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**: $A^* = \{ x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$.
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)

- **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

Everything discussed thus far was deterministic, thus a DFA

Expand this idea to add multiple potential paths

Two types of FSMs
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
- ε is not in Σ
- Does not consume input character

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
NFA Formal Definition

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 $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$
- $q_0$ The starting (initial) state $q_0 \in Q$
- $F$ The set of “Accept” states $F \subseteq Q$
NFA Formal Definition Example

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

\[ Q = \{q_1, q_2, q_3, q_4\} \]

\[ \Sigma = \{\emptyset, 1\} \]

\[ \delta \]

\[ q_0 = q_1 \]

\[ F = \{q_4\} \]
NFA Formal Definition of Computation

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

Let $y_1y_2...y_m$ be a string $w$ where $y_i \in \Sigma$.

$N$ 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

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

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

Wait... Can we say this?!?
Proving the Equivalency of NFAs and DFAs

Proof by Construction:
A method to convert
\[ N = (Q, \Sigma, \delta, q_0, F) \]
into
\[ M = (Q', \Sigma, \delta', q_0', F') \]
Convert NFA to DFA

Proof by Construction. (ignoring $\varepsilon$ for now)

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA that recognizes some language $A$.

Construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ that recognizes the same language $A$.

$Q' = \mathcal{P}(Q)$

ex: $\{\{p_0\}, \{p_0, p_1\}, \{p_0, p_2\}, \{p_0, p_1, p_2\}, \ldots\}$

$q'_0 = \{q_0\}$

$F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

or, $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$

$\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$

where $R \in Q'$ and $a \in \Sigma$.

For $\varepsilon$, define $E(R) = \{q \mid q \text{ can be reached from } R \text{ by traveling along } 0 \text{ or more } \varepsilon \text{ arrows}\}$ where $R \subseteq Q$ and $q \in Q$.

$q'_0 = E(\{q_0\})$

$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$
What we know

We defined DFAs.
We defined Computation using DFAs.
We defined Regular Languages using DFAs.
We defined NFAs.
We defined Computation using NFAs.
We showed equivalence of NFAs to DFAs.

Now What?