THEORY OF AUTOMATA & FORMAL LANGUAGES

DFA: Combined Conditions

Combine two machines using product states

Define → Pair → Transition → Verify

Why combine two machines?

Example: $\Sigma = \{a,b\}$; even number of a AND even number of b.

Machine A: count a

E: even number of a.

O: odd number of a.

Start in E; accept E.

On a: switch E ↔ O.

On b: stay in current state.

Machine B: count b

E: even number of b.

O: odd number of b.

Start in E; accept E.

On b: switch E ↔ O.

On a: stay in current state.

Both machines read the same input simultaneously.

Construct the product states

A pair records the current state of each component machine.

Four possible pairs

EE: even a, even b.

EO: even a, odd b.

OE: odd a, even b.

OO: odd a, odd b.

Initial pair: EE.

Mathematical model

$Q = Q_1 \times Q_2$

$|Q| = 2 \times 2 = 4$

$\delta((p,q),x) = (\delta_1(p,x), \delta_2(q,x))$

Start: (s_1,s_2).

Both components consume x.

A product has at most $|Q_1| \times |Q_2|$ reachable states.

Build the transition table

Reading a toggles the first letter; reading b toggles the second.

Calculate each row

EE on a → OE; on b → EO

EO on a → OO; on b → EE

OE on a → EE; on b → OO

OO on a → EO; on b → OE

Every pair has two transitions.

Complete transition table

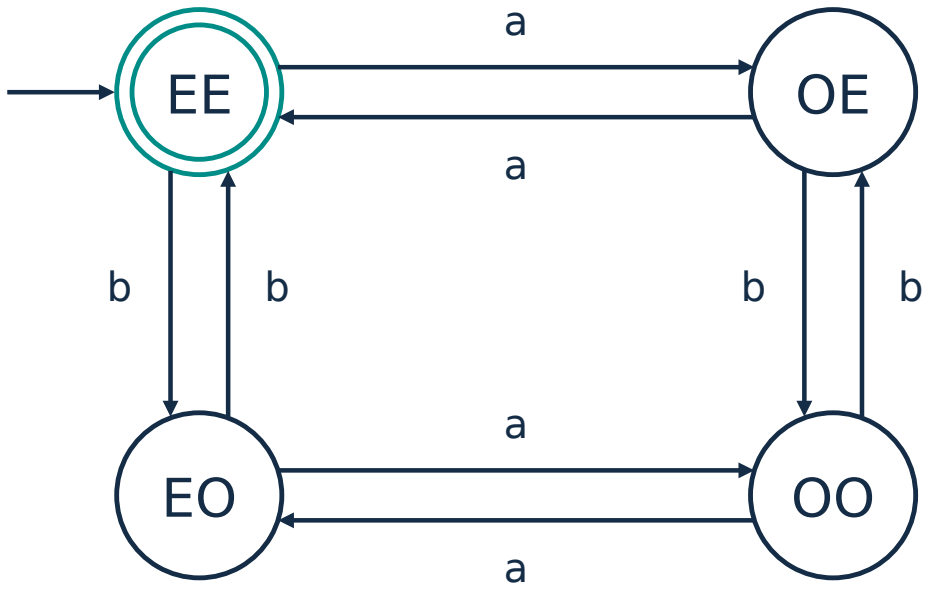
State	a	b
→★ EE	OE	EO
EO	OO	EE
OE	EE	OO
OO	EO	OE

For AND, only EE satisfies both even-count conditions.

Draw the product DFA: AND

L = {w ∈ Σ* : both a and b occur an even number of times}.

Complete DFA



Manual verification

Start at EE.

Read a → OE.

Read b → OO.

Read a → EO; read b → EE.

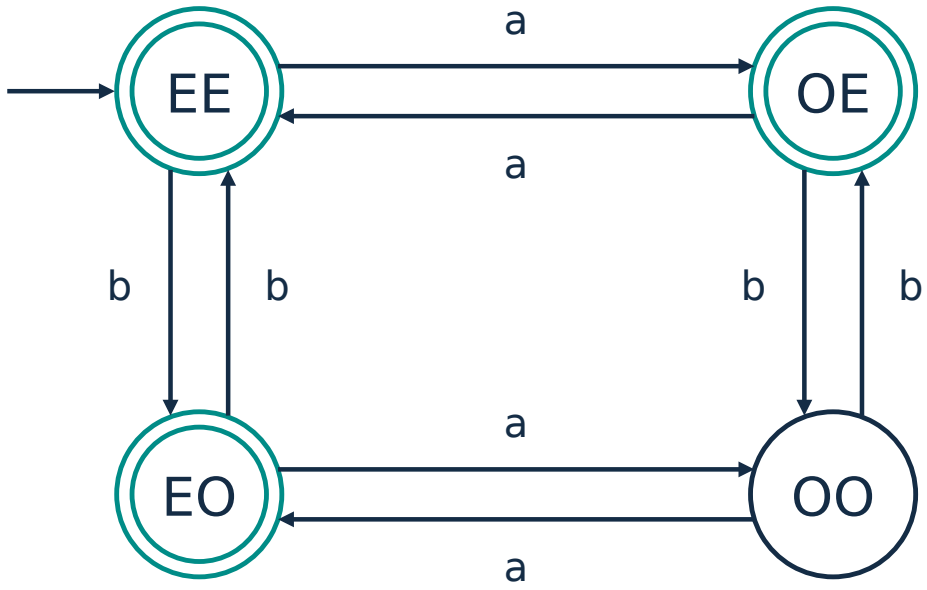
abab: accept; ab: reject.

Only EE is accepting; ε is also accepted.

Change AND to OR

L = {w ∈ Σ* : a occurs evenly OR b occurs evenly}.

Same transitions, new final states



Choose the accepting pairs

EE: both even → accept.

EO: a even → accept.

OE: b even → accept.

OO: neither even → reject.

ab: reject; abb: accept.

OR is inclusive: satisfying both conditions also qualifies.

Complement of a DFA

Complement is always relative to the chosen alphabet Σ.

Original language: both even

F = {EE}.

EE accepts; EO rejects.

OE rejects; OO rejects.

abab: accept; ab: reject.

ε: accept.

Complement: not both even

F′ = Q − F = {EO, OE, OO}.

Keep all transitions unchanged.

abab: reject; ab: accept.

ε: reject.

Requires a complete DFA.

Complement changes accepting states, not the transition rules.

Difference and exclusive OR

The product construction supports other logical conditions too.

Difference: A − B

A: even a; B: even b.

A − B means A AND NOT B.

Accept only EO.

Example: aab → EO → accept.

aa: EE → reject.

Exclusive OR: A XOR B

Exactly one condition holds.

Accept EO and OE.

aab: EO → accept.

ab: OO → reject.

ε: EE → reject.

The same transition graph supports different languages.

Practice and summary

First design component machines, then select product accepting states.

Solve for AND / OR

ε → EE: both accept.

a → OE: AND no; OR yes.

b → EO: AND no; OR yes.

ab → OO: both reject.

aabb → EE: both accept.

General acceptance rules

AND: $F_1 \times F_2$.

OR: $(F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Difference: $F_1 \times (Q_2 - F_2)$.

Complement: $Q - F$.

Remove unreachable states if any.

Next: nondeterministic finite automata (NFA).