Regular Languages & Non-deterministic finite Automata
Regular Expressions and Regular Languages

\[
\text{rexp}_\Sigma ::= \emptyset \mid \lambda \mid s \mid \text{rexp}_\Sigma \text{rexp}_\Sigma \mid \text{rexp}_\Sigma \cup \text{rexp}_\Sigma \mid \text{rexp}_\Sigma^*
\]

with \( s \in \Sigma \)

Deterministic Finite Automata, DFA

**Proposition** Closure under complement, union, intersection

If \( L_1, L_2 \) are accepted by some DFA, then so are

- \( \overline{L_1} = \Sigma^* - L_1 \)
- \( L_1 \cup L_2 \)
- \( L_1 \cap L_2 \).
Kleene's Theorem (announced last lecture)

Theorem The languages accepted by DFAs are exactly the regular languages.

We prove this by

1. If $L = L(M)$, for some DFA $M$, then there is a regular expression $e$ such that $L = L(e)$ (previous lecture).

2. If $L = L(e)$, for some regular expression $e$, then there is a non-deterministic finite automaton with $\lambda$-steps (NFA$_\lambda$) $M$ such that $L = L(M)$. (this lecture)

3. For every NFA$_\lambda$, $M$, there is a DFA $M'$ such that $L(M) = L(M')$ (this lecture).
Non-deterministic finite automaton (NFA)

\[ \delta \]

\begin{align*}
\delta & | q_0 & q_1 & q_2 \\
\text{a} & q_0 & \emptyset & \emptyset \\
\text{b} & \{q_0, q_1\} & q_2 & \emptyset \\
\end{align*}

in shorthand

\[ \delta \]

\begin{align*}
\delta & | q_0 & q_1 & q_2 \\
\text{a} & q_0 & & \\
\text{b} & q_0, q_1 & q_2 &
\end{align*}
Non-deterministic Finite Automata: NFA (formally)

$M$ is a DFA over $\Sigma$ if $M = (Q, \Sigma, q_0, \delta, F)$ with

- $Q$ is a finite set of states
- $\Sigma$ is a finite alphabet
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a finite set of final states
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function

$\mathcal{P}(Q)$ denotes the collection of subsets of $Q$

Reading function $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$ (multi-step transition)

\[
\hat{\delta}(q, \lambda) = \{q\}
\]
\[
\hat{\delta}(q, w\lambda) = \{q'' \mid q'' \in \delta(q', a) \text{ for some } q' \in \hat{\delta}(q, w)\}
\]

The language accepted by $M$, notation $L(M)$, is:

$L(M) = \{w \in \Sigma^* \mid \exists q_f \in F(q_f \in \hat{\delta}(q_0, w))\}$
Suppose we want \( \{w \mid \#_a \text{ even or } \#_b \geq 1\} = L_1 \cup L_2 \)

First idea: put the two machines “nondeterministically” together

The NFA on the right accepts ‘aaa’ which is wrong!
Now we add $\lambda$ transitions or 'silent steps' to NFAs

In a $\text{NFA}_\lambda$ we allow

$$\delta(q, \lambda) = q'$$

for $q \neq q'$. That means

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$$
A finite automaton $M$ is called insulated

(i) if $q_0$ has no in-going arrows
(ii) there is only one final state which has no out-going arrows

**Proposition.** One can insulate any machine $M$ such that
the result $M'$ accepts the same language

**Proof.** By adding states and silent steps, for example
Toolkit building $\text{NFA}_\lambda$s

$L_1 = L$

$L_2$

$L_1 \cup L_2$ insulated

$L_1 L_2$ insulated

$L^*$ insulated
Proposition. For every regular expression $e$ there is an NFA $M_e$ such that

$$L(M_e) = L(e).$$

Proof. Apply the toolkit. $M_e$ can be found ‘by induction on the structure of $e$’: first do this for the simplest regular expressions; then for a composed regular expression compose the automata. ■

Corollary. For every regular language $L$ there is an NFA $M$ that accepts $L$ (so $L(M) = L$).
Avoiding non-determinism

Keep track of where you can go!
A combination is final if one of the members is final.
We show how a NFA can be turned into a DFA

Let \( M \) be a NFA given by \((Q, \Sigma, q_0, \delta, F)\)

Define \( M^+ \) as \((Q^+, \Sigma, q_0^+, \delta^+, F^+)\) by

\[
Q^+ = \mathcal{P}(Q) \\
q_0 = \{q_0\} \\
\delta^+(H, a) = \bigcup_{q \in H} \delta(q, a), \quad \text{for } H \subseteq Q, \\
F^+ = \{H \subseteq Q \mid H \cap F \neq \emptyset\}
\]

Then \( M^+ \) is a DFA accepting the same language as \( M \)
Every NFA \( M \) can be turned into an NFA \( M' \) accepting the same language.

**Corollary.** For every regular language \( L \) there is a DFA \( M \) that accepts \( L \) (so \( L(M) = L \)).