Finite Automata

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

2 states: opened, closed
input:
front sensor but not back
back but not front
both
neither

\[ \Sigma = \{ \text{A, B, C, D} \} \]

today: BAABAB

ends in an accepting state, input string accepted by the automaton
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** \(Q = \{ \text{opened, closed} \}\)
- \(\Sigma\) is a (finite) alphabet \(\Sigma = \{ \#, \& \}\)
- \(\delta: Q \times \Sigma \rightarrow Q\) is the **transition function** ✓
- \(q_0 \in Q\) is the **start state** \(q_0 = \text{cloud}\)
- \(F \subseteq Q\) is the set of **accept states** \(F = \{ \text{closed} \}\)

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

$M$ accepts $w=w_1w_2...w_n$ if there exist states $r_0, r_1, ..., r_n$ in $Q$ such that

- $r_0 = ? q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$
- $r_n \in F$

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
- that contain 001 as a substring
- that are even length and do not contain 00 as a substring

A language that cannot be accepted by a FA?

**YES, e.g.** \[ L = \{ 0^k 1^k \mid k \geq 0 \} \]
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.

What about the class of regular languages?

(any) $FA_{M_A,M_B}$

is the class of regular lang. closed under $\cup$? means: we have two reg. lang. $A,B$ is $A \cup B$ always regular? want an FA for $A \cup B$? YES
Thm 1.25: The class of regular languages is closed under the union operation.

Example: 

\[ A = \{ w \in \{0,1\}^* \mid w \text{ has odd # } 1's \} \]

\[ B = \{ w \in \{0,1\}^* \mid w \text{ has even # } 0's \} \]

Note: 

- \( A \cup B \) construction
- the same except
- \( F_{A \cup B} = F_A \times F_B \)
  \[ = \{ (r_a, r_b) \mid r_a \in F_A \text{ and } r_b \in F_B \} \]

\[ F_{A \cup B} = \{ (r_a, r_b) \mid r_a \in F_A \text{ or } r_b \in F_B \} \]

\[ = F_A \times Q_B \cup Q_A \times F_B \]

\[ \delta((r_a, r_b), \sigma) = (\delta_a(r_a, \sigma), \delta_b(r_b, \sigma)) \quad \forall r_a \in Q_A, \forall r_b \in Q_B, \forall \sigma \in \Sigma \]

Pf sketch: 

\[ M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A) \]

\[ M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B) \]

want to describe \( M_{A \cup B} \):

\[ M_{A \cup B} = (Q_A \times Q_B, \Sigma, \delta_{A \cup B}, (q_{0A}, q_{0B}), F_{A \cup B}) \]

Where 

\[ \delta_{A \cup B}(r_a, r_b, \sigma) = (\delta_a(r_a, \sigma), \delta_b(r_b, \sigma)) \quad \forall (r_a, r_b) \in Q_A \times Q_B, \forall \sigma \in \Sigma \]
Thm 1.26: The class of regular languages is closed under the concatenation operation.

I.e. for any two languages $A, B$ : $A \cdot B$ is regular.

Example:

Want to magically jump to the other automaton (nondeterminism)