**Nondeterminism**

Deterministic: next state uniquely determined by state and input

Nondeterministic: several choices (incl. zero) at any point

<table>
<thead>
<tr>
<th>Deterministic computation</th>
<th>Nondeterministic computation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Deterministic computation" /></td>
<td><img src="image2" alt="Nondeterministic computation" /></td>
</tr>
</tbody>
</table>

Think of it as forking multiple computations in parallel.

---

**Example: Nondeterministic Finite Automaton**

Example: accept binary strings ending in 010

State $q_0$ has two transitions on input 0.

State $q_1$ has no transition on input 0.

run on input string might “get stuck” ⇒ reject

String is accepted if there is some run ending in accepting state.

Any string accepted must end in 010.

Any string ending in 010 is accepted: NFA can stay in initial state until the last 3 symbols (“guesses” when to move).

---

**NFA Definition**

A nondeterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

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

Any relation $R \subseteq A \times B$ corresponds to a function $f_R : A \rightarrow \mathcal{P}(B)$ $f_R(x) = \{y \in B \mid (x, y) \in R\}$.

---

**Equivalence of NFAs and DFAs**

**Subset construction:** consider the set of states that the automaton could be in at any given time

Construct DFA $M = (Q', \Sigma, \delta', q'_0, F')$

- $Q' = \mathcal{P}(Q)$ a state of $M$ is a set of states of $N$
- $\delta'(R, a) = \cup_{r \in R} \delta(r, a)$ for $R \in Q' (R \subseteq Q)$
- $q'_0 = \{q_0\}$ (set with the one initial state)
- $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$ (contains some accept state)
Subset construction: with $\epsilon$-transitions

If NFA is in any given state, it could also be in any state reachable by one or more $\epsilon$-transitions

Define $E(R) = \{q \mid q \text{ reachable from } R \text{ along } \epsilon\text{-transitions}\}$

Transition relation changes to

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

Regular languages, again

We've seen any NFA can be converted to an equivalent DFA.

Corollary: A language is regular if and only if some nondeterministic finite automaton recognizes it.

$\Rightarrow$ If the language is accepted by an NFA, and the NFA has an equivalent DFA, then the language is accepted by a DFA, thus regular.

$
\Leftarrow$ If the language is regular, it's accepted by a DFA. Any DFA is also a NFA.

Closure under Union

Add new initial state with $\epsilon$-transitions to both initial states

![Diagram showing construction of an NFA $N_1 \cup N_2$]

Example: strings with certain lengths

Consider a one-letter alphabet $\Sigma = \{0\}$

Strings which have length multiple of 2 or multiple of 3

![Diagram showing an NFA accepting strings of length multiple of 2 or 3]
Closure under Concatenation

Add $\epsilon$-transitions from all accept states of $N_1$ to initial state of $N_2$

Closure under Kleene Star

Add $\epsilon$-transitions from all accept states to initial state, and new initial (and accepting) state with $\epsilon$-transitions to original one.

Can we construct these without $\epsilon$-transitions?

Yes, since we can convert any $\epsilon$-NFA into one without $\epsilon$:

Union: new initial state, add transitions of both initial states

Concatenation: transitions from all accepting states of $N_1$ to successors of initial state of $N_2$
(including from initial state of $N_1$, if accepting)
if $\epsilon \in L(N_1)L(N_2)$, initial state stays accepting

Star: initial state becomes accepting transitions from any accepting state to corresponding successors of initial state