Non deterministic Finite Automata (NFA)

Recall that for DFAs:

- for each state, exactly one outgoing edge for each \( a \in \Sigma \)
- formalized by type sig \( \delta : Q \times \Sigma \rightarrow Q \)
- extended transition fn \( \delta^* : Q \times \Sigma^* \rightarrow Q \)
- \( \delta^*(s, w) \) is exactly one state
  check if \( \delta^*(s, w) \in A \).

In NFA:

- relax condition on transitions.
- allow any number of transitions per \( a \in \Sigma \).
- also allow special \( \epsilon \)-transitions
  that can be taken for free without reading input.

**Ex.**

![Diagram of an NFA]

Type sig of NFA transition:

\[ \delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q) \]

\[ \delta(s, 0) = \{s, a\} \]  
\[ \delta(s, 1) = \emptyset \]  
*Interpret as "always fail" or "gracefully crash"*

**Try:** run NFA
**Catch:** reject.
tape sig of extended transition is $\delta^*: Q \times \Sigma^* \to \mathcal{P}(Q)$

$\delta^*(s, 01) = \{ b \}$

$\delta^*(s, w)$ might contain states in $A$
might contain states not in $A$
might be empty

What does it mean for an NFA to accept a string?

Define an NFA accepting $w$ to mean
at least one state in $\delta^*(s, w)$ is in $A$
i.e. $\delta^*(s, w) \cap A \neq \emptyset$

Language of NFA $N$: $L(N) = \{ w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset \}$

Several Interpretations

- Magic fairy that tells you which transition to take
  at each step to get to an accepting state
  (if possible)

- Verification: user claiming $w \in L(N)$ has to provide
  proof in the form of seq of transitions

- Many threads in parallel accept if at least one thread
  accepts "Many worlds"

Remarks:

- Every DFA can be interpreted as an NFA.

  DFA $M = (Q, \Sigma, \delta, s, A)$

  build $N = (Q, \Sigma, \delta', s, A)$

  $\delta'(q, a) = \{ \delta(q, a) \}$

- Purely graphically "see" every DFA is an NFA

Next Tuesday: NFA can be converted to a DFA
Ex. Given $w = a_1a_2a_3 \ldots a_n$ NFA for $\exists w \in \emptyset$

Ex. $w = \emptyset$ means read null NFA's for $\exists w \in \emptyset$

Ex. NFA for $\emptyset$

Ex. Often (?) NFA's are smaller than DFA's for the same language.

$L = \{ w | \text{second to last symbol in } w \text{ is } 0 \}$

DFA:

4 states turns out to be optimal, cannot get smaller DFA (why? later)
State label is last two symbols seen (1 means not 0)

NFA:

3 states

Closure properties of NFAs

For convenience, assume that the NFAs given to us have exactly one accepting state.

Normalize if necessary.

Union

$L(N_1) \cup L(N_2)$

Concatenation

$L(N_1) \cdot L(N_2)$
- Concatenation \( L(N_1) \cdot L(N_2) \)

- Kleene Star \( L(N)^* \)

Problem: NFA for \( \Sigma^* \)

\[
\begin{array}{c}
\xrightarrow{0} 0 \\
\xrightarrow{1} 1
\end{array}
\]

\( \epsilon \in \Sigma^* \)

\( \epsilon \notin L(\cup_{i=3} L(N_i)) \)

NFA for \( \emptyset \)

NFA for \( \emptyset \)

New start state
Next, construct a DFA from the NFA (powerset). Then, using Thompson's algorithm, construct an NFA for the same language given any regex. The resulting NFA can be converted to a regex.