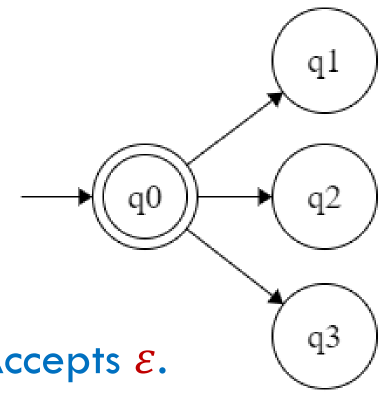
Finite automata and language

- If M is a **FSM** and A is a **language**, The followings are same:
- The language that machine M accepts is A.
- A is the language of machine M.
- M recognizes A. → We prefer this.
- M accepts A. → "accept" is used for both string and language.

Chapter 1

Finite automata and language

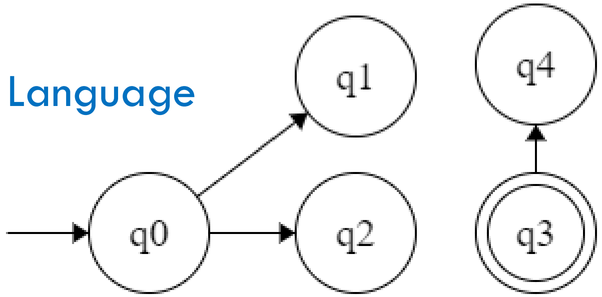
- Empty string:
  - Called epsilon (ε)
  - The start state is a final state.



Accepts ε.

- Empty Language:
  - $\phi = \{\,\}$
  - The accept state is not reachable.

Empty Language



Accept state, q3, is not reachable.

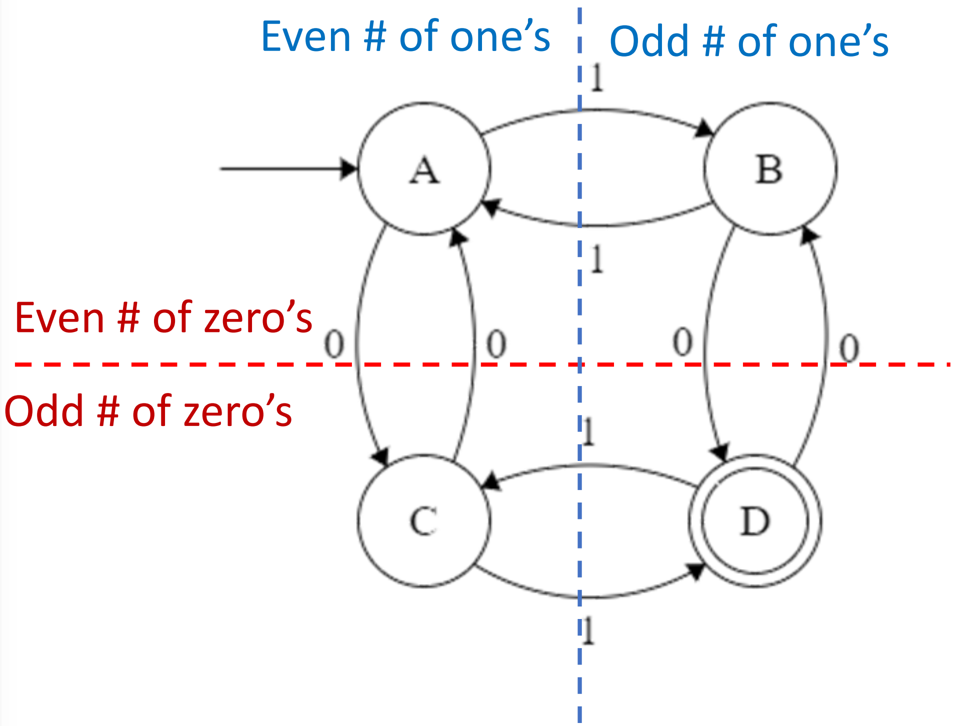
- Note: $\{\varepsilon\} \neq \phi$ and $\varepsilon \neq \phi$

- If a machine **accepts** "no string", then it **recognizes** the "empty language".

Chapter 1

language of machine

- Can you find any pattern for the string that machine M accepts?

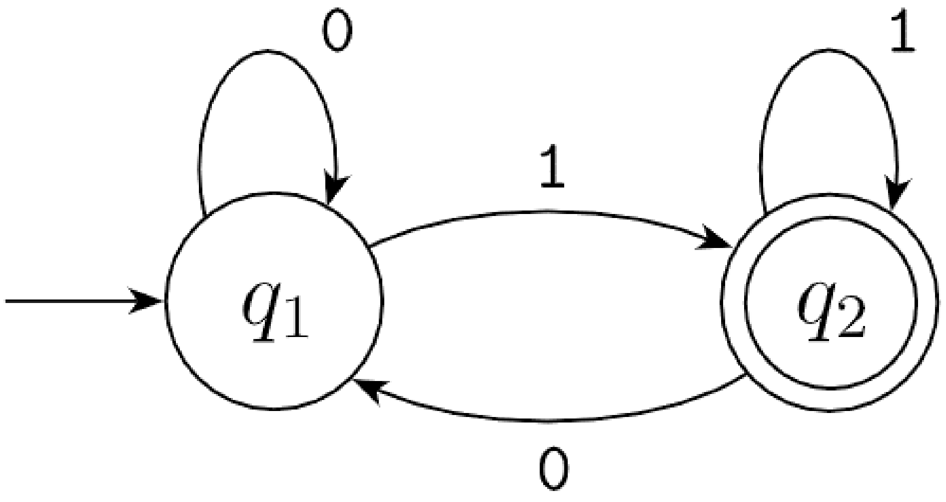


- The accepted strings:
  - 10, 1110,111110, …
  - 10, 1000, 100000,…
  - 01, 0001, 000001, …
  - 01, 0111, 011111, …

- $L(M_1)$ = {w| string w contains odd number of 1's or odd number of 0's.}

language of machine

- Example:

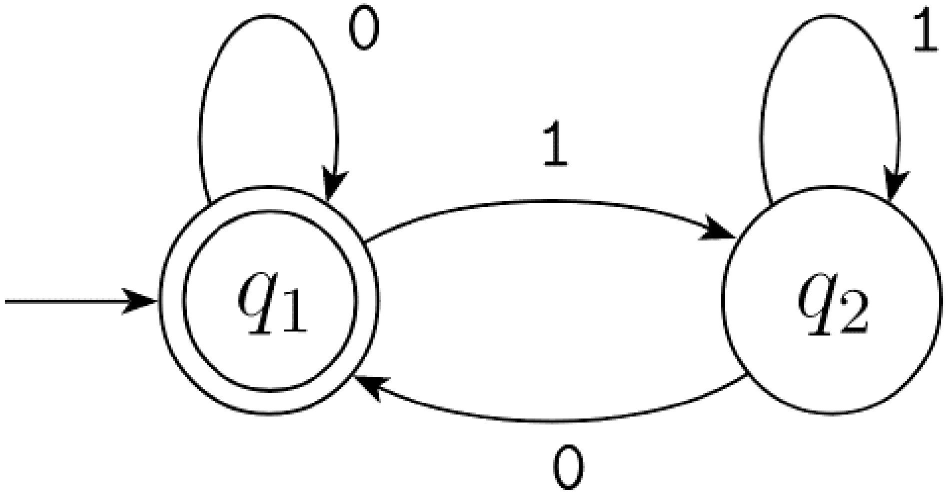


$\Sigma = \{0,1\}$
$Q = \{q_1, q_2\}$
$q_{start} = q_1$
$F = \{q_2\}$

δ	0	1
$\rightarrow q_1$	q_1	q_2
q_2	q_1	q_2

- What is the language of this machine?
- $L(M_2)=\{w \mid w$ ends in a 1$\}$

- language of machine
- Example:



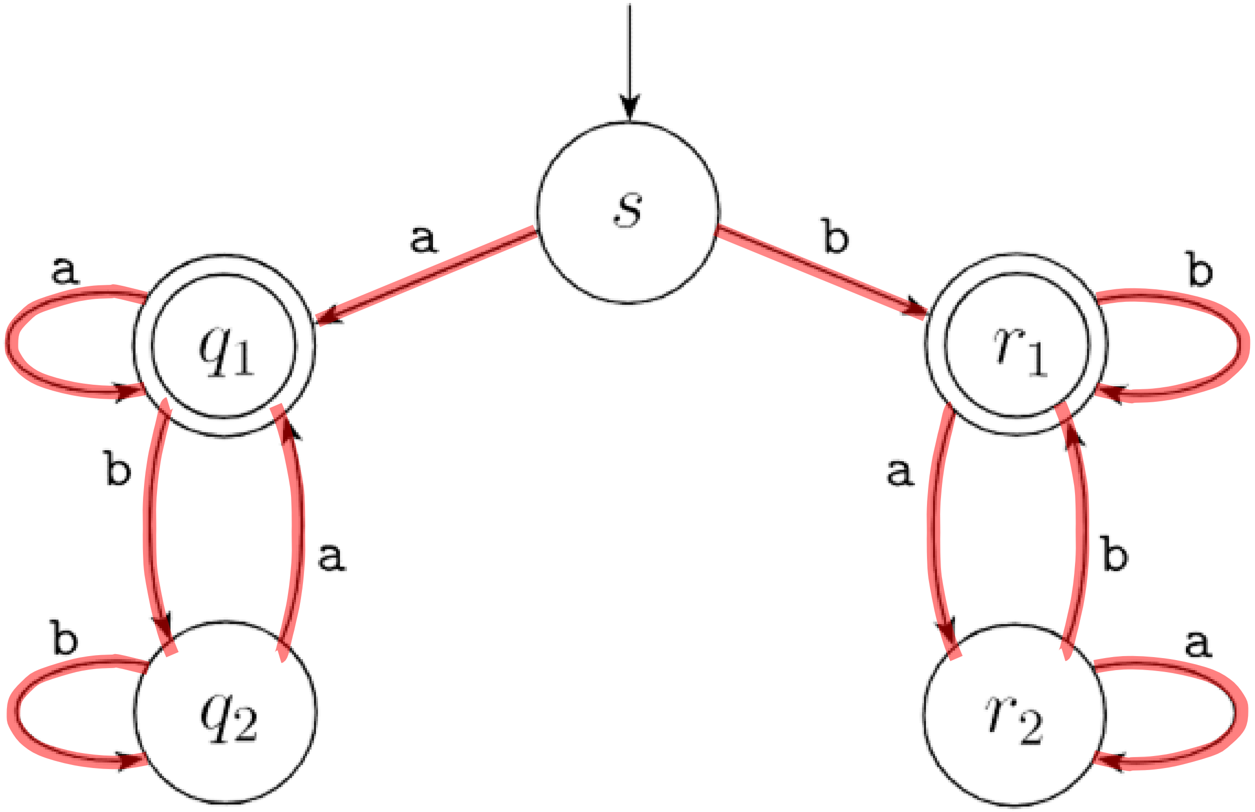
$\Sigma = \{0,1\}$
$Q = \{q_1, q_2\}$
$q_{start} = q_1$
$F = \{q_1\}$

δ	0	1
$\rightarrow \underline{q_1}$	q_1	q_2
q_2	q_1	q_2

- $L(M_3)$={w|w is the empty string ε or ends in a 0}

language of machine

- **Example:**



$\Sigma = \{a, b\}$
$Q = \{q_1, q_2, r_1, r_2\}$
$q_{start} = s$
$F = \{q_1, r_1\}$

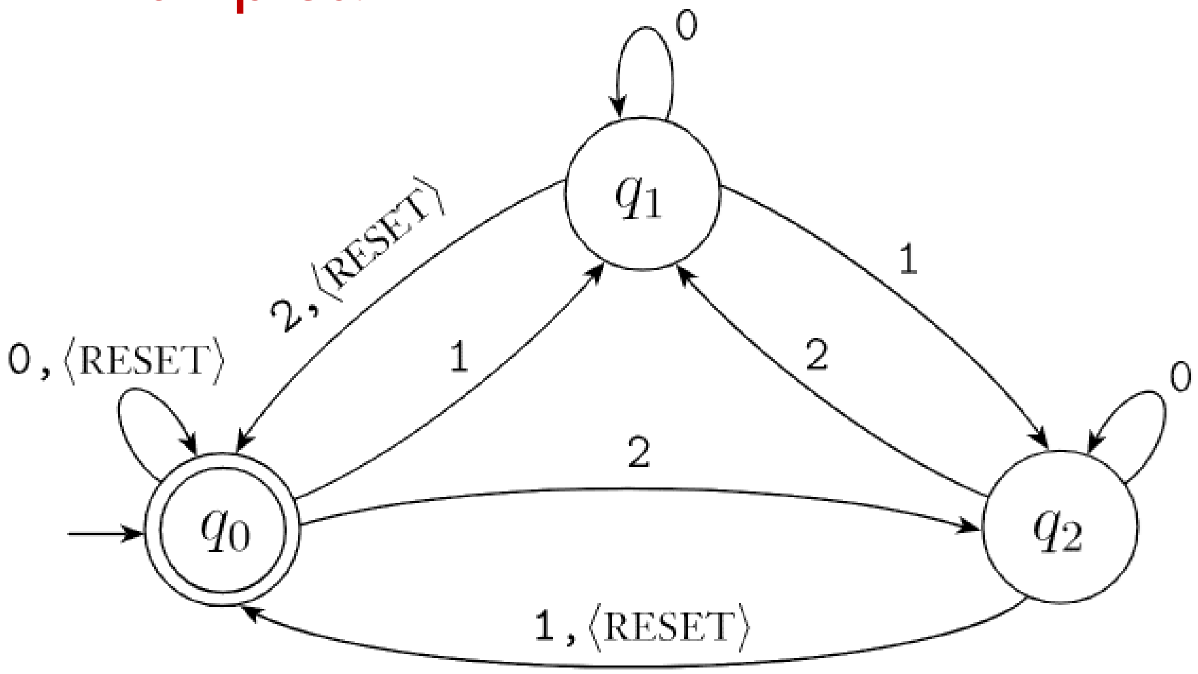
δ	a	b
$\rightarrow s$	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1

- $L(M_4)$ = {w|w start and end with **a** or that start and end with **b**.}
- $L(M_4)$ = {start and end with the same symbol.}

Chapter 1

language of machine

- Examples:



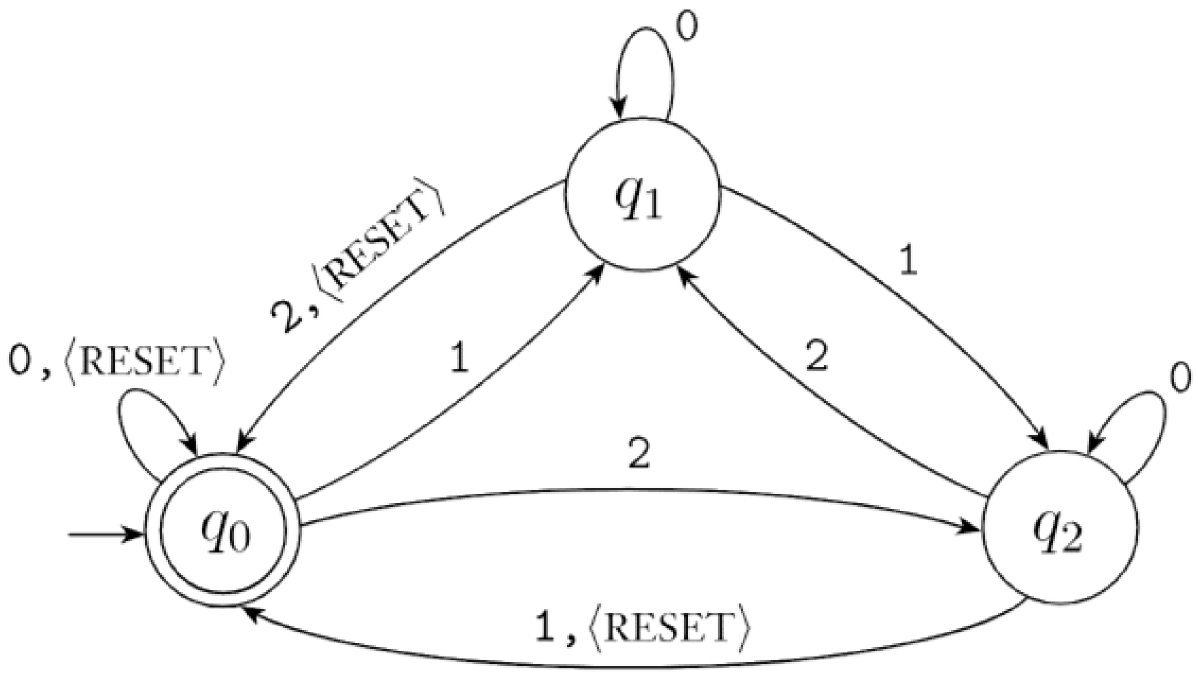
$\Sigma = \{\langle RESET\rangle, 0,1,2\}$
$Q = \{q_0, q_1, q_2\}$
$q_{start} = q_0$
$F = \{q_0\}$

δ	0	1	2	RESET
$\rightarrow q_0$	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_2	q_0	q_1	q_0

- The acceptable strings: <RESET>, 12, 21, 111, 1011, 1101, 1110,…
- $L(M_5)$ = {w|the sum of the symbols in w is 0 modulo 3,except ⟨RESET⟩ that resets the count to 0.}

language of machine

- **Example:**
  - $L(M_5)$ = {w|the sum of the symbols in w is 0 modulo 3,except that ⟨RESET⟩ resets the count to 0}.



- Does M_5 accept **10⟨RESET⟩22⟨RESET⟩012** ?

Designing finite automata

- Design is a creative process.
- cannot be reduced to a simple recipe or formula.
- However, you might find a particular approach helpful when designing various types of automata.
- “Reader as automaton” approach

Designing finite automata

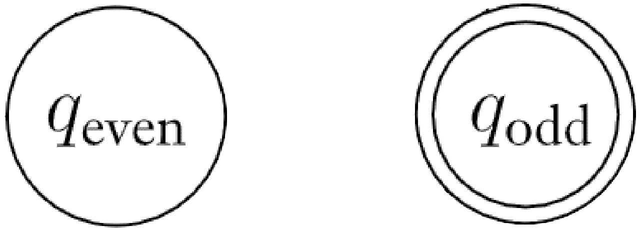
Example:

- construct a **finite automaton E$_1$** to recognize **language** A, where
  - A = {w| binary string w has odd number of 1s.}

- $E_1 = (Q, \Sigma, \delta, q_0, F)$ = ?

1. $\Sigma = \{0,1\}$
2. Q =?
   - even so far, and → q_{even}
   - odd so far. → q_{odd}
   - Q = $\{q_{even}, q_{odd}\}$

- $q_0 = q_{even}$
- $F = \{q_{odd}\}$



Designing finite automata

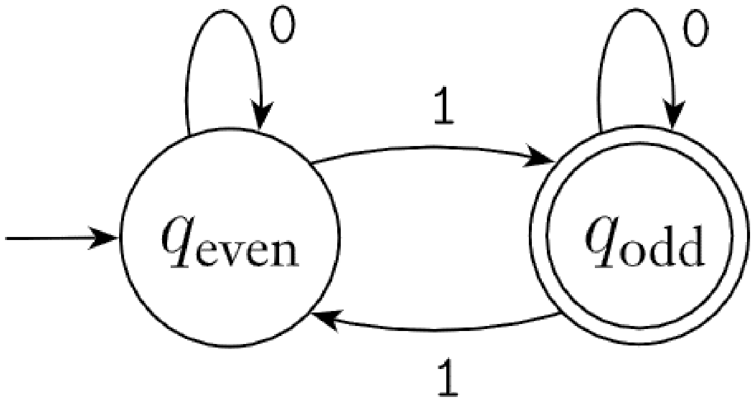
Example:

- construct a **finite automaton E$_1$** to recognize **language** A, where
  - A = {w| binary string w has odd number of 1s.}

- $E_1 = (Q, \Sigma, \delta, q_0, F)$ = ?

1. $\Sigma = \{0,1\}$
2. Q = $\{q_{even}, q_{odd}\}$
3. $q_0 = q_{even}$
4. $F = \{q_{odd}\}$
5. δ =?

δ	0	1
$\rightarrow q_{even}$	q_{even}	q_{odd}
$\underline{q_{odd}}$	q_{odd}	q_{even}



Designing finite automata

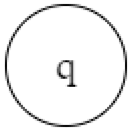
Example:

- construct a **finite automaton** $\mathbf{E_2}$ to recognize **language** A, where
  - A = {w| binary string w has a substring 001.}
  - The strings 0010,1001,001,and1111111001111are all in the language.

Designing finite automata

Design:

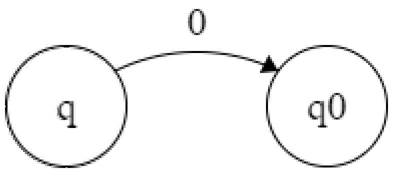
- We start with the minimum number of the states that we need.
- Q =?
  - haven't just seen any symbols of the pattern, → q



Designing finite automata

Design:

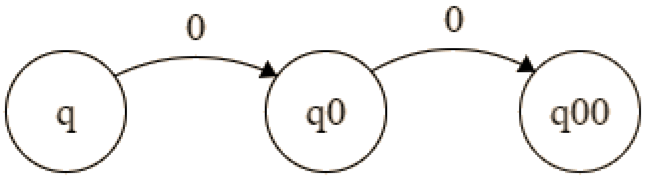
- We start with the minimum number of the states that we need.
- Q =?
  - haven't just seen any symbols of the pattern, → q
  - have just seen a 0, → q_0



Designing finite automata

Design:

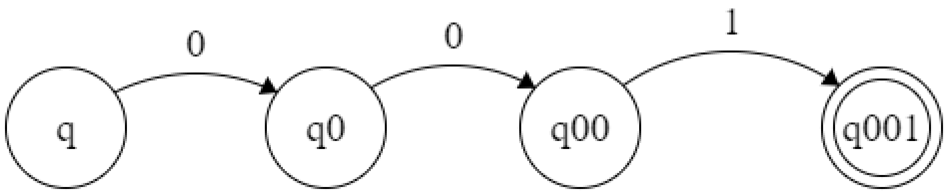
- We start with the minimum number of the states that we need.
- Q =?
  - haven't just seen any symbols of the pattern, → q
  - have just seen a 0, → q_0
  - have just seen 00 → q_{00}



Designing finite automata

Design:

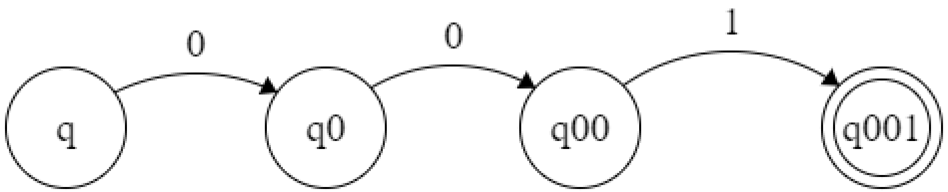
- We start with the minimum number of the states that we need.
- Q =?
  - haven't just seen any symbols of the pattern, → q
  - have just seen a 0, → q_0
  - have just seen 00 → q_{00}
  - have seen the entire pattern 001. → q_{001}



Designing finite automata

Design:

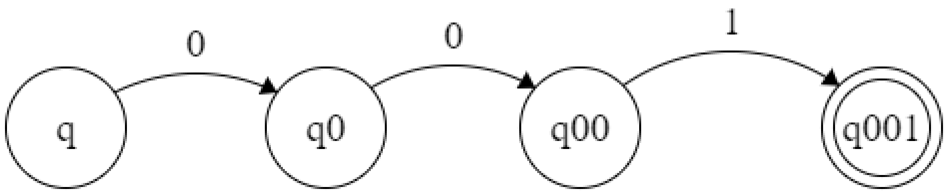
- We start with the minimum number of the states that we need.
- Q =?
  - haven't just seen any symbols of the pattern, → q
  - have just seen a 0, → q_0
  - have just seen 00 → q_{00}
  - have seen the entire pattern 001. → q_{001}
- **Complete the transitions of the automata :**



Designing finite automata

Design:

- We start with the minimum number of the states that we need.
- Q's : Done
- **Complete the transitions of automata :**

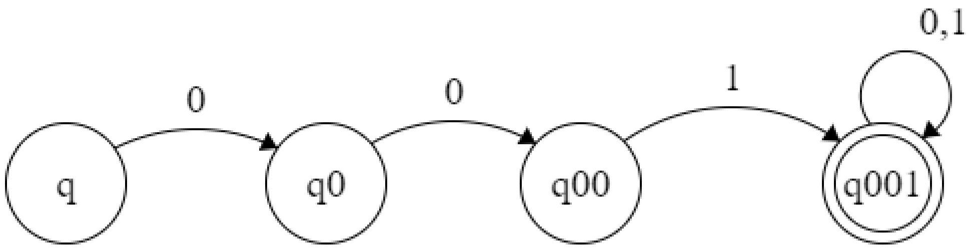


Chapter 1

Designing finite automata

Design:

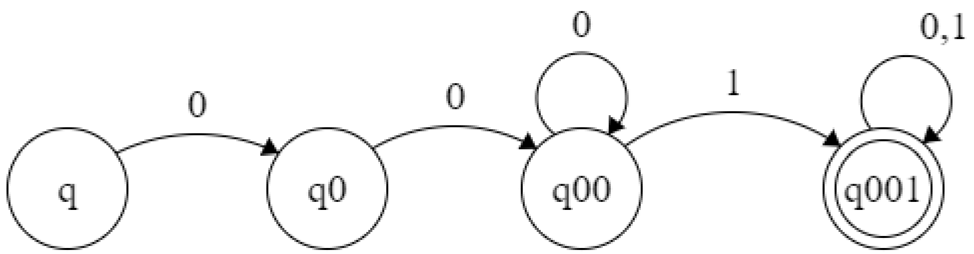
- We start with the minimum number of the states that we need.
- Q's : Done
- **Complete the transitions of the automata :**
  - **q001**



Designing finite automata

Design:

- We start with the minimum number of the states that we need.
- Q's : Done
- **Complete the transitions of the automata :**
  - **q001**
  - **q00**

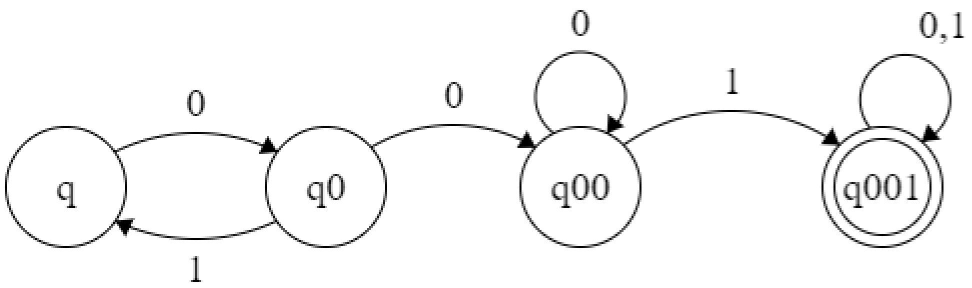


Chapter 1

Designing finite automata

Design:

- We start with the minimum number of the states that we need.
- Q's : Done
- **Complete the transitions of the automata :**
  - **q001**
  - **q00**
  - **q0**

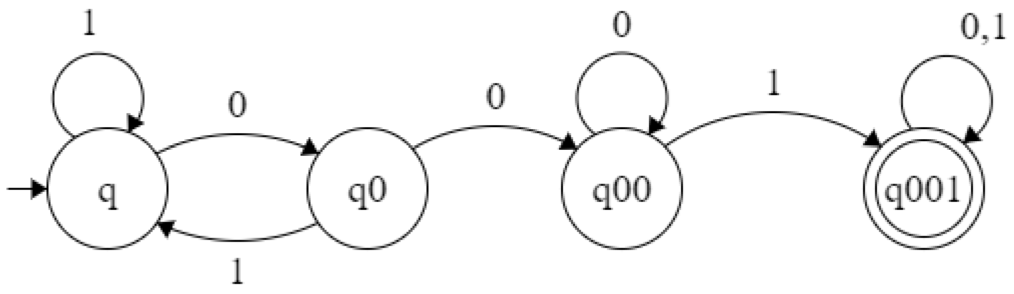


Chapter 1

Designing finite automata

Design:

- We start with the minimum number of the states that we need.
- Q's : Done
- **Complete the transitions of the automata :**
  - **q001**
  - **q00**
  - **q0**
  - **q**



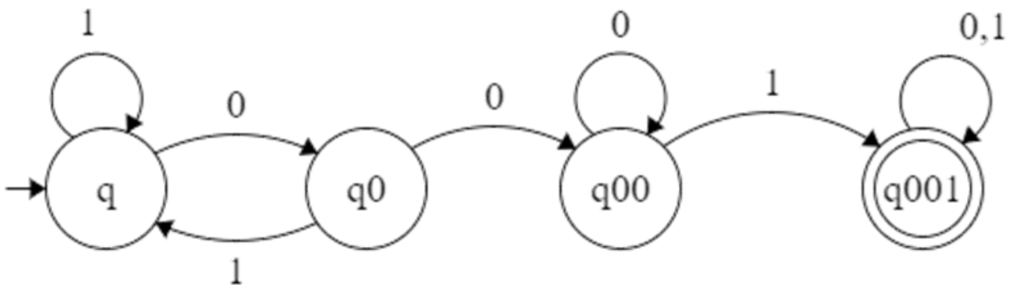
Designing finite automata

Example:

- construct a **finite automaton E₂** to recognize **language** A, where
  - A = {w| the binary string w has a substring 001.}

- $E_2 = (Q, \Sigma, \delta, q_0, F)$ = ?
  - $\Sigma = \{0,1\}$
  - Q = $\{q, q_0, q_{00}, q_{001}\}$
  - $q_0 = q$
  - $F = \{q_{001}\}$

δ	0	1
$\rightarrow q$	q_0	q
q_0	q_{00}	q
q_{00}	q_{00}	q_{001}
q_{001}	q_{001}	q_{001}

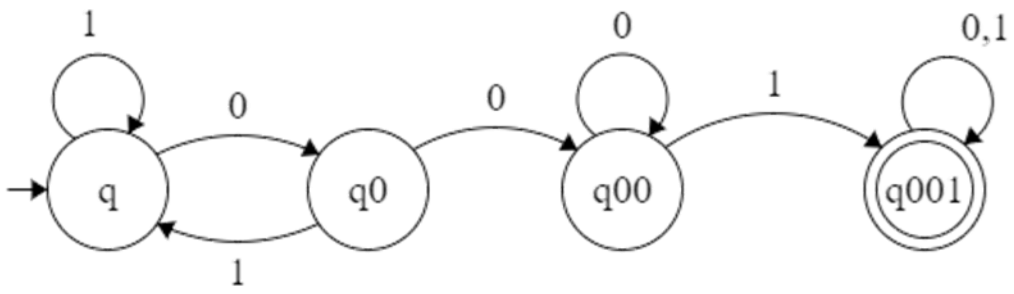


Designing finite automata

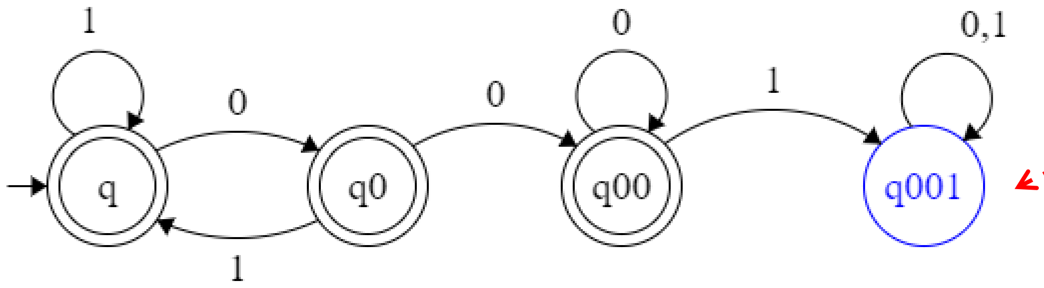
Example 2:

- construct a **finite automaton** $\mathbf{E_3}$ to recognize **language** A, where
  - A = {w| the binary string w does not contain substring 001 in it.}

- It is easier to construct the machine $\mathbf{E_2}$ that accepts any string that contains a substring 001.
  - (previous example)

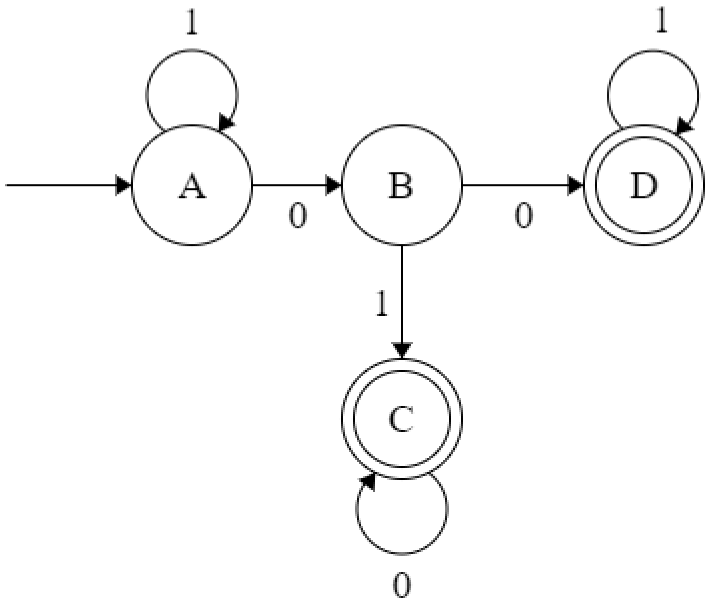


- Then, swap the accepting states with non-accepting states and vise versa.
  - See $\mathbf{E_3}$ Below.

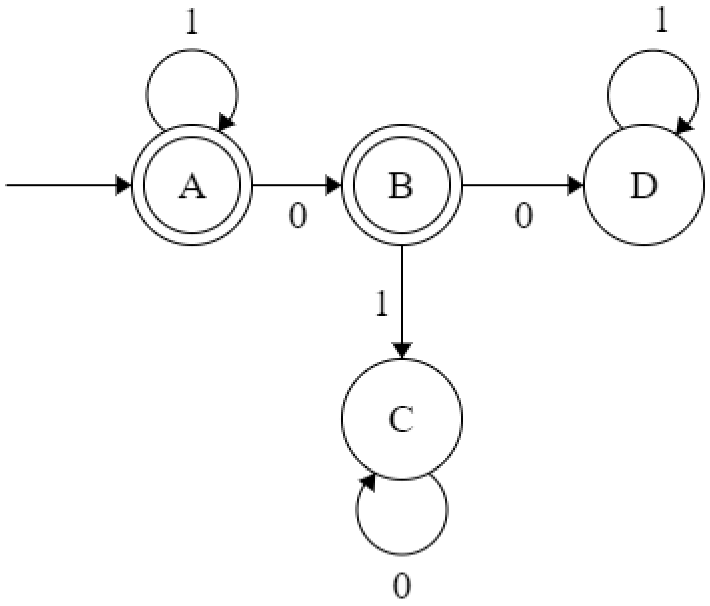


Complementing a language

Let $\boldsymbol{M} = (\mathbf{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \mathbf{q_0}, \mathbf{F})$ be a DFA that accepts a language L. Then a DFA that accepts the complement of L, i.e. (Σ* - L), can be obtained by **swapping** its **accepting states** with its **non-accepting states**, that is $M = (Q, \Sigma, \delta, q_0, \mathrm{Q} - F)$ is a DFA that **accepts** **(Σ* - L)**.



L over Σ



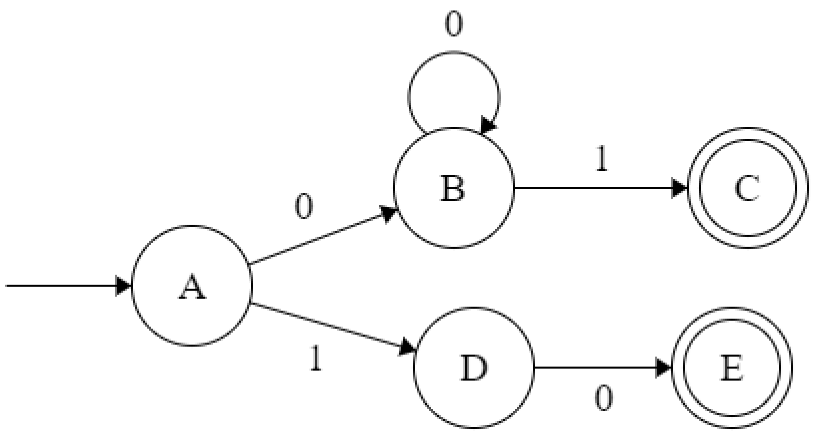
(Σ* - L) = over Σ

Designing finite automata

Question 1: What does this FSM recognize?

- Accepts Strings such as 10, 0^+1, …
- Recognizes A = {w| w is either **10** or a string of **at least one 0 followed by a single 1**.}

Question 2: what happens if w= 11 or 101



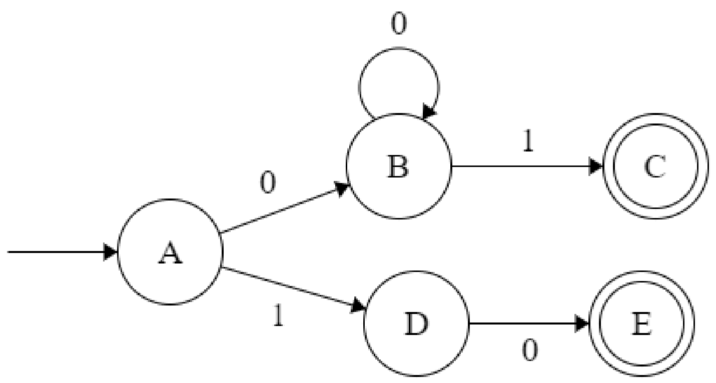
δ	0	1
$\rightarrow A$	B	D
B	B	C
C	?	?
D	E	?
E	?	?

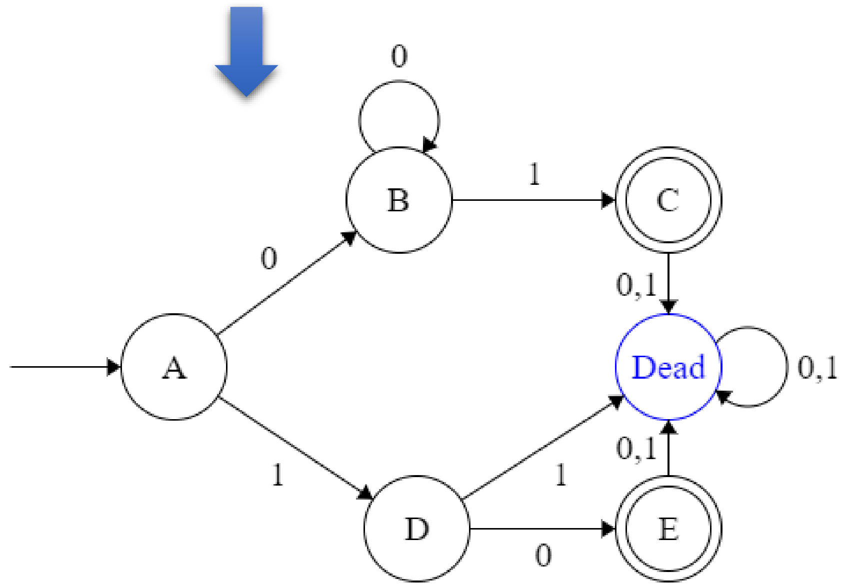
- δ, formally, should be defined. How?

Designing finite automata

Question 1: δ, formally, should be defined. How?

- By adding the a "dead state".





δ	0	1
$\rightarrow A$	B	D
B	B	C
$\underline{C}$	Dead	Dead
D	E	Dead
$\underline{E}$	Dead	Dead
Dead	Dead	Dead

Regular language

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let w=$w_1 w_2 \dots w_n$ be a string where each w_i is a member of the alphabet Σ.
- Then M accepts w if a sequence of states $r_0, r_1, \dots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \dots, n-1$, and
3. $r_n \in F$.

w	w_1	w_2	…	w_n	
State	r_0	r_1	…	r_{n-1}	r_n accept

- We say that M recognizes language L if L = {w|M accepts w}.
- A language is called a regular language if some finite automaton recognizes it.

The regular operations

- Let A and B be languages. We define the regular operations union, intersection, concatenation, and star as follows:
  - Union: $A \cup B \; = \; \{x | x \in A \; or \; x \in B\}$.
  - Intersection: $A \cap B \; = \; \{x | x \in A \; and \; x \in B\}$.
  - Concatenation: $\mathrm{A} \cdot B \; = \; \{x.y | x \in A \; and \; y \in B\}$.
    - attaches a string from A in front of a string from B in all possible ways to get the strings in the new language.
  - Star: $A^* \; = \; \{x_1 x_2 \ldots x_k | k \geq 0 \;\; and \;\; x_i \in A\}$.
    - attaches any number of strings in A together to get a string in the new language.

Chapter 1

The regular languages

- What languages are not regular ?

- **Answer: Any languages that requires memory.**
  - FSM memory is very limited
    - Cannot store the string
    - Cannot count the symbols
  - Will be discussed later.

- **Examples**: Let L is a language over **Σ**={0,1} and,

  - $L = \{ww|w\ is\ a\ binary\ string.\} = \{0101, 010010, 1100111001\}$
  - $L = \{w|w = 1^n0^n, where\ n \in N\} = \{10, 1100, 111000, 11110000\}$

  - These languages are not regular.

The regular languages

Example 3: Is the following language regular?

0 3 6 9 12

- A = {w| the binary string w is a multiple of 3.} = {0,11,110,1001, 1100,…}
- Can we construct a FSM for that?

- If a number is divisible by 3,
  - it can be written as the expression 3X
- If a number is not divisible by 3,
  - it can be written as the expression 3X+1 or 3X+2

- Also
  - If we add a "0" at the right most of a binary integer, k, its value will be doubled, 2k. For example 3 = 11 → 6 = 110
  - If we add a "1" at the right most of a binary integer, k, its value will be doubled, 2k+1. For example 3 = 11 → 7 = 111

The regular languages

Example 3: (continue)

- Q: {R0,R1,R2}   // R0: 3X , R1: 3X+1 , R2: 3X+2
- δ : Transition function?
  - If the new input symbol is 0 or 1, then what will be the new state?
    - 11100101010001?

δ	0	1
R0 : 3x	2(3x) : R0	2(3x)+1= 3(2x)+1 : R1
R1 : 3x +1	2(3x+1) = 3(2x)+2 : R2	2(3x+1)+1 = 3(2x+1) : R0
R2: 3x+2	2(3x+2) = 3(2x)+1 : R1	2(3x+2)+1 = 3(2x')+2 : R2

x' = x+1

Chapter 1

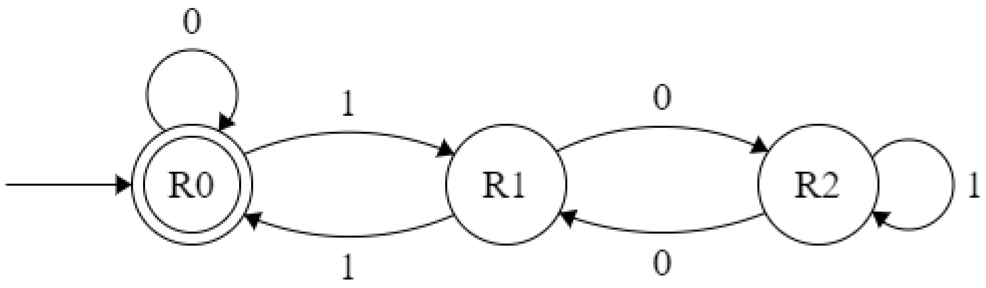
The regular languages

Example 3: (continue)

- Q: {R0,R1,R2} // R0: 3X , R1: 3X+1 , R2: 3X+2
- δ : Transition function?

δ	0	1
→ R0	R0	R1
R1	R2	R0
R2	R1	R2

- Start state: R0
- Final states: {R0}



3k+0 3k+1 3k+2

Chapter 1

The regular operations

- Example 1: Let
  - $Alpahbet\ \Sigma = \{a, c, b, \dots, z\}$
  - $Languages\ over\ \Sigma : A = \{easy, difficult\}, and\ B = \{exam, quiz\}.$

  - $A \cup B = \{easy, difficult, exam, quiz\}.$
  - $A \cdot B = \{easyexam, easyquiz, difficultexam, difficultquiz\}.$
  - $A^* = \{\varepsilon, easy, difficult, easyeasy, easydifficult, difficulteasy,$ $difficultdifficult, easyeasyeasy, , \dots\}.$

The regular operations

- Example 2: Let
  - $Alpahbet\ \Sigma = \{0,1\}$
  - $Languages\ over\ \Sigma : A = \{00,000\}$, and $B = \{11,011,111\}$

  - $A \cup B = \{00,000,11,011,111\}$.
  - $A \cdot B = \{0011,00011,00011,000011,00111,000111\}$.
  - $A^{*} = \{\varepsilon, 00,000,0000,00000,000000\ldots\}$. (kleen star)
  - $A^{+} = \{00,000,0000,00000,000000\ldots\}$. (kleen plus)

The regular operations properties

- Closure
  - Generally speaking, a collection of objects is closed under some operation if applying that operation to members of the collection returns an object still in the collection.
  - If for every x and y in set A, (x ◊ y) is in A, where ◊ is an operation defined on set A, then A is closure under ◊.

- **Example** :
  - N is closed under **multiplication**, but it is not closed under **division**.

The regular operations properties

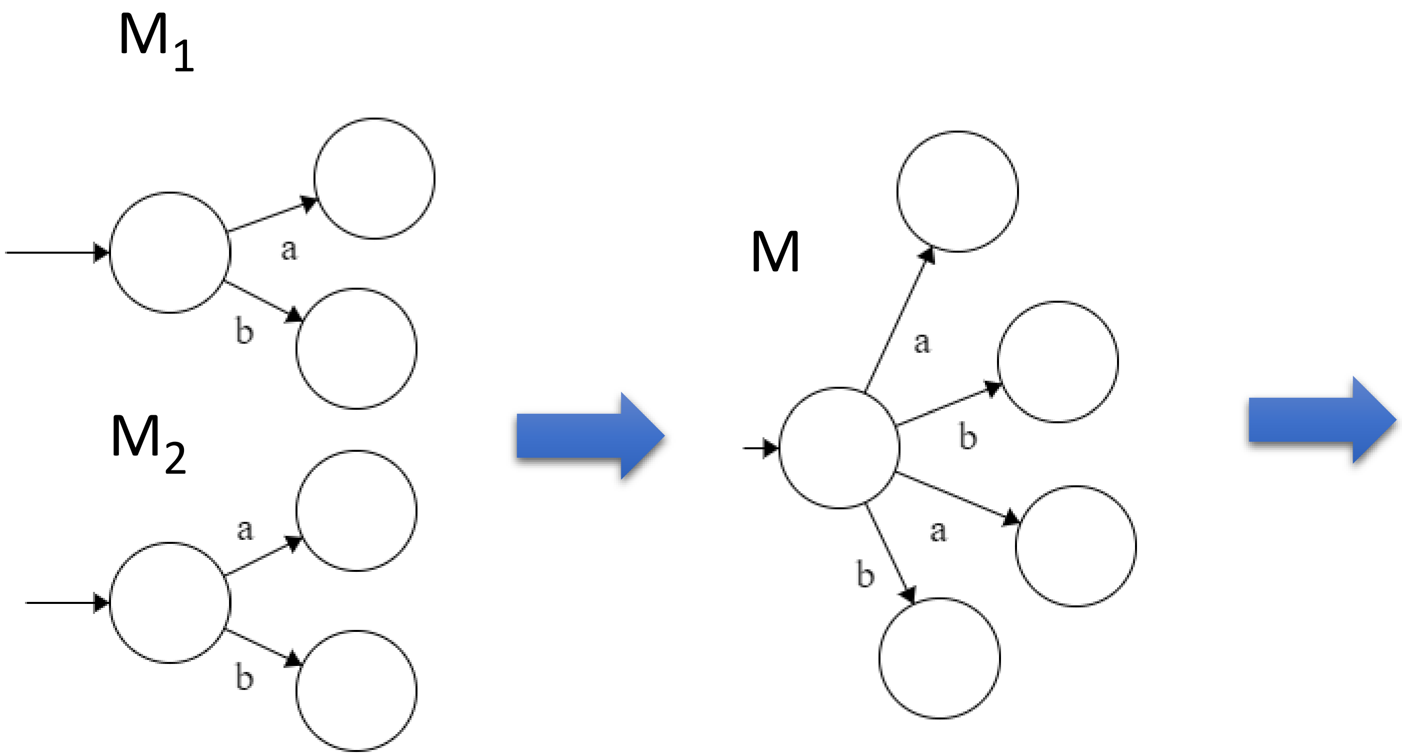
- Theorem
  - **The class of regular languages is closed under the union operation**.
  - In other words, if A1 and A2 are regular languages, so is $A_1 \cup A_2$.

- **Proof** :
  - **This is a proof by construction.**
  - To prove that $A_1 \cup A_2$ is regular, we demonstrate a finite automaton, call it $\mathrm{M} = (Q, \Sigma, \delta, q_0, F)$, that recognizes $A_1 \cup A_2$.
  - Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
  - We construct M from M_1 and M_2.

Chapter 1

The regular operations properties

- Approaches to build the machine:
  1. **Combine the machines**



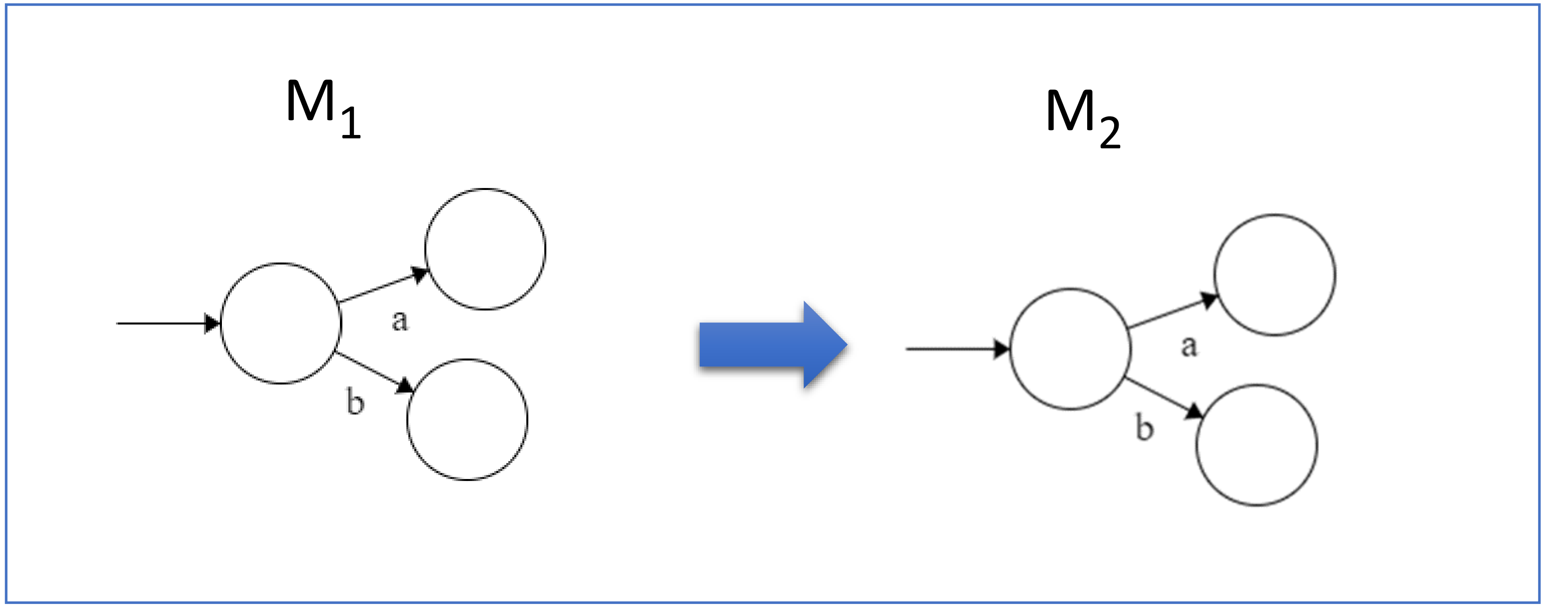
It is not a FSM

Chapter 1

The regular operations properties

- Approaches to build the machine:
  2. **Cascading:** First, Running M_1, then running M_2

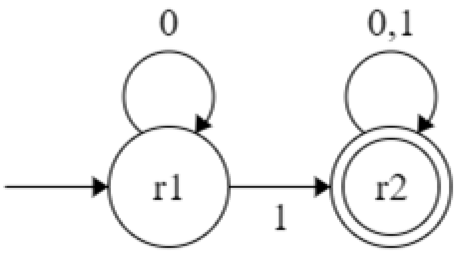
M



Cannot rewind input for M_2

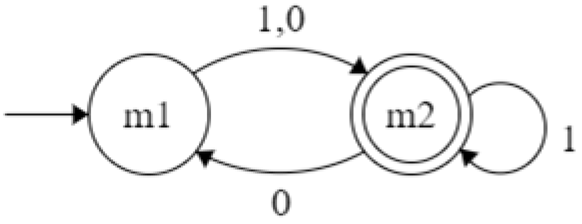
The regular operations properties

3. **Simulate both M_1 and M_2 simultaneously:**
   - Each state in new machine ***M*** represents two states; one from M_1 and one from M_2
   - Finding **Q** : $\{(r_1, r_2) | r_1 \in Q_1 \text{and} r_2 \in Q_2\} = Q_1 \times Q_2$



M_1

δ_1	0	1
→ $r1$	$r1$	$r2$
$r2$	$r2$	$r2$



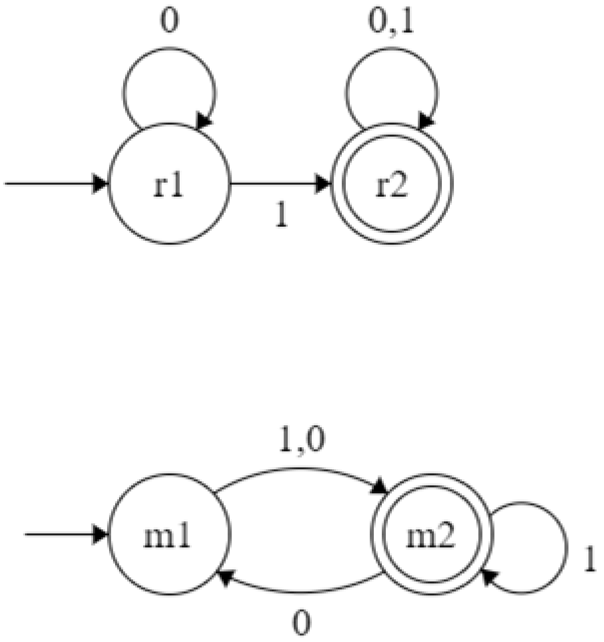
M_2

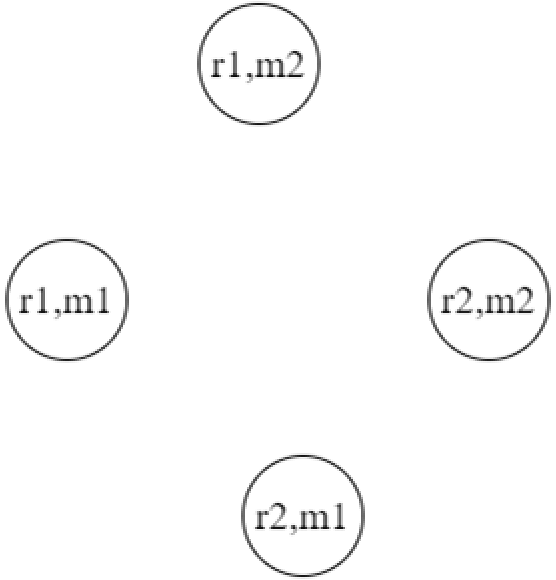
δ_2	0	1
→ $m1$	$m2$	$m2$
$m2$	$m1$	$m2$

$Q = Q_1 \times Q_2$	↓ m1	m2
→ $r1$	$(r1,m1)$	$(r1,m2)$
$r2$	$(r2,m1)$	$(r2,m2)$

The regular operations properties

- Machine M:
  - $Q = Q_1 \times Q_2 = \{(\mathrm{r1}, \mathrm{m1}), (\mathrm{r1}, \mathrm{m2}), (\mathrm{r2}, \mathrm{m1}), (\mathrm{r2}, \mathrm{m2})\}$

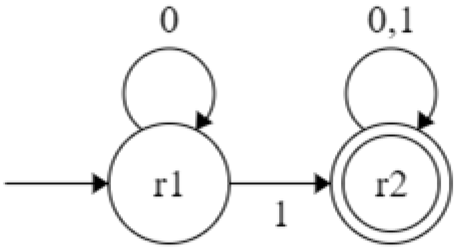


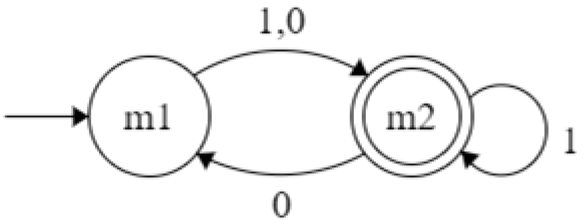


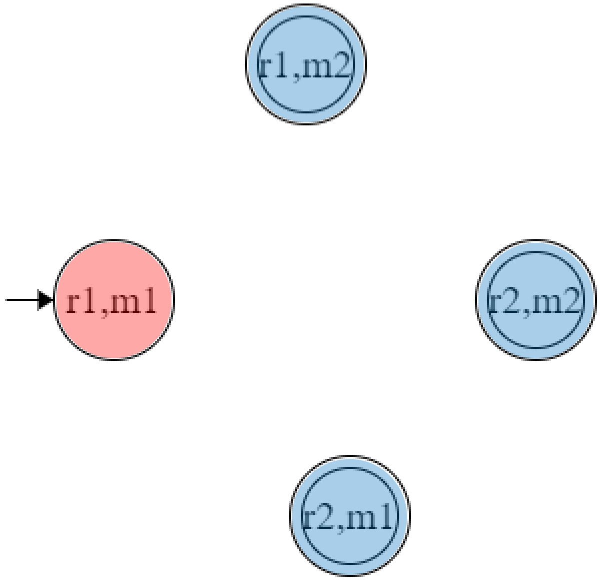
$Q_1 \times Q_2$	m1	m2
→ *r1*	*(r1,m1)*	*(r1,m2)*
r2	*(r2,m1)*	*(r2,m2)*

The regular operations properties

- Machine M:
  - $Q = Q_1 \times Q_2 = \{(\mathrm{r1},\mathrm{m1}),(\mathrm{r1},\mathrm{m2}),(\mathrm{r2},\mathrm{m1}),(\mathrm{r2},\mathrm{m2})\}$
  - Start state : (r1,m1)
  - Accept states : { (r1,m2),(r2,m1),(r2,m2)}



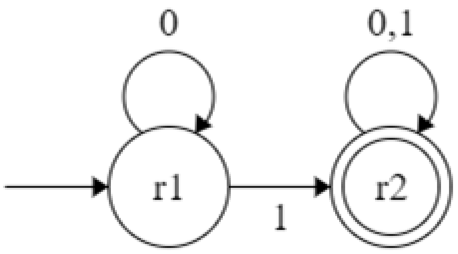




$Q_1 \times Q_2$	m1	m2
→ r1	(r1,m1)	(r1,m2)
r2	(r2,m1)	(r2,m2)

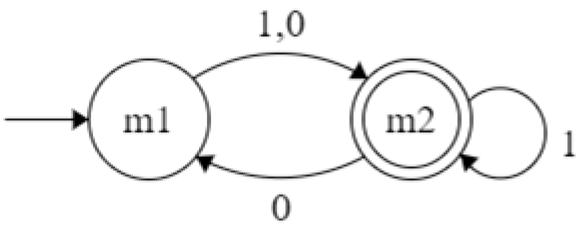
The regular operations properties

- Machine M:
  - $Q = Q_1 \times Q_2 = \{(\mathrm{r1},\mathrm{m1}),(\mathrm{r1},\mathrm{m2}),(\mathrm{r2},\mathrm{m1}),(\mathrm{r2},\mathrm{m2})\}$
  - Start state : (r1,m1)
  - Accept states : { (r1,m2),(r2,m1),(r2,m2)}
  - **the transition function ?** **δ((r_1,r_2),a)** = (δ$_1$(r_1,a),δ$_2$(r_2,a))



M_1

δ_1	0	1
→ $r1$	$r1$	$r2$
$r2$	$r2$	$r2$



M_2

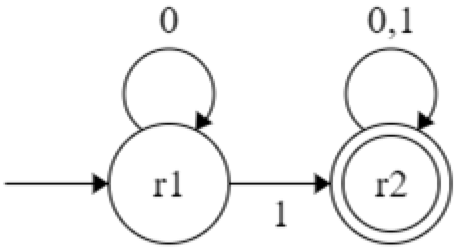
δ_2	0	1
→ $m1$	$m2$	$m2$
$m2$	$m1$	$m2$

δ	0	1
(r1,m1)	(r1,m2)	(r2,m2)
(r1,m2)	(r1,m1)	(r2,m2)
(r2,m1)	(r2,m2)	(r2,m2)
(r2,m2)	(r2,m1)	(r2,m2)

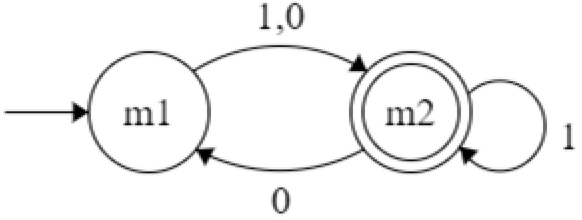
The regular operations properties

- Machine M:
  - $Q = Q_1 \times Q_2 = \{(\mathrm{r1},\mathrm{m1}),(\mathrm{r1},\mathrm{m2}),(\mathrm{r2},\mathrm{m1}),(\mathrm{r2},\mathrm{m2})\}$
  - Start state : (r1,m1)
  - Accept states : { (r1,m2),(r2,m1),(r2,m2)}
  - **the transition function ?**

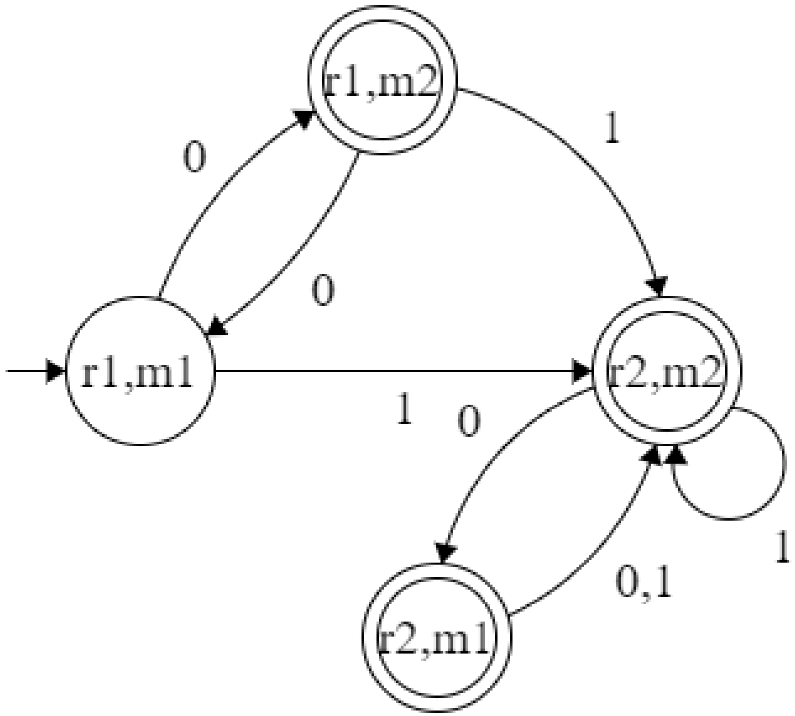
δ	0	1
→ (r1,m1)	(r1,m2)	(r2,m2)
(r1,m2)	(r1,m1)	(r2,m2)
(r2,m1)	(r2,m2)	(r2,m2)
(r2,m2)	(r2,m1)	(r2,m2)



$L(M_1) = L_1$



$L(M_2) = L_2$



$L(M) = L_1 \cup L_2$

The regular operations properties

- **Theorem**
  - **The class of regular languages is closed under the union operation**.
  - In other words, if A1 and A2 are regular languages, so is $A_1 \cup A_2$.

- **Proof** (cont.):
  - **proof by construction :** The language of the following machine is $A_1 \cup A_2$.
  - **FSM** $\mathrm{M} = (Q, \Sigma, \delta, q_0, F)$
    - Σ, the Alphabet
    - Q= $\{(r_1, r_2) \mid r_1 \in Q_1$ and $r_2 \in Q_2\}$.
    - $\boldsymbol{\delta}$ **,the transition function :** $\delta((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$
    - $q_0 = (q_1, q_2)$.
    - F= $\{(r_1, r_2) \mid r_1 \in F_1$ or $r_2 \in F_2\}$ = $(F_1 \times Q_2) \cup (Q_1 \times F_2)$.

  - Since we could construct a FSM that recognizes $A_1 \cup A_2$ is regular, then $A_1 \cup A_2$ is regular.

The regular operations properties

- **Theorem**
  - **The class of regular languages is closed under the union operation**.
  - In other words, if A1 and A2 are regular languages, so is $A_1 \cup A_2$.

- **Proof** (cont.):
  - **proof by construction :** The language of the following machine is $A_1 \cup A_2$.
  - **FSM** $\mathrm{M} = (Q, \Sigma, \delta, q_0, F)$
    - Σ, the Alphabet
    - Q= $\{(r_1, r_2) | r_1 \in Q_1 \text{and} r_2 \in Q_2\}$.
    - **δ ,the transition function :**$\delta((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$
    - $q_0 = (q_1, q_2)$.
    - F= $\{(r_1, r_2) | r_1 \in F_1 \text{or } r_2 \in F_2\}$ = $(F_1 \times Q_2) \cup (Q_1 \times F_1)$ $\neq \boldsymbol{F_1 \times F_2}$.

  - Since we could construct a FSM that recognizes $A_1 \cup A_2$ is regular, then $A_1 \cup A_2$ is regular.

The regular operations properties

- **Theorem**
  - **The class of regular languages is closed under the intersection operation**.
  - In other words, if A1 and A2 are regular languages, so is $A_1 \cap A_2$.

- **Proof** (cont.):
  - **proof by construction :** The language of the following machine is $A_1 \cap A_2$.
  - **FSM** $\mathrm{M} = (Q, \Sigma, \delta, q_0, F)$
    - Σ, the Alphabet
    - Q= $\{(r_1, r_2) | r_1 \in Q_1 \text{and} r_2 \in Q_2\}$.
    - $\boldsymbol{\delta}$ **,the transition function :**$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
    - $q_0 = (q_1, q_2)$.
    - F= $\{(r_1, r_2) | r_1 \in F_1 \text{or } r_2 \in F_2\}$ = $F_1 \times F_2$.

  - Since we could construct a FSM that recognizes $A_1 \cap A_2$ is regular, then $A_1 \cap A_2$ is regular.