THEORY OF AUTOMATA & FORMAL LANGUAGES

Systematic Construction of DFA

From verbal conditions to complete state diagrams

Describe → Design → Draw → Verify

A systematic design method

Start with the language, not with arbitrary circles.

Define the problem

1. Identify the alphabet Σ.
2. Describe valid strings.
3. List accepted and rejected cases.
4. Identify what must be remembered.
5. Assign a state to each situation.

Complete the machine

6. Choose the initial state.
7. Mark accepting states.
8. Give one rule per state and symbol.
9. Draw the transition diagram.
10. Trace boundary and typical inputs.

Every state should have a clear meaning.

Example 1: strings starting with a

$\Sigma = \{a,b\}$; $L = \{w : w \text{ begins with } a\}$.

State meanings

q_0: nothing read yet (initial).

q_1: first symbol was a (final).

q_2: first symbol was b (dead).

ε is rejected.

Once the first symbol is known, keep it.

Transition table

State	a	b
→ q_0	q_1	q_2
★ q_1	q_1	q_1
q_2	q_2	q_2

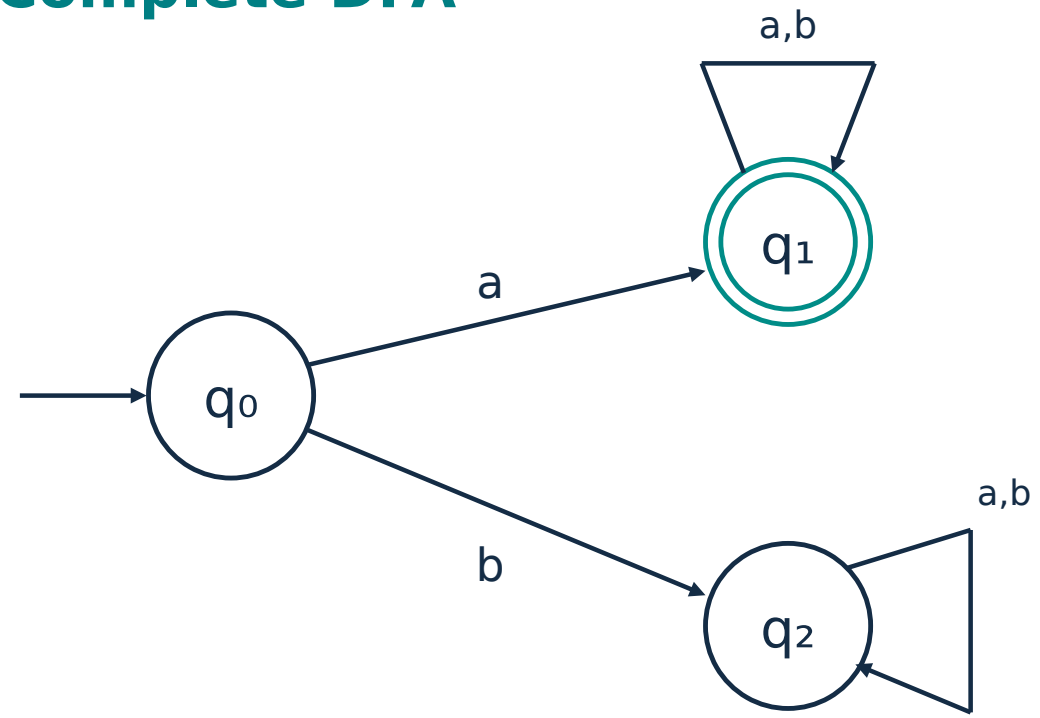
$F = \{q_1\}$

A dead state cannot reach an accepting state.

Example 1: draw and verify

The first symbol decides the result; all later symbols preserve it.

Complete DFA



Manual verification

ab: $q_0 \to q_1 \to q_1$: accept

ba: $q_0 \to q_2 \to q_2$: reject

a: $q_0 \to q_1$: accept

b: $q_0 \to q_2$: reject

ε: q_0: reject

A loop labelled a,b represents two transitions.

Example 2: strings ending with ab

$\Sigma = \{a,b\}$; remember the longest suffix matching a prefix of ab.

State meanings

q_0: no matching suffix (initial).

q_1: current string ends with a.

q_2: current string ends with ab.

Only q_2 is accepting.

The next input may destroy the match.

Transition table

State	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
★ q_2	q_1	q_0

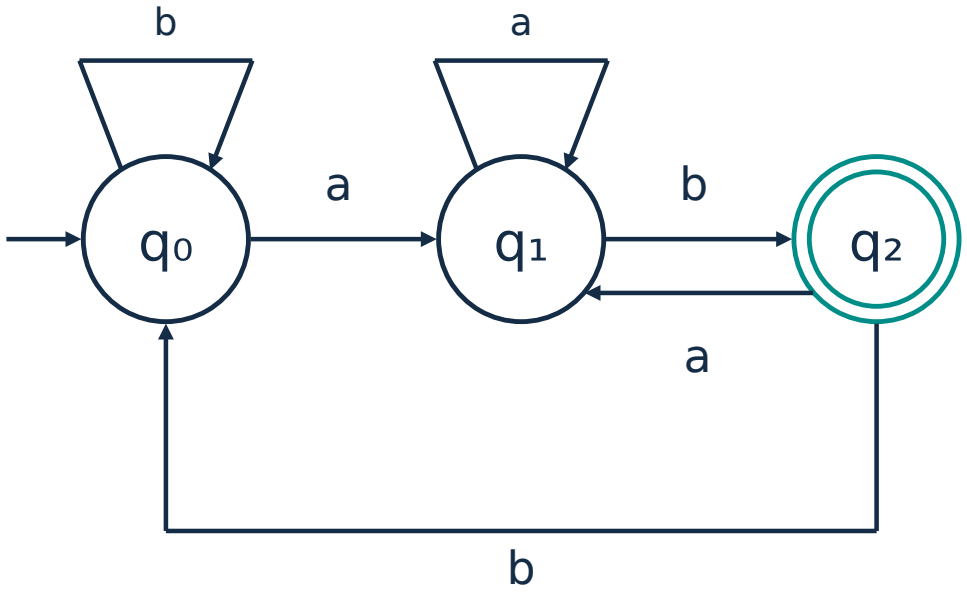
$F = \{q_2\}$

The accepting state is not necessarily permanent.

Example 2: draw and verify

The final two symbols determine membership.

Complete DFA



Manual verification

aab: $q_0 \to q_1 \to q_1 \to q_2$

Result: accept.

aba: $q_0 \to q_1 \to q_2 \to q_1$

Result: reject.

ε, a, b: all rejected.

A visit to q_2 is insufficient unless the input ends there.

Example 3: strings containing ab

$\Sigma = \{a,b\}$; once ab occurs, it remains a substring.

State meanings

q_0: no ab, no trailing a.

q_1: no ab, trailing a.

q_2: ab has occurred (final).

Start at q_0.

After reaching q_2, stay there.

Transition table

State	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
★ q_2	q_2	q_2

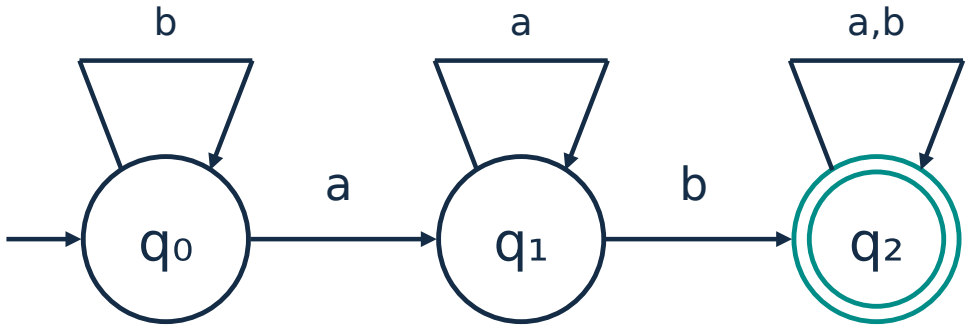
F = $\{q_2\}$

Only the outgoing rules of q_2 differ from the suffix example.

Example 3: draw and verify

A detected substring cannot disappear when more symbols arrive.

Complete DFA



Manual verification

aba: $q_0 \to q_1 \to q_2 \to q_2$

Result: accept.

bba: $q_0 \to q_0 \to q_0 \to q_1$

Result: reject.

ab, bab, aabb: accepted.

An accepting sink state is useful for permanent conditions.

Practice: distinguish prefix, suffix and substring

Compare the conditions before designing a machine.

Required prefix ab

q_0: start; a → q_1, b → dead

q_1: seen a; b → q_2, a → dead

q_2: final; loop on a,b

Dead: loop on a,b

Only ab... is accepted.

Check your understanding

String aba: prefix yes, suffix no.

String bab: prefix no, suffix yes.

String aab: prefix no, suffix yes.

All three contain ab.

ε fails all three conditions.

Next: DFA construction for counting and modular constraints.