Finite Automata

- basic computational model: limited amount of memory
- example: controller for an automatic door
Finite Automata

Formal definition:

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

- \(Q\) is a finite set of **states**
- \(\Sigma\) is a (finite) alphabet
- \(\delta : Q \times \Sigma \rightarrow Q\) is the **transition function**
- \(q_0 \in Q\) is the **start state**
- \(F \subseteq Q\) is the set of **accept states**

Pictorial representation: **state diagram**
Another (more abstract) example:
- accept all strings over \{0,1\} that start with 1 and end with 0

\begin{itemize}
\item $q_3 \rightarrow$ all strings starting with 0
\item $q_1 \rightarrow$ all strings that start and end with 1
\item $q_2 \rightarrow$ all strings that start with 1 and end with 0
\item $q_0 \rightarrow$ \emptyset
\end{itemize}
Let $M=(Q, \Sigma, \delta, q_0, F)$ be a FA. The **language of $M$ (accepted/recognized by $M$)** is $L(M)$.

Formally: need the definition of **computation**:

- $w_i \in \Sigma$

**$M$ accepts $w=w_1w_2...w_n$** if there exist states $r_0, r_1, ..., r_n$ in $Q$ such that
- $r_0 = \lambda q_0$
- $\delta (r_i, w_{i+1}) = r_{i+1}$ $\forall i \in \{0, 1, 2, ..., n-1\}$
- $r_n \in F$

A language is **regular** if there exists a FA that recognizes it.
Designing FAs

Examples - languages over \{0,1\} consisting of strings:
- with odd number of 1’s \( \leftrightarrow M_1 \)
- that contain 001 as a substring \( \leftrightarrow M_2 \)
- that are even length and do not contain 00 as a substring \( \leftrightarrow M_3 \)

A language that cannot be accepted by a FA?
\{ a^k \mid k \text{ is a prime} \} \quad \{ a^k b^k \mid k \geq 0 \} \quad \{ w w \mid w \in \Sigma^* \}
Regular operations

Let \( A \) and \( B \) be languages. The following three language operations are called the **regular operations**:

- **union**: \( A \cup B \)
- **concatenation**: \( A.B \)
- **star**: \( A^* \)

The natural numbers are **closed under multiplication** but not division. \( x \cdot y \in \mathbb{N} \quad \forall x, y \in \mathbb{N} \)

What about the class of regular languages?
**Thm 1.25**: The class of regular languages is closed under the union operation. (as well as closed under the intersection)

Means: the union of any two reg. lang. is a reg. lang.

Or: given two FA $M_A$ and $M_B$, we want to construct a FA $M$ that recognizes the union of $L(M_A)$ and $L(M_B)$

Example:

- $M_A$: even 1's $\xrightarrow{1} \text{odd 1's}$
- $M_B$: even 0's $\xrightarrow{0} \text{odd 0's}$

**Formal construction of $M$:**

$M = (Q, \Sigma, \delta, q_0, F)$

where

- $Q = Q_A \times Q_B$
- $q_0 = (q_{A0}, q_{B0})$
- $\delta((p,q),a) = (\delta_A(p,a), \delta_B(q,a))$

$[F = F_A \times F_B \leftarrow \text{would prove closed under intersection}]$

$F = F_A \times Q_B \cup Q_A \times F_B$

$\forall (p,q) \in Q \quad \forall a \in \Sigma$
Thm 1.26: The class of regular languages is closed under the concatenation operation.

\begin{itemize}
  \item have: \( M_A, M_B \) two FA
  \item want: to construct a FA \( M \) s.t. \( L(M) = L(M_A) \cdot L(M_B) \)
\end{itemize}

\begin{center}
\begin{tikzpicture}
\node[state,initial] (q0) at (0,0) {};\node[state, accepting] (q1) at (2,0) {};
\draw (q0) -- node[above] {0} (q1);
\draw (q0) -- node[below] {1} (q1);
\draw (q0) -- node[below] {1} (q1);
\draw (q0) -- node[above] {0} (q1);
\end{tikzpicture}
\end{center}

\text{run } M_A \quad \text{run } M_B