THEORY OF AUTOMATA & FORMAL LANGUAGES

Nondeterministic Finite Automata

Recognizing a language through alternative paths

Branch → Track → Verify → Decide

From DFA to NFA

The same input can allow several possible next states.

DFA: one next state

Each state and symbol has one destination.

Exactly one path for each input.

Every symbol has a defined transition.

Accept if that path ends in F.

Example: $\delta(q_0,a) = q_1$.

NFA: a set of next states

Zero, one or many destinations.

Several paths may exist for one input.

A missing transition ends that path.

Accept if any complete path ends in F.

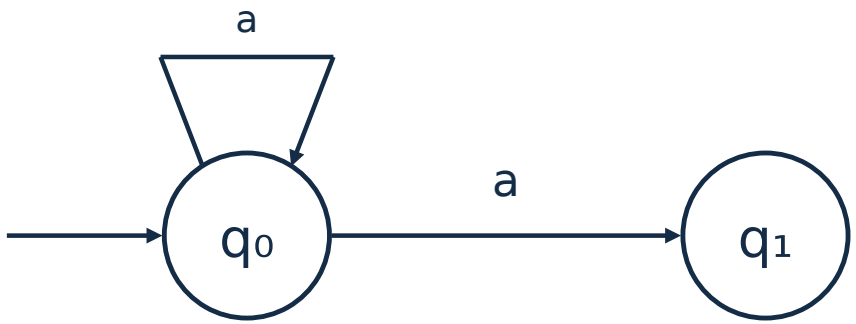
Example: $\delta(q_0,a) = \{q_0,q_1\}$.

Nondeterminism means alternative possibilities, not random choices.

Understand branching

Example: from q_0, reading a may lead to q_0 or q_1.

Two possible paths



Two possible destinations on a

Interpretation

Start with active set $\{q_0\}$.

Read a: follow both a transitions.

New active set = $\{q_0, q_1\}$.

Both paths have consumed one a.

Do not read a twice on either path.

Track all possibilities together, rather than guessing a single path.

Formal mathematical model

An NFA without ε-transitions has the same five components as a DFA.

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

Q: finite set of states.

Σ: finite input alphabet.

$q_0 \in Q$: initial state.

$F \subseteq Q$: accepting states.

$\delta: Q \times \Sigma \to 2^Q$.

What changes in δ?

2^Q: set of all subsets of Q.

For $Q = \{q_0, q_1\}$:

$2^Q = \{\varnothing, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$.

$\delta(q_0, a)$ may be any such subset.

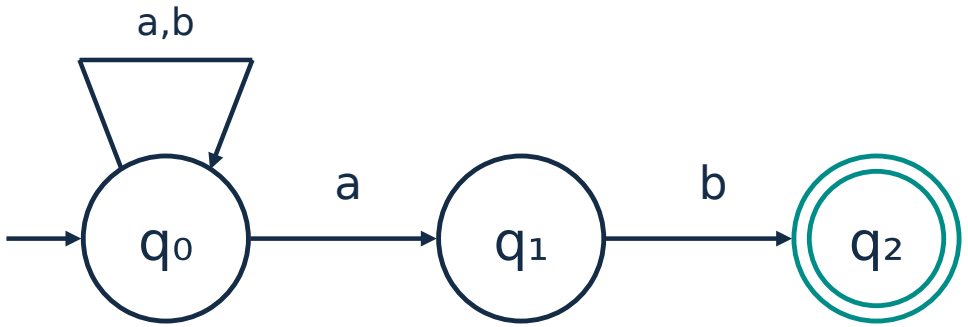
$\varnothing$ means no next state.

Here we use ordinary NFAs; ε-transitions will be introduced separately.

Construct an NFA: strings ending in ab

Σ = {a,b}; regular expression (a|b)*ab.

Complete NFA



No unlabelled or implied transitions

State meanings and rules

q_0: scan any prefix.

q_1: a possible final a was read.

q_2: the final ab was matched.

q_0 loops on a,b; also a → q_1.

q_1 on b → q_2; no other edges.

Only q_2 is accepting. The match must finish exactly at the end of input.

Read the transition table

Each table entry is a set of states, including the empty set.

Transition table

State	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
★ q_2	$\emptyset$	$\emptyset$

Initial active set = $\{q_0\}$.

Update the active set

Let S be the current set of states.

Read the next symbol x.

New $S = \bigcup \delta(q,x)$, for $q \in S$.

Collect destinations; remove repeats.

Accept after input iff $S \cap F \neq \emptyset$.

No outgoing edge means that particular path cannot consume the symbol.

Accepted example: aab

Trace every possible state after each input symbol.

Active-state calculation

Start: $S_0 = \{q_0\}$.

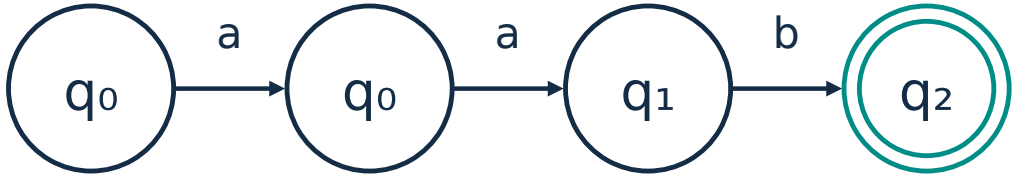
Read a: $S_1 = \{q_0, q_1\}$.

Read a: $S_2 = \{q_0, q_1\}$.

Read b: $S_3 = \{q_0, q_2\}$.

$S_3 \cap F = \{q_2\} \neq \emptyset$.

One successful path



One accepting path: $q_0 \to q_0 \to q_1 \to q_2$

Accept aab because at least one complete path ends in q_2.

Rejected example: aba

Reaching an accepting state early does not guarantee acceptance.

Active-state calculation

Start: $S_0 = \{q_0\}$.

Read a: $S_1 = \{q_0, q_1\}$.

Read b: $S_2 = \{q_0, q_2\}$.

Read a: $S_3 = \{q_0, q_1\}$.

$S_3 \cap F = \varnothing$.

Why does it fail?

q_2 was reached after ab.

But one symbol a remains.

q_2 has no outgoing transition on a.

Other paths survive but are not final.

Therefore aba is rejected.

An accepting path must consume the whole string.

Practice and summary

Use the same NFA for strings ending in ab.

Try these inputs

ε → $\{q_0\}$: reject.

a → $\{q_0,q_1\}$: reject.

ab → $\{q_0,q_2\}$: accept.

bab → $\{q_0,q_2\}$: accept.

abb → $\{q_0\}$: reject.

Key rules

A DFA is a special case of an NFA.

An NFA may have missing edges.

Branches consume the same input.

One successful complete path suffices.

DFA and NFA recognize regular languages.

Next: systematic NFA construction and ε-transitions.