THEORY OF AUTOMATA & FORMAL LANGUAGES

Deterministic Finite Automata

From language conditions to state transition machines

Understand → Construct → Trace → Verify

From a language to a machine

Recall: a language is a set of strings over an alphabet.

The recognition problem

$\Sigma = \{0, 1\}$

$L = \{w : w \text{ ends in } 1\}$

Accept: 1, 01, 101

Reject: ε, 0, 110

Need a rule for every input.

How a machine works

Start in a known state.

Read one symbol at a time.

Update the state using a rule.

After the last symbol, decide.

Accept only in a final state.

A DFA recognizes strings; it does not generate them.

What does a state remember?

Example: strings ending in 1 over $\Sigma = \{0, 1\}$.

Meaning of each state

q_0: no input yet, or last symbol 0

q_1: last symbol is 1

Start at q_0 (ε is rejected).

Only q_1 is accepting.

Earlier symbols need not be stored.

Transition rules

At q_0, read 0 → q_0

At q_0, read 1 → q_1

At q_1, read 0 → q_0

At q_1, read 1 → q_1

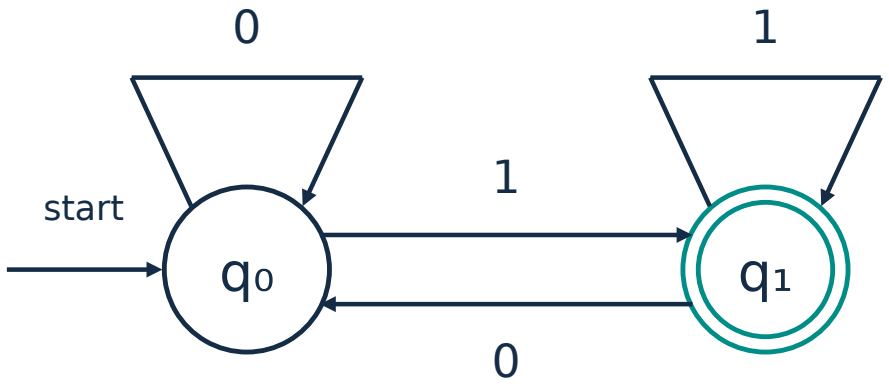
Every state has a rule for 0 and 1.

State meaning determines the transitions.

State transition diagram

A directed arrow is a transition; its label is the input symbol.

Diagram



Double circle = accepting state

How to read the diagram

Incoming arrow: initial state q_0

Double circle: accepting state q_1

Loop: stay in the same state.

Arrow $q_0 \to q_1$ reads 1.

Arrow $q_1 \to q_0$ reads 0.

Exactly one outgoing transition per symbol from every state.

Formal mathematical model

A DFA is specified by a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$.

The five components

$Q = \{q_0, q_1\}$: finite set of states

$\Sigma = \{0, 1\}$: input alphabet

$\delta : Q \times \Sigma \to Q$: transition function

$q_0 \in Q$: initial state

$F = \{q_1\}$: set of final states

Transition table

Current state	0	1
→ q_0	q_0	q_1
★ q_1	q_0	q_1

Example: $\delta(q_1, 0) = q_0$

→ means initial; ★ means final.

Deterministic means one unique next state for each state-symbol pair.

Trace an accepted string

Input w = 101; read from left to right.

Step-by-step trace

Start: q_0

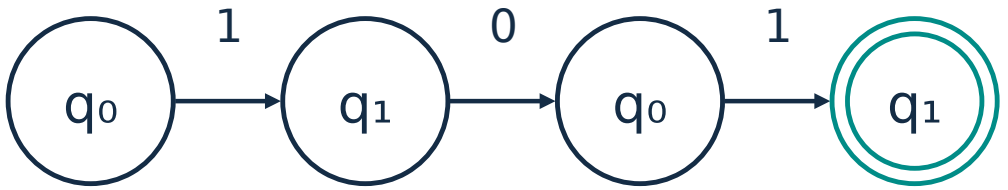
Read 1: $\delta(q_0, 1) = q_1$

Read 0: $\delta(q_1, 0) = q_0$

Read 1: $\delta(q_0, 1) = q_1$

End: $q_1 \in F \rightarrow$ ACCEPT

Why it works



Final state q_1: accepted

Only the state after consuming the entire string determines acceptance.

Trace a rejected string

Input w = 110; the same DFA processes it.

Step-by-step trace

Start: q_0

Read 1: $\delta(q_0, 1) = q_1$

Read 1: $\delta(q_1, 1) = q_1$

Read 0: $\delta(q_1, 0) = q_0$

End: $q_0 \notin F \rightarrow$ REJECT

Boundary cases

ε: no transition; stay at q_0.

Therefore ε is rejected.

0: ends at $q_0 \rightarrow$ reject.

1: ends at $q_1 \rightarrow$ accept.

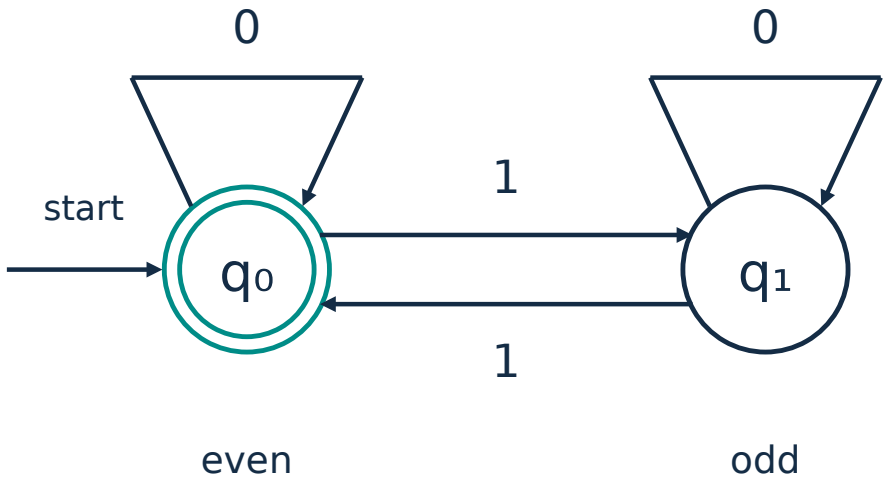
Visiting q_1 early is not enough.

Rejection does not mean that the machine stopped before reading all symbols.

A second DFA: even number of 1s

The state now remembers parity rather than the last symbol.

Diagram



Double circle = accepting state

Construct the rules

q_0: even count (initial, final)

q_1: odd count

Reading 0 does not change parity.

Reading 1 toggles parity.

Trace 101: $q_0 \to q_1 \to q_1 \to q_0$.

ε is accepted: zero occurrences of 1 is an even count.

Practice and common mistakes

Apply the state meanings before drawing transitions.

Practice: ends in 1

Input 001 → q_1: accept

Input 100 → q_0: reject

Input ε → q_0: reject

Why two states? Track last symbol.

Every state needs both input rules.

Practice: even number of 1s

Input 00 → q_0: accept

Input 11 → q_0: accept

Input 111 → q_1: reject

Input ε → q_0: accept

Final states depend on the language.

Next: systematic DFA construction for prefixes, suffixes and substrings.