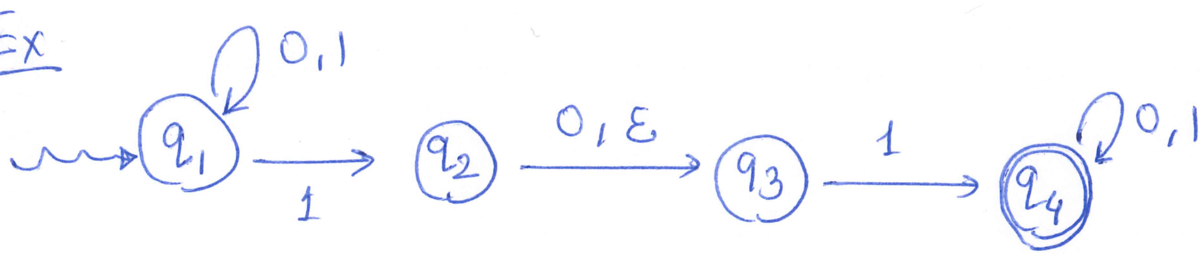
Non-Deterministic Finite Automata (NFA)

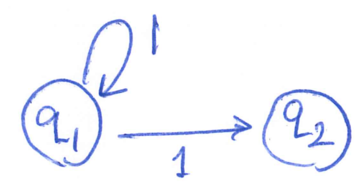
In a DFA, the current state and the input symbol read, determine a unique next state. In an NFA, one or more or none next states are possible and in addition, a state may be changed without reading an input symbol (referred to as "ε-move").

Ex

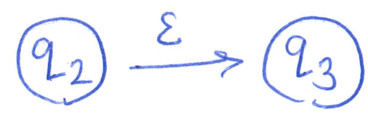


Note

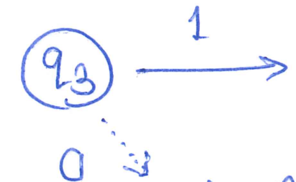
- More than one moves.



- ε-moves.



- Move missing

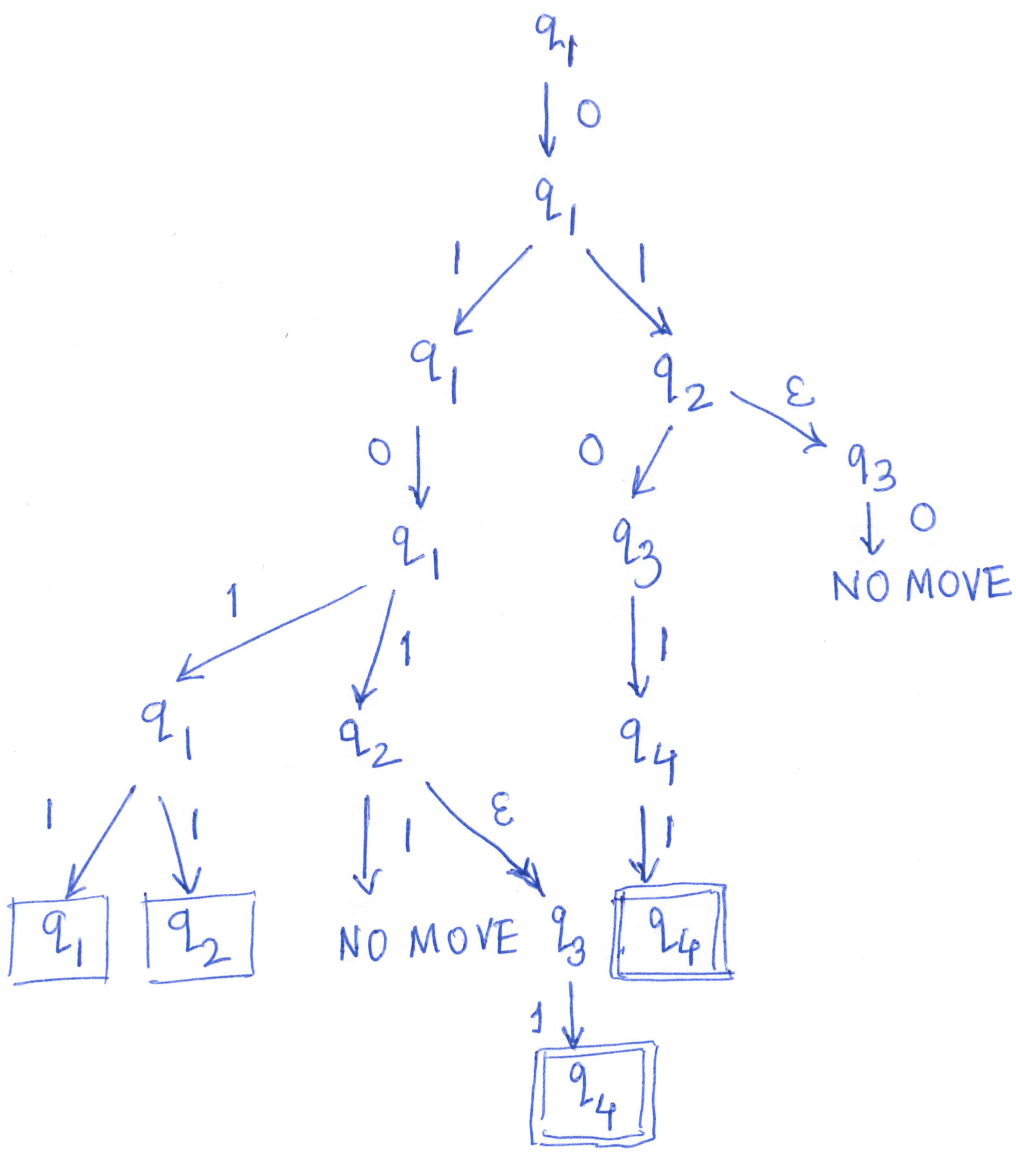


No move here.

In state q_1 and input $\underline{1}$, the NFA can stay in state q_1 <u>or</u> change state to q_2.

On given input, an NFA has several possible computation paths, forming a computation tree.

E.g. on input <u>01011</u>



Thus on input <u>01011</u> the NFA computation path

- Either has no move at some step, tantamount to REJECT
- or ends in states q_1 or q_2 and REJECTS
- or ends in state q_4 and ACCEPTS (since q_4 is designated as an accept state.)

Def An NFA N is said to accept input $x \in \Sigma^*$ **iff** there exists at least one computation path of N on input x that ends in an accept state.

Note The above NFA accepts 101, 11, 01011, ...

It does not accept 100, ...

(verify!)

It is not difficult to see that it accepts precisely those strings that have <u>11</u> or <u>101</u> as a consecutive substring.

We'll see that every NFA N has an equivalent DFA M, in the sense that N and M accept precisely the same set of inputs, i.e. that $L(N) = L(M)$ where $L(N)$, the language $\frac{\text{recognized}}{\text{accepted}}$ by N is

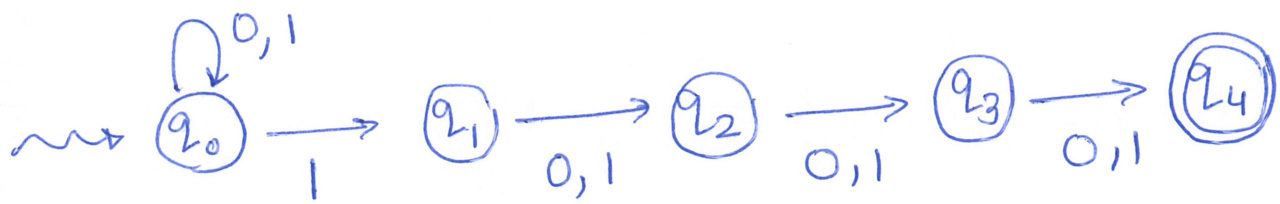
$$L(N) = \left\{ x \in \Sigma^* \;\middle|\; \begin{array}{l} N \text{ accepts } x, \text{ i.e. } N \text{ has} \\ \text{at least one computation} \\ \text{path on } x \text{ that accepts.} \end{array} \right\}$$

Constructing NFA is often easier.

Ex. Let $\Sigma = \{0,1\}$,

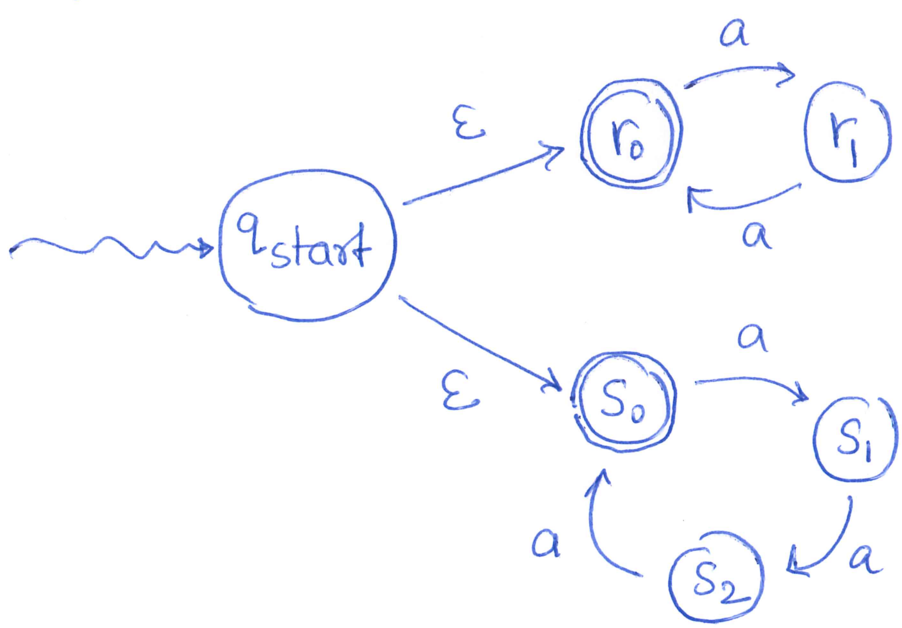
$L = \{ x \mid$ the fourth symbol of x from the end is $\underline{1} \}$.

L is regular and the DFA recognizing it must have at least 16 states. However an NFA is easily constructed.



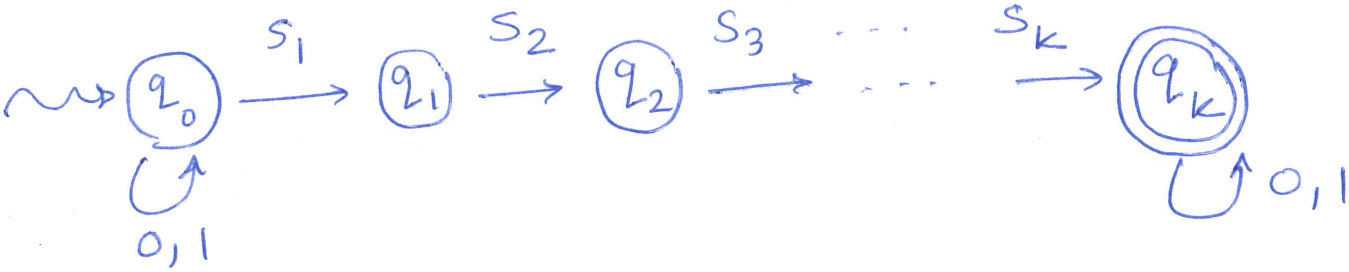
Ex. Let $\Sigma = \{a\}$.

$L = \{ x \mid |x|$ is multiple of 2 or 3 $\}$.



Ex. Let $\Sigma = \{0,1\}$. $s_1 s_2 \cdots s_k \in \{0,1\}^k$ be a fixed string/pattern,

$L = \{ x \mid x \text{ has } s_1 s_2 \cdots s_k \text{ as consecutive substring} \}$.



Formal Definition of NFA

Def An NFA N is a 5-tuple $N = (Q, \Sigma, \delta, q_1, F)$ where

- Q is a finite set of states.
- Σ " alphabet
- q_1 is the start state.
- $F \subseteq Q$ is the subset of accept states.
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow \mathcal{P}(Q)$

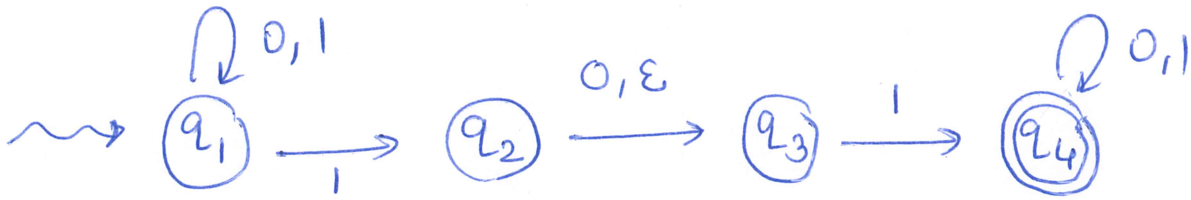
is the transition function.

Note

- $\delta(q, a)$ is the subset of next states on current state q and input symbol a. This allows one or many or none possible moves, none if $\delta(q, a) = \phi$.

- $\delta(q, \varepsilon)$ allows ε-moves.

Ex.



- $Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$,
  $F = \{q_4\}$.

-

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

Acceptance by an NFA (formal definition)

Def Let $N = (Q, \Sigma, \delta, q_1, F)$ be an NFA and $x \in \Sigma^*$ be an input. N accepts x iff

- x can be written as $x = x_1 x_2 .. x_n$ where $x_i \in \Sigma \cup \{\varepsilon\}$ for $1 \leq i \leq n$ and
- there exist states $r_1, r_2, .., r_{n+1} \in Q$ such that

① $r_1 = q_1$

② $r_{i+1} \in \delta(r_i, x_i)$ for $i = 1, 2, .., n$

③ $r_{n+1} \in F$.

Def $L(N) = \{ x \in \Sigma^* \mid N \text{ accepts } x \}$.

—— x ——

Equivalence of DFAs and NFAs.

Observation Every DFA is also an NFA.

Justification Evident.

Formally $M = (Q, \Sigma, q_1, \delta, F)$, a DFA, is also an NFA where $\delta(q, a)$ is thought of as a singleton ~~stat~~ set for all $q \in Q$, $a \in \Sigma$ and $\delta(q, \varepsilon) = \phi$.

We now focus on proving that:

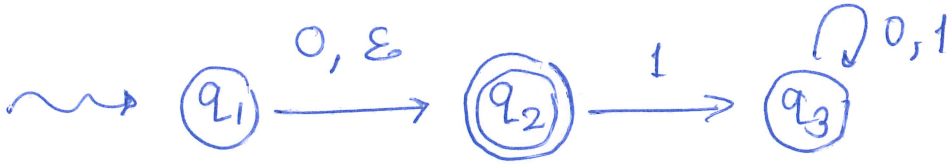
Theorem Every NFA has an equivalent DFA.

Proof The proof consists of two steps.

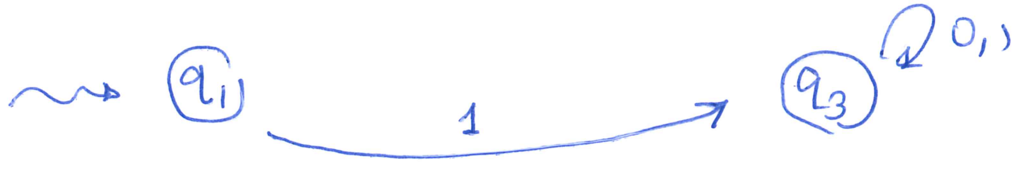
- First, remove ε-moves from the NFA.
- then, construct an equivalent DFA by the "subset construction".

Step ①: Removing ε-moves

Idea Suppose our NFA is

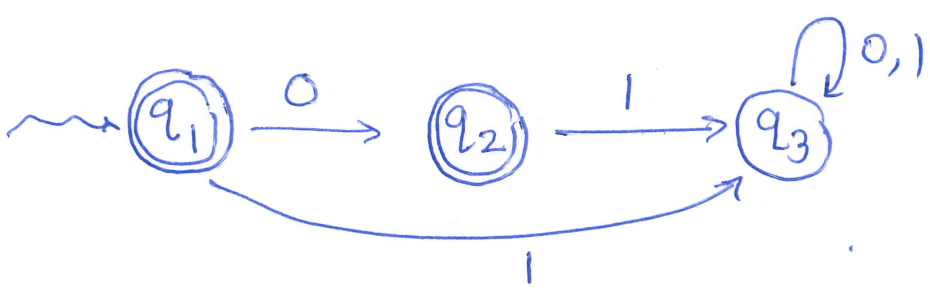


We remove the ε-move $q_1 \xrightarrow{\varepsilon} q_2$. Since in state q_1 and input 1, one could have first moved to q_2 "for free" and then to q_3, we should add the move:



Moreover, any input that makes the NFA end in state q_1 is accepted since one could then move to the accept state q_2 "for free". Thus if we were to remove the ε-move, we better make q_1 an accept state to preserve this "functionality".

Hence, an equivalent NFA (without ε-moves) is:



Actual Construction

Given an NFA, order its set of states Q arbitrarily. Go over the states $q \in Q$ in that order, one by one, and for each state $q \in Q$

Remove-All-ε-moves-Incoming-into-(q):

Remove-All-ε-moves-Incoming-into(q).

① Remove ε-self-loop $q \xrightarrow{\varepsilon} q$ if any.

② $\forall q'$ such that $q' \neq q$ and $q' \xrightarrow{\varepsilon} q$,

- { $\forall q''$ $\forall a \in \Sigma \cup \{\varepsilon\}$ such that

$$q' \xrightarrow{\varepsilon} q \xrightarrow{a} q'',$$

add the move $q' \xrightarrow{a} q''$.

}

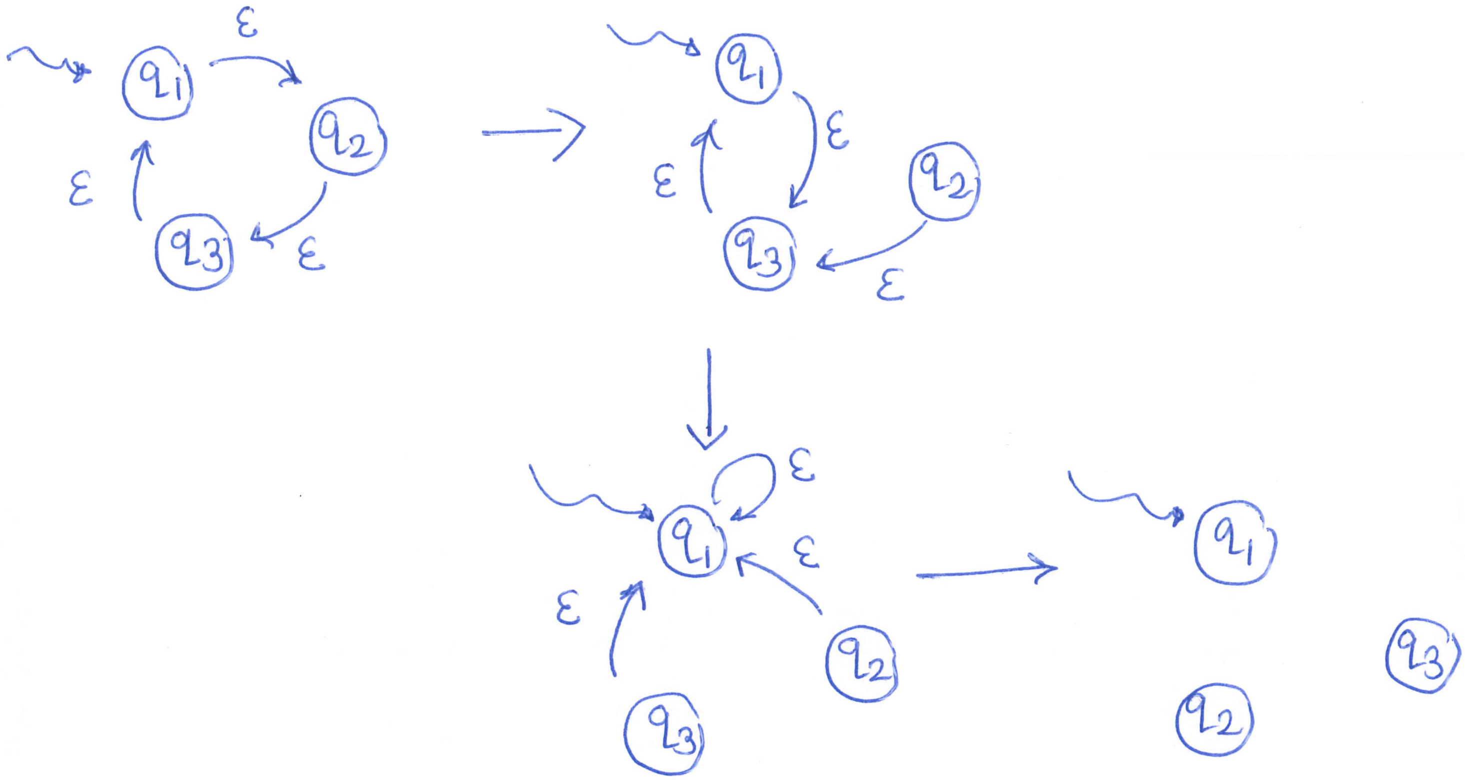
- If q is an accept state, make q' also an accept state.

- Delete the ε-move $q' \xrightarrow{\varepsilon} q$.

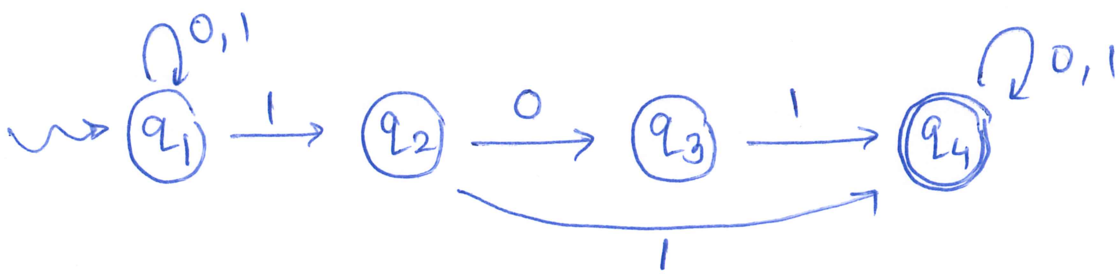
Exercise Convince yourself that

- the above operation leads to an equivalent NFA.
- once all incoming ε-moves into q are removed, no incoming ε-moves into q are added subsequently. (WHY ?!)

Note The construction "automatically" gets rid of "ε-cycles" if any. E.g.



We'll now assume that the given NFA has no ε-moves. E.g.



We wish to construct an equivalent DFA. The idea is to "remember" the set of all states that the NFA could possibly be in, after reading any input (prefix).

E.g. on input 01010, the subset changes as

$$\{q_1\} \xrightarrow{0} \{q_1\} \xrightarrow{1} \{q_1, q_2\} \xrightarrow{0} \{q_1, q_3\} \xrightarrow{1} \{q_1, q_2, q_4\} \xrightarrow{0} \{q_1, q_3, q_4\}.$$

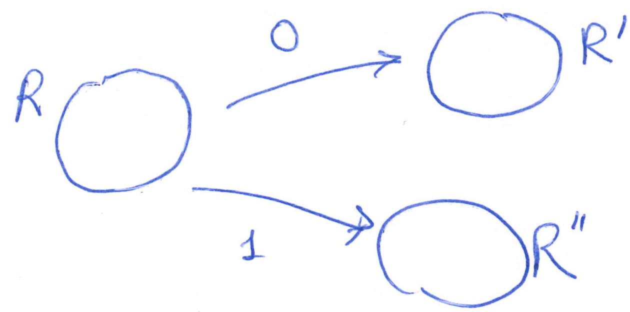
Thus, after reading 01010, the NFA could have been in any of the states $\{q_1, q_3, q_4\}$. In particular, it could have been in the accept state q_4 and so the NFA accepts the input 01010.

So we can simulate the NFA by a DFA such that

- States of the DFA are all possible subsets of the set of states of the NFA.

- The DFA "remembers" the subset of all states that the NFA could be in.

- Transitions of DFA are defined as:



where $R, R', R'' \subseteq Q$, Q is the set of states of NFA,

and (say) R' is all states that the NFA can move to on input 0 from some state in R.

- If R contains some accept state of NFA, then R is designated as accept state of DFA.

Formal construction

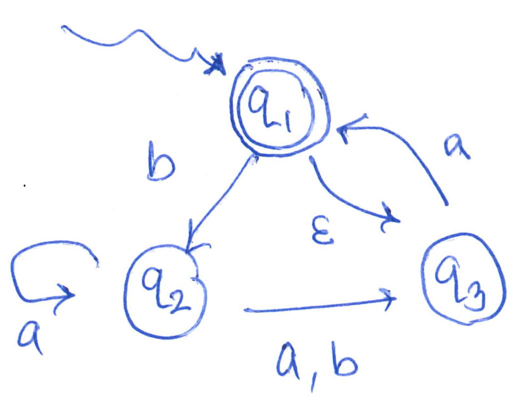
Let $N = (Q, \Sigma, \delta, q_1, F)$ be an NFA with no ε-moves. A DFA
$M = (Q', \Sigma, \delta', q_1', F')$ that is equivalent to N is constructed as follows:

- $Q' = \mathcal{P}(Q)$.
- $q_1' = \{q_1\}$.
- $F' = \{R \mid R \subseteq Q, \; R \cap F \neq \phi\}$.
- For $R \subseteq Q$ and $a \in \Sigma$,

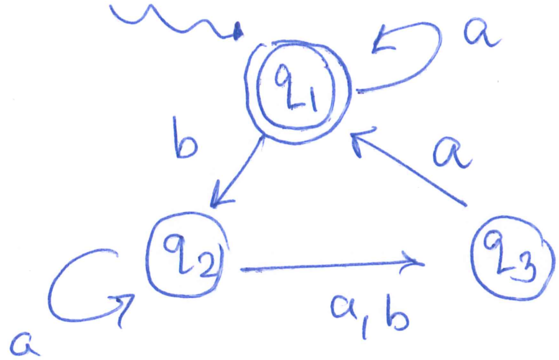
$$\delta'(R, a) = \bigcup_{q \in R} \delta(q, a).$$

$\blacksquare$

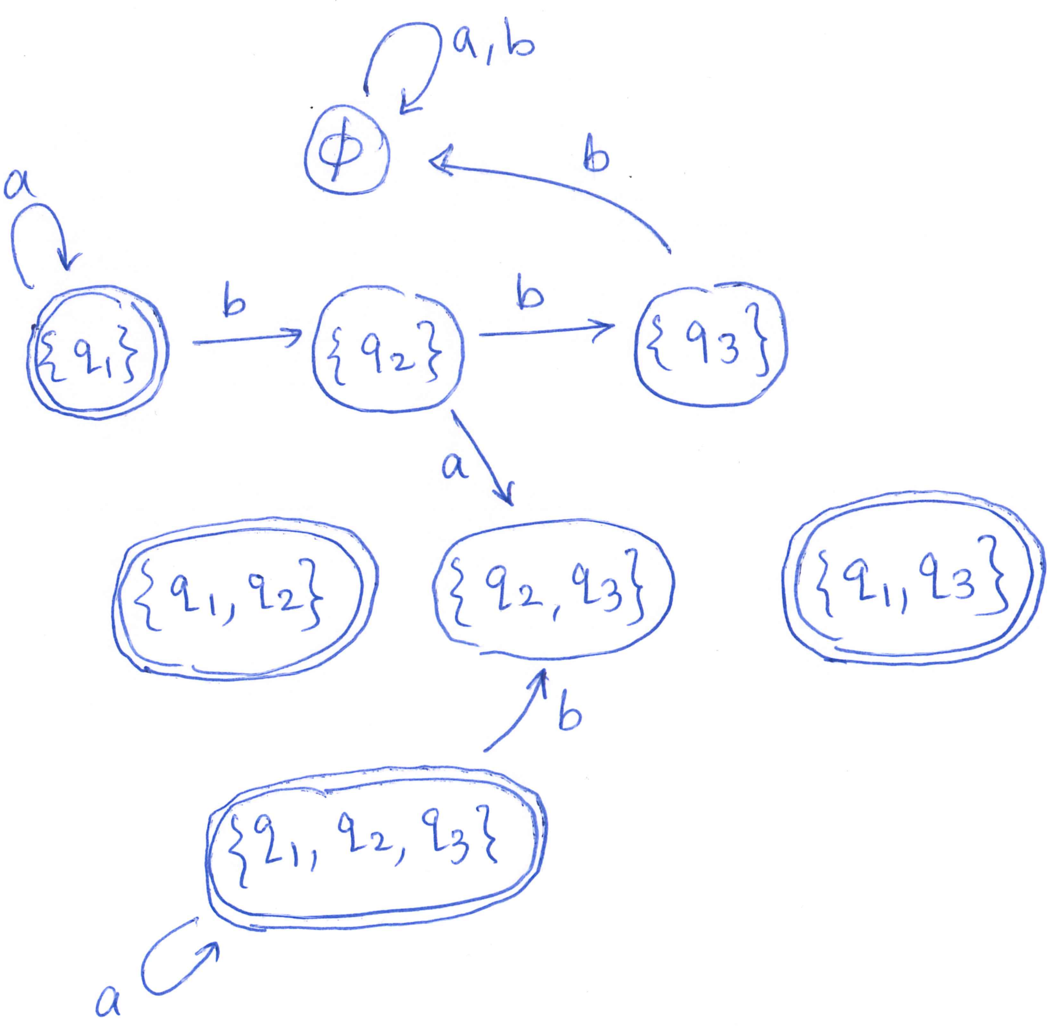
Example Given NFA



We can first remove the ε-move and get



Then the subset construction gives a DFA:



<u>Exercise</u> Complete all the transitions of this DFA.

<u>Note</u> Given k-state NFA, the construction gives 2^k-state DFA.