Nondeterministic Finite Automata

sara Sorahi

Introduction to the Theory of Computation

Why Do We Need Nondeterminism?

Deterministic finite automata (DFAs):

- have exactly one transition per symbol,
- are often difficult to design,
- require remembering information at all times.

Idea: What if the machine could explore *several possibilities at once*?

What Nondeterminism Means

Nondeterminism is:

- not randomness,
- not probability,
- not physical guessing.

An NFA accepts a string if *there exists at least one computation path* that leads to acceptance.

Formal Definition of an NFA

An NFA is a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

Key difference from DFA:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$$

- transitions return *sets of states*
- ε-moves are allowed

Acceptance in an NFA

A string is accepted if:

- at least one computation path
- consumes the entire input
- and ends in an accepting state.

Acceptance is **existential**. Rejection requires that *all* paths fail.

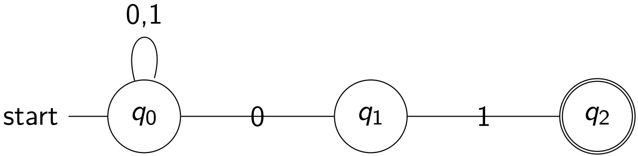
Example 1: Strings Ending in 01

Language:

$$L = \{w \in \{0,1\}^* \mid w \text{ ends with } 01\}$$

The NFA guesses where the final 01 begins.

NFA Graph: Ending in 01



Wrong guesses die; one correct guess is enough.

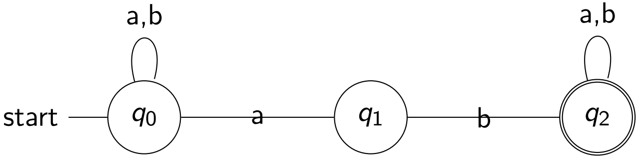
Example 2: Contains Substring ab

Language:

$$L = \{w \in \{a, b\}^* \mid w \text{ contains } ab\}$$

The automaton branches when it sees a.

NFA Graph: Contains ab



Once accepted, the automaton stays accepting.

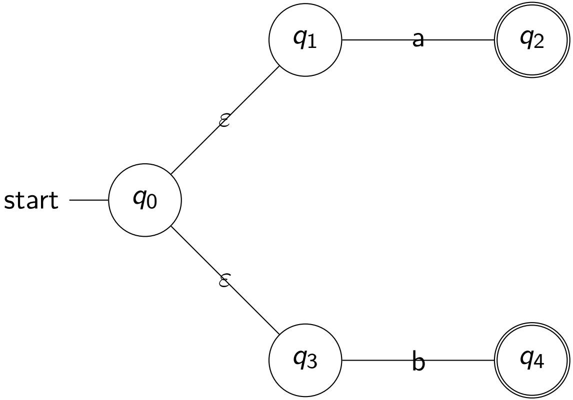
Example 3: Contains aa OR bb

Language:

$$L = \{w \mid w \text{ contains } aa \text{ or } bb\}$$

NFAs naturally express logical OR.

NFA Graph: `aa` OR `bb`

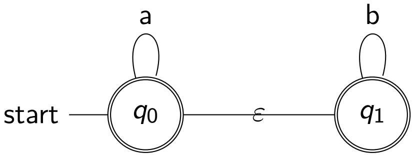


Acceptance requires success in either branch.

Example 4: Strings of the Form $a^* b^*$

The NFA guesses when the input switches from as to bs.

NFA Graph: $a^* b^*$

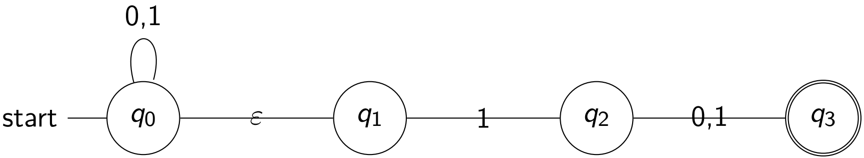


ε-moves change state without consuming input.

Example 5: Third Symbol from the End Is 1

The NFA guesses a future boundary and verifies locally.

NFA Graph: Third from the End



Only correct guesses survive to acceptance.

Conceptual Takeaway

Nondeterminism allows parallel exploration of possibilities, but acceptance requires only one successful path.

Why NFA to DFA?

NFAs are:

- easier to design
- intuitive and compact

DFAs are:

- deterministic
- executable step by step

Question: Does nondeterminism give us more computational power?

Central Result

Theorem. For every NFA, there exists an equivalent DFA that accepts exactly the same language.

Meaning:

- NFAs and DFAs accept the same languages
- Nondeterminism adds convenience, not power

Key Idea of Subset Construction

Recall:

- An NFA may be in multiple states at once
- A DFA must be in exactly one state

Idea: One DFA state represents **a set of NFA states**.

How DFA States Are Formed

After reading some input, an NFA may be in:

$$\{q_1, q_3, q_4\}$$

The DFA will have:

$$\text{one state representing } \{q_1, q_3, q_4\}$$

DFA states $=$ subsets of NFA states

Accepting States Rule

A DFA state is accepting if:

- it contains **at least one accepting NFA state**

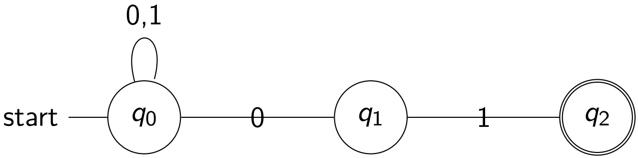
Existential acceptance is preserved.

Example 1: Ending in 01 (NFA Idea)

NFA strategy:

- scan freely
- whenever a 0 appears, guess it starts 01

Example 1: NFA Graph



Example 1: DFA Transitions

Start state:

$$\{q_0\}$$

- on 0 $\rightarrow \{\mathrm{q}_0, q_1\}$
- on 1 $\rightarrow \{\mathrm{q}_0\}$

Accepting DFA states:

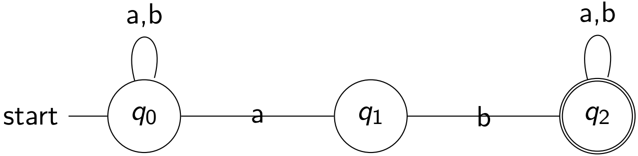
$$\text{any set containing } q_2$$

Example 2: Contains `ab` (NFA Idea)

NFA strategy:

- scan normally
- on `a`, branch and wait for `b`

Example 2: NFA Graph



Example 2: DFA Transitions

From $\{q_0\}$:

- $\texttt{a} \rightarrow \{\mathtt{q}_0, q_1\}$

- $\texttt{b} \rightarrow \{\mathtt{q}_0\}$

From $\{q_0, q_1\}$:

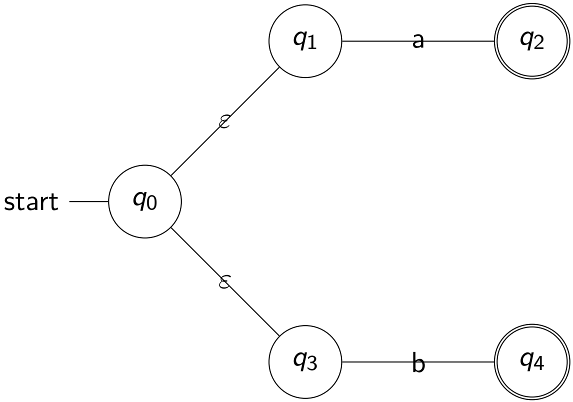
- $\texttt{b} \rightarrow \{\mathtt{q}_0, q_2\}(\textit{accepting})$

Example 3: `aa` OR `bb` (NFA Idea)

NFA strategy:

- use ε to branch
- one branch checks `aa`
- one branch checks `bb`

Example 3: NFA Graph



Example 3: DFA Start State

ε-closure of q_0:

$$\{q_0, q_1, q_3\}$$

- on `a` $\rightarrow$ $\{\mathtt{q}_2\}$
- on `b` $\rightarrow$ $\{\mathtt{q}_4\}$

Both are accepting DFA states.

Practice and Useful Links

Recommended tools:

- JFLAP — visual NFA/DFA construction
- Automata Tutor (University of Stuttgart)
- Sipser, Chapter 1 — worked examples

Practice tip: Always write the **set of NFA states** before drawing the DFA.