Modeling Computation

Introduction to Formal Languages and Automata

NFAs
DFA that accepts all strings in \( \{a, b\}^* \) that contain \( aa \) (2 consecutive \( as \)).
DFA that accepts all strings in \( \{a, b\}^* \) that contain \( bb \) (2 consecutive \( bs \)).

Same as the previous
DFA that accepts all strings in \( \{a, b\}^* \) that contain \( aa \) or \( bb \).

\[ \text{Not a DFA!} \]

It is a **Nondeterministic Finite Automaton**.
Nondeterministic Finite Automata (NFA)

- \( M = (Q, \Sigma, \Delta, s, F) \)

- \( Q \) is a finite set of states

- \( \Sigma \) is a finite input alphabet

- \( \Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q \) is the transition relation. 
  \( \varepsilon \) is the empty string, \( \Delta \) is a relation, not a function.

- \( s \in Q \) is the initial state

- \( F \subseteq Q \) is the set of favorable states
Example: NFA that accepts all strings in \( \{a, b, c\}^* \) that contain \textit{abababa}.
Example: NFA that accepts all strings in \( \{0,1\}^* \) that end with 111 or 000.

\[ L = \{ \mathbf{w} \mid \mathbf{w} \text{ ends with 111 or 000} \} \]
Language Recognition by FSMs

- The language recognized or accepted by the machine $M$, denoted $L(M)$, is the set of all strings that are accepted by $M$.

- Two automata are equivalent if they accept the same language.
Compare NFAs to DFAs

• Which is more computationally powerful?

They have the same power. They are equivalent.

• Which is easier to design?

NFA
**Theorem:** For each NFA $A = (Q_N, \Sigma, \Delta, s_N, F_N)$, there exists a DFA $B = (Q_D, \Sigma, \delta, s_D F_D)$ equivalent to $A$.

**Proof:** by construction (sketch)

- Each state in $B$ corresponds to a **set** of states in $A$
  - Subset construction problem
- For $T \subseteq Q_N$, $\epsilon - close(T)$ is the set of states reachable from a state in $T$ using $\epsilon$-jumps
- $s_D = \epsilon - close(s_N)$
- $Q_D = \{S \subseteq Q_N \mid S = \epsilon - close(S)\}$, $\epsilon$-closed subsets of $Q_N$
  - Do lazy evaluation to find
- $F_D = \{S \mid (S \in Q_D) \land (S \cap F_N \neq \emptyset)\}$, states that contain at least 1 favorable state of $A$
- $\delta(s, a)$ for $a \in \Sigma$ and $s \in Q_D$ is computed as
  - Let $s = \{p_1, ..., p_k\}$
  - Let $\bigcup_{i=1}^k \Delta(p_i, a) = \{r_1, ..., r_m\}$
  - $\delta(s, a) = \epsilon - close(\{r_1, ..., r_m\})$
Example: Convert NFA to DFA

\[ S_D = \epsilon - \text{close}(s) = \{ s, p, q, r \} \]

<table>
<thead>
<tr>
<th>S</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S, p, q</td>
<td>S, r, p, q, t</td>
</tr>
<tr>
<td>B</td>
<td>S, p, q, r, f</td>
<td>S, p, q, t</td>
</tr>
<tr>
<td>C</td>
<td>S, p, q, t</td>
<td>S, p, q, r</td>
</tr>
<tr>
<td>D</td>
<td>S, p, q</td>
<td>S, p, q, r, f</td>
</tr>
<tr>
<td>E</td>
<td>S, p, q, f</td>
<td>S, p, q, r, f</td>
</tr>
</tbody>
</table>