CS 6041
Theory of Computation

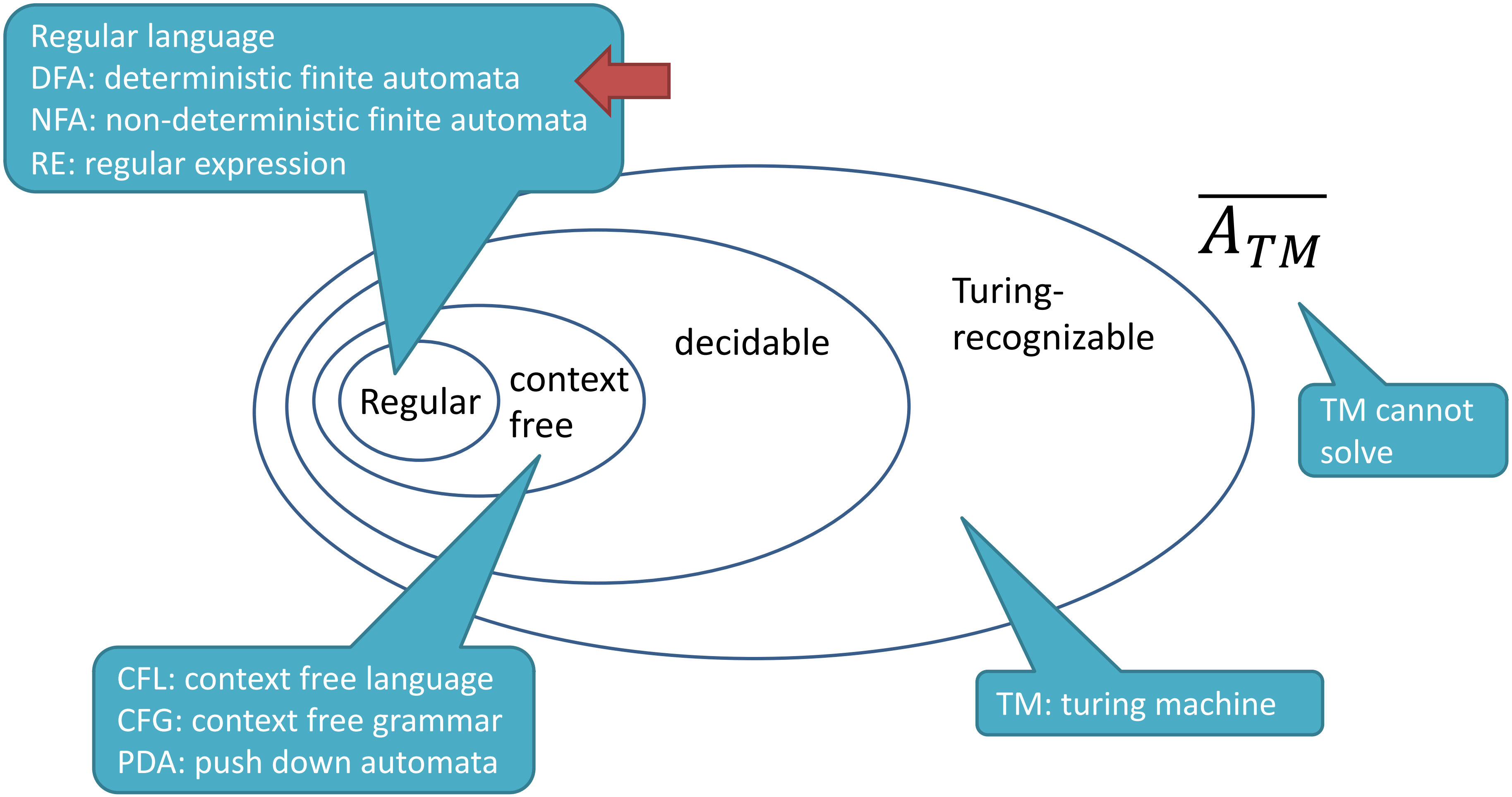
Deterministic finite automata

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Where are we now?



Outline

- Finite Automata
  - Definition
  - Example
  - Language of DFA
  - Computation for DFAs
- Design DFAs
  - Example
  - Regular language
  - Regular operation

Finite Automata

- Definition: DFA is a 5-tuple $M=(Q,\Sigma,\delta,q_0,F)$
- Language on M: $L(M) = \{w \in \Sigma^* \mid \delta(q_0,w) \in F\}$

  Input string w

  Final state is accept

- DFA practice:
  - definition → graph/language
  - graph → language
  - language → graph

  This class

Designing finite automata

$M=(Q,\Sigma,\delta,q_0,F)$

- State:
  - All states, start state, accept state, etc.
- Transition:
  - from one state to another state based on the input

Design a DFA for a language

- Step 1: list all possible states
- Step 2: draw all the transitions between the states
- Step 3: add start and accept states

Example 1: language - - > figure

- $L(E_1)=\{ w \mid w \text{ has odd amount of 1s } \}, \Sigma=\{0,1\}$

Step 1: define states

Example 1: language - - > figure

- $L(E_1)$={ w | w has odd amount of 1s }, Σ={0,1}

q_{even}: even amount of 1s
q_{odd} : odd amount of 1s

Step 1: define states

Example 1: language - - > figure

- $L(E_1)$={ w | w has odd amount of 1s }, Σ={0,1}

q_{even}: even amount of 1s
q_{odd} : odd amount of 1s

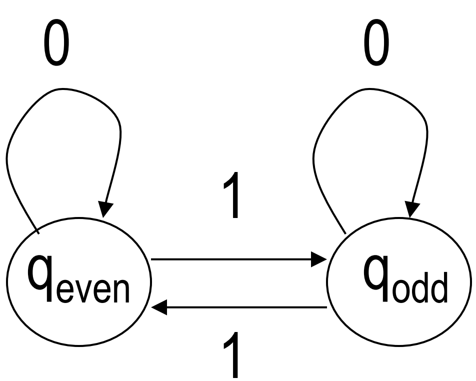
Step 1: define states

Step 2: define transitions

Example 1: language - - > figure

- $L(E_1)=\{\ w \mid w$ has odd amount of 1s $\}$, $\Sigma=\{0,1\}$

q_{even}: even amount of 1s
q_{odd} : odd amount of 1s



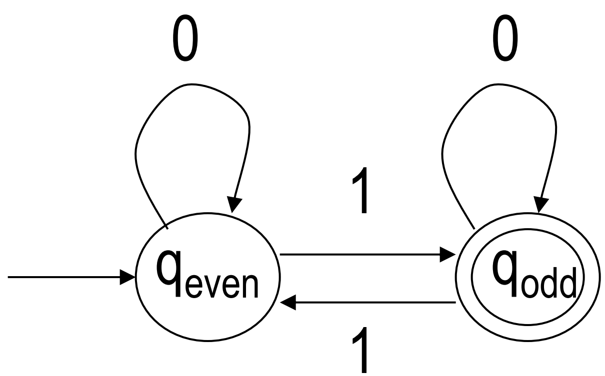
Step 1: define states

Step 2: define transitions

Example 1: language - - > figure

- $L(E_1)=\{ w \mid w \text{ has odd amount of 1s} \}, \Sigma=\{0,1\}$

q_{even}: even amount of 1s
q_{odd} : odd amount of 1s



Step 1: define states

Step 2: define transitions

Step 3: define start state and accept states

Example 2: language - - > figure

- $L(E_2)=\{ w \mid w \text{ has substring } 001 \}, \Sigma=\{0,1\}$

Example 2: language - - > figure

- $L(E_2)=\{ w \mid w \text{ has substring } 001 \}, \Sigma=\{0,1\}$

q: empty string
q_0: has substring 0 (for $\underline{0}01$)
q_{00}: has substring 00 (for $\underline{00}1$)
q_{001}: has substring 001

Example 2: language - - > figure

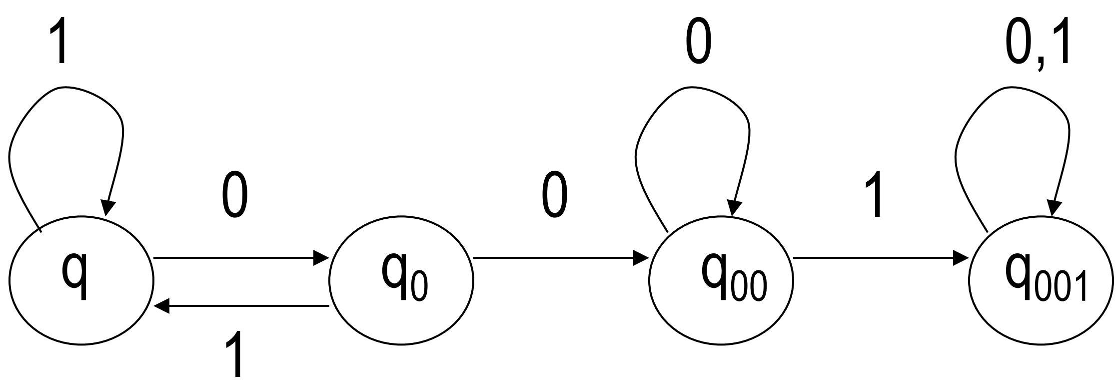
- $L(E_2)$={ w | w has substring 001 }, Σ={0,1}

q: empty string (no 0, 00, 001)
q_0: has substring 0 (for <u>0</u>01)
q_{00}: has substring 00 (for <u>00</u>1)
q_{001}: has substring 001

Example 2: language - - > figure

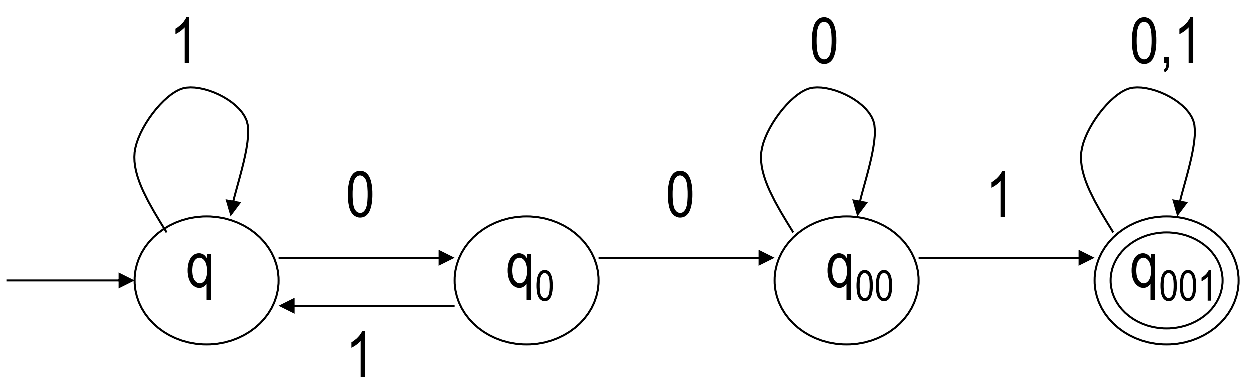
- $L(E_2)$={ w | w has substring 001 }, Σ={0,1}

q: empty string (no 0, 00, 001)
q_0: has substring 0 (for <u>0</u>01)
q_{00}: has substring 00 (for <u>00</u>1)
q_{001}: has substring 001



Example 2: language - - > figure

- $L(E_2)=\{\ w \mid w$ has substring $001\ \}$, $\Sigma=\{0,1\}$



Example 3: language - - > figure

- L = Set of all strings that start with 0, $\Sigma=\{0,1\}$

  $= \{0, 00, 01, 000, 010, \ldots\}$

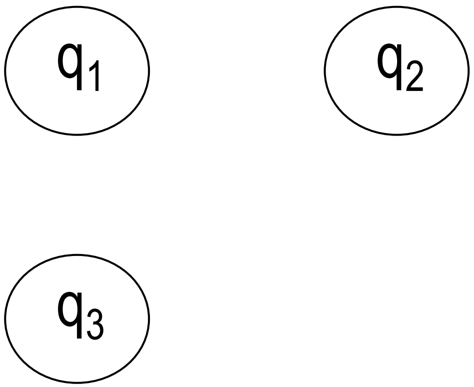
Can anyone draw the DFA?

Example 3: language - - > figure

- L = Set of all strings that start with 0, Σ={0,1}

  = {0, 00, 01, 000, 010, ...}

q_1: ε
q_2: start with 0
q_3: start with 1

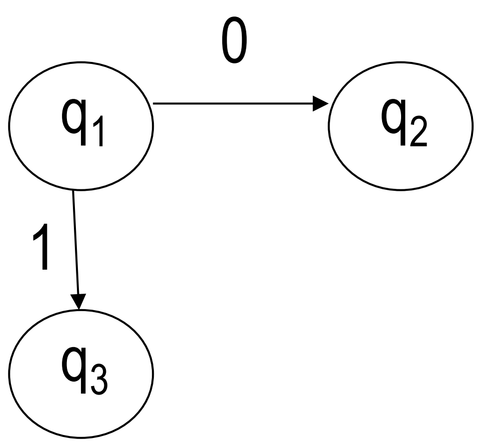


Example 3: language - - > figure

- L = Set of all strings that start with 0

  = {0, 00, 01, 000, 010, ...}

q_1: ε
q_2: start with 0
q_3: start with 1

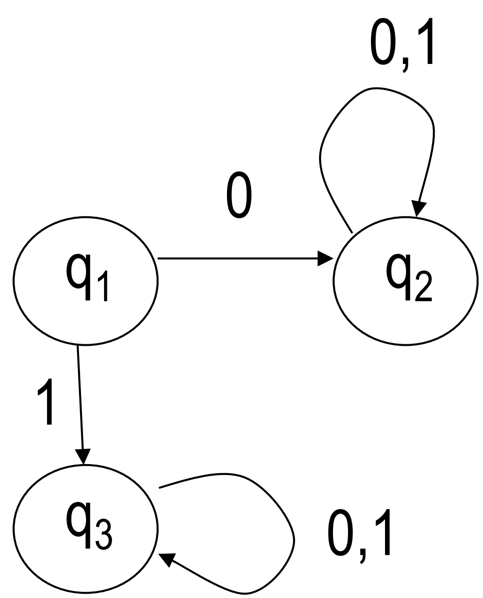


Example 3: language - - > figure

- L = Set of all strings that start with 0

  = {0, 00, 01, 000, 010, ...}

q_1: ε
q_2: start with 0
q_3: start with 1

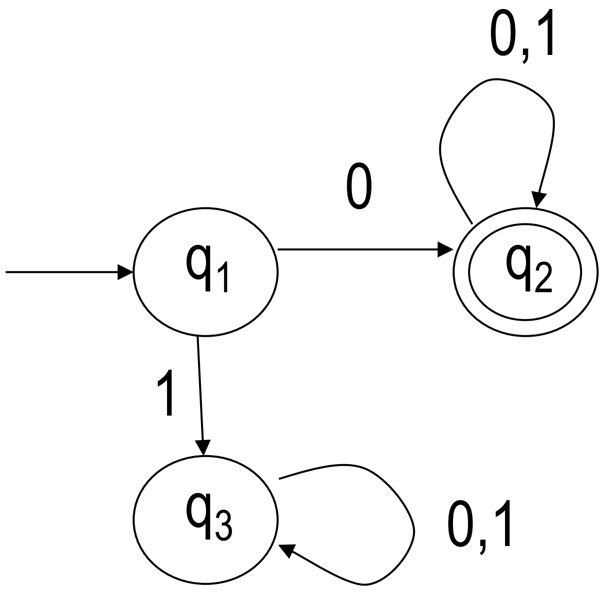


Example 3: language - - > figure

- L = Set of all strings that start with 0

  = {0, 00, 01, 000, 010, ...}

q_1: ε
q_2: start with 0
q_3: start with 1



Example 4: language - - > figure

- L = Set of all strings over {0,1} that of length is 2

  = {00, 01, 10, 11}

Can anyone draw the DFA?

Example 4: language - - > figure

- L = Set of all strings over {0,1} that of length is 2

  = {00, 01, 10, 11}

q_1: ε
q_2: length is 1
q_3: length is 2
q_4: length is 3 or more

Example 4: language - - > figure

- L = Set of all strings over {0,1} that of length is 2

  = {00, 01, 10, 11}

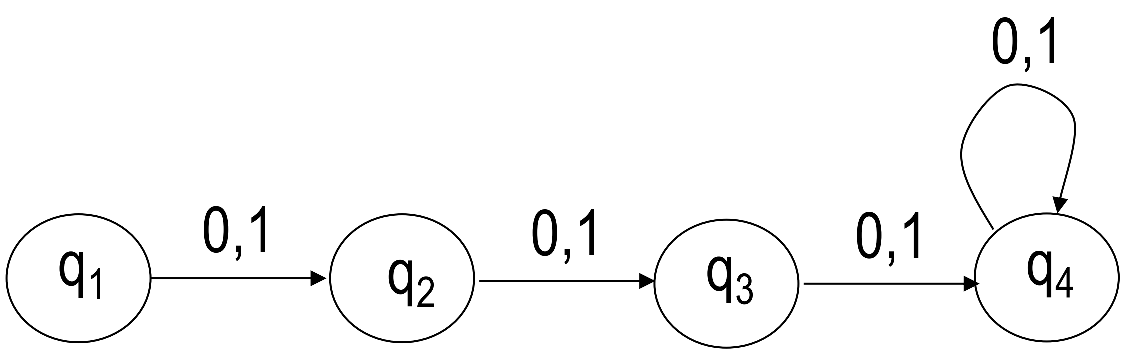
q_1: ε
q_2: length is 1
q_3: length is 2
q_4: length is 3 or more

Example 4: language - - > figure

- L = Set of all strings over {0,1} that of length is 2

  = {00, 01, 10, 11}

q_1: ε
q_2: length is 1
q_3: length is 2
q_4: length is 3 or more

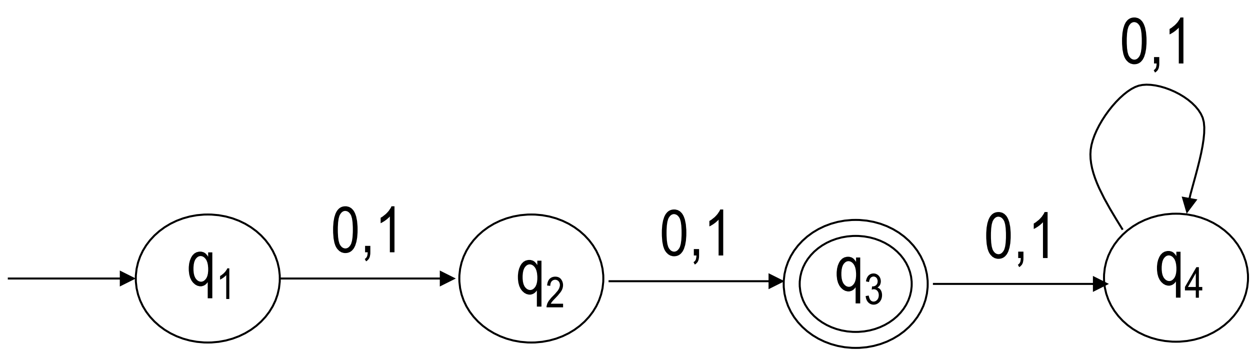


Example 4: language - - > figure

- L = Set of all strings over {0,1} that of length is 2

  = {00, 01, 10, 11}

q_1: ε
q_2: length is 1
q_3: length is 2
q_4: length is 3 or more



Example 5: language - - > figure

- L = Set of strings over {a,b} that contains string aabb in it

Can anyone draw the DFA?

Example 5: language - - > figure

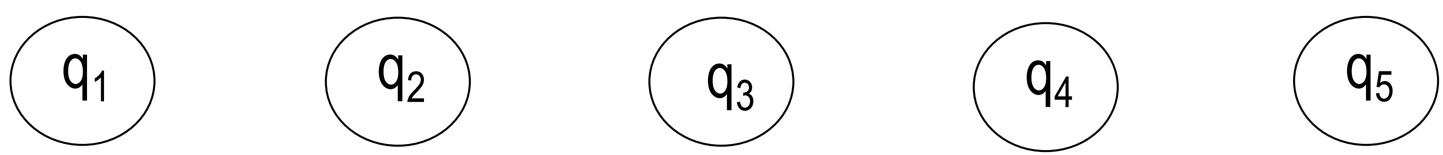
- L = Set of strings over {a,b} that contains string aabb in it

q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb

Example 5: language - - > figure

- L = Set of strings over {a,b} that contains string aabb in it

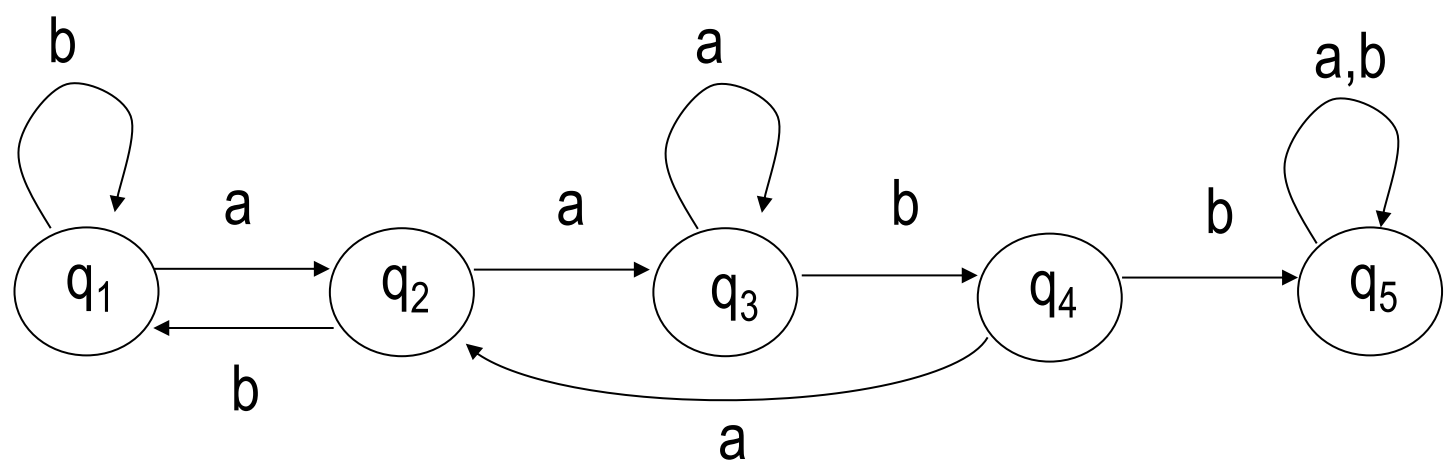
q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Example 5: language - - > figure

- L = Set of strings over {a,b} that contains string aabb in it

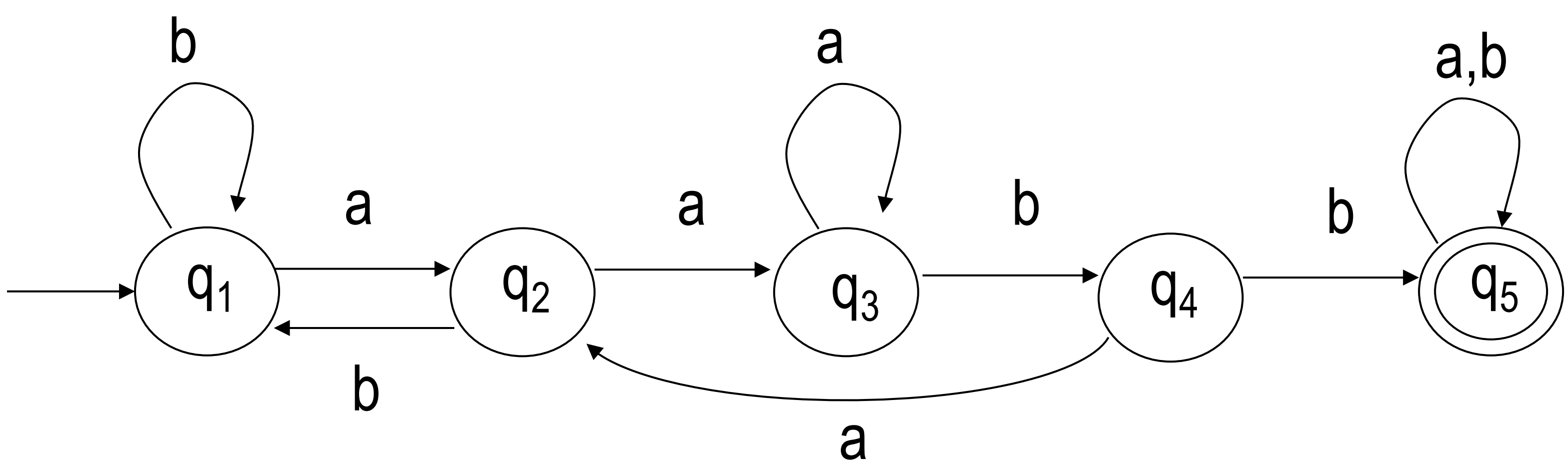
q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Example 5: language - - > figure

- L = Set of strings over {a,b} that contains string aabb in it

q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Example 6: language - - > figure

- L = Set of strings over {a,b} that does not contain string aabb in it

Can anyone draw the DFA?

Example 6: language - - > figure

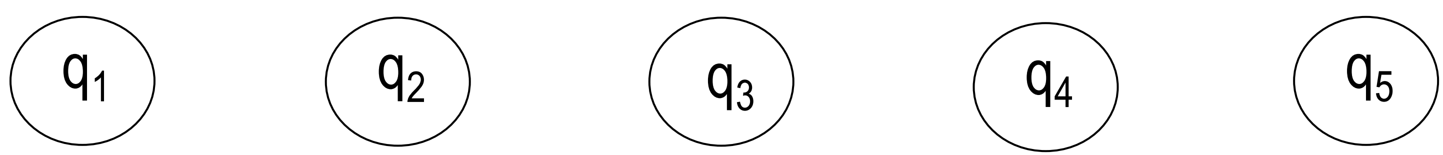
- L = Set of strings over {a,b} that does not contain string aabb in it

q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb

Example 6: language - - > figure

- L = Set of strings over {a,b} that does not contain string aabb in it

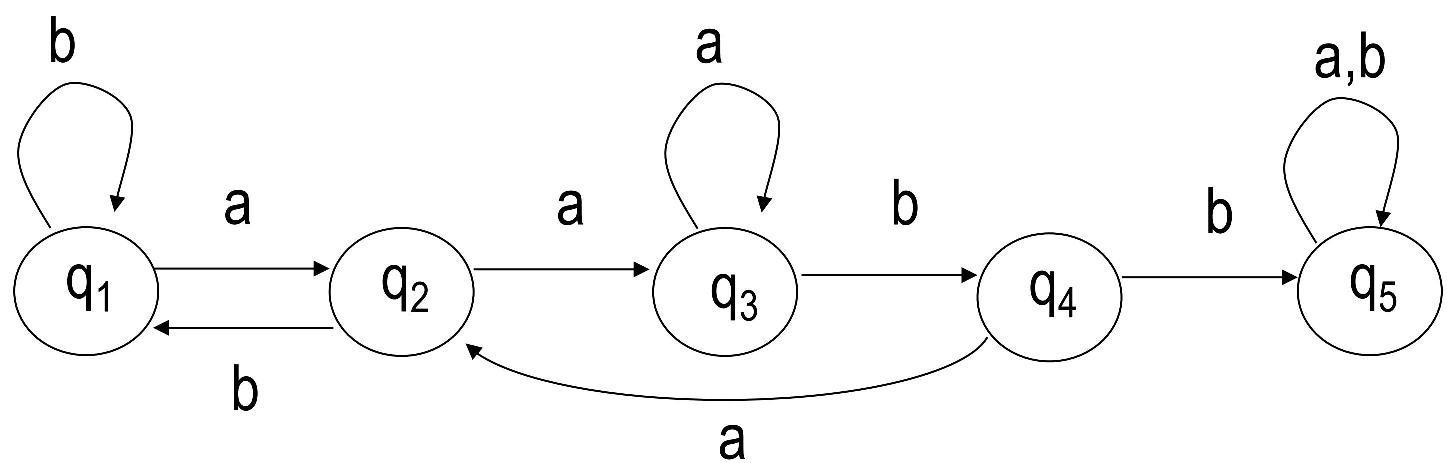
q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Example 6: language - - > figure

- L = Set of strings over {a,b} that does not contain string aabb in it

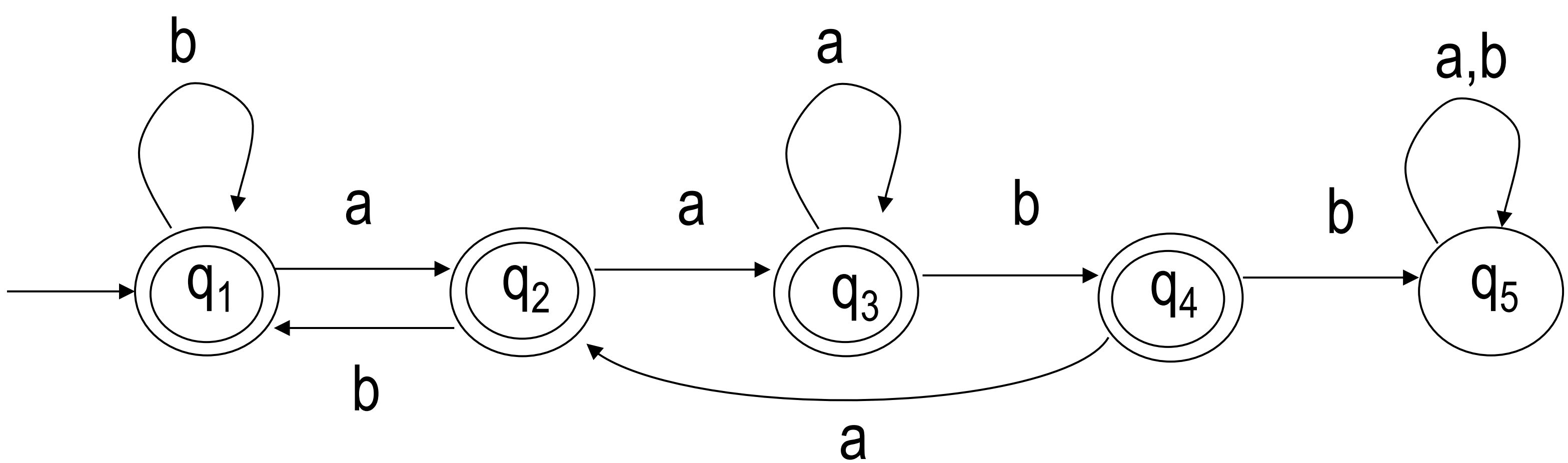
q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Example 6: language - - > figure

- L = Set of strings over {a,b} that does not contain string aabb in it

q_1: contains nothing
q_2: contains a
q_3: contains aa
q_4: contains aab
q_5: contains aabb



Regular language

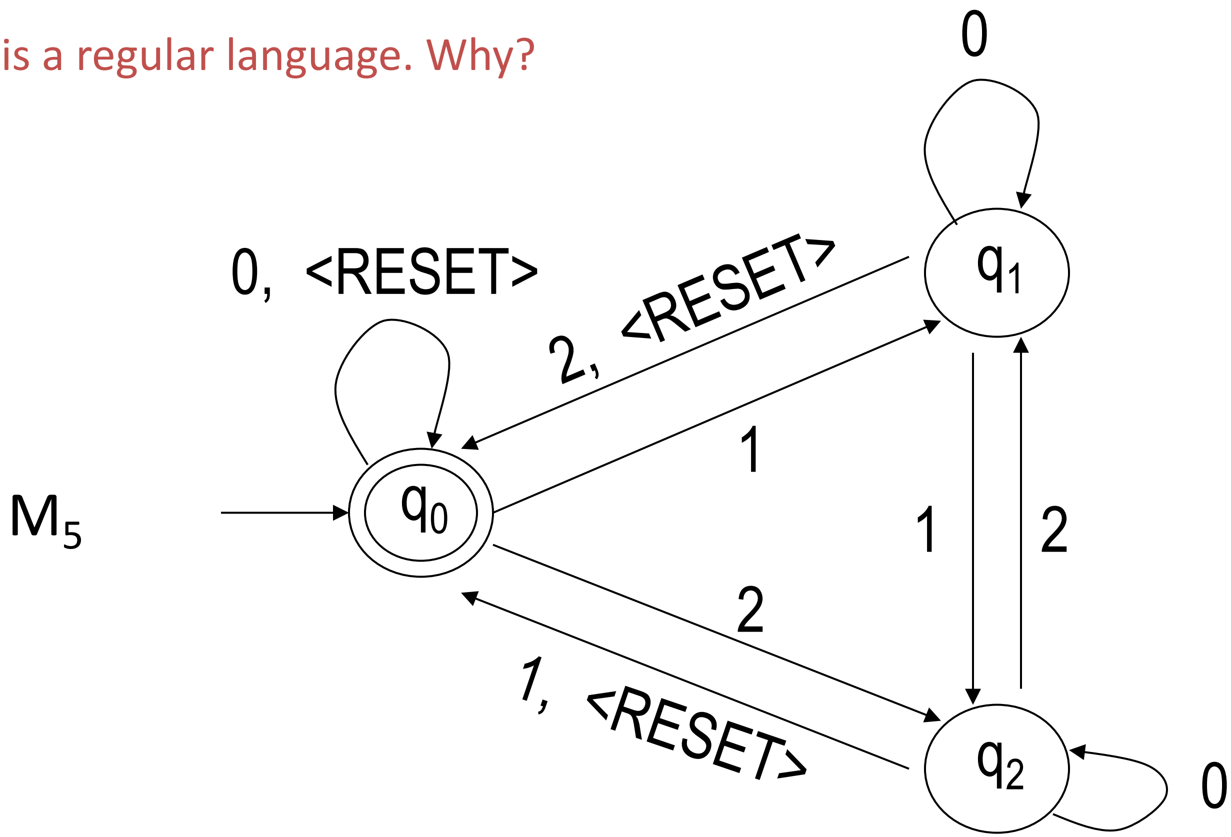
- A language is called a regular language if some finite automaton recognizes it

- Regular language:
  - $L=L(M)$
  - M is finite automaton

Regular language example

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

Why said $L(M_5)$ is a regular language. Why?



Regular language

- What languages are not regular?
  - Not recognized by any DFAs
  - Require memory
    - Memory for DFA is limited, it only stores its current state
    - It cannot store or count strings

- E.g., $a^n b^n$ is not regular language

Regular operations

- If A and B are languages
  - Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
  - Concatenation: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$
  - Star: $A^* = \{x_1 x_2 \dots x_k | k \geq 0 \text{ and each } x_i \in A\}$
- Examples:

$$A = \{\texttt{good}, \texttt{bad}\} \quad B = \{\texttt{boy}, \texttt{girl}\}$$

$$A \cup B = \{\texttt{good}, \texttt{bad}, \texttt{boy}, \texttt{girl}\},$$

$$A \circ B = \{\texttt{goodboy}, \texttt{goodgirl}, \texttt{badboy}, \texttt{badgirl}\}, \text{ and}$$

$$A^* = \{\varepsilon, \text{good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad}, \ldots\}.$$

Regular operations

- Theorem: the regular languages are closed under regular operations
  - Union, $A \cup B$
  - Concatenation, $A \cap B$
  - Star, A^*
  - Complement, $\bar{A}$
  - Boolean operation, $\mathrm{AND}{:}\wedge,\ OR{:}\vee,\ XOR{:}\oplus$

Regular operations

	DFA	PDA	TM
Union	close	?	?
Concatenation	close	?	?
Star	close	?	?
Complement	close	?	?
Boolean operation	close	?	?

Close under the union operation

- Theorem: regular language is closed under the union operation

Proof by construction → create a DFA for it

- Proof:

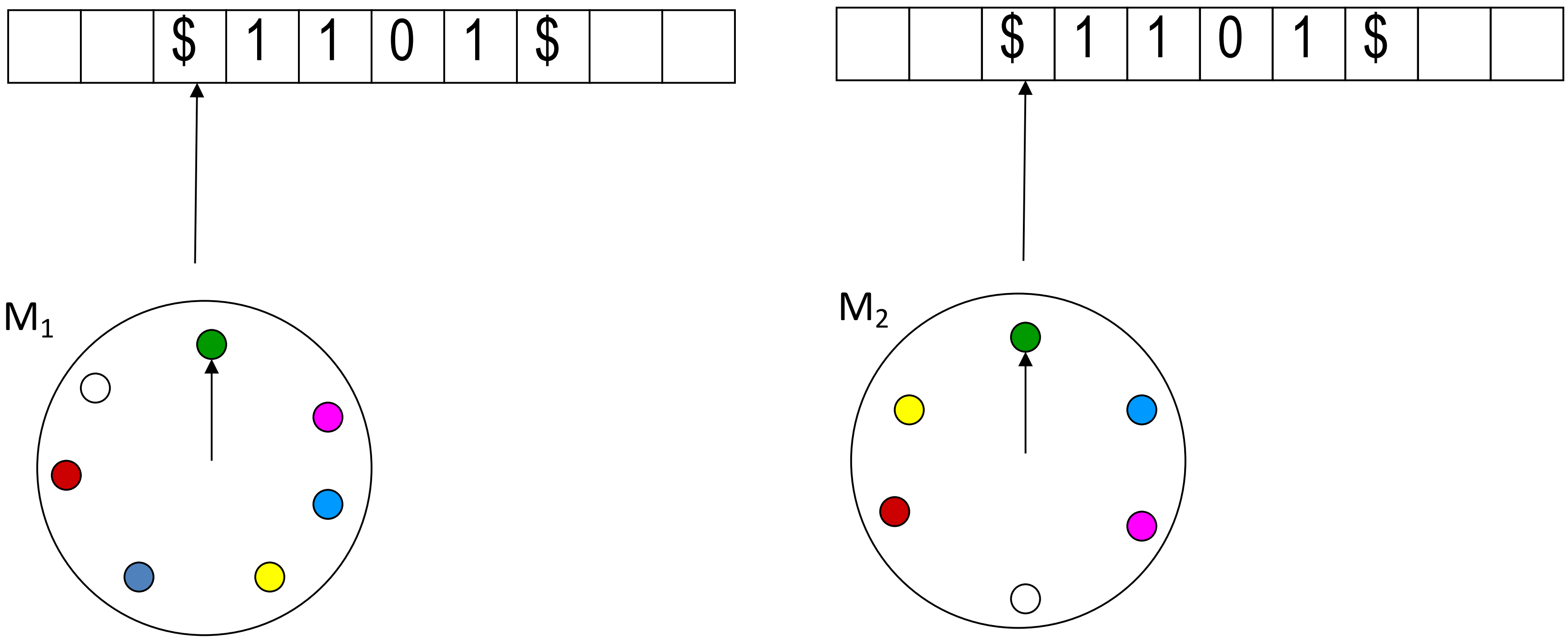
Let $L_i=L(M_i)$ is a regular language, $M_i=(Q_i,\Sigma,\delta_i,q_i,F_i)$, i=1,2. We need to build a finite automata to recognize $L_1 \cup L_2$

Build $M_3=(Q_3,\Sigma,\delta_3,q_3,F_3)$.
$Q_3=Q_1 \times Q_2$;
$\delta_3((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$;
$q_3=(q_1,q_2)$;
$F_3=(F_1 \times Q_2) \cup (Q_1 \times F_2)$.

$L(M_3) = L_1 \cup L_2$, so $L_1 \cup L_2$ is still regular language

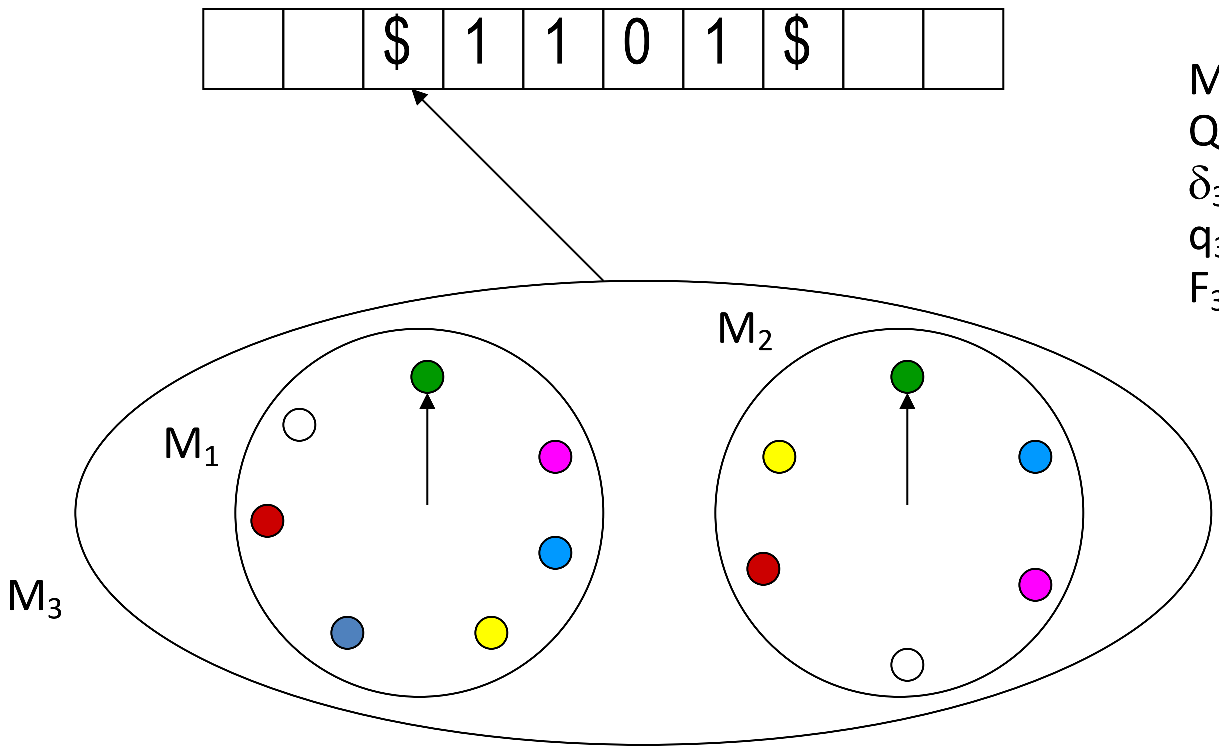
Close under the union operation

- Theorem: regular language is closed under the union operation



Close under the union operation

- Theorem: regular language is closed under the union operation



$M_3=(Q_3,\Sigma,\delta_3,q_3,F_3)$.
$Q_3=Q_1\times Q_2$;
$\delta_3((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$;
$q_3=(q_1,q_2)$;
$F_3=(F_1\times Q_2)\cup(Q_1\times F_2)$.

Close under concatenation operation

- Theorem: regular language is closed under the concatenation operation

Proof by construction → create a DFA for it

- Proof:

Let $L_i=L(M_i)$ is a regular language, $M_i=(Q_i,\Sigma,\delta_i,q_i,F_i)$, i=1,2. We need to build a finite automata to recognize $L_1 \cap L_2$

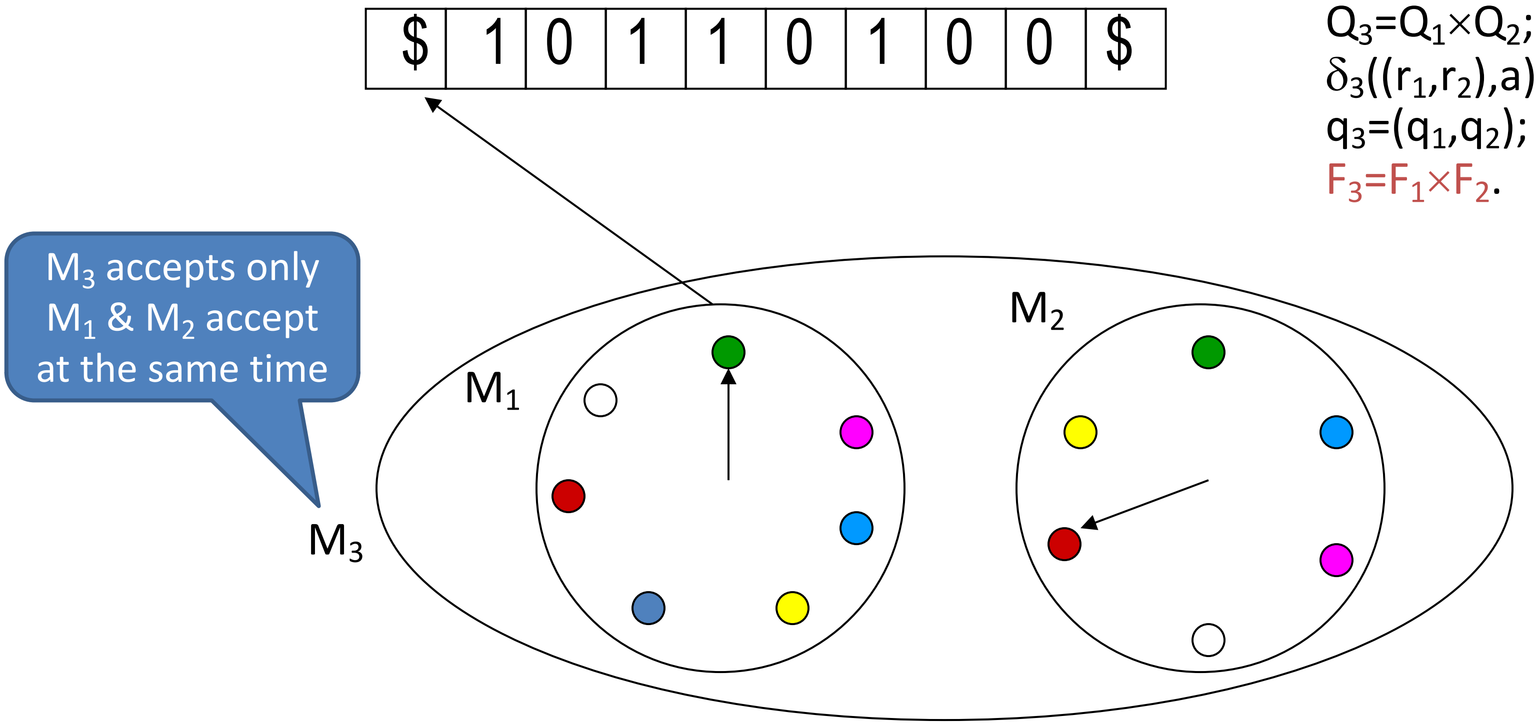
Build $M_3=(Q_3,\Sigma,\delta_3,q_3,F_3)$.
$Q_3=Q_1 \times Q_2$;
$\delta_3((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$;
$q_3=(q_1,q_2)$;
$F_3=F_1 \times F_2$.

$L(M_3) = L_1 \cap L_2$, so $L_1 \cap L_2$ is still regular language

Close under concatenation operation

- Theorem: regular language is closed under the concatenation operation

$M_3=(Q_3,\Sigma,\delta_3,q_3,F_3)$.
$Q_3=Q_1\times Q_2$;
$\delta_3((r_1,r_2),a)=(\delta_1(r_1,a),\delta_2(r_2,a))$;
$q_3=(q_1,q_2)$;
$F_3=F_1\times F_2$.



Draw DFA online

- http://madebyevan.com/fsm/

  - Add a state: double-click on the canvas
  - Add an arrow: shift-drag on the canvas
  - Move something: drag it around
  - Delete something: click it and press the delete key (not the backspace key); On Laptop/Macbook, please press “Fn” + “Delete/backspace”.
  - Make accept state: double-click on an existing state
  - Add start state: shift-drag on the canvas to one state
  - Type numeric subscript: put an underscore before the number (like "S_0")
  - Type greek letter: put a backslash before it (like "\beta")