Nondeterministic finite automata

Featured Student Cub/Event

Theorem (stated in last lecture)

The class of regular languages is closed under the concatenation operation

IOW: if A_1 and A_2 are regular languages, so is $A \circ B$.

Proof attempt (stated in last lecture)

- Start with FA M_1 and M_2 recognizing the regular languages A_1 and A_2.
- Construct a FA M that must accept an input if it can be broken into two pieces, where M_1 accepts the first piece and M_2 accepts the second piece.
- Problem: M doesn't know know where to break the input.

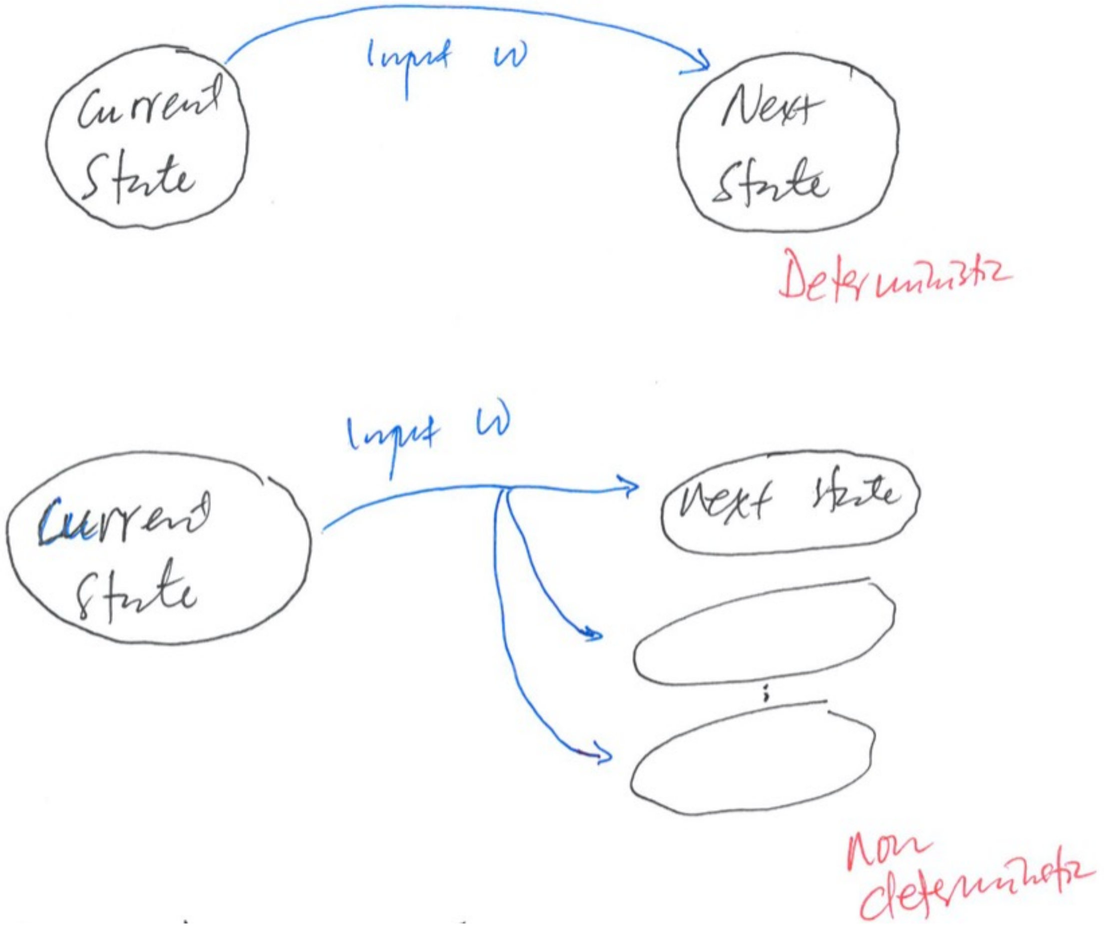
$$w = s_1 s_2 s_3 \cdots s_n$$

- We need a different strategy – nondeterminism

Nondeterminism

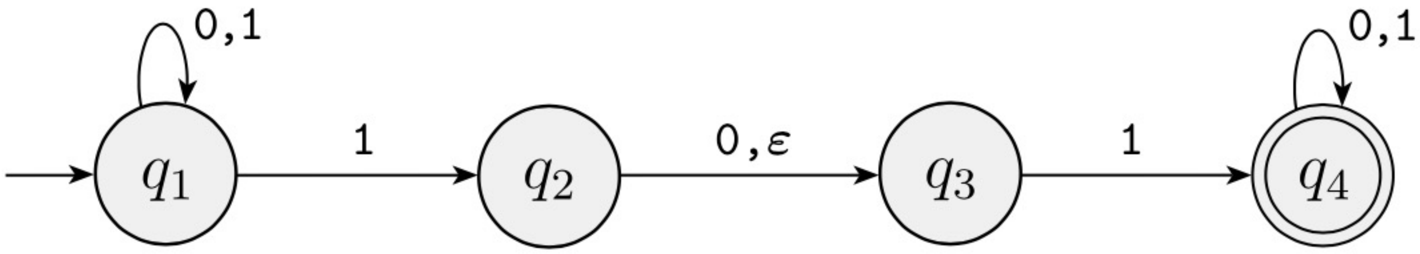
- In a **nondeterministic** machine, several choices may exist for the next state at any time.

- Nondeterminism is a generalization of determinism, so every DFA is automatically an NFA.



Additional features in NFA compared to DFA

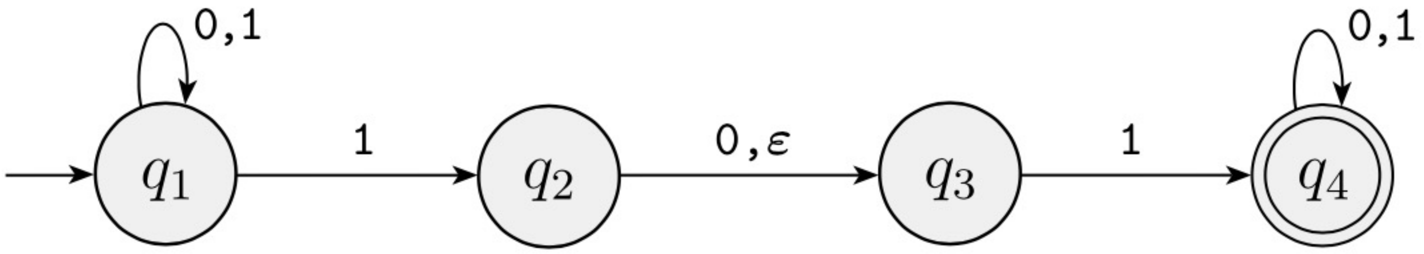
- Difference 1
  - DFA: every state has exactly one exiting arrow for each symbol
  - NFA: not the case here.
    - q_1 has one exiting arrow for 0, but it has two for 1
    - q_2 has one arrow for 0, but it has none for 1
  - In general, in an NFA a state may have zero, one or many exiting arrows for each alphabet symbol.



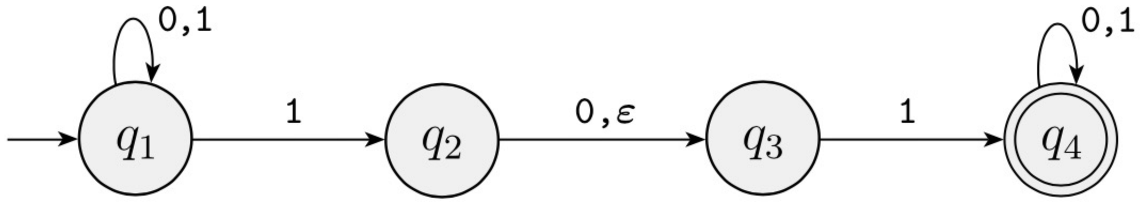
NFA N_1

Additional features in NFA compared to DFA

- Difference 2
  - DFA: labels on transition arrows are symbols from the alphabet
  - NFA: the example N_1 violates this
    - It has an arrow labeled ε
  - In general, an NFA may have arrows labelled with members of the alphabet or ε.
  - Zero, one, or many arrows may exit from each state with label ε.

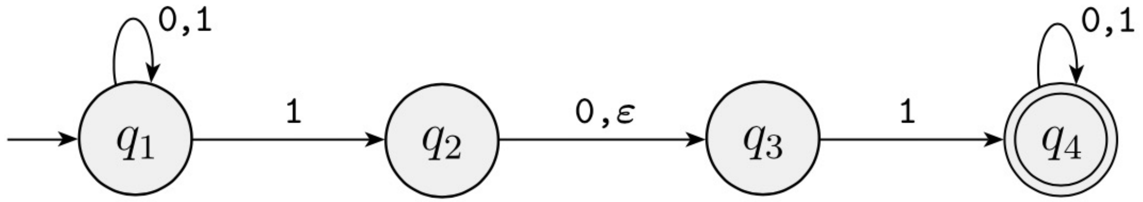


NFA N_1



How does an NFA compute?

- Suppose we are running an NFA on an input string and we have come to a state with multiple ways to proceed.
- For example, say we are in state q_1 in the NFA N_1 and the next input symbol is a 1.
- After reading that symbol, the machine splits into multiple copies of itself and follows <u>all</u> possibilities in parallel.
- Each copy of the machine takes one of the possible ways to proceed and continues as before.
- If there are subsequent choices, the machine splits again.
- If the next input symbol doesn't appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies.
- Finally, if <u>any one</u> of the copies of the machine is an accept state at the end of the input, the NFA accepts the input string.



How does an NFA compute?

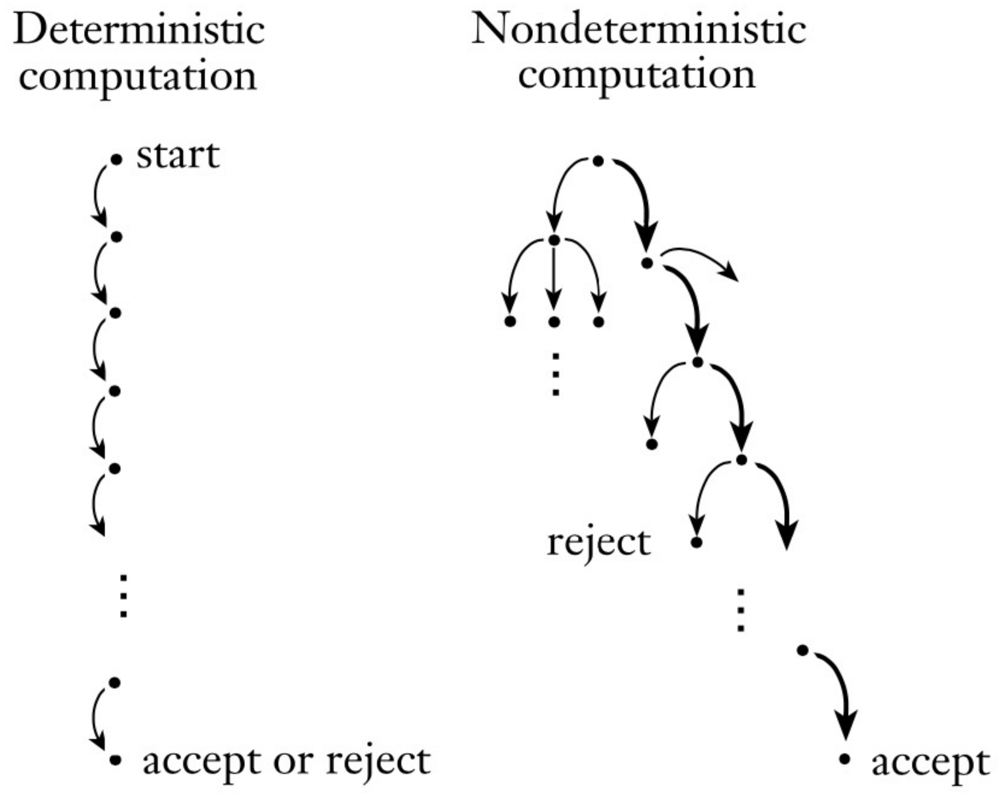
- If a state with an ε symbol on an exiting arrow is encountered, something similar happens
- Without reading any input, the machine splits into multiple copies, one following each of the exiting ε-labeled arrows and one staying at the current state.
- Then the machine proceeds nondeterministically as before.

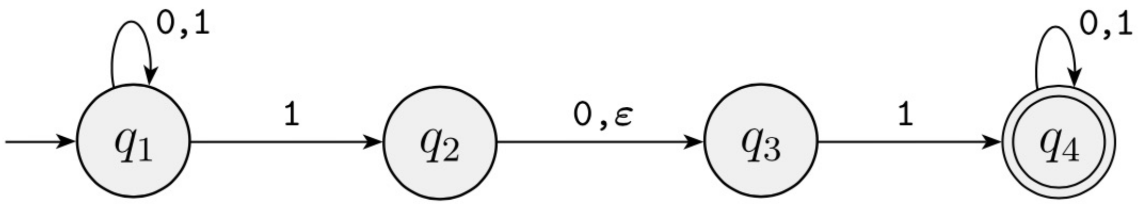
Views of nondeterminism

- **Kind of parallel computation**
  - When the NFA splits to follow several choices, that corresponds to a process “forking” into several children, each processing separately
  - If at least one of these processes accepts, then the entire computation accepts.

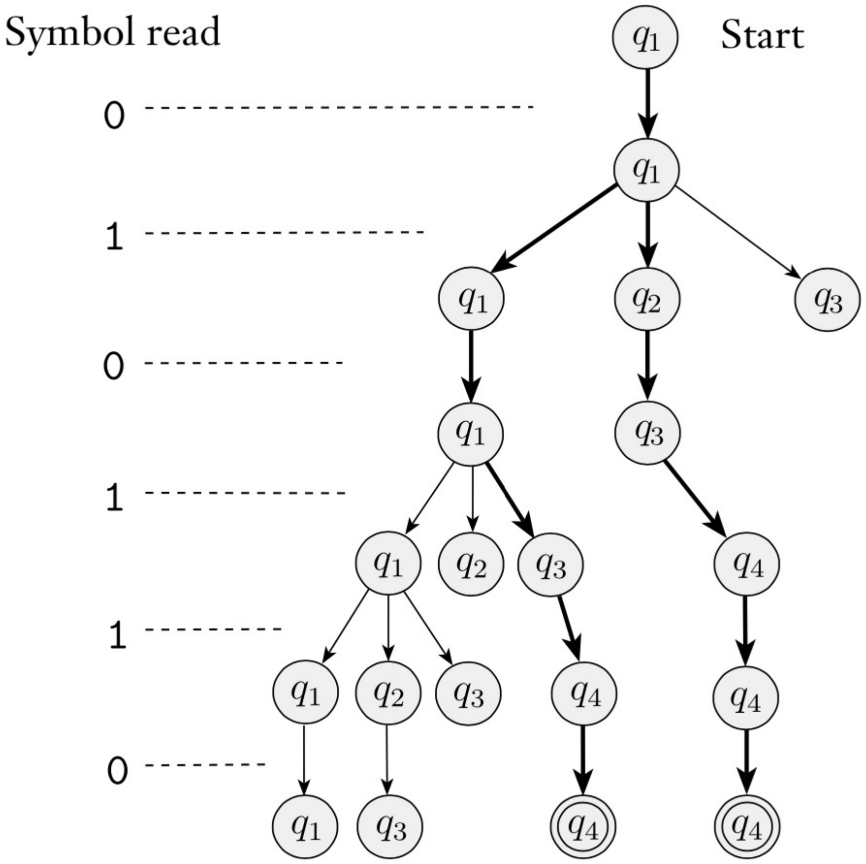
- **Tree of possibilities**
  - Root corresponds to start of the computation
  - Each branching point in the tree corresponds to a point in the computation at which the machine has multiple choices
  - Machine accepts if at least one of the branches ends in an accept state

Visual illustration of tree of possibilities view





Sample run: computation of N_1 on input 010110



What does N_1 do on input 010?

Convince yourself that N_1 accepts all strings that contain either 101 or 11 as a substring

Nondeterministic FA are useful in several respects

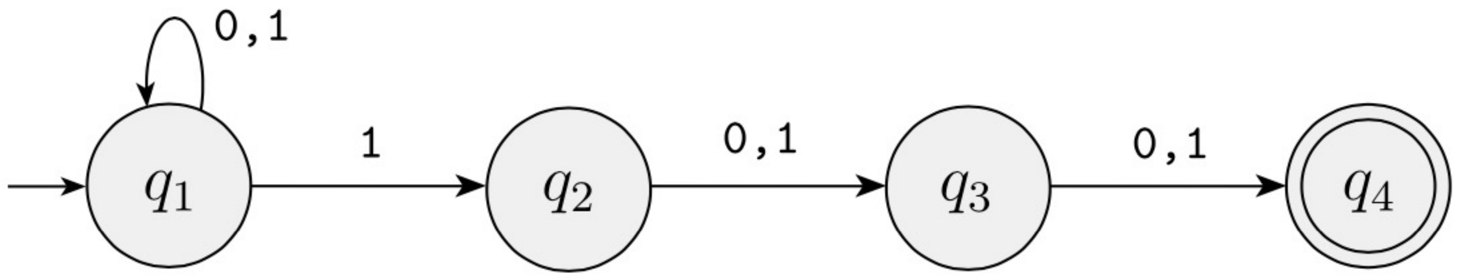
- As we will see, **every NFA can be converted into an equivalent DFA**, and *constructing NFAs is sometimes easier than directly constructing DFAs*

- An NFA maybe *much smaller than its deterministic counterpart, or its functioning may be easier to understand*

- Nondeterminism in FA is also *a good introduction to nondeterminism in more powerful computational models*

Examples of NFA: Example 1

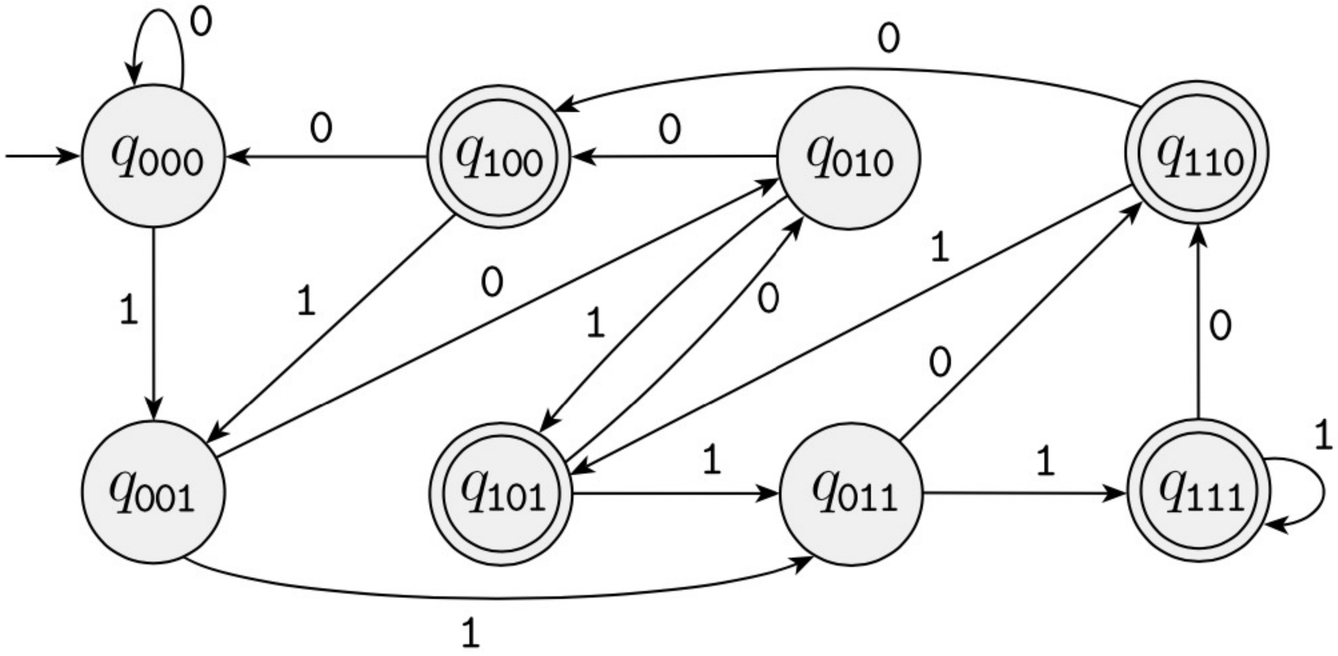
- Let A be the language consisting of all strings over {0,1} containing a 1 in the third position from the end (e.g. 000100 is in A but 0011 is not).
- The following four-sate NFA N_2 recognizes A.



One good way to view the computation of this NFA is to say that it stays in the start state q_1 until it "guesses" that it is three places from the end.
At that point, if the input is a 1, it branches to state q_2 and uses q_3 and q_4 to "check" on whether its guess was correct.

DFA equivalent of N_2

- Every NFA can be converted into an equivalent DFA, but sometimes the DFA may have many more states.
- The smallest DFA for the language A in Example 1 contains 8 states, and it is a lot more complex to understand



Example 2

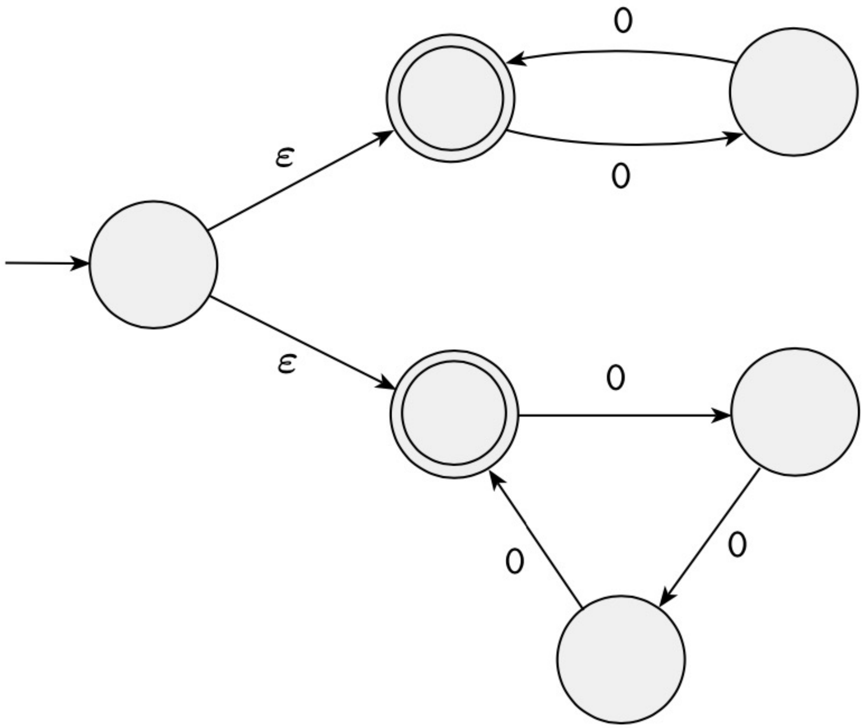
The NFA N_3 at the right has input alphabet $\{0\}$ consisting of a single symbol (unary alphabet).

This machine determines convenience of having ε arrows.

What language does the machine recognize?

0^k where k is a multiple of 2 or 3.

N_3



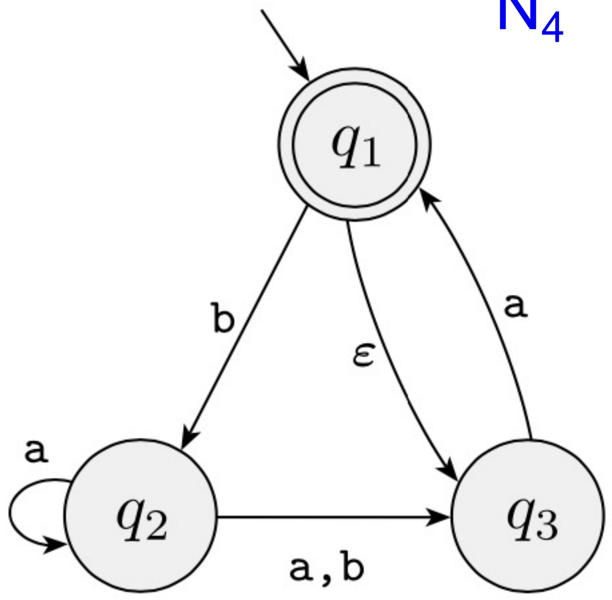
Example 3

Exercise: convince yourself that N_4

accepts the strings
ε, a, baba, baa

but doesn't accept the strings
b, bb, babba.

N_4



Formal definition of NFA

Similar to DFA, but differs in one essential way:

the type of the transition function

- DFA

  (state, input symbol) -----> (next state)   ONE

- NFA

  (state, input symbol)

  or   ------> (set of possible next states)

  (state, the empty string)

Formal definition of NFA

- To write a formal definition, we set up some additional notations.
  - For any set Q we write $P(Q)$ to be the collection of all possible subsets of Q
  - For any alphabet Σ we write Σ_ε to be $\Sigma \cup \{\varepsilon\}$

  - Now we can write the formal description of the type of the transition function in an NFA as $\delta: Q \times \Sigma_\varepsilon \longrightarrow P(Q)$

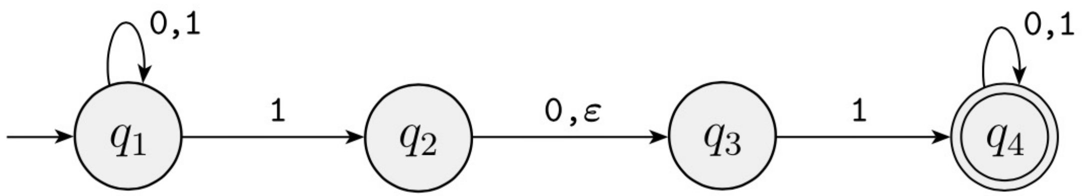
Formal definition of NFA

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.

Recall the NFA 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 of an NFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that 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 Σ_ε and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

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