language is Where,

- {q0, q1} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q1} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	0	1
q0	q0	q1
q1	–	q2

Introduction to NFA (Non-deterministic Finite Automata)

NFA Example 10

Construct an NFA with ∑ = {0, 1} for the language L = $\{(01)^n \mid \mathbf{n \geq 1}\}$.

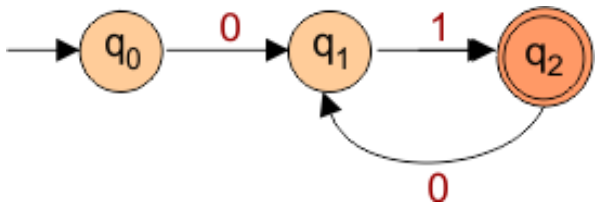
Solution

The language-generated strings will be like as

When n=1, the string will be 01 because $(01)^1$=01.

When n=2, the string will be 0101 because $(01)^2$=0101.

By choosing a random value, i.e., n=5, then $(01)^5$=0101010101. And so on.



So, the NFA transition diagram for the above language is Where,

- {q0, q1, q2} refers to the set of states
- {0,1} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	0	1
q0	q0q1	q0
q1	–	q2
q2	–	–

NFA Example 11

Construct an NFA with ∑ = {0, 1} for the language L = $\{(01)^n \mid \mathbf{n \geq 0}\}$.

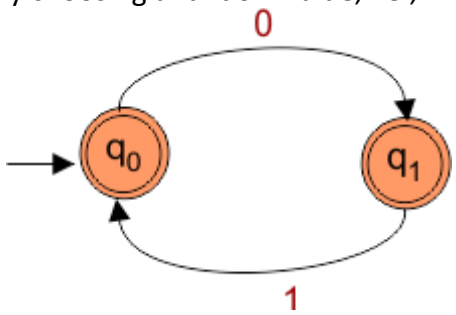
Solution

The language-generated strings will be like as

When n=1, the string will be 01 because $(01)^0$ = null.

If n=1, the string will be 01 because $(01)^1$=01.

By choosing a random value, i.e., n=4, the result will be $(01)^4$=01010101. And so on.



So, the NFA transition diagram for the above language is Where,

- •q0, q1} refers to the set of states
- •{0,1} refers to the set of input alphabets
- •δ refers to the transition function
- •q0 refers to the initial state
- •{q1} refers to the set of final states

- **Transition Table for the above Non-Deterministic Finite Automata is-**

States	0	1
q0	q1	–
q1	–	q0

Introduction to NFA (Non-deterministic Finite Automata)

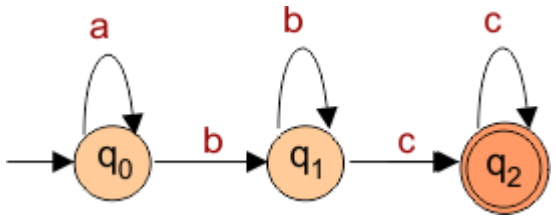
NFA Example 12

Construct an NFA with ∑ = {a,b,c} where strings **contain some "a's" followed by some "b's" followed by some c's.**

Solution

The language-generated strings will be like as

L={abc, aabbcc, aaabbcc, aabbbcc, aaaabbbbccc, …… }



So, the NFA transition diagram for the above language is Where,

- {q0, q1, q2} refers to the set of states
- {a,b,c} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	a	b	c
q0	q0	q1	–
q1	–	q1	q2
q2	–	–	q2

Introduction to NFA (Non-deterministic Finite Automata)

❑ Formal definition of ∈-NFA

❑ Practice problems in ∈-NFA

1. Draw a Finite Automata which accepts the string “ab”.
2. Draw a Finite Automata which accepts the string “a or b”.
3. Draw a Finite Automata that accepts the string “a, b or c”.
4. Draw a Finite Automata that accepts the string “a*”.
5. Construct an NFA with ∑ = {a,b,c, ε} where strings contain some “a’s” followed by some “b’s” followed by some “c’s.

Introduction to NFA (Non-deterministic Finite Automata)

❑ Epsilon NFA (∈-NFA)

Epsilon NFA (∈-NFA) is similar to the NFA but with a little difference in that it has an epsilon (∈) transition that NFA doesn't have. A Transition that does not consume any input symbol is called an ε-transitions.

- **∈-NFA s**tate diagrams usually label Epsilon with the Greek letter "∈". It is also called lambda.
- ∈-transitions give a convenient way of designing the machine systems.
- Due to ε-transitions, the first string of language may be empty.

Formal Definition of Epsilon-NFA

The formal definition of **∈**-NFA is represented through 5-tuple (Q, ∑, δ, q_0, F) where,

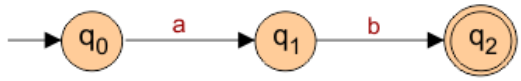
- **Q** is a finite set of all states ($q_0, q_1, q_2, \ldots, q_n$) where n is a finite number.
- **∑** Is a finite set of symbols called the alphabet. i.e. {0, 1}.
- **δ : Q x (∑ U ∈)** → 2^Q is a total function called as transition function.
- **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
- **F** is a set of final states where F will be a subset (⊆) of Q.

❑ Practice problems of ∈-NFA:

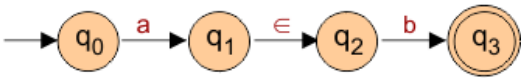
Example 01

Draw a Finite Automata which accepts the string "ab".

NFA



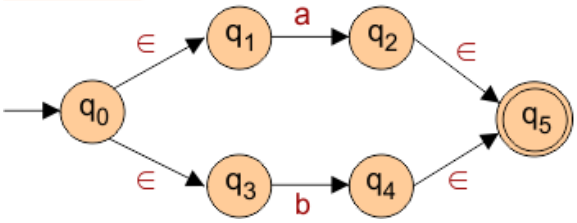
∈- NFA



Example 02

Draw a Finite Automata which accepts the string "a or b".

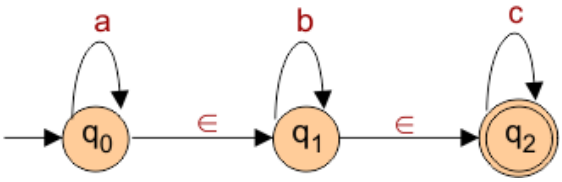
∈- NFA



Example 03

Draw a Finite Automata that accepts the string "a, b or c".

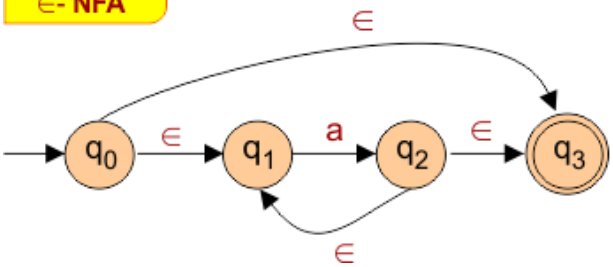
∈- NFA



Example 04

Draw a Finite Automata that accepts the string "a*".

∈- NFA



Introduction to NFA (Non-deterministic Finite Automata)

Example 5

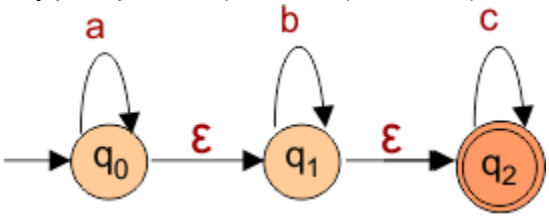
Construct an NFA with **∑ = {a,b,c, ε}** where strings **contain some "a's" followed by some "b's" followed by some "c's.**

Note: It is now possible to transition through ε because it is being held by ∑.

Solution

The language-generated strings will be like as

L={**ε, abc,** aabbcc, aaabbcc, aabbbcc, aaaabbbbccc, …… }



So, the NFA transition diagram for the above language is Where,

- {q0, q1, q2} refers to the set of states
- {a,b,c} refers to the set of input alphabets
- δ refers to the transition function
- q0 refers to the initial state
- {q2} refers to the set of final states

Transition Table for the above Non-Deterministic Finite Automata is-

States	a	b	c
q0	q0	q1	–
q1	–	q1	q2
q2	–	–	q2

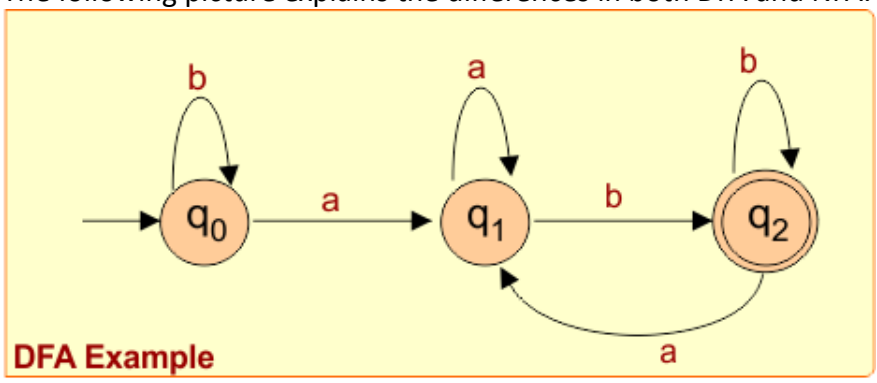
Introduction to NFA (Non-deterministic Finite Automata)

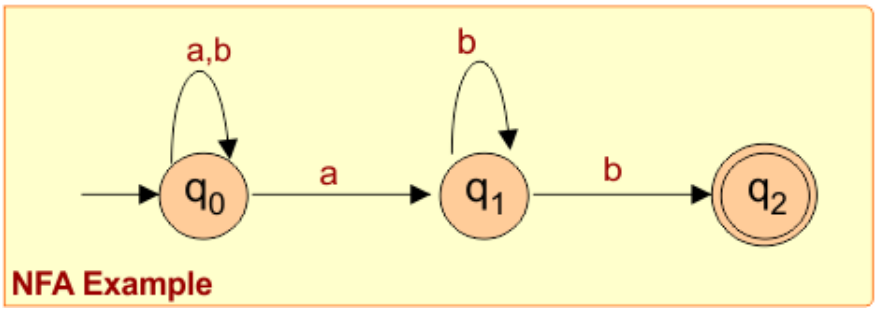
❑ Difference Between DFA and NFA

There is not much difference between DFA and NFA because they both accept regular languages. The basic difference in both of these is their transition function is different.

- In DFA, The transition function (δ) takes two arguments (state, symbol) as input and returns **a single state as output.**
- In NFA, The transition function (δ) takes two arguments (state, symbol) as input and **can return multiple states.**

The following picture explains the differences in both DFA and NFA.





Key difference in DFA and NFA

Let explain all Key difference in detail for DFA and NFA

1. Transition Function

Both NFA and DFA can be defined using 5 tuples. The only **difference** between the 5 tuples is the **transition function.** The transition functions for NFA and DFA are given below

DFA five tuples are (Q, Σ, δ, q0, F) where transition function is

$\delta : Q \times \Sigma \Rightarrow Q$

NFA five tuples are (Q, Σ, δ, q0, F) where transition function is

$\delta : Q \times (\Sigma \cup \varepsilon) \Rightarrow 2^Q$

In DFA transition function, when an input is given at any state Q, then the next possible output state belongs to Q also.

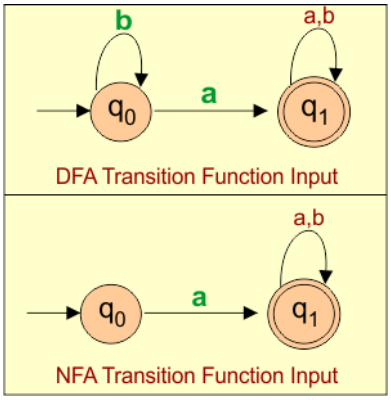
But In NFA transition function, when an input is given at any state Q, then the next possible output states belongs to 2^Q.

2. Transition Function Input

In a DFA diagram it is compulsory to go to a state for every input symbol where as in NFA diagram it is not compulsory to go to a state for every input symbol.

Introduction to NFA (Non-deterministic Finite Automata)

Example: Let suppose Sigma (input symbols) values are {a,b}, states {q0, q1}, q0 initial and q2 is final state for both NFA and DFA.



Note: The above given NFA and DFA diagrams are **not equivalent**, selected just for understanding the given concept

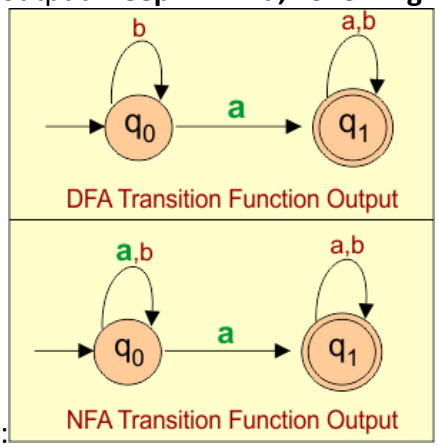
Explanation

- **For DFA:** at state "q0" and q1, it is necessary to tell the path for all input symbols "a" and "b".
- **For NFA:** at state "q0" and q1, it is not necessary to tell the path for all input symbols "a" and "b". As in above diagram, only path of input symbol "a" at state "q0" is mentioned, **but "b" is missing.**

3. Transition Function Output

In DFA, there is only one option to go for next state against an input symbol, from any state. So output is exactly one state. But In NFA, there are multiple options to go to the next state against a single input symbol, from any state. So, output leads towards many states.

For example: Look at the following two NFA and DFA diagrams to explain the transition function output**. Keep in mind, Following** NFA and DFA diagrams are **not equivalent**



:

Explanation:

- In DFA diagram the transition for input "a" at state q0 go to next state "q1". So, output is q1 against the input "a" at state q0.
- But In NFA diagram the transition for input "a" at state "q0" go to next state "q1" and loop itself at a time. So, output is q0q1 against the input "a" at state q0.

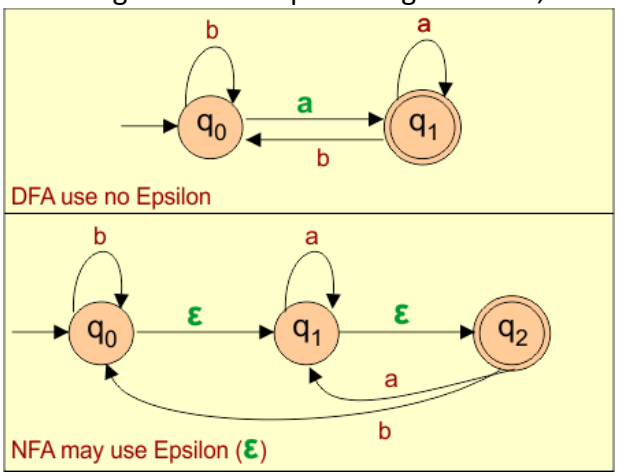
Introduction to NFA (Non-deterministic Finite Automata)

4. Epsilon (Empty) String

As epsilon (e) means nothing or empty.

- DFA is a machine that cannot move to another state without using any input symbol. That's why DFA cannot go to next state on epsilon moves.
- But NFA is a machine that can move to another state without using epsilon as input symbol.

Following is the descriptive diagram for it,



In above diagrams,

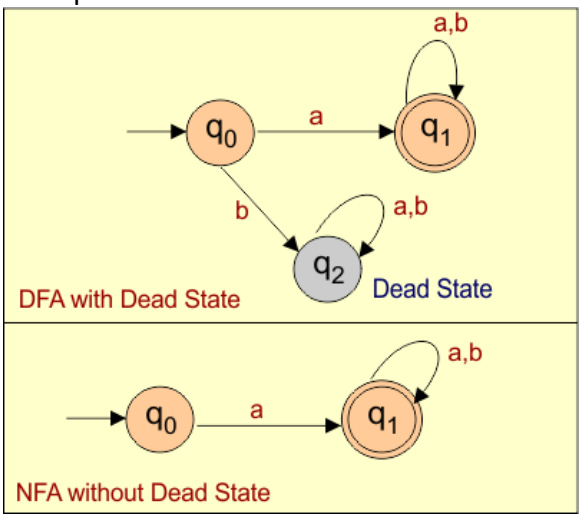
- For DFA, Input "a" is compulsory at state "q0" to reach at final state.
- For NFA, by using epsilon (empty) as input we can reach at final state.

5. Trap State / Dead State

A dead state is also called **trap state, reject state or halt state**. Once the transition enters into a dead state, there is no way for it to go out anywhere for further progress. It is often used in DFA model where the automaton should halt or reject certain inputs. So,

- Dead states is more common concept in DFAs
- But less common in NFA, it can also be identified in NFAs.

Example: Draw a machine for NFA and DFA for the regular expression R = a(a+b)*



Explaintion: q2 is dead state in DFA diagram because once the transition enters into the q2, it will halt there because there is no chacnce to reach at final state.

Introduction to NFA (Non-deterministic Finite Automata)

6. NFA and DFA Conversions

Every DFA is and NFA but not vice versa. But there is an equivalent DFA for every NFA.

NFA to DFA conversion

Every NFA can be converted to its equivalent DFA

- **NFA to equivalent DFA conversions** is done by using methods such as the subset construction algorithm.
- The NFA-to-DFA conversion process involves creating subsets of NFA states, with each subset representing a state in the DFA. The DFA transitions are determined by the original NFA transitions.
- While converting NFA to DFA, if the NFA maximum states are n, then the resulting DFA can maximum numbers of **states are 2^n.**
- After conversion, the resulting DFA may have redundant or unreachable states. DFA minimization algorithms can be applied to **reduce the number of states** in the DFA.
- The DFA constructed from an NFA recognizes the **same language as the original NFA.**
- While converting NFA to DFA, the number of states in DFA can grow exponentially up to 2^n. This attribute leads towards **exponential in the worst case.**

DFA to NFA Conversion

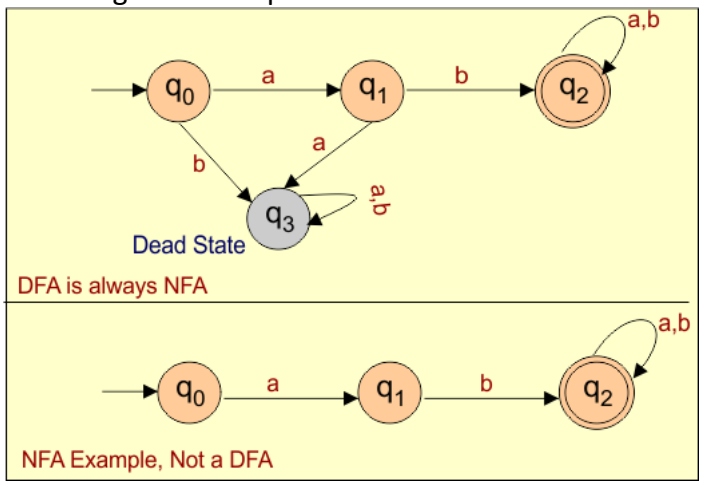
Every DFA is already an NFA, So no need to convert it into NFA. If we think to minimize DFA then it is happened by removing **the possible redundant transitions**. After minimizing DFA, it may be a NFA but no longer a DFA.

- **Converting a DFA to an NFA** does not result in an increase in the number of states
- **The NFA obtained from the DFA** recognizes the same language as the original DFA.
- **Time and space complexity** is linear, it may remain same or bit reduced.

7. Every DFA is NFA But Not Vice Versa

Every DFA is an NFA that's why we can convert every DFA to its equivalent NFA but not vice versa.

Following are two equivalent DFA and NFA which accept all strings starting with "ab".



- DFA diagram meets all the conditions of NFA diagram
- but NFA diagram does not satisfy all the rules for DFA construction. As, in NFA diagram at state q0, the only the transition for input "a" is made. There is no transitions is made for input "b", so this NFA is not a DFA.

Introduction to NFA (Non-deterministic Finite Automata)

8. Digital Computer Usage

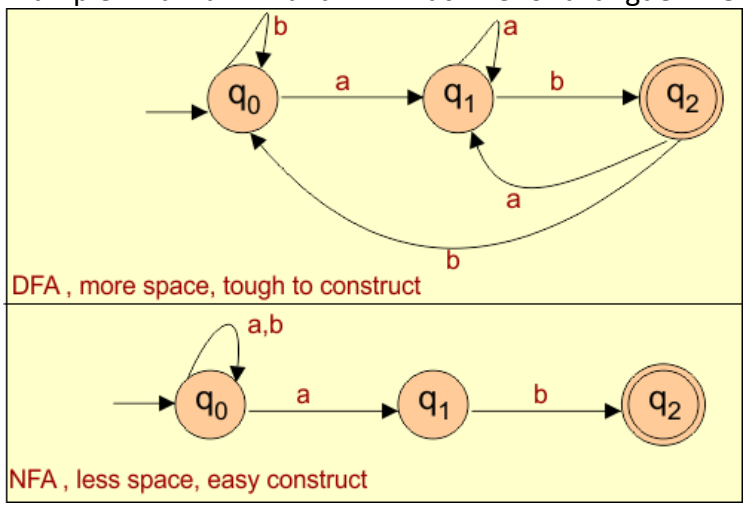
NFAs can have multiple next states for a single input symbol. This introduces non-determinism, which is not desirable for digital computers.

- All digital computers are deterministic because DFA provides a clear path for each input.
- NFA does not provide a path for each input, so NFA is not used in digital computers.

9. Space and Construction

As DFA may constrains some dead states and necessary transitions against each input which leads toward move space and complexity to construct DFA machines. And on the other hand, NFA generally contains no dead state. So, NFA consume less space and easy to construct as compare to DFA.

Example: Draw a NFA and DFA machine for a langue where each string end with “ab”.



Two equivalent NFA and DFA example which show the more space consumption in terms of DFA as compare to NFA.

10. Time and Space complexity

While converting NFA to DFA, the number of states in DFA can grow exponentially up to 2^n. This attribute leads towards exponential in the worst case.

While converting DFA to NFA, Time and space complexity is linear, it may remain same or bit reduced because the number of states are almost same.

Similarities in NFA and DFA

- **Four tuples out of five** are similar for both DFA and NFA except transition.
- Both DFA and NFA **accept the regular languages**
- NFA and DFA contains **finite number of states**
- NFA and DFA both can hold **more than one final states**
- **Power of DFA and NFA** is same because both types of automata are equivalent in terms of the languages they can recognize. This equivalence theorem is known as the **“subset construction”**. It states that every NFA has an equivalent DFA that recognizes the same language, and vice versa.
- **All DFA’s are NFA’s but not vice versa.**

Introduction to NFA (Non-deterministic Finite Automata)

Summary Table for DFA and NFA Difference

Term	DFA	NFA
Stand For	DFA stands for **Deterministic Finite Automata.**	NFA stands for Nondeterministic Finite Automata.
For Every Input	**For each sigma** value (0, 1…n) there must be a provided path from each state.	**For each sigma value** (0, 1…n), there is no need to specify the path from each state
Digital Computers	DFA provides a path for each input, so **all digital computers are deterministic.**	NFA does not provide a path for each input, so NFA is not used in digital computers.
Possible Transitions	There is only **one** possible transition from **one state on the one input symbol.** It means multiple choices corresponding to an input are not available.	There is **more than one** possible transition from **one state on the same input symbol.** It means multiple choices are available corresponding to a single input.
Epsilon	**DFA cannot use Empty String**. (Epsilon) transition.	NFA can use an Empty String (Epsilon) transition.
Dificulty	DFA is more difficult to construct and understand.	NFA is easier to construct and understand.
Visibility	All DFA are NFA.	All NFA are not DFA.
Space	DFA requires more space. Because it has normally more states.	NFA requires less space than DFA. Because it usually has fewer states
Dead State	Dead State are more common in DFA	Dead State are less common in NFA
Complexity	DFA to NFA conversion, time and space complexity remains almost same or bit reduced	NFA to DFA conversion, time and space complexity increased to exponentional.

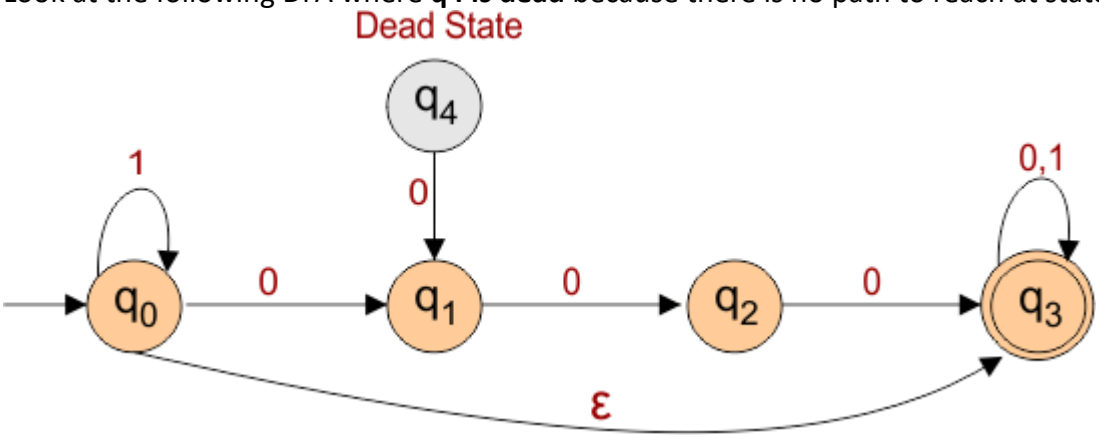
Introduction to NFA (Non-deterministic Finite Automata)

❑Dead State in TOC

When there is no path to reach a certain state, then that state will be considered a dead state in TOC. It may be **found in NFA** but cannot be used in DFA**.** It is also known as a **dead configuration or trap state.**

Example 1 : [q4 is deat state]

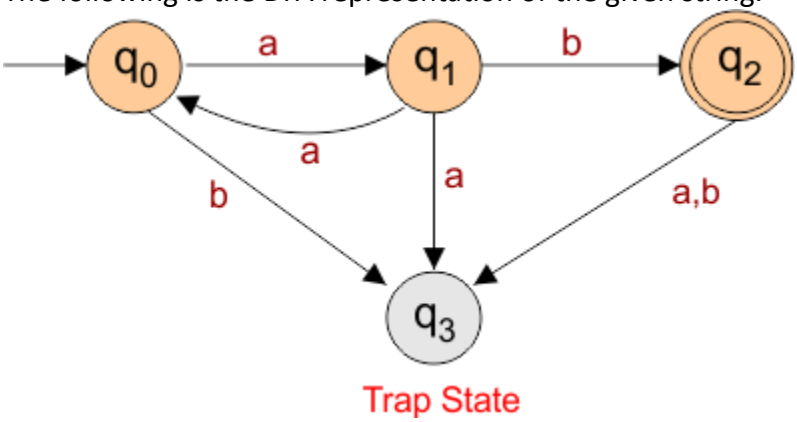
Look at the following DFA where **q4 is dead** because there is no path to reach at state q4.



Note: If dead state is non-final then it can be removed without any problem from NFA graph
The trap state considered the receiving inputs as rejected inputs. **This concept is used in DFA. It is also known as trap configuration.**
As we know, in the case of DFA, there must be a dedicated path for each input of Sigma. So, some of the inputs have to be rejected according to requirements.

Example 02: Draw a DFA that accepts only the string "aaab".

The following is the DFA representation of the given string.



According to DFA, the initial state is q0, and it changes to q1 through input "a", but for input "b", it does not need for self-loop. So, input "b" transition goes to the trap state (q3).
Therefore, the transition of input "b" at state q0, input "a" at state q1, and both input "a" and "b" at q2 leads toward the trap state (q3).