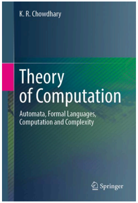
Nondeterministic Finite Automata

Prof. (Dr.) K.R. Chowdhary

Former Professor & Head, Department of Computer Sc. & Engineering
MBM Engineering College, Jodhpur

K.R. Chowdhary

Theory of Computation



Automata, Formal Languages, Computation and Complexity

Focuses on pedagogy in its writing, that represents a refreshing approach

Ensures comprehensive and enjoyable learning

Undergone a rigorous classroom testing

Introduction to NFA

A nondeterministic FA starts in one state. It accepts input, if at least one sequence of transitions leads to a final state. Intuitively: it always "guesses right."

- It allow several transitions for a single (state, symbol) pair
- Simultaneous transitions with the same input symbol
- may change states without reading any input
- all possible input alternatives are not necessary at every state

This generalization allows more than one execution of the same word (this is nondeterminism). A word is accepted iff one of these executions ends in a final state.

NFA: Formal Definition

- Let M be an *NFA*:

$$M = (Q, \Sigma, \delta, s, F) \tag{1}$$

$$\delta : Q \times \Sigma \to 2^Q, \tag{2}$$

  Therefore, $\delta(q_0, q) \in 2^Q$

- The language accepted by *NFA* is:

$$L = L(M) = \{w \in \Sigma^* \mid (q_0, w) \vdash_M^* (\{\dots, q_f, \dots\}, \varepsilon), q_f \in F\} \tag{3}$$

- Since, an *NFA* moves to a set of states, it simulates more than one *DFA* in parallel. Collection of all these states is called *state space*.
- If entire state space is searched in sequential way, it will require back tracking, resulting to a slower process(in DFA).
- Because *NFA* accepts null(ε) input also, the generalized transition functions is modified as:

$$\delta = Q \times \Sigma \cup \{\varepsilon\} \to 2^Q. \tag{4}$$

For any given regular expression, an NFA will have lesser number of states. Consider the regular expressions $(ab + aba)^*$.

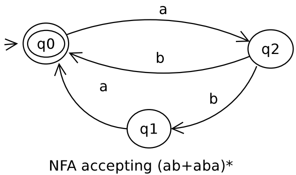


Figure 1: NFA for the regular expression $(ab + aba)^*$.

The *nfa* has far 3 states. Also, all inputs are not allowed, e.g., $aba \Rightarrow q_0 q_1 q_0 q_1$, and $\delta(q_0, abb) = \phi$.

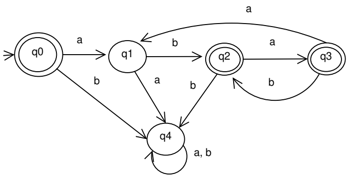


Figure 2: DFA for the regular expression $(ab + aba)^*$.

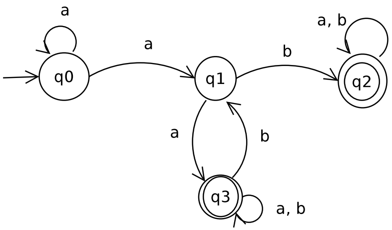


Figure 3: NFA accepting $w = aab$.

$$(q_0, aab) \vdash (\{q_0, q_1\}, ab)$$
$$\vdash (q_0, ab) \cup (q_1, ab)$$
$$\vdash (\{q_0, q_1\}, b) \cup (\{q_3\}, b)$$
$$\vdash (q_0, b) \cup (q_1, b) \cup (q_3, b)$$
$$\vdash (\phi, \varepsilon) \cup (q_2, \varepsilon) \cup (\{q_1, q_3\}, \varepsilon)$$
$$\vdash (\{q_1, q_2, q_3\}, \varepsilon)$$

Since in the final state set $\{q_1, q_2, q_3\}$, the $q_2, q_3 \in F$, string abb is accepted by the *NFA*.

- Are *NFAs* more powerful than than *DFA*? Can they recognize a language that *DFA* cannot recognize?
- If one of the paths in *NFA* leads to final state, input is accepted. In fact there are other paths also for the same input !!. So, are the class of languages accepted by *DFA* are subset of that by *NFA*?
- An *NFA* is in fact not powerful than DFA.

For every NFA there is an equivalent DFA

Theorem

For every NFA their exists an equivalent DFA, which accepts the same language as the NFA.

Proof.

Let the *NFA* be $M_N = \{Q, \Sigma, \delta, s, F\}$, and let there exists a *DFA*, $M_D = \{Q', \Sigma, \delta', s', F'\}$, which accepts the same language. Machine M_D can be constructed from M_N as follows:

1. For every state $q_i \in Q$ obtain its closure due to ε transitions as: $C(q_i) := \{q_i\}$;
2. *while there is a transition* $\delta(q_i, \varepsilon) = r_k$

   $C(q_i) := C(q_i) \cup \{r_k\}$;
3. $Q' = 2^Q, S' = C(s)$; $F' = \{\ldots, q_f, \ldots\}, \forall q_f \in F$
4. δ' : if there is transition $\delta(q_i, a) = q_j$, and in the *DFA* we have $\{\ldots, q_i, \ldots\}$ and $\{\ldots, q_j, \ldots\}$ state sets, then keep a transition between these state sets labeled as a.

□

Obtain a DFA for given NFA

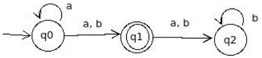


Figure 4: NFA

We generate all next state sets at every state, starting with q_0, for all the alphabet symbols. Then next states for all these states:

$$\delta(q_0, a) = \{q_0, q_1\}$$
$$\delta(q_0, b) = \{q_1\}$$

$$\begin{aligned}\delta(\{q_0, q_1\}, a) &= \delta(q_0, a) \cup \delta(q_1, a)\\ &= \{q_0, q_1\} \cup \{q_2\}\\ &= \{q_0, q_1, q_3\}.\end{aligned}$$

$$\delta(\{q_0, q_1\}, b) = \{q_1, q_2\}$$
$$\delta(\{q_1, 1\}, a) = \{q_2\}$$
$$\delta(\{q_1\}, b) = \{q_2\}$$

$$\delta(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}$$
$$\delta(\{q_0, q_1, q_2\}, b) = \{q_1, q_2\}$$

$$\begin{aligned}\delta(\{q_1, q_2\}, a) &= \delta(q_1, a) \cup \delta(q_2, a)\\ &= \{q_2\} \cup \{\phi\}\\ &= \{q_2\}.\end{aligned}$$

$$\begin{aligned}\delta(\{q_1, q_2\}, b) &= \{q_2\}\\ \delta(\{q_2, a\}) &= \phi\\ \delta(\{q_2\}, b) &= \{q_2\};\end{aligned}$$

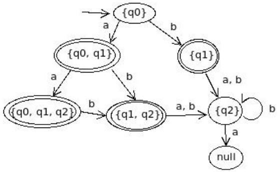


Figure 5: Equivalent DFA for a given NFA.

Since it does not exist, we can drop the null state.

Bibliography

Chowdhary, K.R. (2025). Variants of Finite Automata. In: Theory of Computation. Springer, Singapore.
https://doi.org/10.1007/978-981-97-6234-7_3