Discrete Mathematics

NFA — Spring 2025

Konstantin Chukharev

§1 Non-determinism

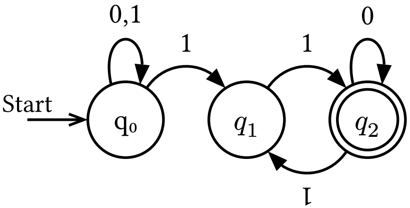
Non-deterministic Finite Automata

Definition 1: A *non-deterministic finite automaton* (NFA) is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where

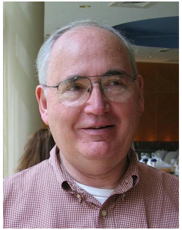
- Q is a *finite* set of states,
- Σ is an *alphabet* (finite set of input symbols),
- $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$ is a *transition function*,
- $q_0 \in Q$ is an *initial* (*start*) state,
- $F \subseteq Q$ is a set of *accepting* (*final*) states.

Note: $\delta : (q, c) \mapsto \underbrace{\{q^{(1)}, \ldots, q^{(n)}\}}_{\text{non-determinism}}$

	0	1
q0	q0	q0, q1
q1		q2
q2	q2	q1




Michael Rabin


Dana Scott

Non-Determinism

Definition 2: A model of computation is *deterministic* if at every point in the computation, there is exactly *one choice* that can make.

Note: The machine accepts if *that* series of choices leads to an accepting state.

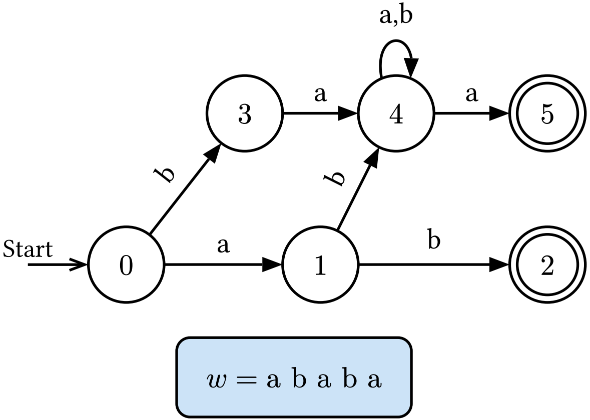
Definition 3: A model of computation is *non-deterministic* if the computing machine may have *multiple decisions* that it can make at one point.

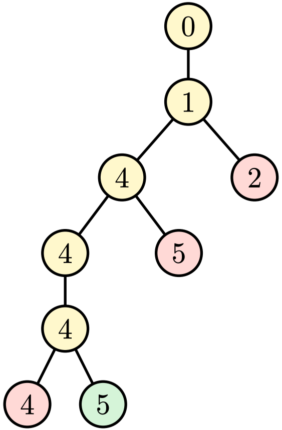
Note: The machine accepts if *any* series of choices leads to an accepting state.

Intuition on non-determinism:

1. Tree computation
2. Perfect guessing
3. Massive parallelism

Tree Computation





- At each *decision point*, the automaton *clones* itself for each possible decision.
- The series of choices forms a directed, rooted *tree*.
- At the end, if *any* active accepting (green) states remain, we *accept*.

Perfect Guessing

- We can view nondeterministic machines as having *magic superpowers* that enable them to *guess* the *correct choice* of moves to make.
- Machine can always guess the right choice if one exists.
- No physical implementation is known, yet.

Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states *at once*.
- Each symbol read causes a transition on every active state into each potential state that could be visited.
- Non-deterministic machines can be thought of as machines that can try any number of options in parallel (using an unlimited number of “processors”).

Computation Model

Reachability relation for NFA is very similar to DFA's:

$$\langle q, x \rangle \vdash_{\text{DFA}} \langle r, y \rangle \quad \text{iff} \quad \begin{cases} x = cy & \text{where } c \in \Sigma \\ r = \delta(q, c) \end{cases}$$

$$\langle q, x \rangle \vdash_{\text{NFA}} \langle r, y \rangle \quad \text{iff} \quad \begin{cases} x = cy & \text{where } c \in \Sigma \\ r \in \delta(q, c) \end{cases}$$

Definition 4: An NFA *accepts* a word $w \in \Sigma^*$ iff $\langle q_0, w \rangle \vdash^* \langle f, \varepsilon \rangle$ for some $f \in F$.

Definition 5: A language *recognized* by an NFA is a set of all words accepted by the NFA.

$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \langle q_0, w \rangle \vdash^* \langle f, \varepsilon \rangle, f \in F \}$$

Rabin–Scott Powerset Construction

Any NFA can be converted to a DFA using *Rabin–Scott subset construction*.

$\mathcal{A}_{\mathrm{N}} = \langle \Sigma, Q_{\mathrm{N}}, \delta_{\mathrm{N}}, q_0, F_{\mathrm{N}} \rangle$

- $Q_{\mathrm{N}} = \{q_1, q_2, \ldots, q_n\}$
- $\delta_{\mathrm{N}} : Q_{\mathrm{N}} \times \Sigma \longrightarrow \mathcal{P}(Q_{\mathrm{N}})$

$\mathcal{A}_{\mathrm{D}} = \langle \Sigma, Q_{\mathrm{D}}, \delta_{\mathrm{D}}, \{q_0\}, F_{\mathrm{D}} \rangle$

- $Q_{\mathrm{D}} = \mathcal{P}(Q_{\mathrm{N}}) = \{\emptyset, \{q_1\}, \ldots, \{q_2, q_4, q_5\}, \ldots, Q_{\mathrm{N}}\}$
- $\delta_{\mathrm{D}} : Q_{\mathrm{D}} \times \Sigma \longrightarrow Q_{\mathrm{D}}$
- $\delta_{\mathrm{D}} : (A, c) \mapsto \{r \mid \exists q \in A.\, r \in \delta_{\mathrm{N}}(q, c)\}$
- $F_{\mathrm{D}} = \{A \mid A \cap F_{\mathrm{N}} \neq \emptyset\}$

ε-NFA

Definition 6: *Epsilon closure* of a state q, denoted $E(q)$ or ε-clo(q), is a set of states reachable from q by ε-transitions.

$$E(q) = \varepsilon\text{-clo}(q) = \left\{ r \in Q \mid q \xrightarrow{\varepsilon} r \right\}$$

This definition can be extended to the *sets of states*. For $P \subseteq Q$:

$$E(P) = \bigcup_{q \in P} E(q)$$

Note: $q \in \varepsilon$-clo(q) since each state has an *implicit* ε-loop.

Example: For the following NFA, epsilon closure of q is ε-clo$(q) = \{q, r, s\}$.

$$q \xrightarrow{\varepsilon} r \xrightarrow{\varepsilon} s$$

From ε-NFA to NFA

To construct NFA from ε-NFA:

1. Perform a transitive closure of ε-transitions.
   - After that, accepted words contain *no two consecutive* ε-transitions.
2. Back-propagate accepting states over ε-transitions.
   - After that, accepted words *do not end* with ε.
3. Perform symbol-transition back-closure over ε-transitions.
   - After that, accepted words *do not contain* ε-transitions.
4. Remove ε-transitions.
   - After that, you get an NFA.

Kleene's Theorem

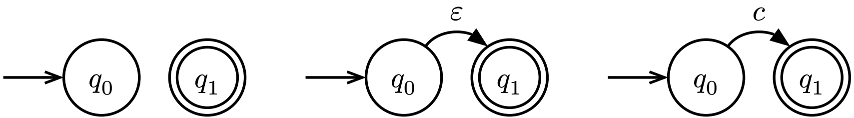
Theorem 1: REG $=$ AUT.

Proof *(REG $\subseteq$ AUT)*: *For every regular language, there is a DFA that recognizes it.*

Proof by induction over the *generation index k*. Show that $\forall k.\, \mathrm{Reg}_k \subseteq \mathrm{AUT}$.

Another name of this part: Thompson's construction (NFA from regular expression).

Base: $k = 0$, construct automata for $\mathrm{Reg}_0 = \{\emptyset, \{\varepsilon\}, \{c\}$ for $c \in \Sigma\}$, *showing* $\mathrm{Reg}_0 \subseteq \mathrm{AUT}$:



$L = \emptyset$ $L = \{\varepsilon\}$ $L = \{c\}$

Induction step: $k > 0$, already have automata for languages $L_1, L_2 \in \mathrm{Reg}_{k-1}$.

$\square$

Kleene's Theorem [2]

Proof *(AUT $\subseteq$ REG)*: *The language recognized by a DFA is regular.*

TODO: Kleene's algorithm (regular expression from DFA): Given a deterministic automaton $\mathcal{A}$, we can construct a regular expression for the regular language recognized by $\mathcal{A}$. $\square$