Computation theory

Lecture 2

Brwa R. Hassan

State machines

State machines—also known as finite automata—are foundational models in computation theory (a branch of theoretical computer science). They provide a mathematical framework for modeling systems that process inputs and transition between a finite number of states based on those inputs.

Basic Concepts:-

- State: A condition or mode in which a system operates. At any moment, the machine is in exactly one state.
- Transition: A change from one state to another, triggered by an input symbol.
- Alphabet (Σ): A finite set of input symbols.
- Start State: The initial state before any input is processed.
- Accept (Final) States: States that indicate successful recognition of a pattern or language

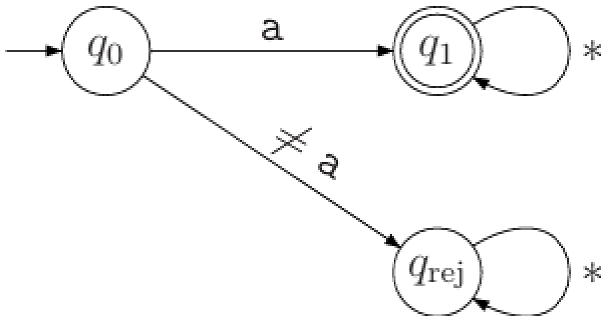
Types of finite automata:-

- Deterministic Finite Automaton (DFA)
- Nondeterministic Finite Automaton (NFA)

State machines

Here is a simple state machine (i.e., finite automaton) M that accepts all strings starting

with a.



Here * represents any possible character.

Notice key pieces of this machine: three states, q0 is the start state (arrow coming in),

q1 is the final state (double circle), transition arcs.

To run the machine, we start at the start state. On each input character, we follow the

corresponding arc. When we run out of input characters, we answer “yes” or “no”, depending

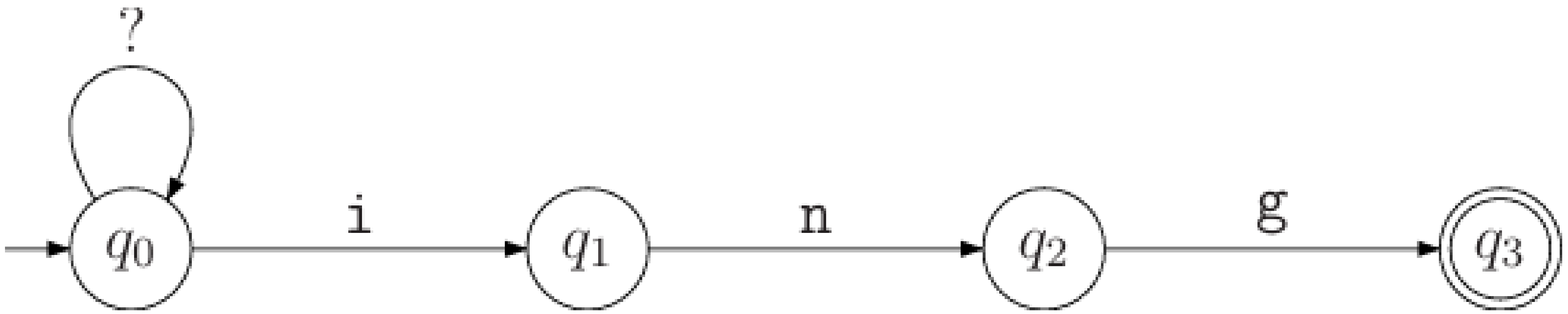
on whether we are in the final state.

The language of a machine M is the set of strings it accepts, written L(M). In this case

L(M) = {a,aa,ab,aaa,…….} .

Another example

- Here is a simple state machine (i.e., finite automaton) M that accepts all ASCII strings ending with ing.

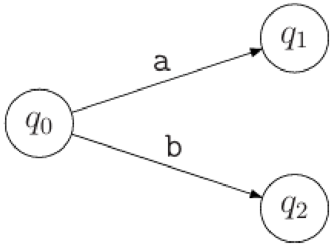


Notice key pieces of this machine: four states, q0 is the start state (arrow coming in), q3 is the final state (double circle), transition arcs.

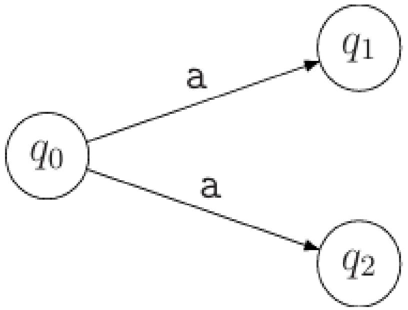
The language of a machine M is the set of strings it accepts, written L(M). In this case L(M) = {walking,flying,ing,…..} .

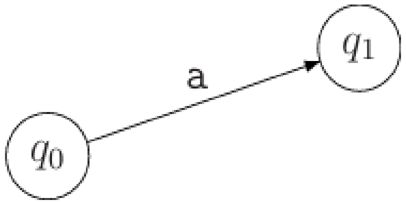
DFA- deterministic finite automata

- Each node in a deterministic machine has exactly one outgoing transition for each character in the alphabet. That is, if the alphabet is {a,b} , then all nodes need to look like



Both of the following are bad, where q1=/ q2 and the right-hand machine has no outgoing transition for the input character b.





DFA- deterministic finite automata

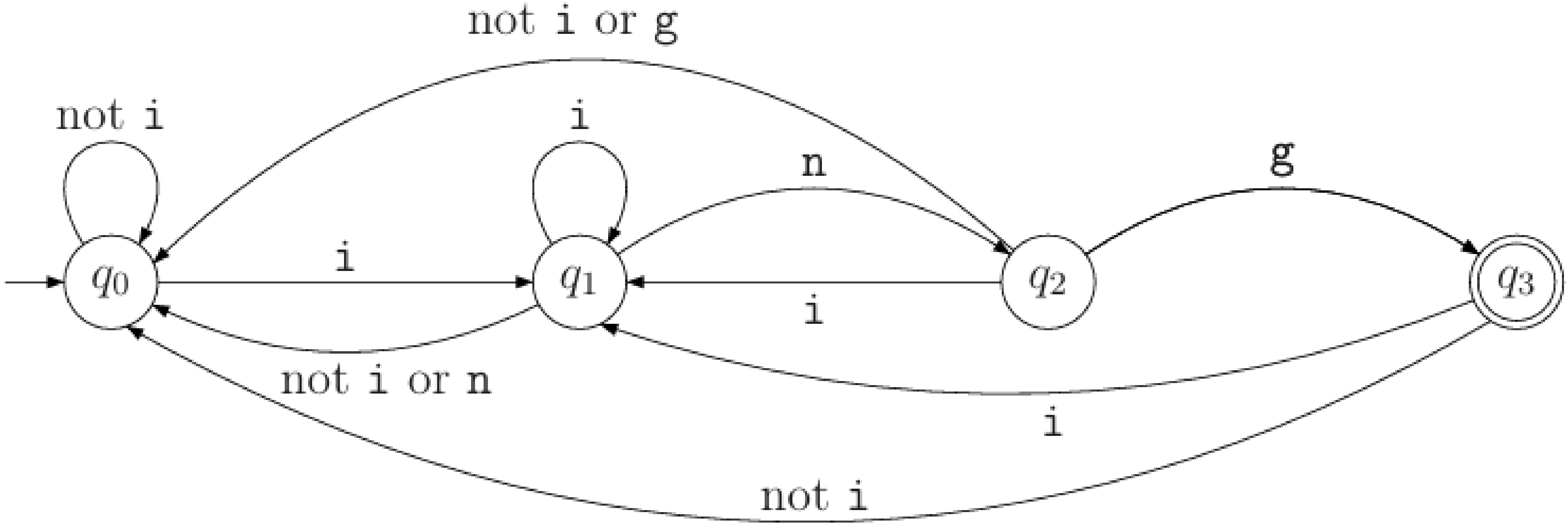
- **Definition**: For each state and input symbol, there is exactly one transition to a next state.

- **Formal Definition:**

A 5-tuple $(Q,\Sigma,\delta,q0,F)$, where:

- Q : Finite set of states
- Σ : Input alphabet
- $\delta:Q\times\Sigma\to Q$: Transition function
- $q0\in Q$: Start state
- $F\subseteq Q$: Set of accept states

-ing detector would be redrawn as:

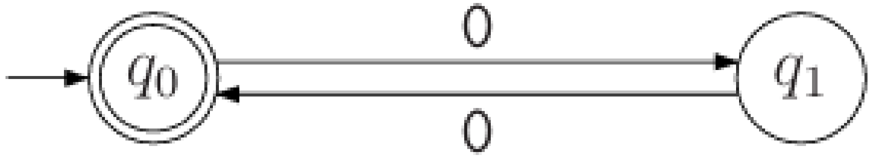


More examples of DFAs

Number of characters is even

Input: $\Sigma = \{0\}$.

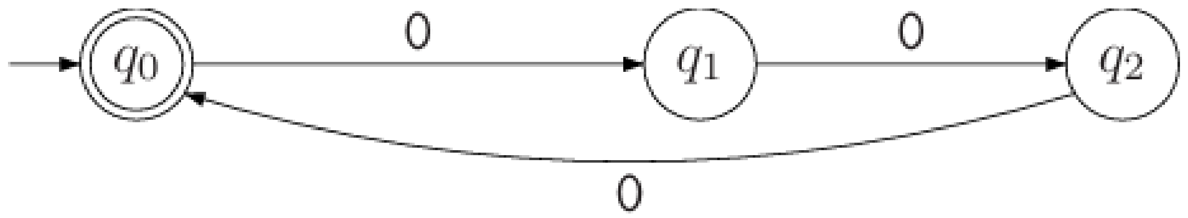
Accept: all strings in which the number of characters is even.



More examples of DFAs

Number of characters is divisible by 3

Input: $\Sigma = \{0\}$.
Accept: all strings in which the number of characters is divisible by 3.

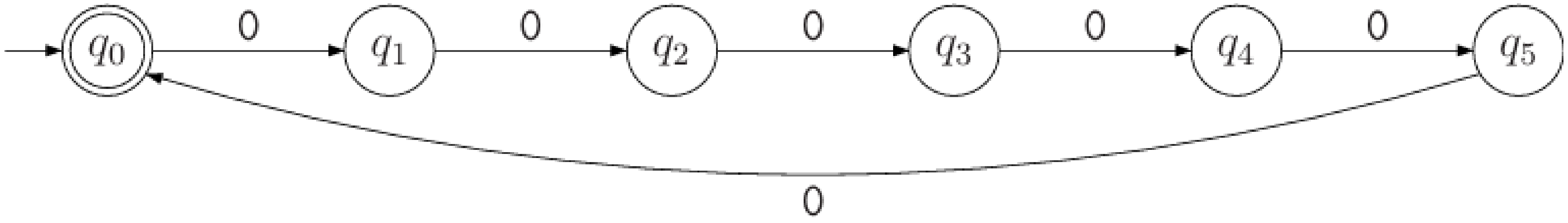


More examples of DFAs

Number of characters is divisible by 6

Input: $\Sigma = \{0\}$.

Accept: all strings in which the number of characters is divisible by 6.

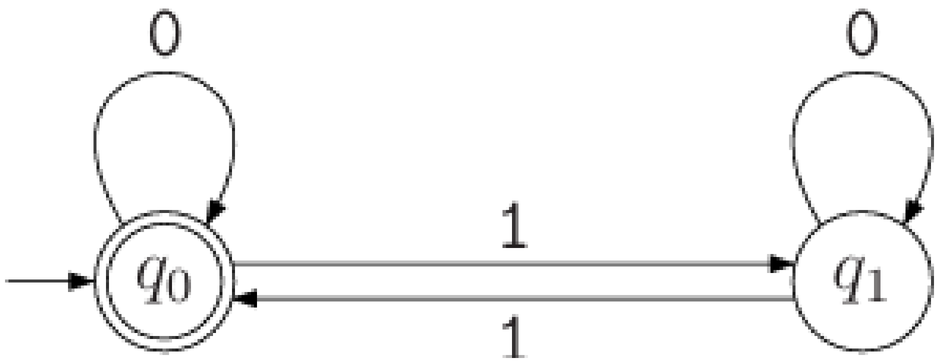


More examples of DFAs

Number of ones is even

Input is a string over $\Sigma = \{0, 1\}$.

Accept: all strings in which the number of ones is even.

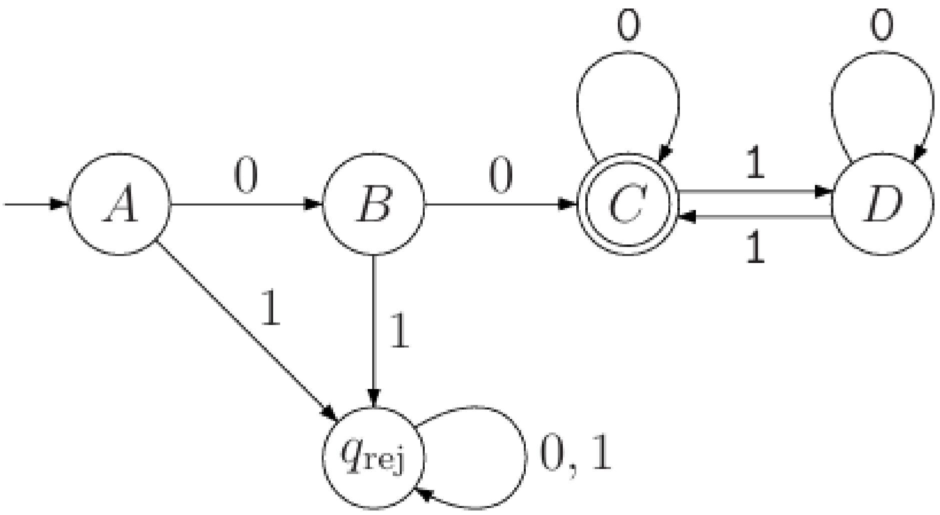


More examples of DFAs

Number of ones is even

The input is strings over $\Sigma = \{0, 1\}$.

Accept: all strings of the form $00w$, where w contains an even number of ones.



state table

- A state table (also called a transition table) in a Deterministic Finite Automaton (DFA) is a tabular representation of the transition function δ. It shows, for each state and each input symbol, the next state the DFA will move to.
- **Structure of a DFA State Table**

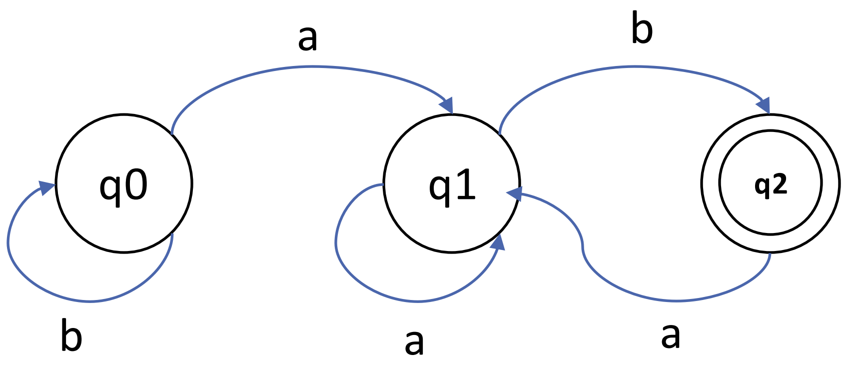
Current State	Input Symbol \`a\`	Input Symbol \`b\`	...
q_0	q_1	q_2	...
q_1	q_0	q_3	...
...	...	...	...

- Rows represent the current states.
- Columns represent the input symbols from the alphabet Σ.
- Cells contain the next state after reading the corresponding input symbol from the current state.

Example

DFA that accepts strings over {a, b} ending with "ab"

- States: q_0 (start), q_1, q_2 (accepting)
- Alphabet: {a, b}
- Transition function:



Current St	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
* q_2	q_1	q_0

Some acceptable strings { ab, aab, bab, aaab, bbaab}

Rejected strings { (empty string), a, b, aba, abb, ba}

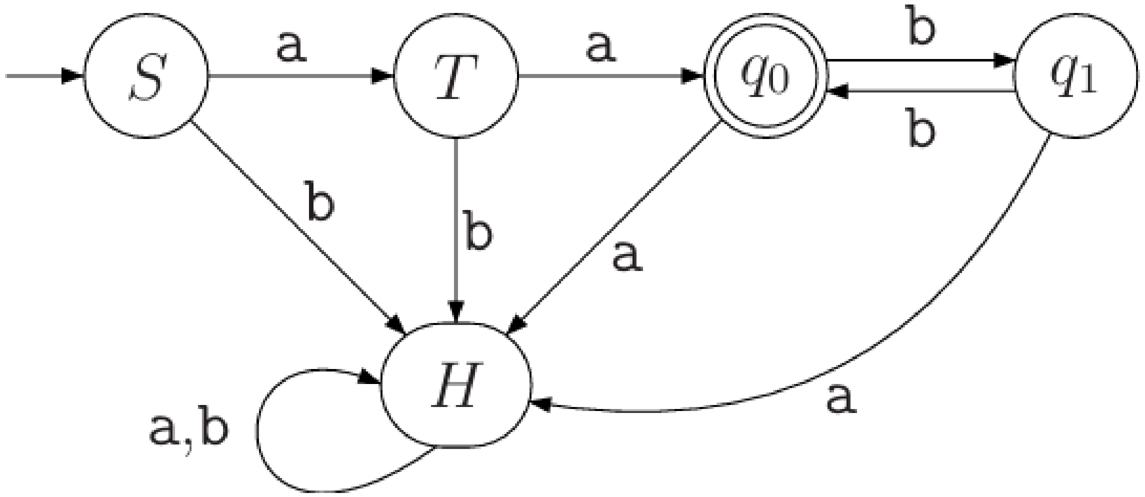
Languages that depend on k

$$\mathtt{aab}^{2i}$$

Consider the following language:

$$L_2 = \left\{ \mathtt{aab}^n \;\middle|\; n \text{ is a multiple of } 2 \right\}.$$

Its finite automata is



Continue

This automata formally is the tuple $(Q, \Sigma, \delta, S, F)$.

1. $Q = \{S, T, H, q_0, q_1\}$ - states.
2. $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ - alphabet.
3. $\delta : Q \times \Sigma \rightarrow Q$, described in the table.

	a	b
S	T	H
T	q_0	H
H	H	H
q_0	H	q_1
q_1	H	q_0

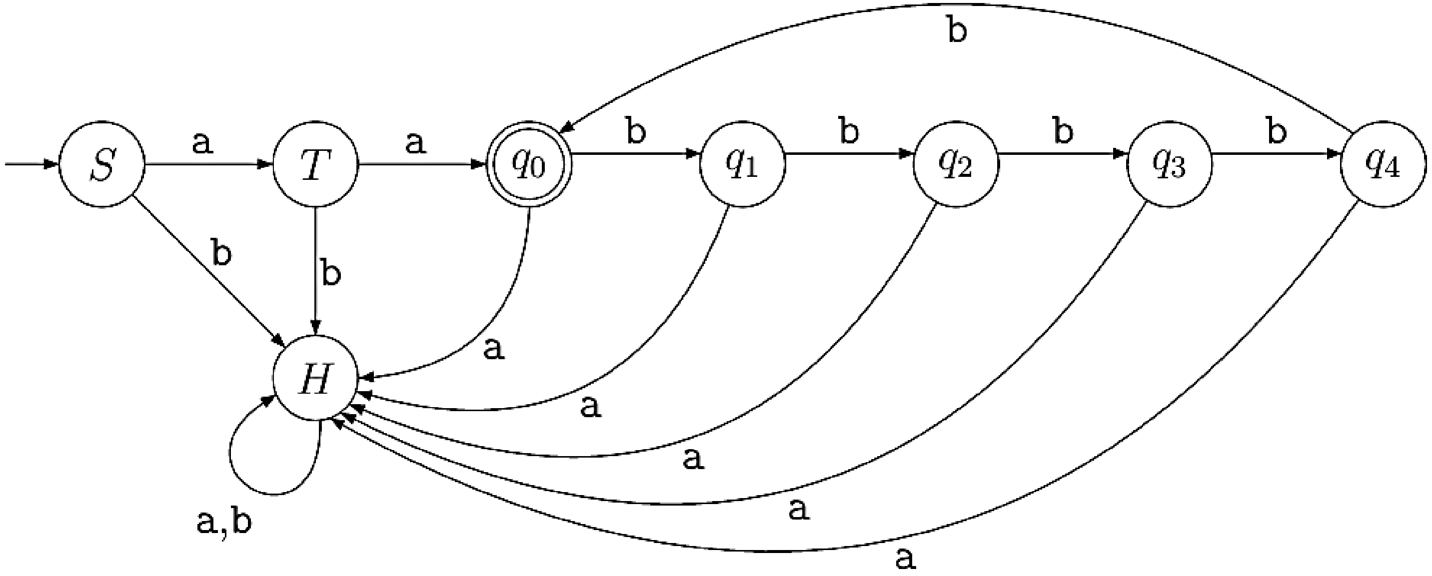
Languages that depend on k

$$\mathtt{aab}^{5i}$$

Consider the following language:

$$L_5 = \left\{ \mathtt{aab}^n \,\middle|\, n \text{ is a multiple of } 5 \right\}.$$

Its finite automata is



Continue

1. $Q = \{S, T, H, q_0, q_1, q_2, q_3, q_4\}$ - states.
2. $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ - alphabet.
3. $\delta : Q \times \Sigma \rightarrow Q$ - see table.
4. S is the start state.
5. $F = \{q_0\}$ is the set of accepting states.

δ	a	b
S	T	H
T	q_0	H
H	H	H
q_0	H	q_1
q_1	H	q_2
q_2	H	q_3
q_3	H	q_4
q_4	H	q_0

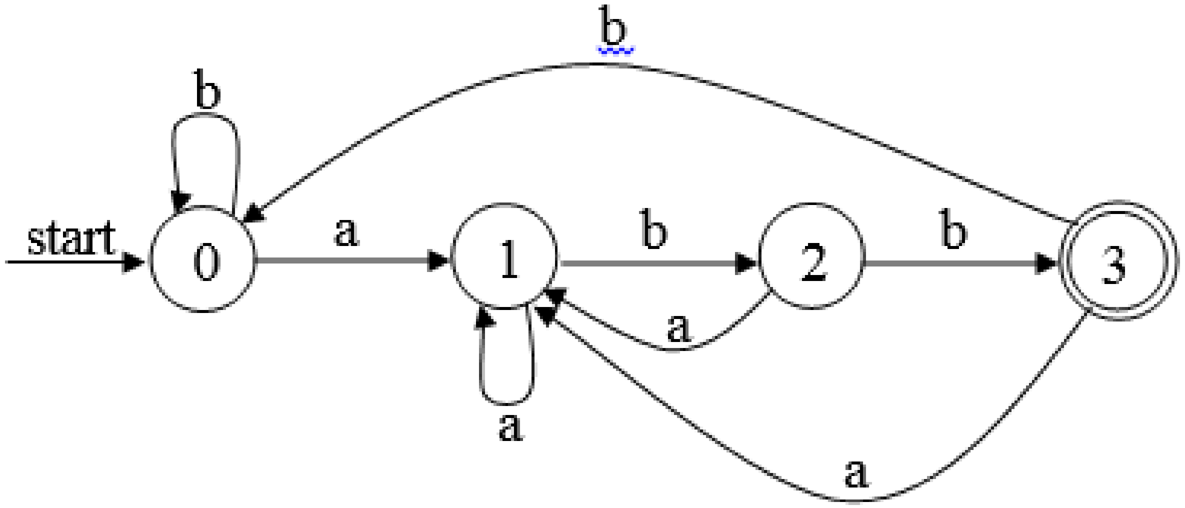
Class work

Make a DFA that recognizes the language (a|b)*abb

∑={a,b}

solution

DFA:



Transition table:

State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

Thank you