Regular Languages

I used to hate writing assignments, but now I enjoy them.

I realized that the purpose of writing is to inflate weak ideas, obscure poor reasoning, and inhibit clarity.

With a little practice, writing can be an intimidating and impenetrable fog! Want to see my book report?

The dynamics of interbeing and monological imperatives in Dick and Jane: A study in psychic transrelational gender modes.

Academia, here I come!
Definition:

A language is a **Regular Language** iff some **Finite State Machine** recognizes it.
Intuition Building:

What would make a language NOT Regular?

Answer: It requires memory.
A Few Examples (not limited to):

Non-Regular languages

- $0^n1^n$
- $0^m1^n, m > n$
- $w = (\emptyset|1)^*$
- where $\#_0(w) = \#_1(w)$
- $ww$ or $ww^R$

Regular Languages

- Contains a substring
- $0^*1^*0^+$
- Any finite language
- Binary numbers divisible by 3
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:** $A^* = \{ x_1x_2\ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$. 
Question: Are Regular Languages closed under Union, Concatenation, and Star?
Theorem: Regular Languages are Closed Under Union

Intuition:
- In terms of sets, what does Union mean?
- How can this be applied to two FSMs?
- Can you build a machine that models this behavior?

Define:
- $Q_3 = Q_1 \times Q_2$
- $\Sigma$ (no change needed)
- $q_3 = (q_1, q_2)$
- $F_3 = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
  Note: NOT $(F_1 \times F_2)$
- $\delta_3((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$
  where $a \in \Sigma$ and $(p, q) \in Q_3$.

Construct $M_3 = M_1 \cup M_2$

What else would this proof need?
NFAs and DFAs
Reminder: Regular Operations

Let $A$ and $B$ be languages. We define the regular operations \textit{union}, \textit{concatenation}, and \textit{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:** $A^* = \{x_1x_2\ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$. 
Proofs that Regular Languages Are Closed Under…

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

But...

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

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 \textbf{state} is known, and the current \textbf{inputs} are known, there may be \textbf{multiple possible} future states.”

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

- Multiple edges with the same label
- \( \varepsilon \) (optional) edges
  - \( \varepsilon \) is not in \( \Sigma \)
  - 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
  \( \delta: Q \times \Sigma \varepsilon \rightarrow \mathcal{P}(Q) \)
- \( q_0 \) The starting (initial) state
  \( q_0 \in Q \)
- \( F \) The set of “Accept” states
  \( F \subseteq Q \)
NFA Formal Definition Example

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

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

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

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

\[ \Sigma = \{ 0, 1 \} \]

\[ \delta(q_0, 0) = q_1 \]

DO YOU REMEMBER WHAT MUST COME NEXT?
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

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

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

Wait... Can we say this?!!