NFAs and DFAs

And you thought we couldn’t come up with any more acronyms....
Reminder: Regular Operations

Let A and B be languages. We define the regular operations union, concatenation, and star as follows:

- **Union**: \( A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \).
- **Concatenation**: \( A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \).
- **Star**: \( A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \).
Proofs that Regular Languages Are Closed Under...

- **Union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
  - Proof by Construction (last class)

**BUT...**

- **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star:** $A^* = \{x_1x_2...x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

WE NEED BETTER TOOLS!
Deterministic vs. Nondeterministic Finite Automata

Two types of FSMs

Everything discussed thus far was deterministic, thus a DFA

Expand this idea to add multiple potential paths
What is Determinism?

Are computers deterministic?

“If the current state is known, and the current inputs are known, then the future state is also known.”

- No Choices
- No Randomness
- No Oracles
- No Errors/Cheating

According to our previous definition, FSM == DFA
What is Nondeterminism?

“If the current state is known, and the current inputs are known, there may be multiple possible future states.”

- Random choice?
- Parallel choice and simultaneous execution?
Relaxing The Requirements

Multiple edges with the same label
ε (optional) edges

Only need one path to an accept state

What could go wrong?

How do we know which path to try?

- Try them all
- Always make the right choice
A Nondeterministic Finite Automaton is a 5-tuple
\[ N = (Q, \Sigma, \delta, q_0, F) \]
where:
- \( Q \): Set of states (finite)
- \( \Sigma \): Alphabet of symbols (finite)
- \( \delta \): The transition function
  \[ \delta: Q \times \Sigma \epsilon \rightarrow P(Q) \]
- \( q_0 \): The starting (initial) state
  \[ q_0 \in Q \]
- \( F \): The set of “Accept” states
  \[ F \subseteq Q \]
NFA Formal Definition

\[ N = (Q, \Sigma, \delta, q_0, F) \]

\[ Q = \{q_1, q_2, q_3, q_4\} \]
\[ \Sigma = \{0, 1\} \]
\[ \delta \]
\[ q_0 = q_1 \]
\[ F = \{q_4\} \]
NFA Formal Definition of Computation

Let $N = (Q, \Sigma, \delta, q_0, F)$

Let $y_1y_2...y_m$ be a string $w$ where $y_i \in \Sigma_{\varepsilon}$

$N$ accepts $w$ if there is a sequence of states $r_0, r_1, r_2...r_m$ in $Q$ such that:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $0 \leq i < m$, and
3. $r_m \in F$

$N$ “recognizes” language $A$ if

$A = \{w \mid N$ accepts $w\}$

We now have a NONDETERMINISTIC tool that we can use to understand Regular Languages!

Can we say this?!?
Proof of Equivalency

Of NFA and DFA

Proof by construction: A method to convert this:

\[ N = (Q, \Sigma, \delta, q_0, F) \]

Into this:

\[ M = (Q', \Sigma, \delta', q'_0, F') \]
Regular Operations

Union, Concatenation, and Star

3 Proofs by Construction (Demo)
Regular Operations: Union
Regular Operations: Concatenation
Regular Operations: Star