MTL783: Theory of Computation 1st Semester, 2025-2026

Lecture 04 — 04/08/2025

Lecturer: Prof. Minati De *Scribe: Team 04*

Scribed by:

1. Aniket Anand (2022MT11259)
2. Aditya Arya (2021MT60958)
3. S. Madhur Goyal (2022MT11269)
4. Abhinav (2022MT11940)
5. Aditya Thomas (2021MT60944)

1 Overview

In the last lecture, we discussed the basics of regular languages and deterministic finite automata (DFAs).

In this lecture, we introduce **non-deterministic finite automata (NFAs)**, compare their expressive power with DFAs, and cover the *subset construction* for converting NFAs to DFAs.

2 Regular-Language Warm-up

Question: Show that the language

$$L = \{awa \mid w \in \{a,b\}^*\}$$

is regular.

Sketch: The DFA in Fig. 1 accepts precisely those strings whose first and last symbols are a.

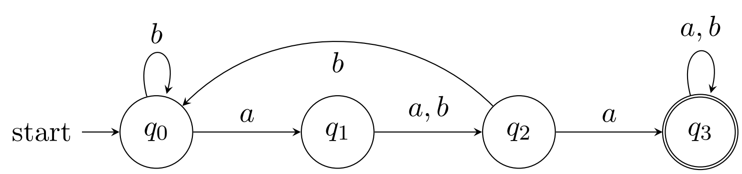

Figure 1: DFA for $L = \{awa\}$. States with double circles are final.

3 Non-Deterministic Finite Automaton (NFA)

Definition 1 (NFA)**.** *A Non-deterministic Finite Automaton (NFA) is a 5-tuple* $N = (Q, \Sigma, \delta, q_0, F)$*, where:*

- Q *– a finite set of states,*
- Σ *– a finite input alphabet,*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ *– the transition function,*
- $q_0 \in Q$ *– the initial state,*
- $F \subseteq Q$ *– the set of accepting (final) states.*

Example 1: Accepting $\lambda, 10, 1010, \ldots$

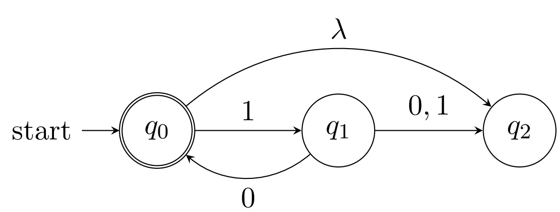

Figure 2: NFA accepting $\lambda, 10, 1010, 101010, \ldots$.

Explanation:

- **Accepts** λ**:** Initial state q_0 is accepting, and no input is needed.
- **Accepts** $10, 1010, 101010, \ldots$**:** Each repetition of "10" moves the automaton along $q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{\lambda} q_0$, looping back.
- **Rejects strings starting with 0:** There is no transition from q_0 on 0.
- An NFA is allowed to have transitions undefined (like $\delta(q_0, 0)$); this is valid.
- Even if some computation paths fail, a string is accepted if *any path* ends in a final state.

—

Example 2: Multiple transitions from same state

Behavior:

- From q_0, input a can lead to both q_2 and q_4 (nondeterminism).

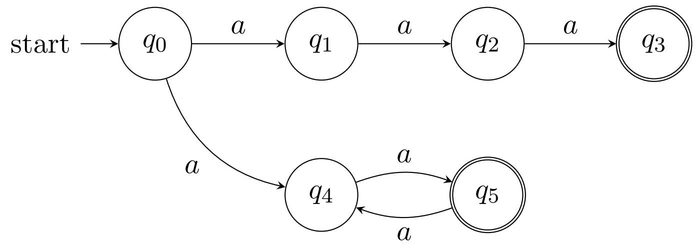

Figure 3: NFA with two choices after reading 'a' from q_0.

- $aaaa$ is accepted through path $q_0 \xrightarrow{a} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_4 \xrightarrow{a} q_5$; however, $aaaaa$ is not accepted.
- Take the case of aaa:
  - aaa is not accepted through path $q_0 \xrightarrow{a} q_4 \xrightarrow{a} q_5 \xrightarrow{a} q_4$.
  - aaa is accepted through path $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$.
  - Hence aaa will be **accepted** since there is a possible path where it ends at a final stage
- b is accepted via path $q_0 \xrightarrow{b} q_1$.

4 Extended Transition Function

Definition: Let $\delta^*(q_i, \omega) = Q_j$ where $q_i \to q_j$ (all such) where we have a labeled path ω.

$$\delta^*(q_i, \omega) \text{ contains } q_j \text{ iff } \exists \text{ a walk in the transition graph from } q_i \text{ to } q_j \text{ labeled } \omega.$$

[Try to define it recursively ⇒ You will be in trouble?]

Example

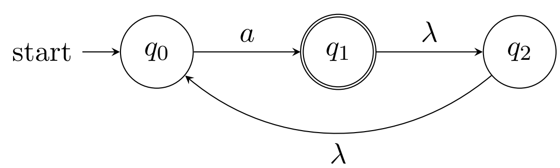

Figure 4: Example NFA with λ-moves for extended transition function.

At q_1, reading $[a]$: pointer at a, we read an empty string λ to reach q_2.

$$\delta^*(q_1, a) = \{q_1, q_2, q_0\}$$
$$\delta^+(q_2, \lambda) = \{q_0, q_2\}$$
$$\delta^*(q_2, aa) = \{q_0, q_1, q_2\}$$

Definition 2 (Language of an NFA)**.** *The language L accepted by an NFA $N = (Q, \Sigma, \delta, q_0, F)$ is defined as the set of all strings accepted by N. Formally,*

$$L(N) = \{\omega \in \Sigma^* \mid \delta^*(q_0, \omega) \cap F \neq \varnothing\}.$$

Example

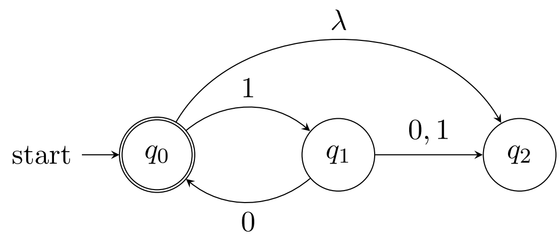

Figure 5: Example NFA N_2 with λ-moves.

We have:

$$L(N_2) = \{(10)^n \ : \ n \geq 0\}$$

Constructed DFA with $L(N_2)$ language

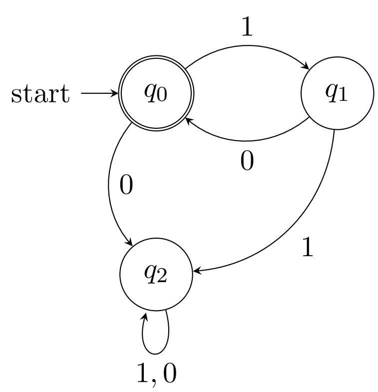

Figure 6: DFA equivalent to N_2 accepting $\{(10)^n : n \geq 0\}$.

5 Equivalence of DFA and NFA

Theorem 3 (Subset Construction)**.** *For every NFA $N = (Q, \Sigma, \delta, q_0, F)$, there exists a DFA $M = (Q', \Sigma, \delta', q_0', F')$ such that*

$$L(M) = L(N).$$

Two finite acceptors are said to be acceptors M_1 and M_2, and are said to be **equivalent** if and only if they both accept the same language, i.e.,

$$L(M_1) = L(M_2)$$

$$[\text{Every DFA is an NFA}] \quad \Rightarrow \quad \text{NFA has some non-deterministic property.}$$

Theorem: Let L be the language accepted by an NFA

$$M_N = (Q_N, \Sigma, \delta_N, q_{0N}, F_N)$$

Then there exists a DFA

$$M_D = (Q_D, \Sigma, \delta_D, \{q_{0D}\}, F_D)$$

such that

$$L = L(M_D)$$

An example of this equivalence is shown below.

5.1 Example

Fig. 7 shows conversion of an NFA to an equivalent DFA.

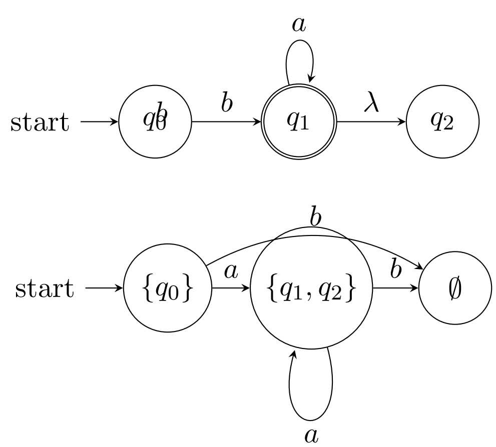

Figure 7: NFA (top) and equivalent DFA (bottom) via subset construction.

References

[1] Peter Linz, *An Introduction to Formal Languages and Automata*, 6th Edition, Jones & Bartlett Learning, 2016.