1 Regular languages

1.1 Deterministic Finite Automaton

Σ a finite non-empty set called an **alphabet**. Typically $\Sigma = 0, 1$.
A **string** w is a finite sequence of symbols -aka. alphabets) in Σ.
We have:

$$\epsilon := \text{zero symbols}$$

$$|w| = \text{number of symbosl in } w$$

$$\Sigma^k := \{w : w \text{ string over } \Sigma \mid |w| = k\}$$

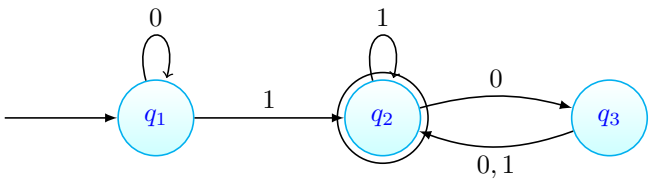
$$\Sigma^\star := \bigcup_{k \geq 0} \Sigma^k,\ \Sigma := \bigcup_{k \geq 1} \Sigma^k$$

A **language** L over Σ is a subset of $\Sigma^\star$. The concatenation of $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ is

$$xy = x_1 \dots x_n y_1 \dots y_m$$

Note : A language is called a **problem** when strings are given some interpretation (e.g. view $w \in \Sigma^\star$ as the binary representation of an integer).

An **automaton** is an abstract model of computation :



Terminology:

- Transition diagram
- Start arrow
- Accept state

It reads an input $w = w_1 \dots w_n$ from left to right. It follows the arcs according to the w_i starting at the start state. It accepts IFF it ends up in an accepting state after reading the whole input. It rejects otherwise.

Formally, a **Deterministic Finite Automaton (DFA)** is a 5-tuple $\mathcal{D} = (Q, \sigma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to Q$ is a transition function
- $q_0 \in Q$ is the start state

- $F \subseteq Q$ is the set of the accepting states

We define the **Extended Transition Function** $\hat{\delta}$ of D as follows :

$$\hat{\delta}(q, \epsilon) := q,\ \hat{\delta}(q, xa) := \delta(\hat{\delta}(q, x), a)$$

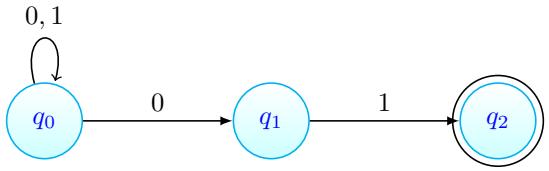
We define the **language of** D as follows :

$$\mathcal{L}(\mathcal{D}) := \{w \in \Sigma^\star \mid \hat{\delta}(q_0, w) \in F\}$$

We say that D recognizes a language L if $L = \mathcal{L}(\mathcal{D})$ We say $L \subseteq \Sigma^\star$ is **regular** if $L = \mathcal{L}(\mathcal{D})$ for some D

1.2 Non-Deterministic Finite Automaton

Nondeterminism : A generally useful concept in studying computation. It gives the power of being in several states at once. Example:



This automaton accepts any string ending in 01.

Formally : A **NFA** is a 5-tuple $\mathcal{N} = (Q, \Sigma, \delta, q_0, F)$ where Q, Σ, q_0, F are defined as previously and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function.

The extended transition function is:

$$\hat{\delta}(q, \epsilon) = \{q\},\ \hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)$$

We let:

$$\mathcal{L}(\mathcal{N}) = \{w \in \Sigma^\star \mid \hat{\delta}(q, xa) \cap F \neq \emptyset\}$$

1.3 Equivalence of DFAs and NFAs

Now, we want to find a way to convert an **NFA** $\mathcal{N}$ into a **DFA**, ie. find some $\mathcal{D}$ DFA such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{D})$

Construction :

Let $\mathcal{N} = (Q_\mathcal{N}, \Sigma, \delta_\mathcal{N}, q_0, F)$. Consider $\mathcal{D} = (Q_\mathcal{D}, \Sigma, \delta_\mathcal{D}, q_0, F)$ as follows :

- $Q_\mathcal{D} = \mathcal{P}(Q_\mathcal{N})$
- $\delta_\mathcal{D}$ so that : $\forall a \in \Sigma, \forall s \in Q_\mathcal{D}$, we have :

$$\delta_\mathcal{D}(s, a) := \bigcup_{p \in S} \delta_\mathcal{N}(p, a)$$

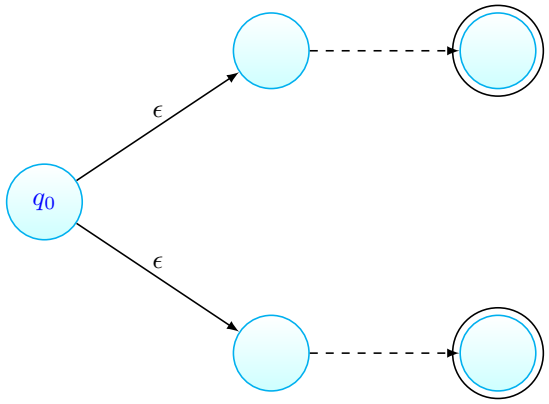
- $F_\mathcal{D} = \{s \subseteq Q_\mathcal{N} \mid s \cap F_\mathcal{N} \neq \emptyset\}$

Theorem : These automaton denote the same language. ■

Note : When converting, the number of states grows exponentially.

1.4 $\mathcal{E}$-NFA

Idea : Allow transition without reading any input symbol (ie. reading ϵ).
Example : One can simply build an automaton denoting the union of two regular languages.



Definition : $\mathcal{E} - Closure(q) = \{q' \in Q \mid q'$ can be reached from q by following only ϵ-transitions over multiple transitions$\}$. We can hence extend the definition of the extended transition function to ϵ-NFAs by having:

$$\hat{\delta}(q, \epsilon) := \mathcal{E} - Closure(q)$$