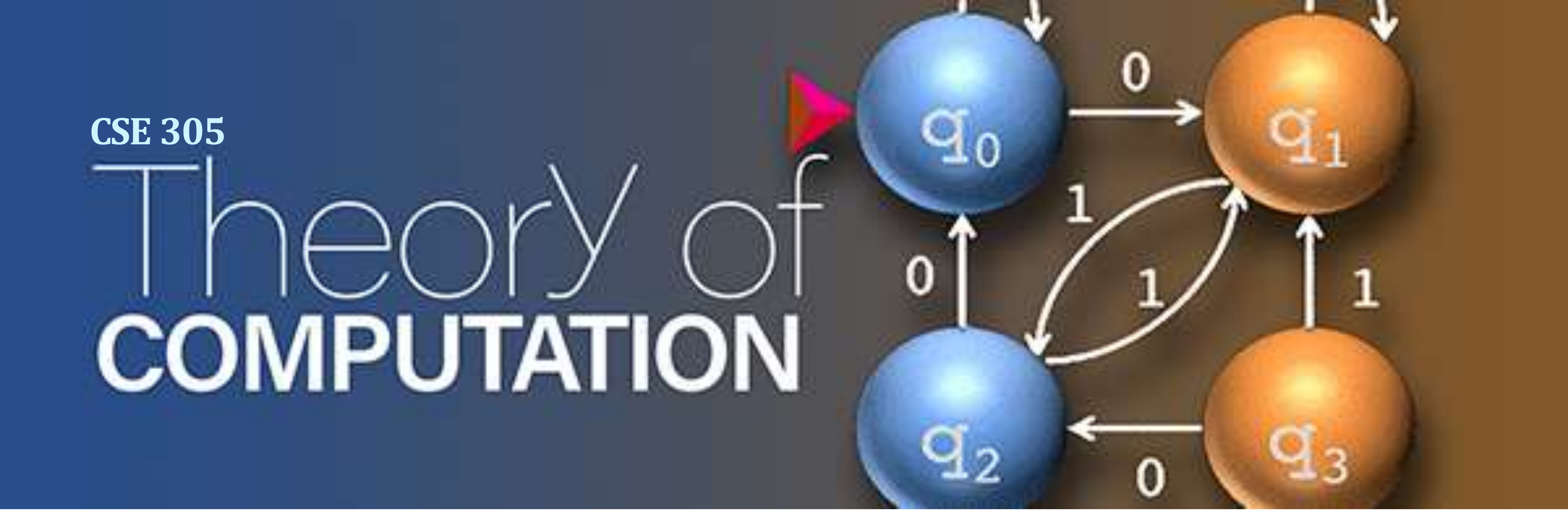


Lecture 16
Finite-State Automata (3)

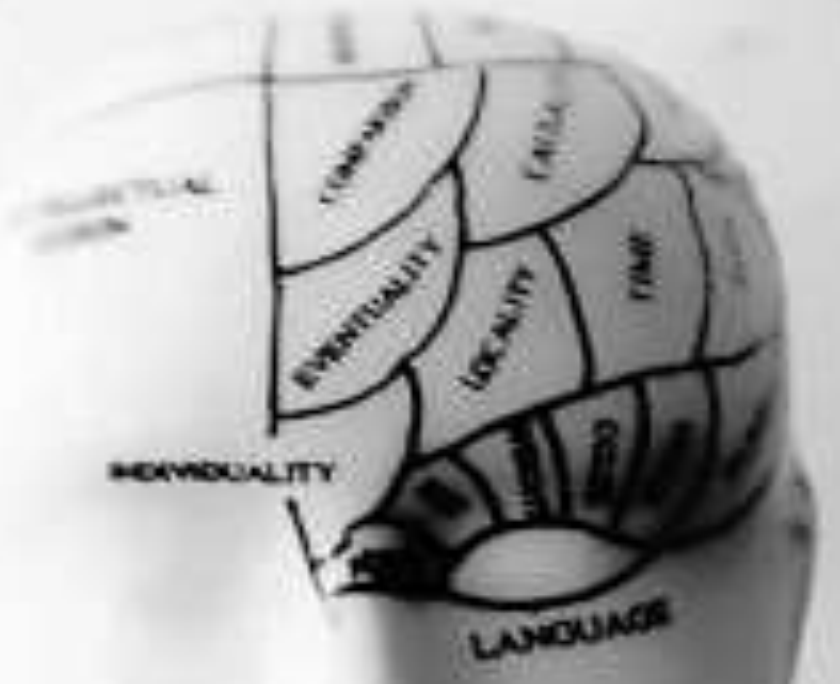
Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

www.mijanrahman.com

Contents

Finite State Automata



Properties of Transition Function

- A transition function is defined on every state for every input symbol.
- Transition Function (δ) is defined as $\delta = Q \times \Sigma \rightarrow Q$.

  Where,

  Q is set of all states.

  Σ is set of input symbols.

Properties of transition functions:

- **Property 1:** $\delta(q, a) = q$. It means the state of a system can be changed by an input symbol, *a*.
- **Property 2:** For all strings w and input symbol a,

  $$\delta(q, aw) = \delta(\delta(q, a), w)$$

  $$\delta(q, wa) = \delta(\delta(q, w), a)$$

- It means the state after the automaton consumes or reads the first symbol of a string *aw* and the state after the automaton consumes a prefix of the string *wa*.

Extended Transition Function

- **The extended transition function δ* of an automaton M tells us what state M ends up in after processing an entire string of characters.**
- In fact, the definition of δ* is what tells us what we mean when we say "process a string".
- **δ* is similar to δ but different in important ways:**
  1. δ* inputs entire strings, while δ inputs single characters.
  2. Each automaton has its own definition of δ; the definition of δ* is the same for every automaton (although it depends on δ).

- **The definition of δ* is different for different kinds of automata (DFA, NFA, etc).**

Extended Transition Function

- **Extended transition function for DFA:**

  - Intuitively, when a DFA processes the empty string, it doesn't do anything: if it started in state q, then it stays in state *q*.
  - To process the string ***xa***, the DFA would first process the substring ***x***, and then take one more step with the character aa (using the automaton's single step transition function).

- This intuition leads to the following definition:

Definition: extended transition function for DFA

Given an DFA M, we define the **extended transition function** $\hat{\delta} : Q \times \Sigma^* \to Q$ inductively by $\hat{\delta}(q, \varepsilon) = q$, and $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$.

Extended Transition Function

- **Extended transition function for DFA:**

- Thus, an extended transition function (δ^*) takes two arguments. The first argument is a state q and the second argument is a string w.
- It returns a state just like the transition function δ. It can be defined as the state in which the FA ends up, if it begins in state q and receives string x of input symbols.

- We define δ^* recursively as follows:
  1. For any $q \in Q$, $\delta^*(q, \epsilon) = q$
  2. For any $q \in Q$ and $a \in \Sigma$,
  3. $\delta^*(q, ya) = \delta(\delta^*(q,y), a)$

Language of a DFA

- **A DFA defines a language.** The set of all strings that result in a sequence of state transition from start state to an accepting state is the language defined by the DFA.
- **The language is denoted as L(M) for a DFA M = (Q, Σ, δ, F q_0), and is defined by:**

  L(M) = {w:δ*(q_0, w) is in F}.

- Here δ* is the extended transition function. The language represented by a DFA is regular.
- In order to accept a language L, the FA has to accept all the strings in L and reject all the strings in L'(compliment of a language, i.e. strings not in the language).

Minimization of DFA

- **Minimization of DFA means reducing the number of states from given FA. Thus, we get the FSM(finite state machine) with redundant states after minimizing the FSM.**

- **Construct a minimum state automata equivalent to given automata:** We have to follow the various steps to minimize the DFA. These are as follows:

  - **Step 1:** Remove all the states that are unreachable from the initial state via any set of the transition of DFA.
  - **Step 2:** Draw the transition table for all pair of states.
  - **Step 3:** Now split the transition table into two tables T1 and T2. T1 contains all final states, and T2 contains non-final states.

Minimization of DFA

- **Step 4:** Find similar rows from T1 such that:

  1. $\delta\ (q, a) = p$
  2. $\delta\ (r, a) = p$

  That means, find the two states which have the same value of a and b and remove one of them.

- **Step 5:** Repeat step 3 until we find no similar rows available in the transition table T1.
- **Step 6:** Repeat step 3 and step 4 for table T2 also.
- **Step 7:** Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

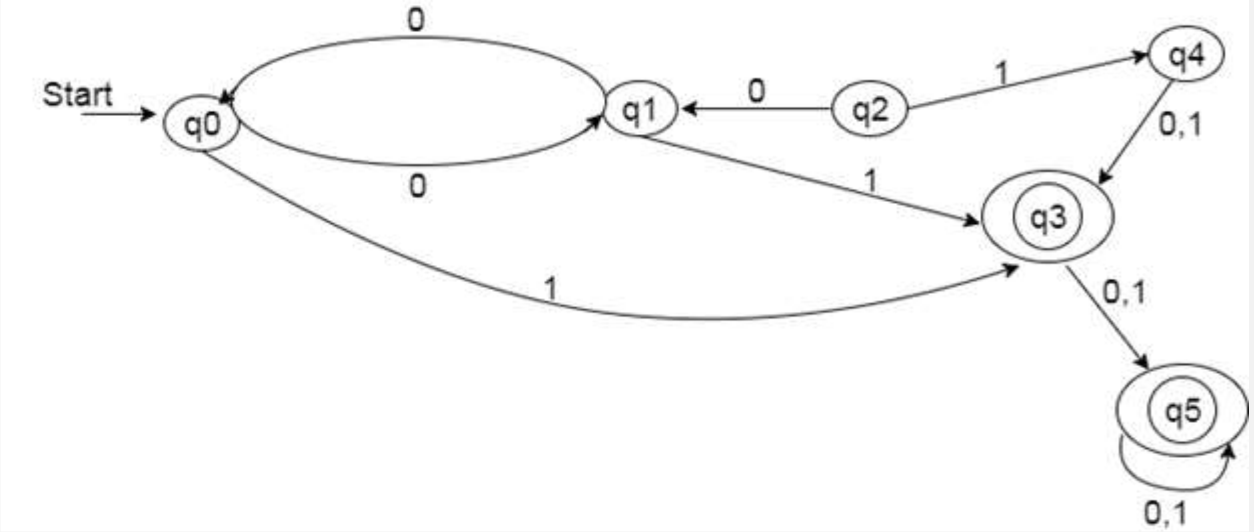
Minimization of DFA

- **Example-1: Consider a DFA given in the following figure. Construct a minimum state automata equivalent to given automata.**

- **Step 1:** In the given DFA, q2 and q4 are the unreachable states so remove them.
- **Step 2:** Draw the transition table for the rest of the states.

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q5	q5
*q5	q5	q5

- **The given DFA:**



Minimization of DFA

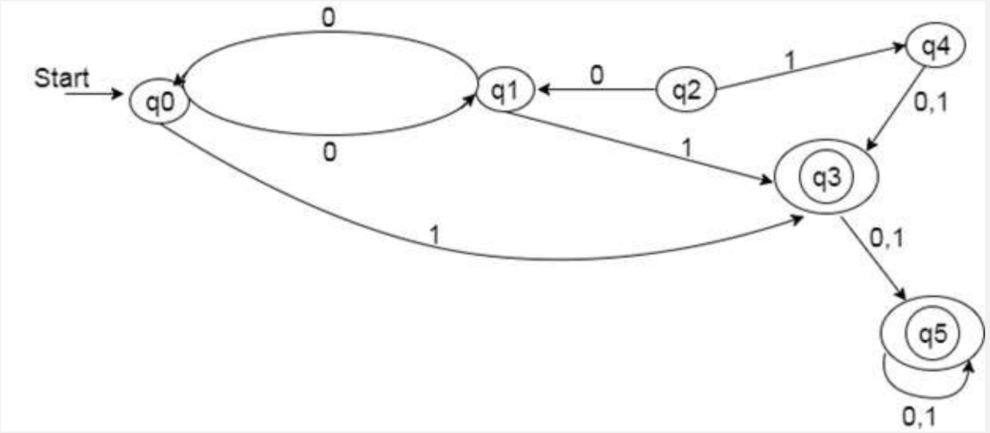
- **Step 3:** Now divide rows of transition table into two sets as, T1 and T2:
  1. One set (T1) contains those rows, which start from non-final states:

State	0	1
q0	q1	q3
q1	q0	q3

  2. Another set (T2) contains those rows, which starts from final states.

State	0	1
q3	q5	q5
q5	q5	q5

- **The given DFA:**



Minimization of DFA

- **Step 4:** Set 1 (T1) has no similar rows so set 1 will be the same.
- **Step 5:** In set 2, row 1 and row 2 are similar since q3 and q5 transit to the same state on 0 and 1. So skip q5 and then replace q5 by q3 in the rest.

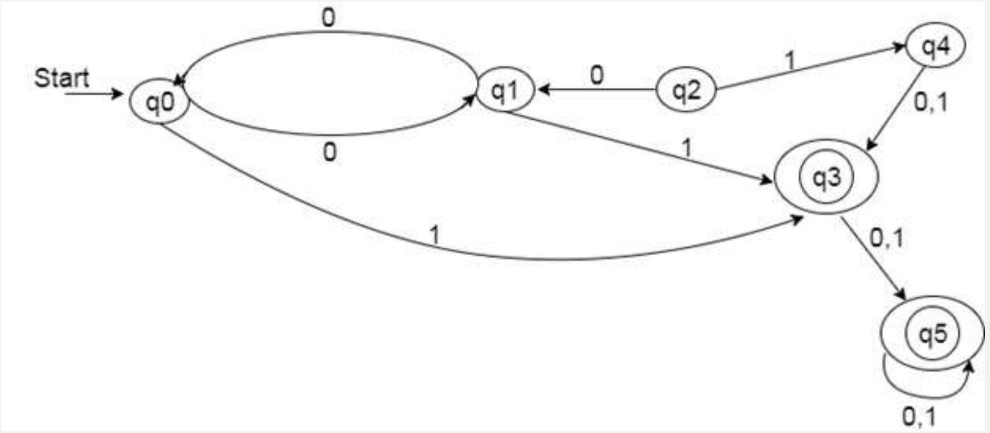
State	0	1
q3	q3	q3

- **Step 6:** Now combine set 1 and set 2 as:

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q3	q3

- **Now it is the transition table of minimized DFA.**

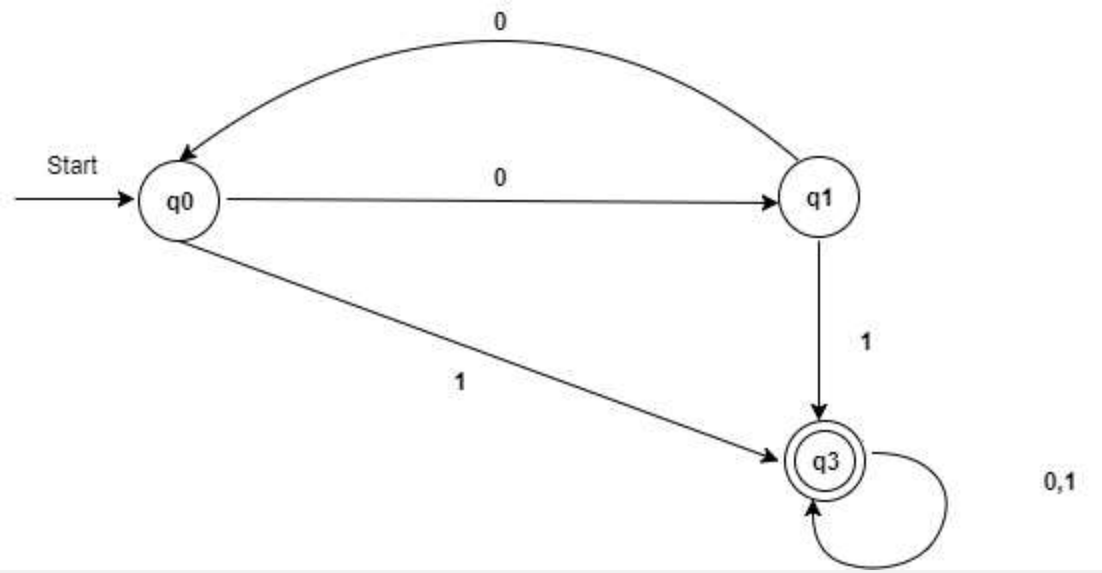
- **The given DFA:**



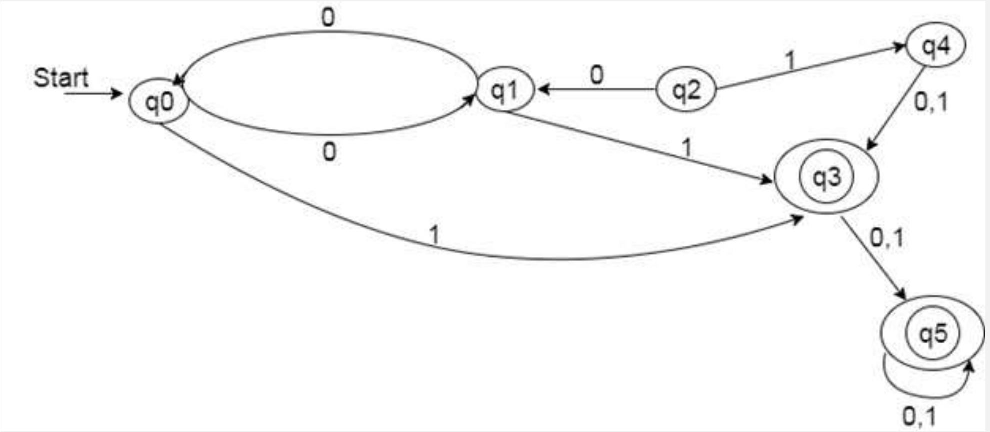
Minimization of DFA

- **Step 7: The following is the transition diagram of minimized DFA.**

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q3	q3



- **The given DFA:**

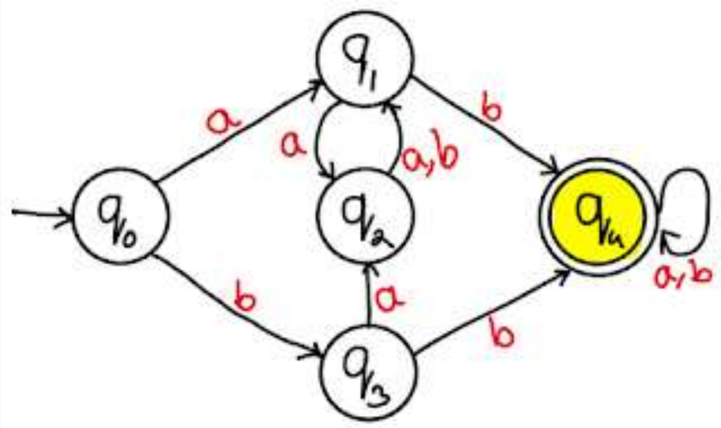


Minimization of DFA

- **Example-2: Consider a DFA given in the following figure. Construct a minimum state automata equivalent to given automata.**
- Transition table for the given automata:

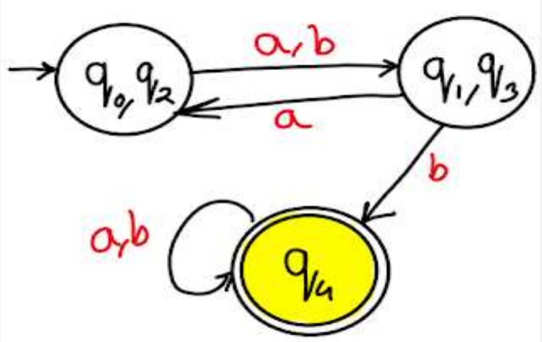
State	Input = a	Input = b
->q0 Initial state	q1	q3
q1	q2	q4
q2	q1	q1
q3	q2	q4
q4 Final state	q4	q4

- **The given DFA:**



- After minimization, we have three states of the minimized DFA:
  1. {q4},
  2. {q0,q2},
  3. {q1,q3}

- **The minimized DFA:**



Minimization of DFA

- **Hints:** Split final states and non final states.

  A0 = {q4}

  A1 = {q0,q1,q2,q3}

  A0 cannot be partition further.

  In A1,

  q0 is 1 equivalent to q2 for input a, but not equivalent to q1 and q3.

  q1 is 1 equivalent to q3 for input a and b, but not to q0 and q2.

  So, A1 can be partitioned as,

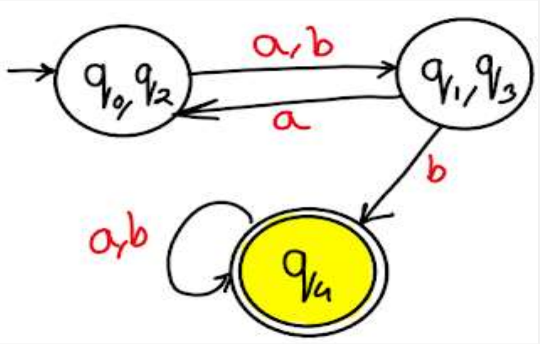
  B0 = {q0, q2}

  B1 = {q1, q3}

- Thus, the set of states: {q4}, {q0,q2}, {q1,q3}
- After minimization, we have three states of the minimized DFA:
  1. {q4},
  2. {q0,q2},
  3. {q1,q3}

State	Input = a	Input = b
->{q0,q2} Initial state	{q1,q3}	{q1,q3}
{q1,q3}	{q0,q2}	{q4}
{q4} Final state	{q4}	{q4}

- **The minimized DFA:**



?

THE END