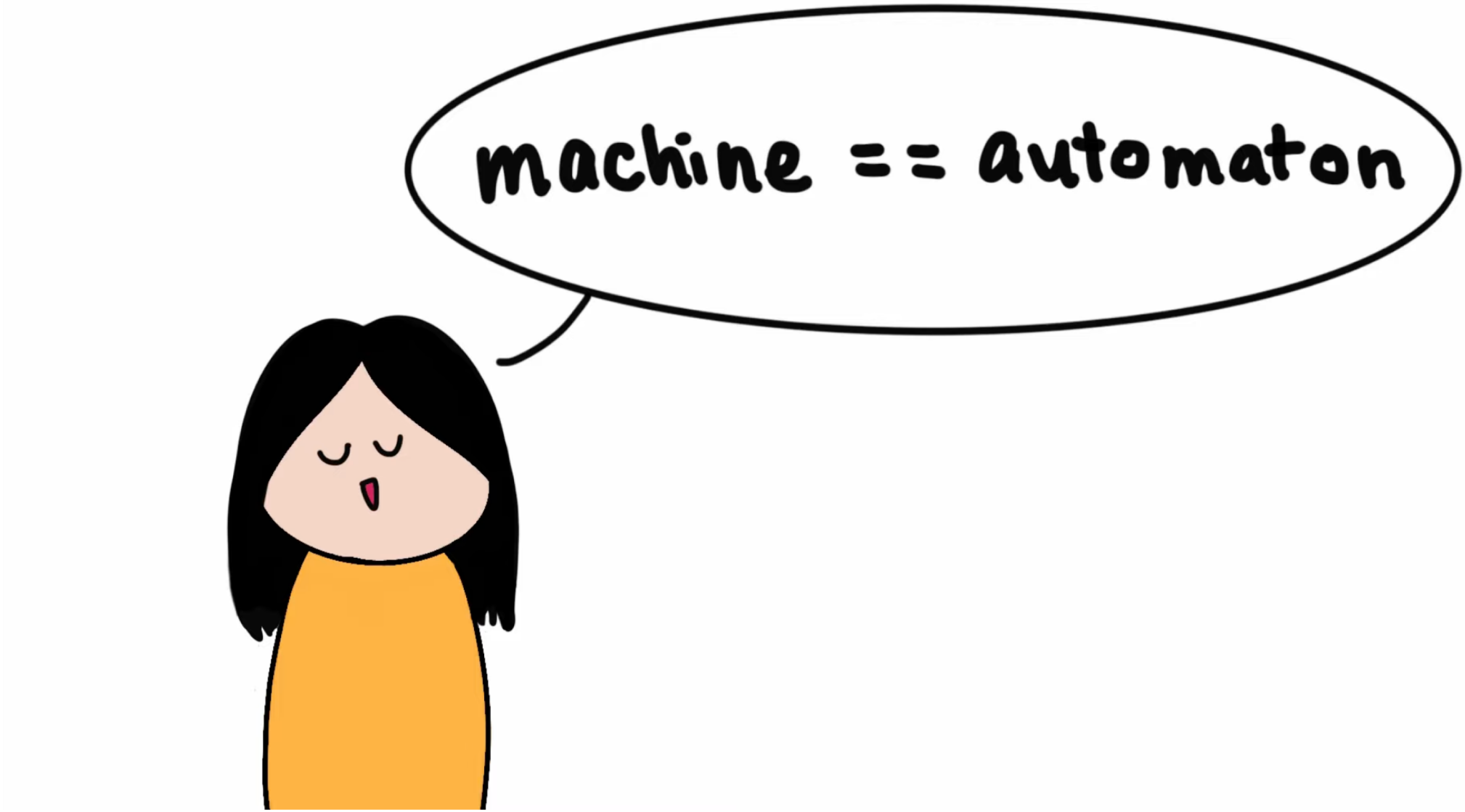
CS F351: Theory of Computation

02 - Deterministic Finite Automata

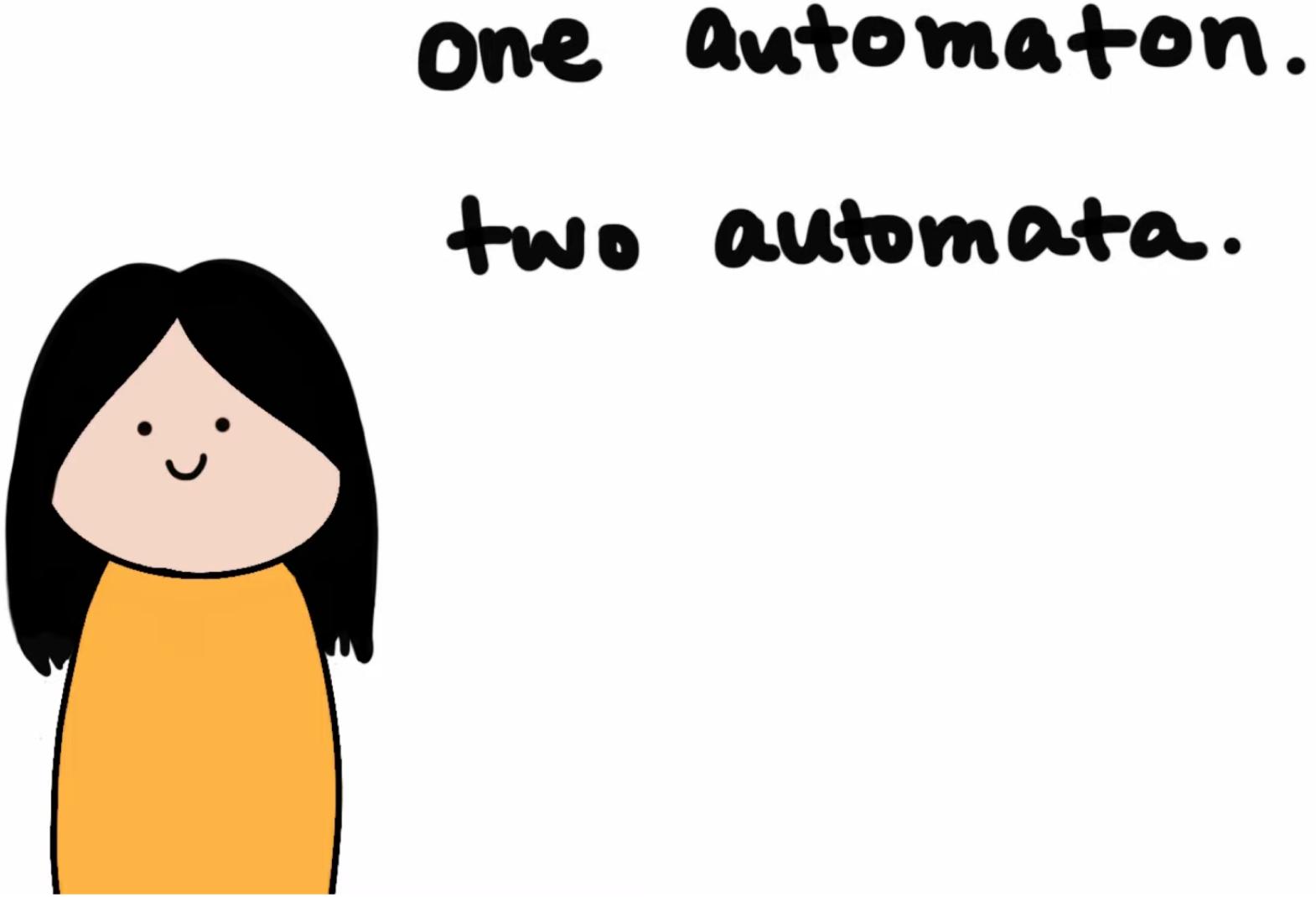
BITS Pilani
Dubai Campus

Dr Elakkiya R, AP/CS

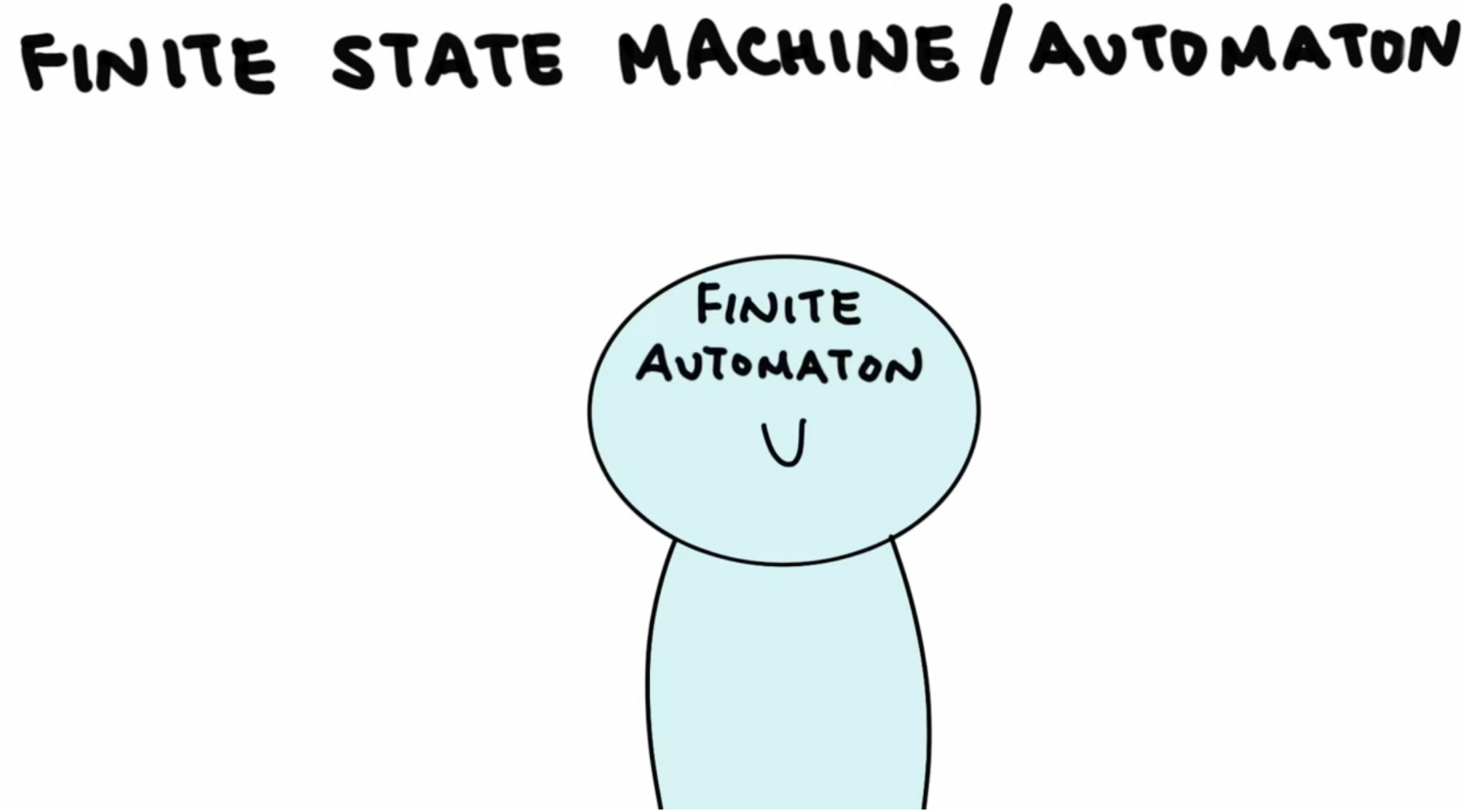
Introduction: Understanding of DFA



Introduction: Understanding of DFA



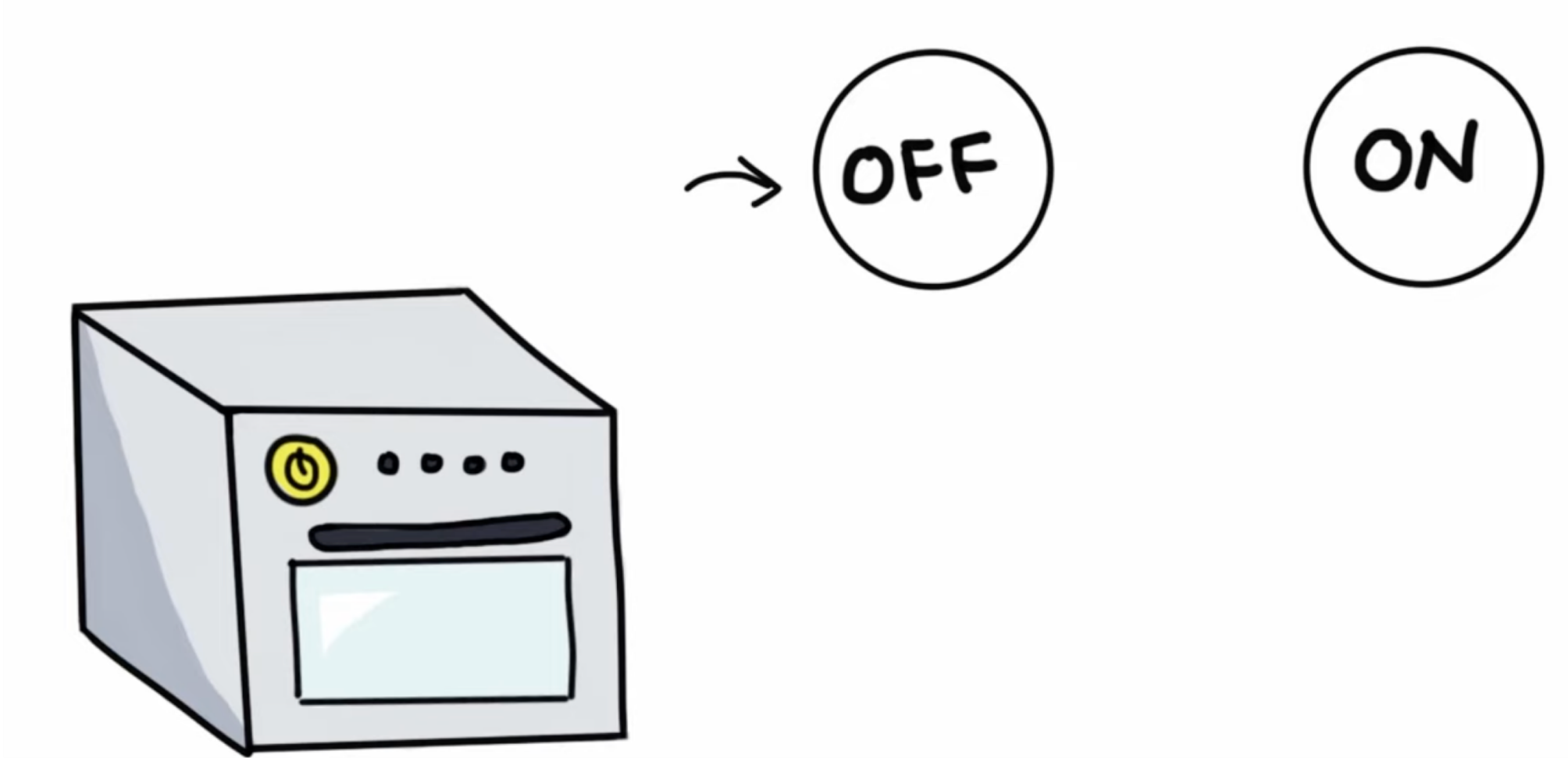
Introduction: Understanding of DFA



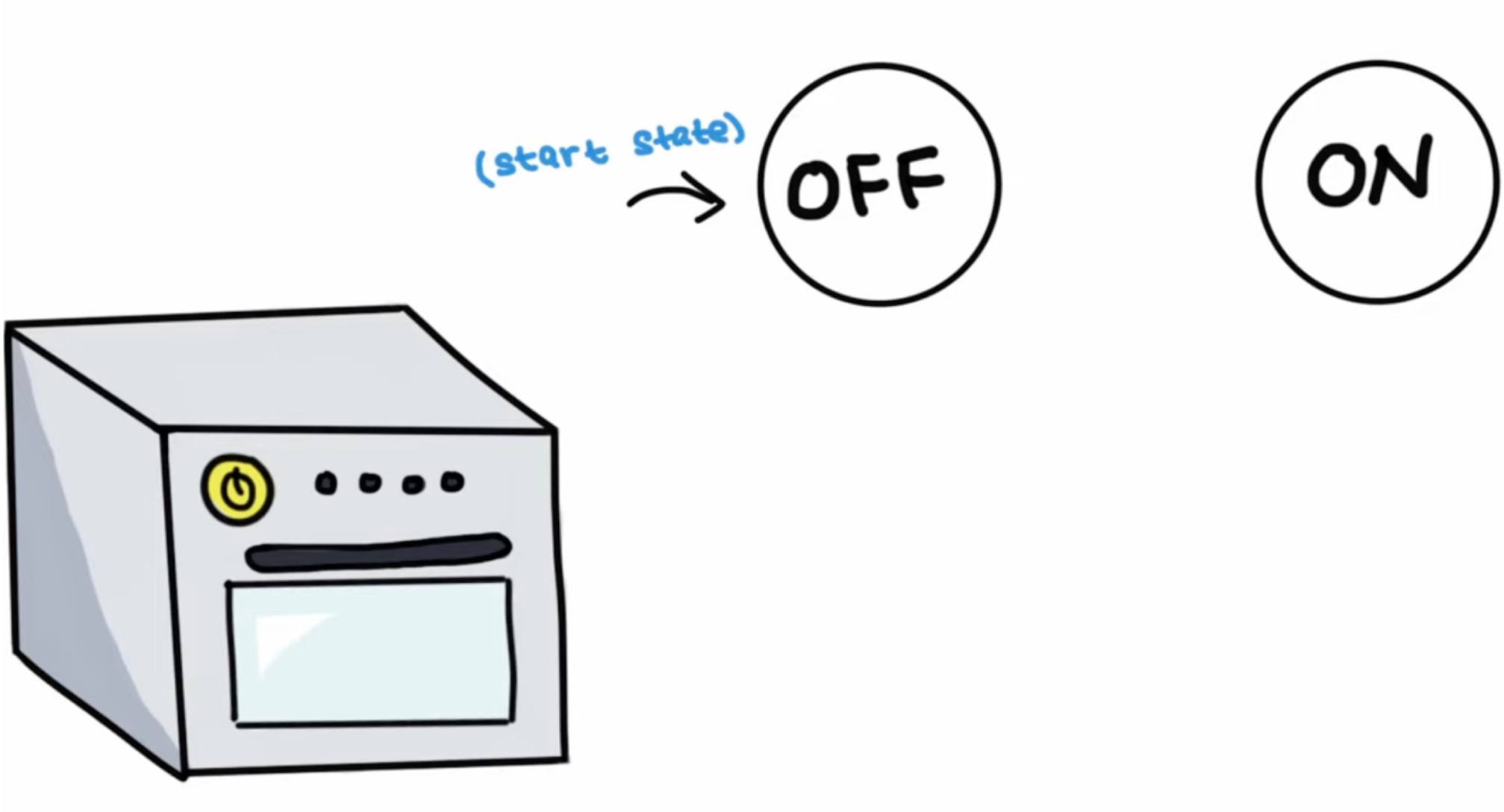
Introduction: Understanding of DFA

Introduction: Understanding of DFA

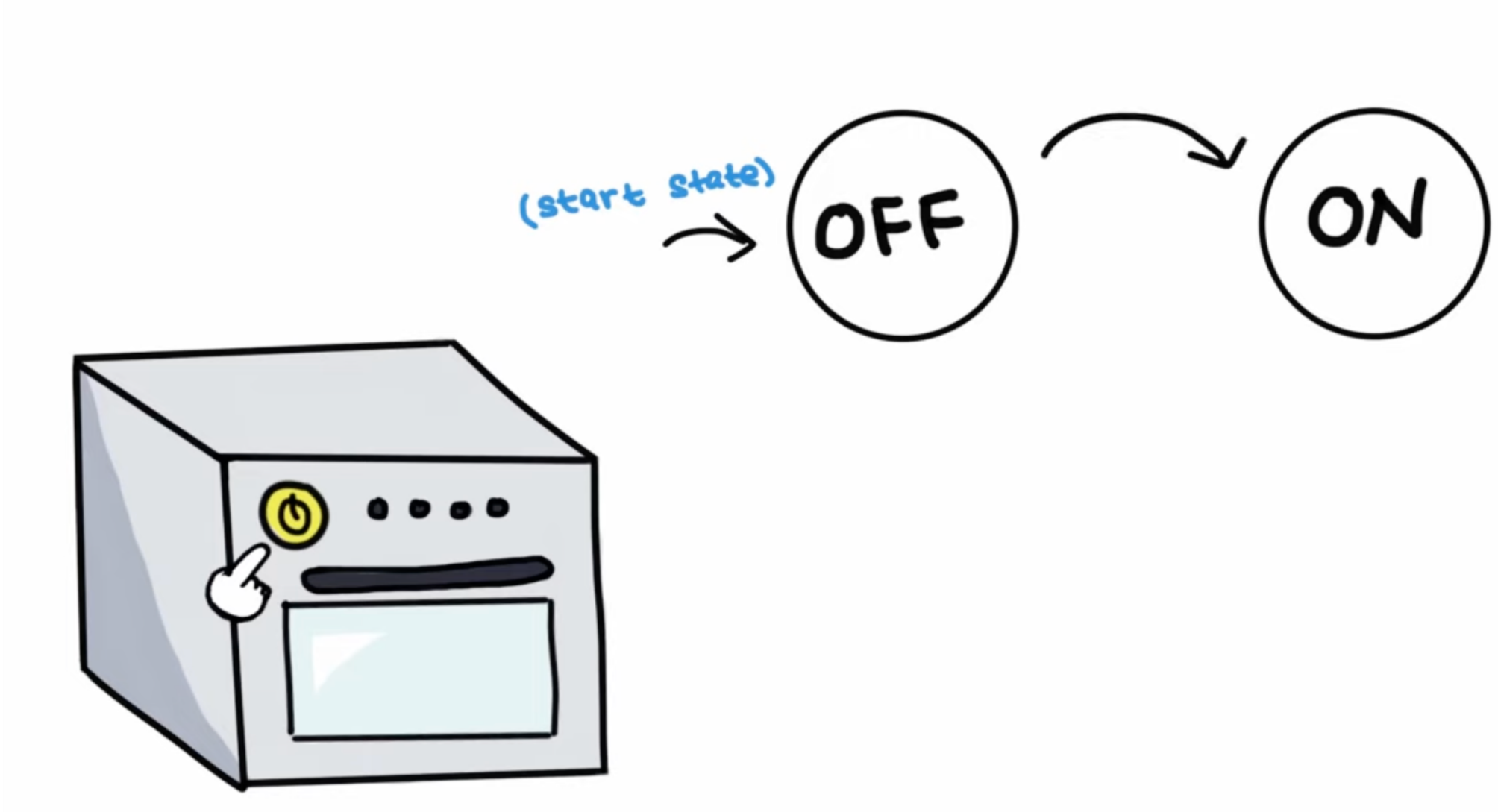
Introduction: Understanding of DFA



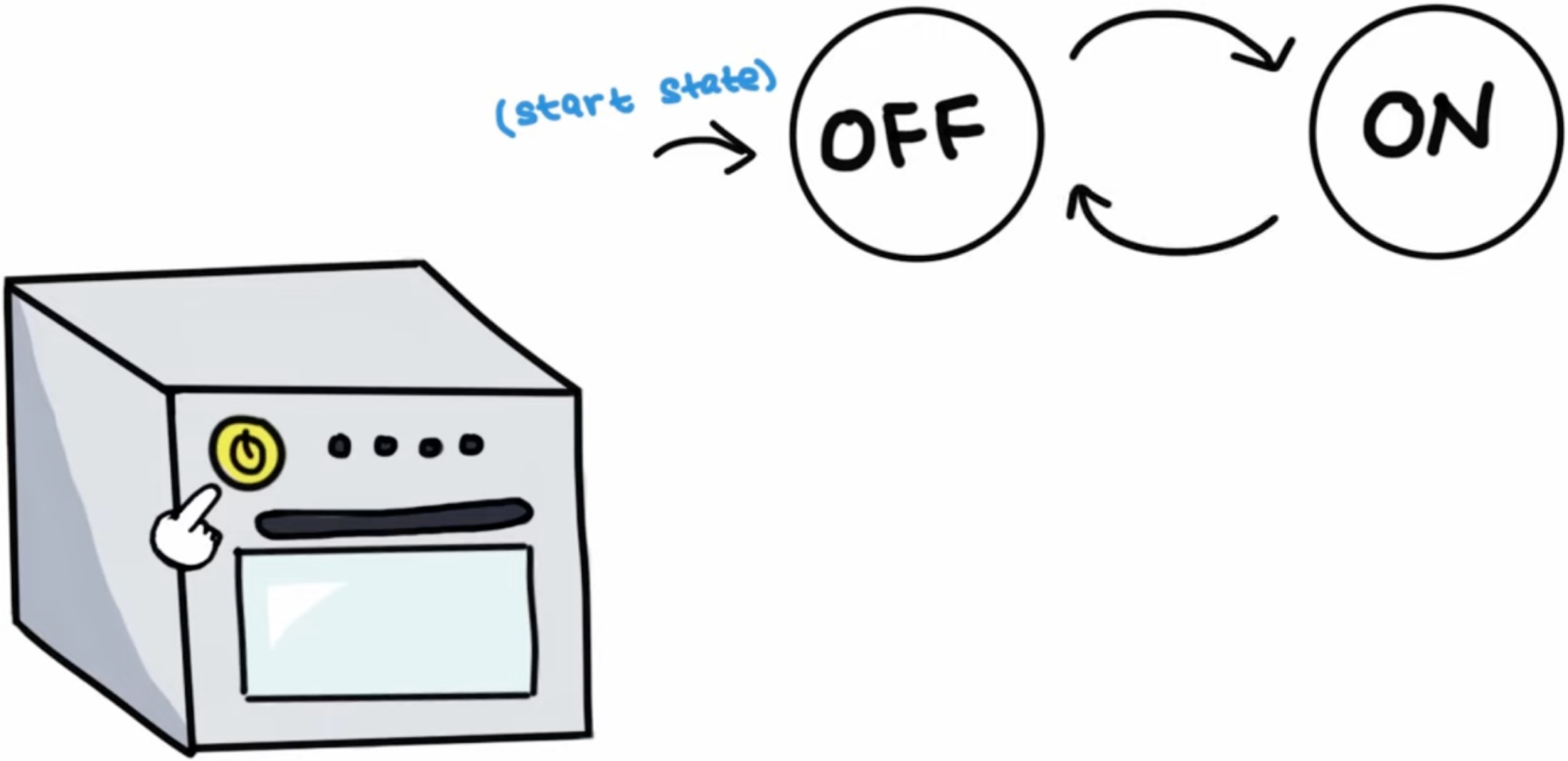
Introduction: Understanding of DFA



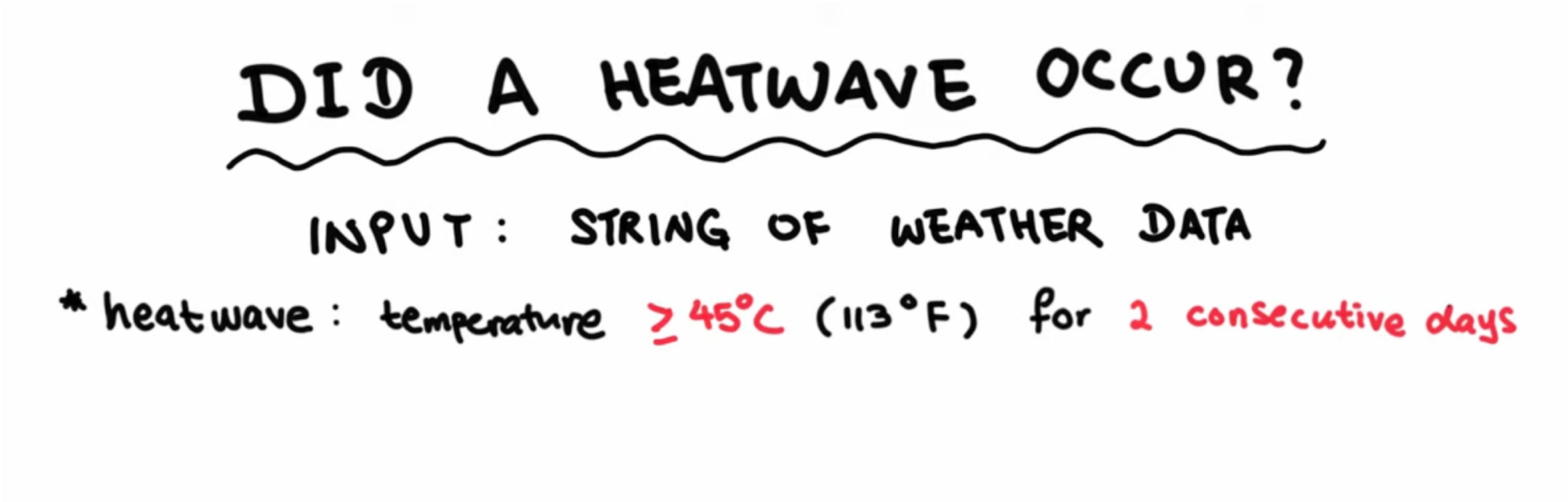
Introduction: Understanding of DFA



Introduction: Understanding of DFA



Introduction: Understanding of DFA



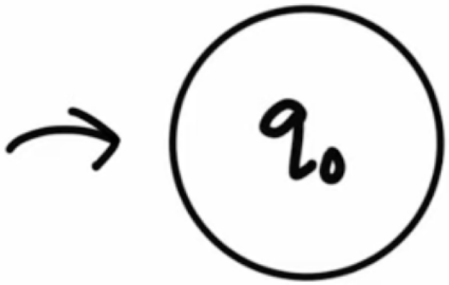
Introduction: Understanding of DFA

innovate achieve lead

DID A HEATWAVE OCCUR?

INPUT : STRING OF WEATHER DATA

* heatwave : temperature $\geq 45°C$ (113°F) for 2 consecutive days



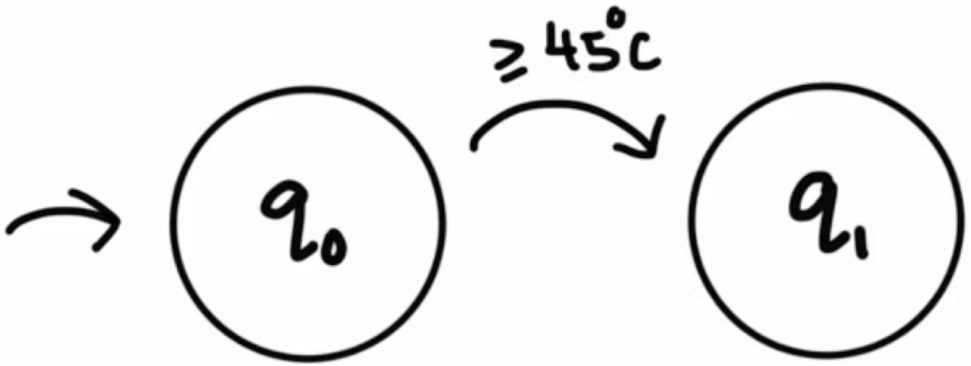
Introduction: Understanding of DFA

innovate achieve lead

DID A HEATWAVE OCCUR?

INPUT : STRING OF WEATHER DATA

* heatwave : temperature $\geq 45^{\circ}C$ ($113^{\circ}F$) for 2 consecutive days



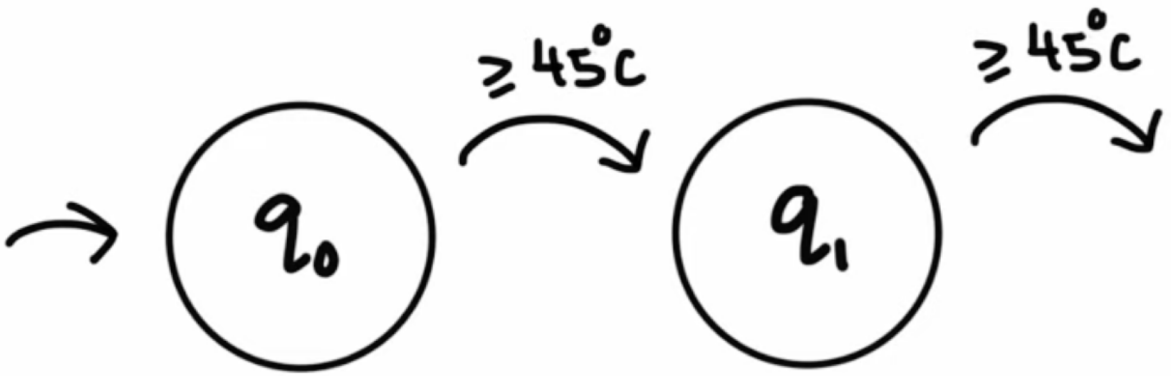
Introduction: Understanding of DFA

innovate achieve lead

DID A HEATWAVE OCCUR?

INPUT : STRING OF WEATHER DATA

* heatwave : temperature $\geq 45^{\circ}C$ (113°F) for 2 consecutive days



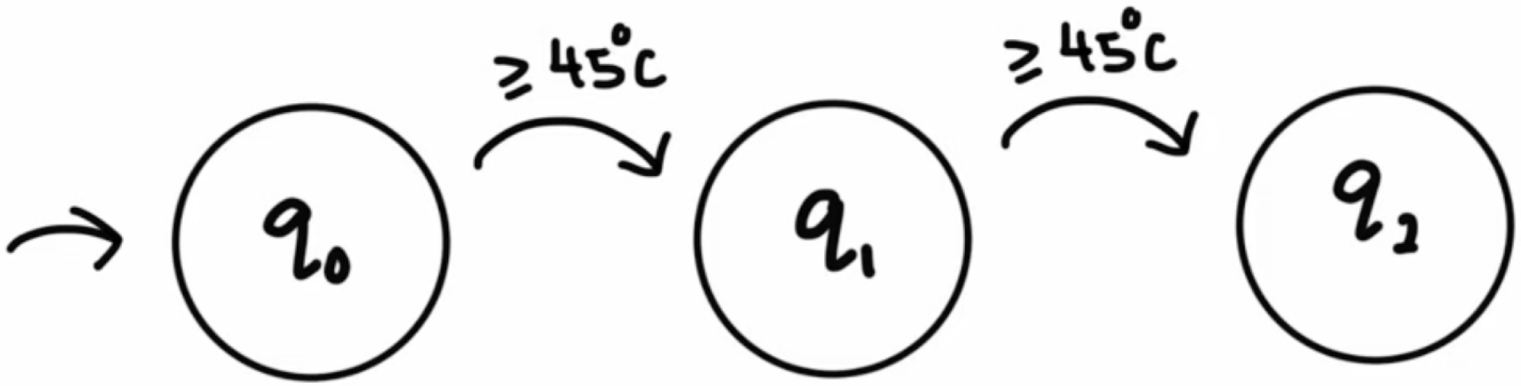
Introduction: Understanding of DFA

innovate achieve lead

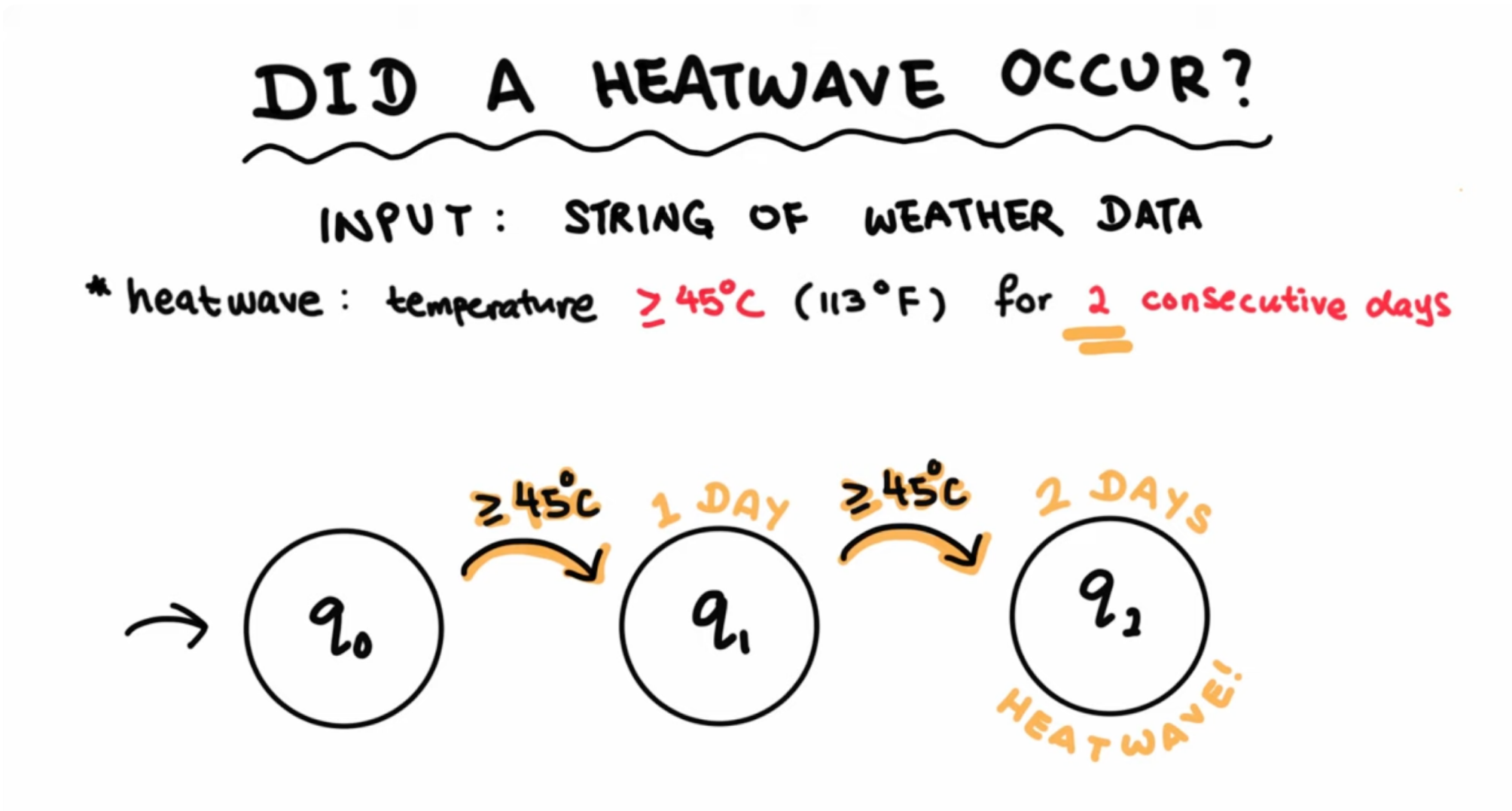
DID A HEATWAVE OCCUR?

INPUT : STRING OF WEATHER DATA

* heatwave : temperature $\geq 45^{\circ}C$ ($113^{\circ}F$) for 2 consecutive days



Introduction: Understanding of DFA



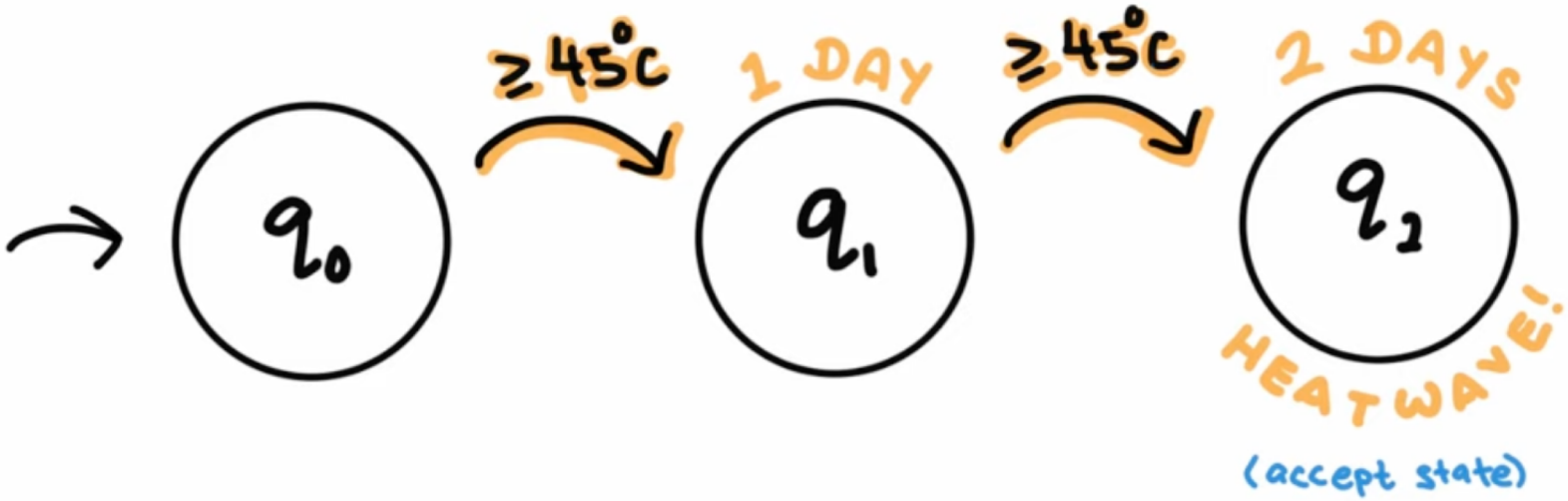
Introduction: Understanding of DFA

innovate achieve lead

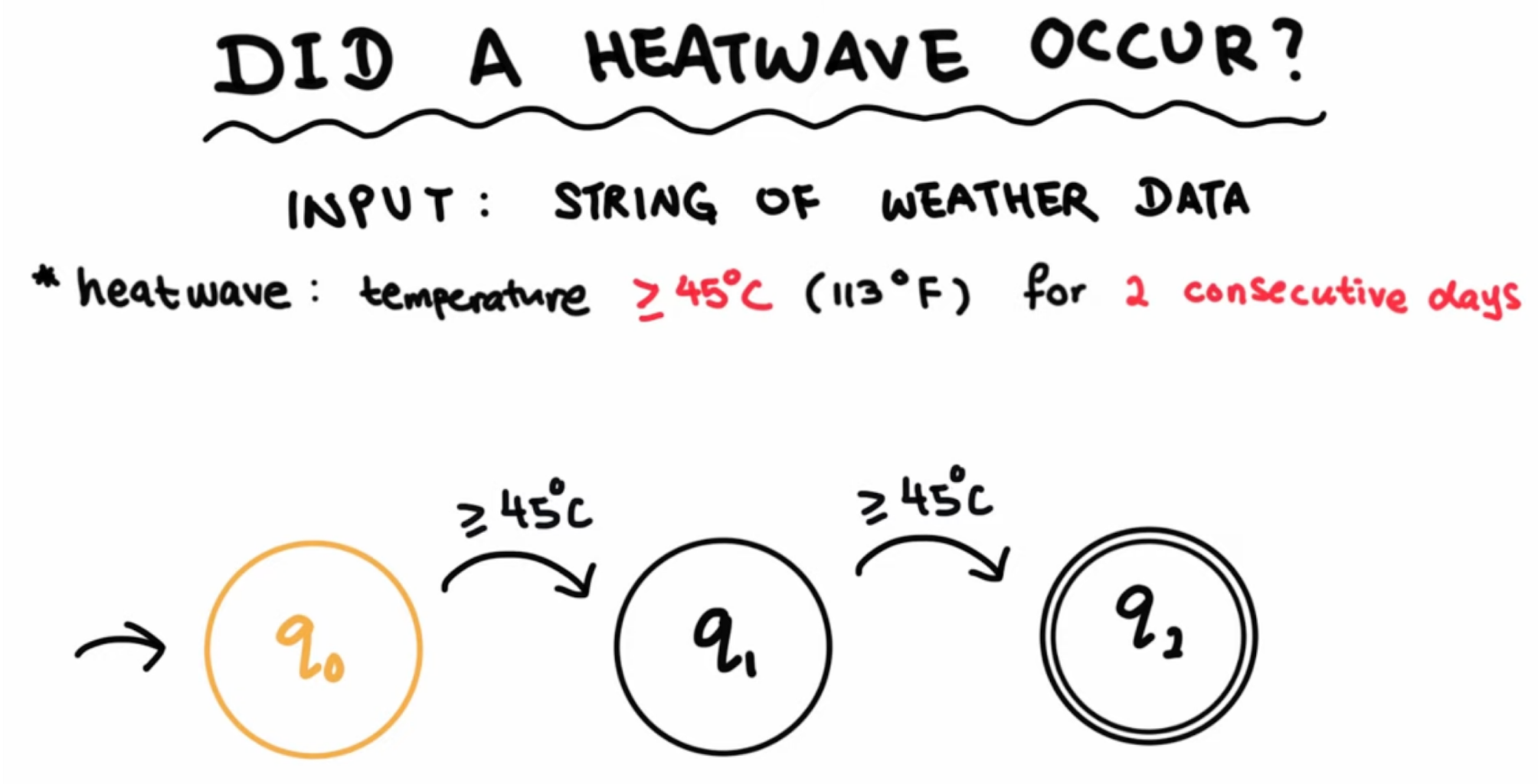
DID A HEATWAVE OCCUR?

INPUT : STRING OF WEATHER DATA

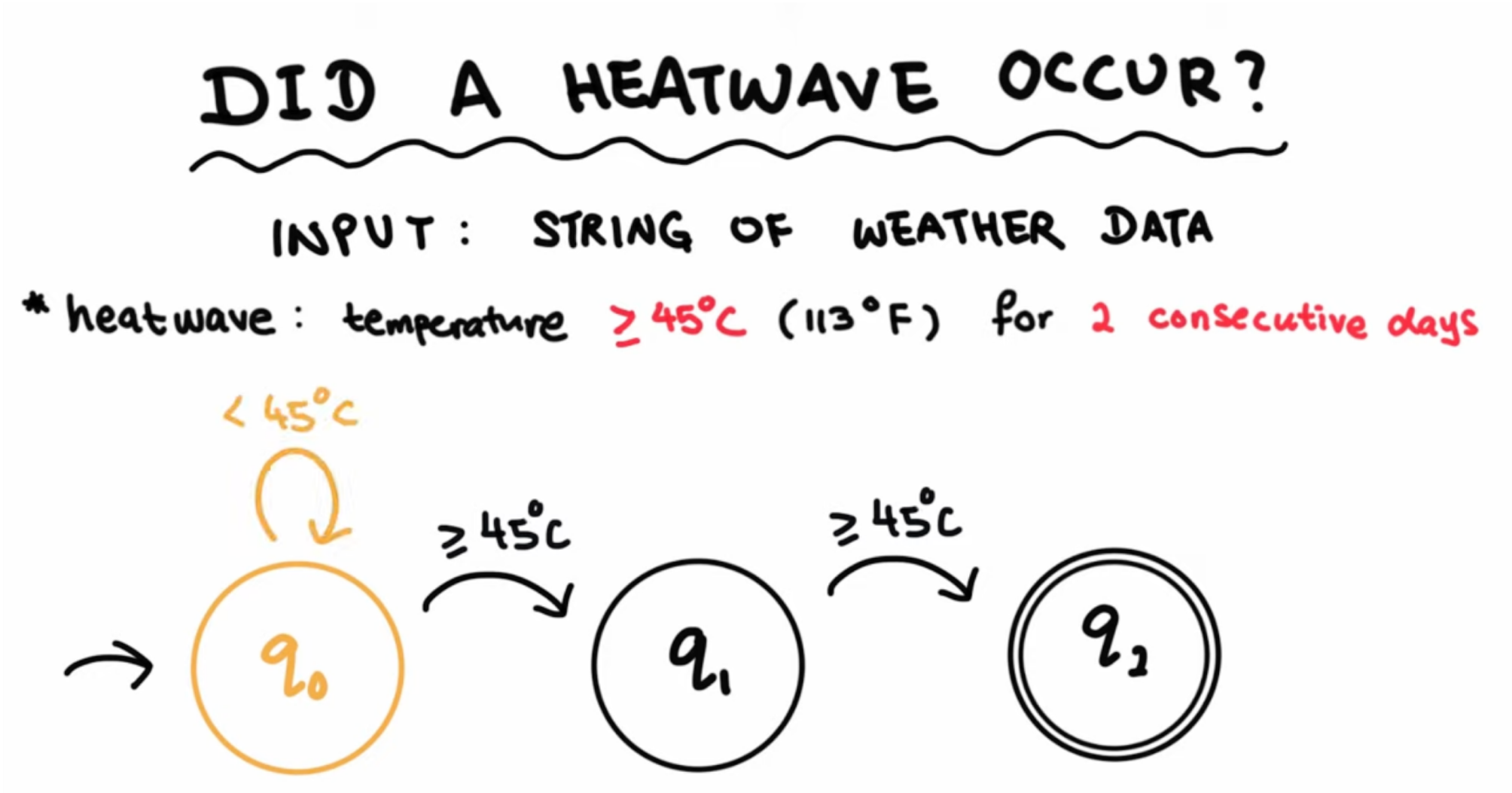
* heatwave : temperature $\geq 45^{\circ}C$ ($113^{\circ}F$) for 2 consecutive days



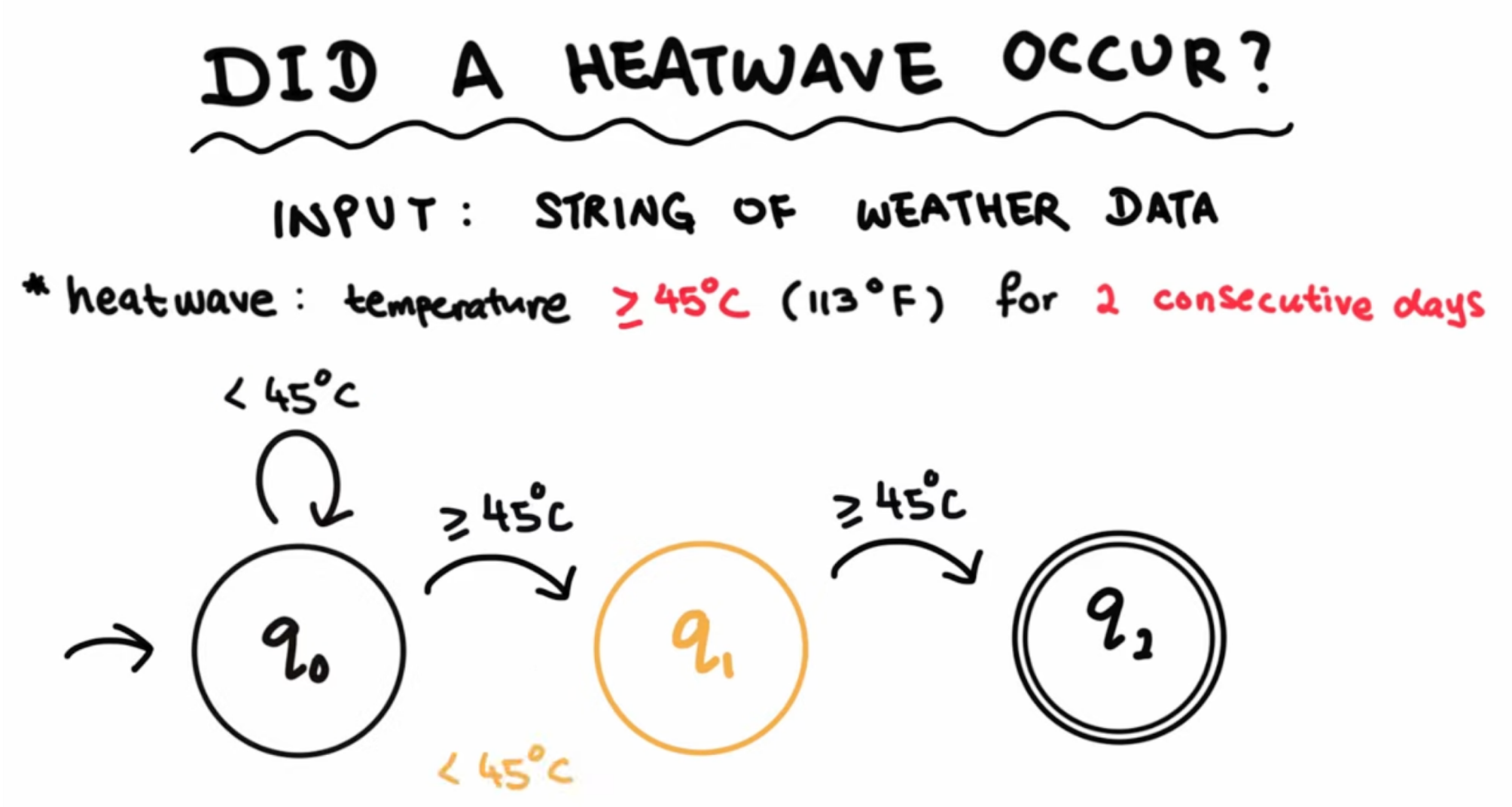
Introduction: Understanding of DFA



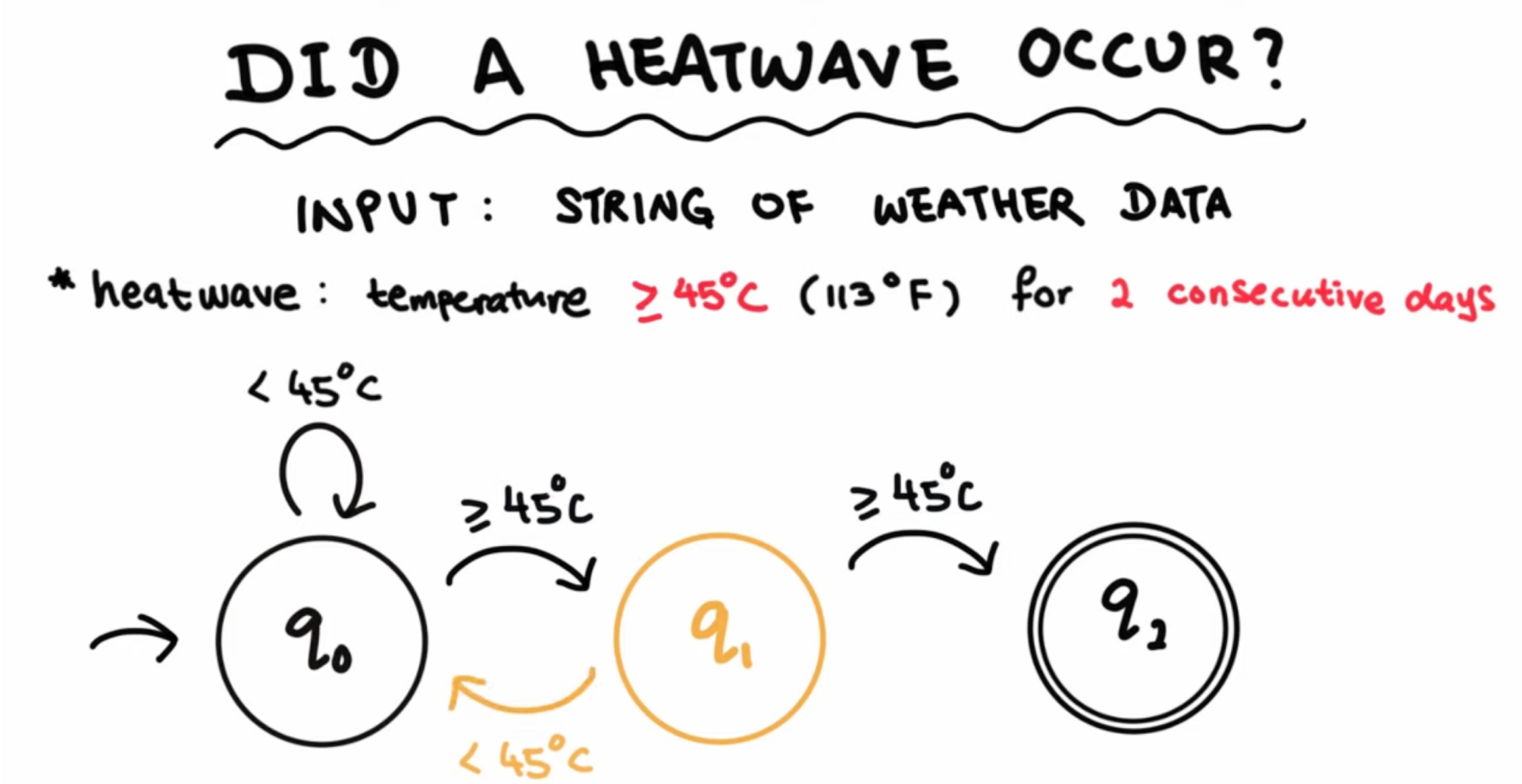
Introduction: Understanding of DFA



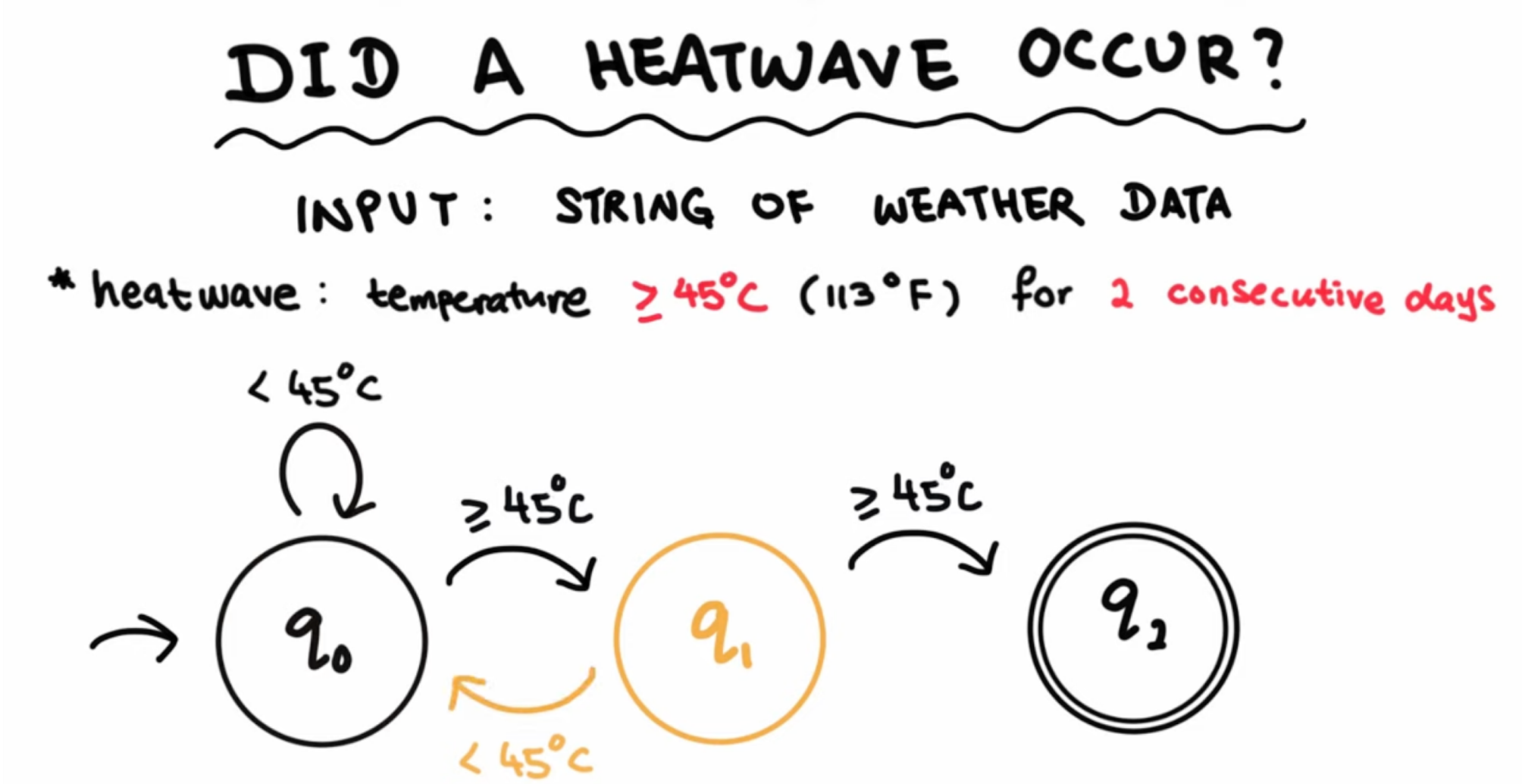
Introduction: Understanding of DFA



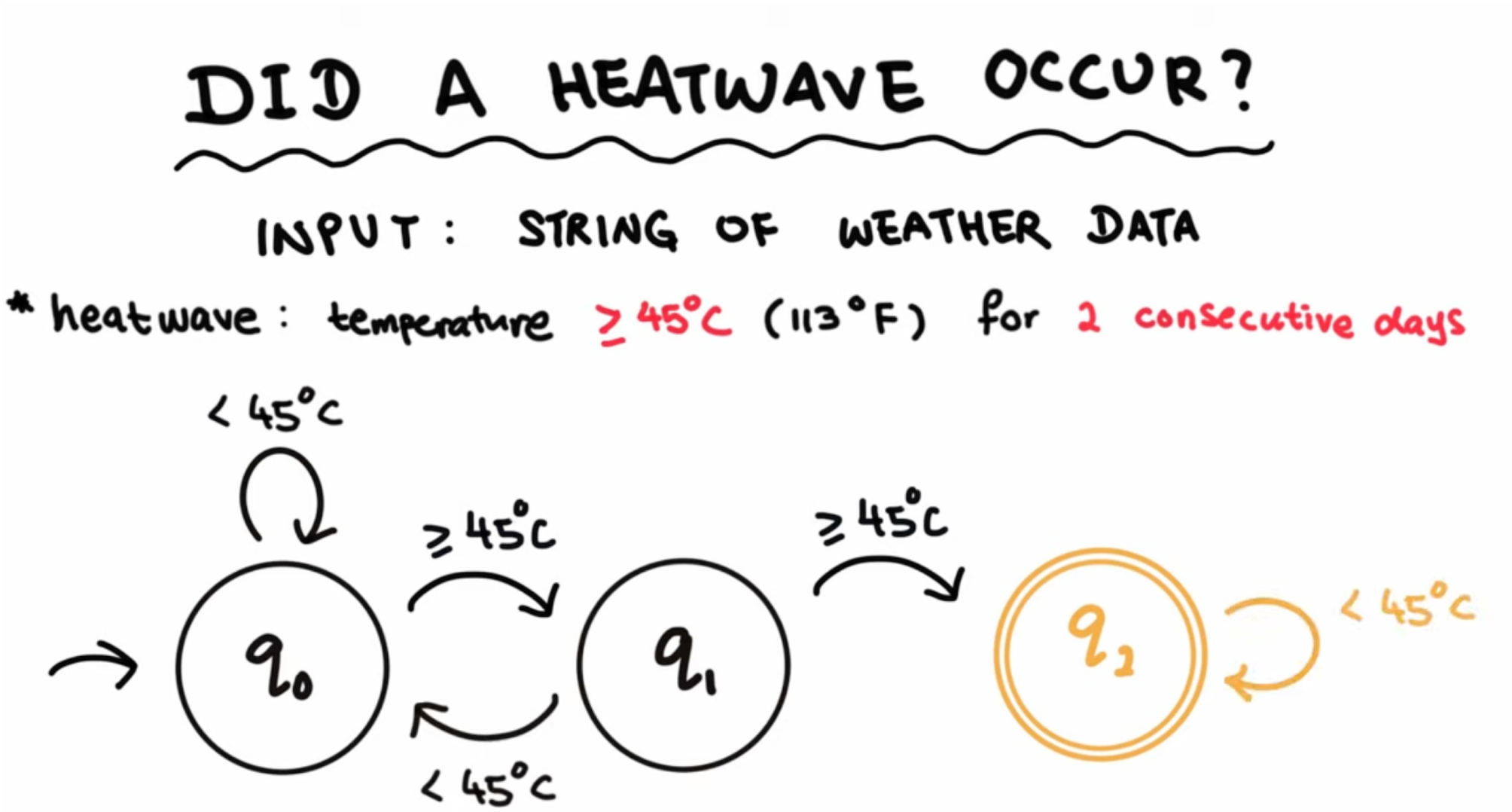
Introduction: Understanding of DFA



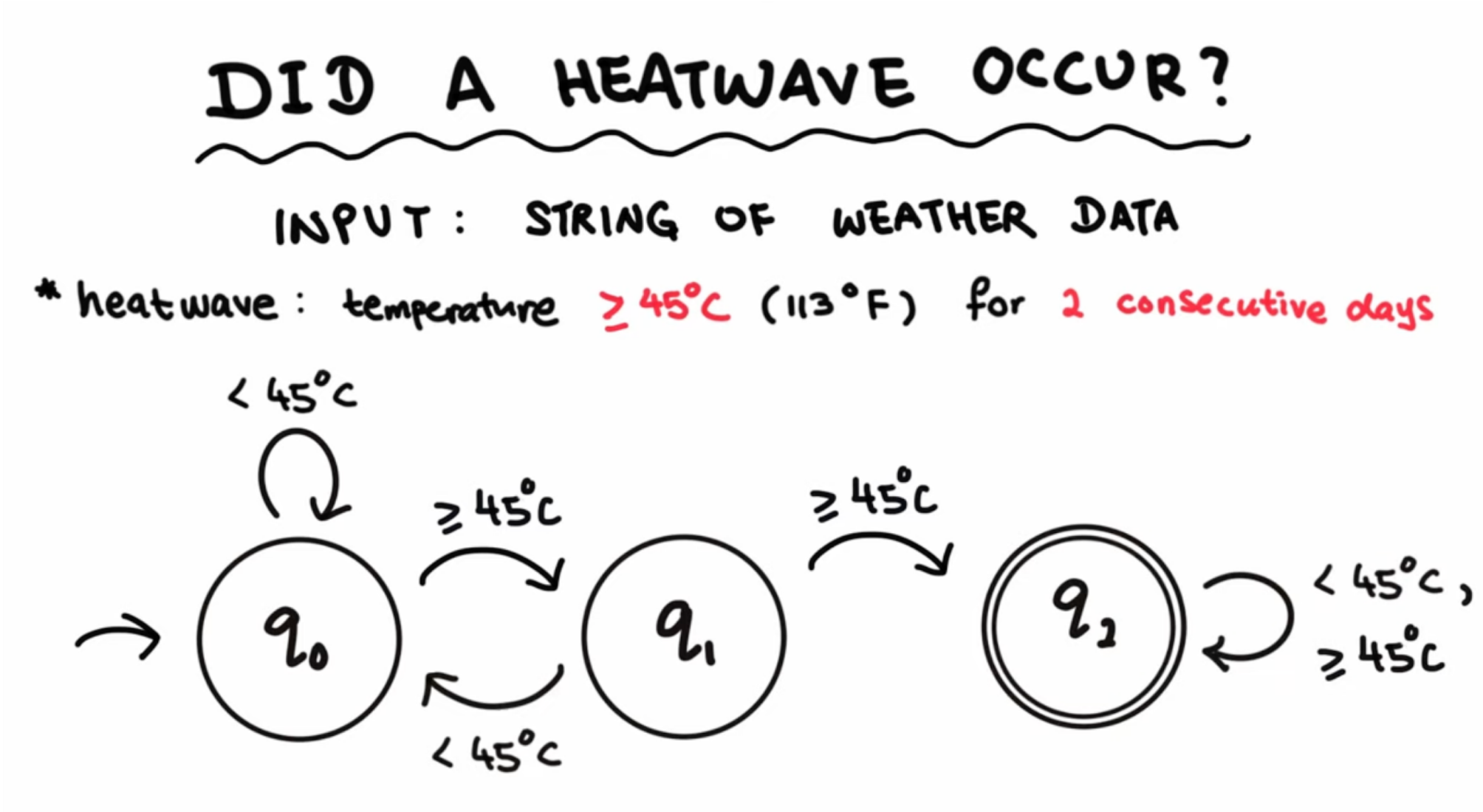
Introduction: Understanding of DFA



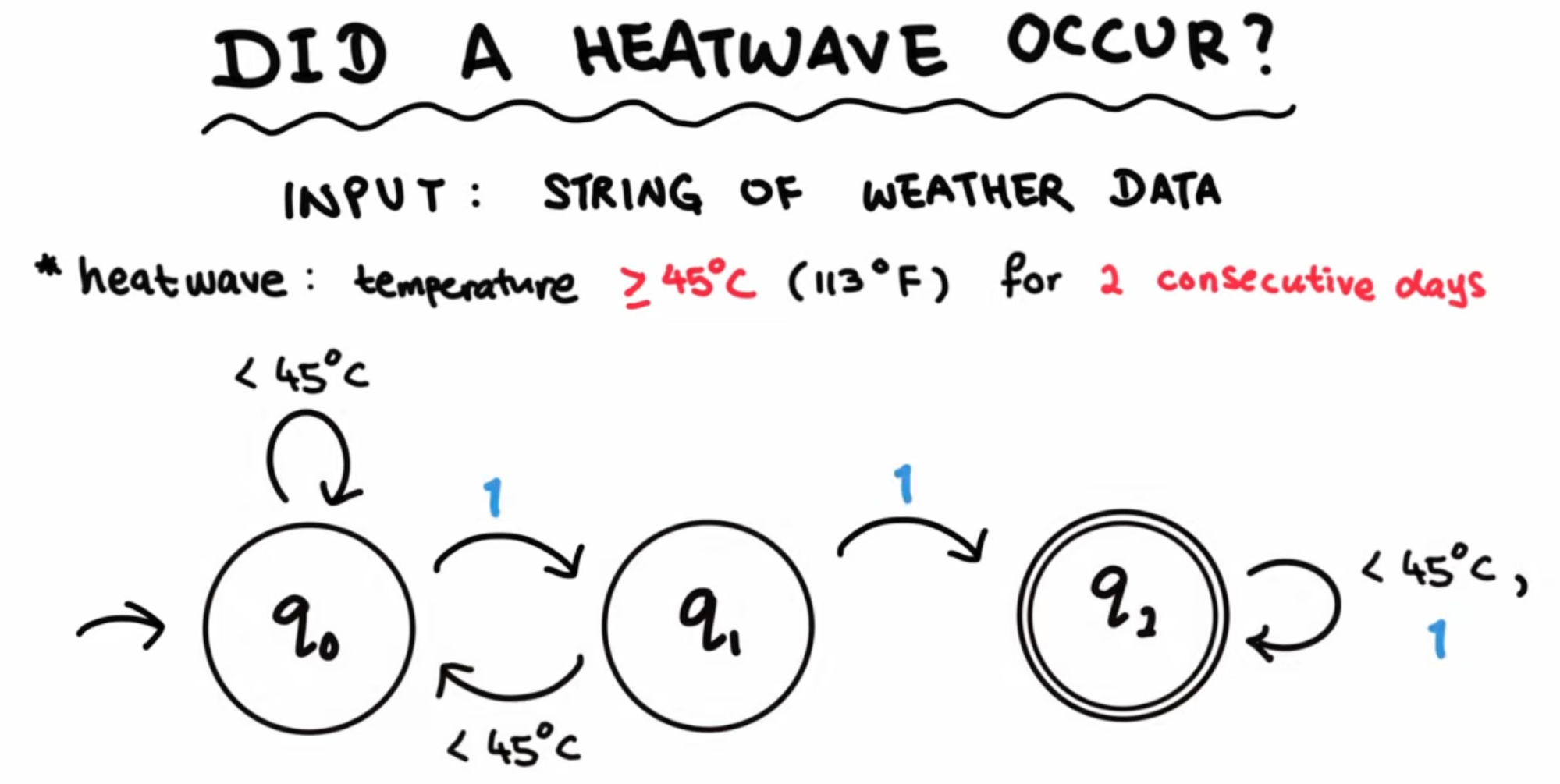
Introduction: Understanding of DFA



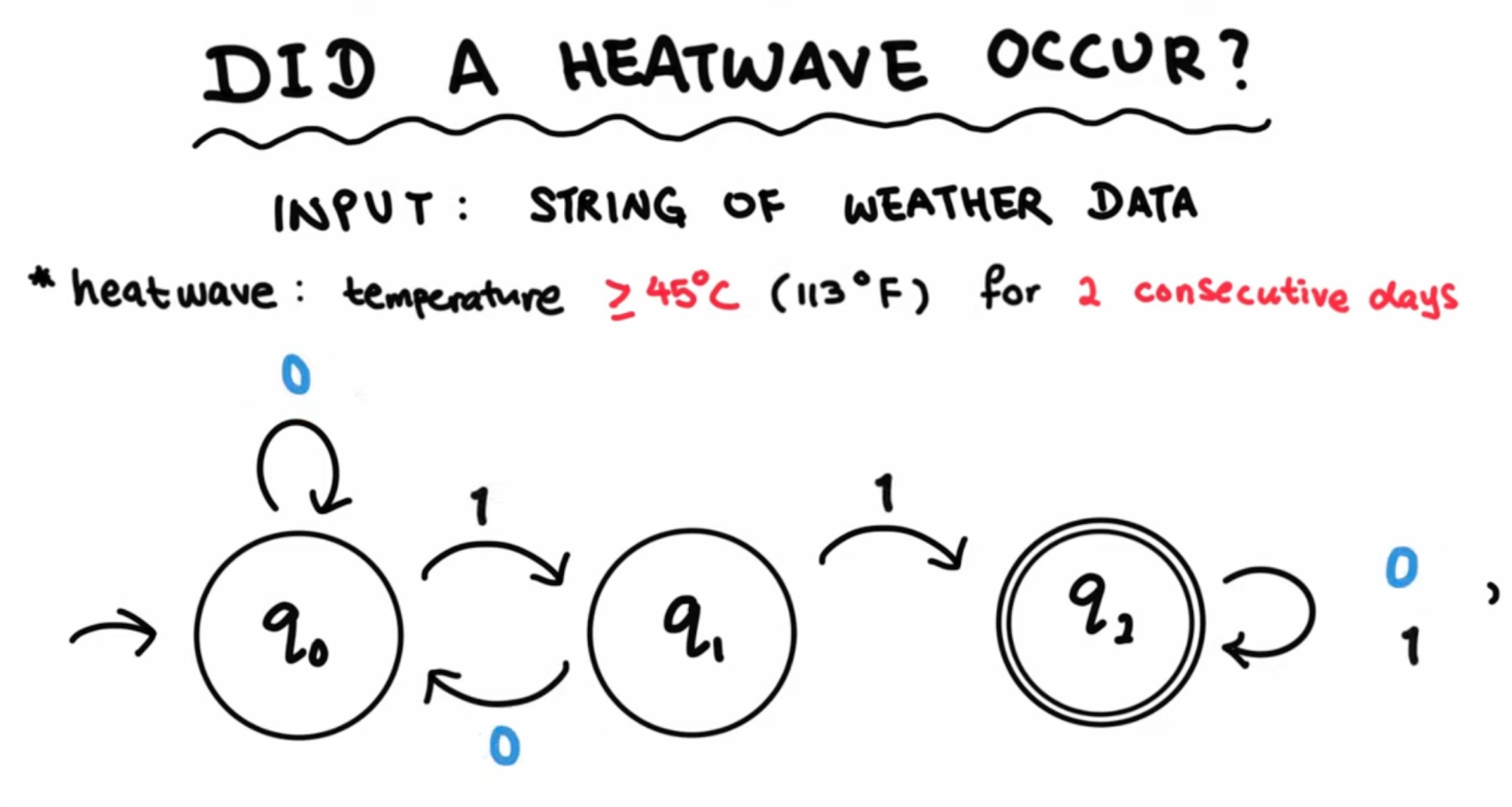
Introduction: Understanding of DFA



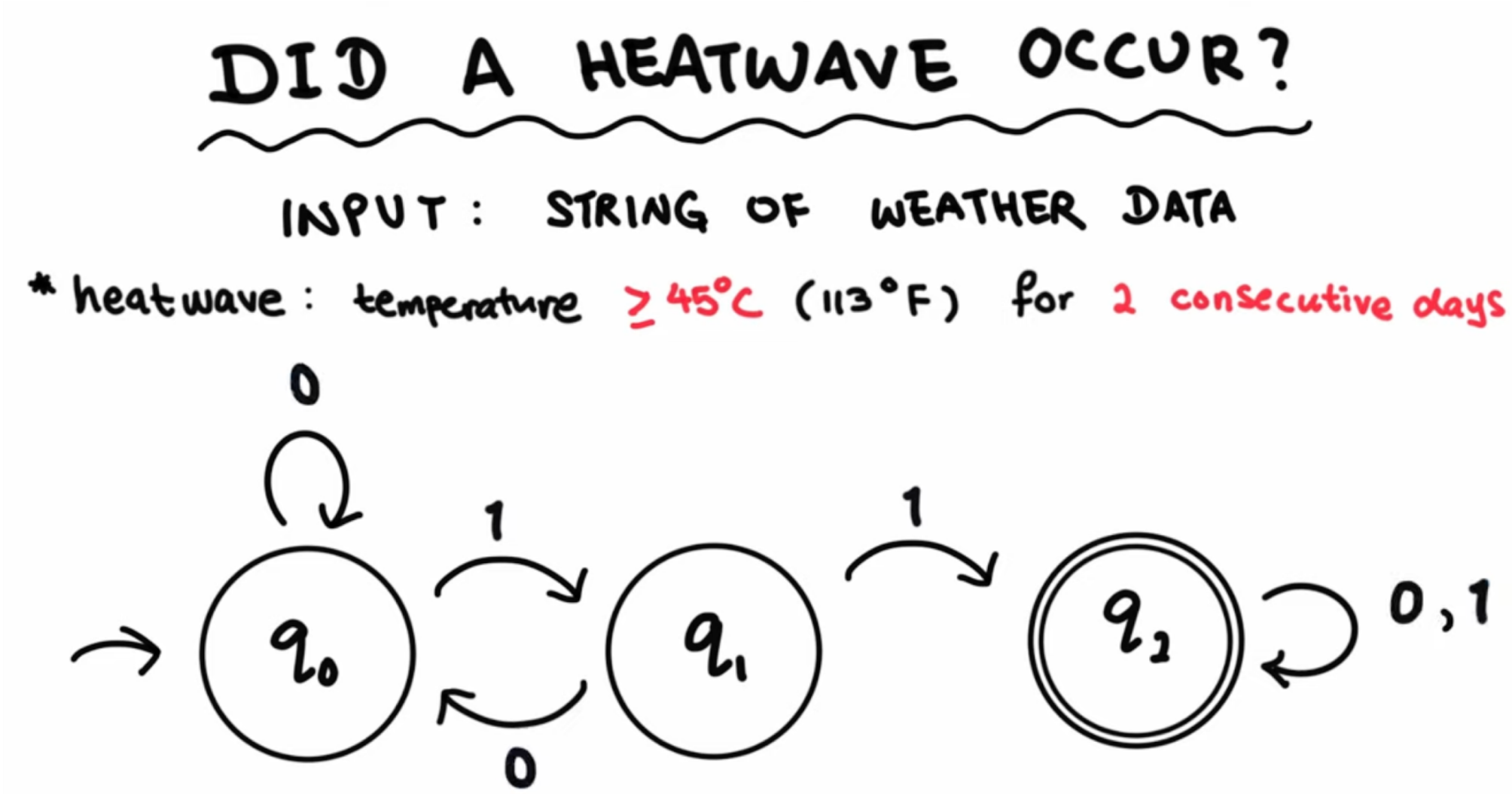
Introduction: Understanding of DFA



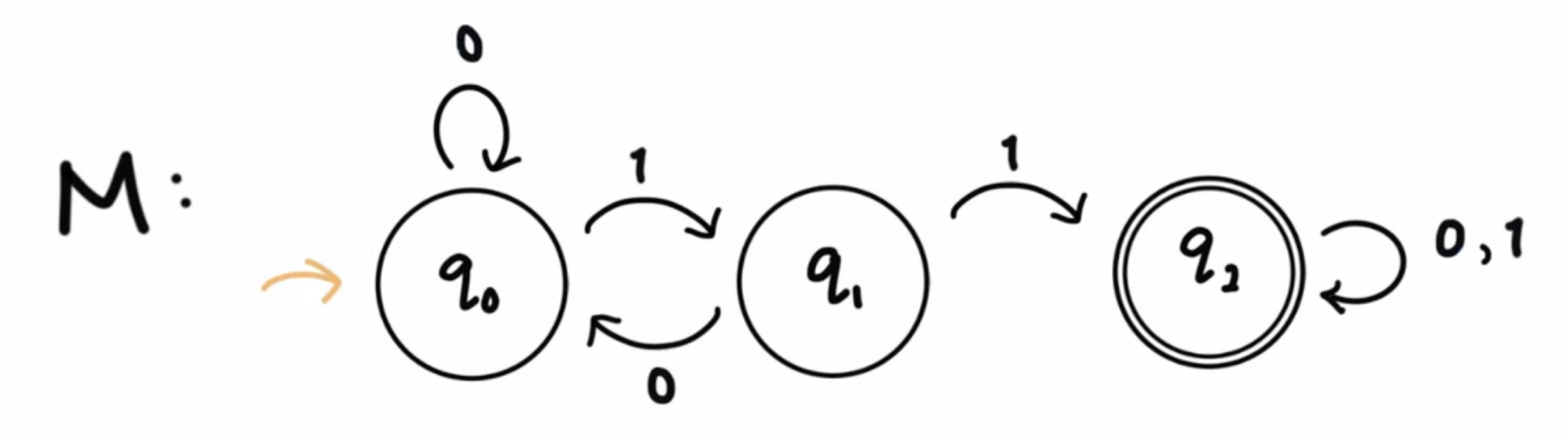
Introduction: Understanding of DFA



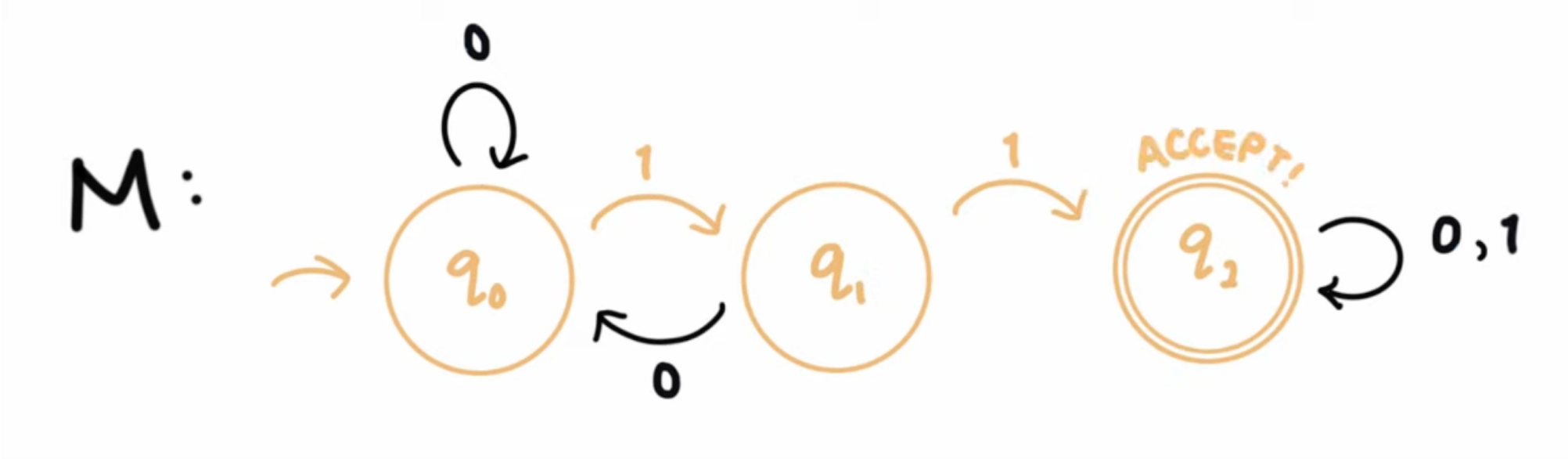
Introduction: Understanding of DFA



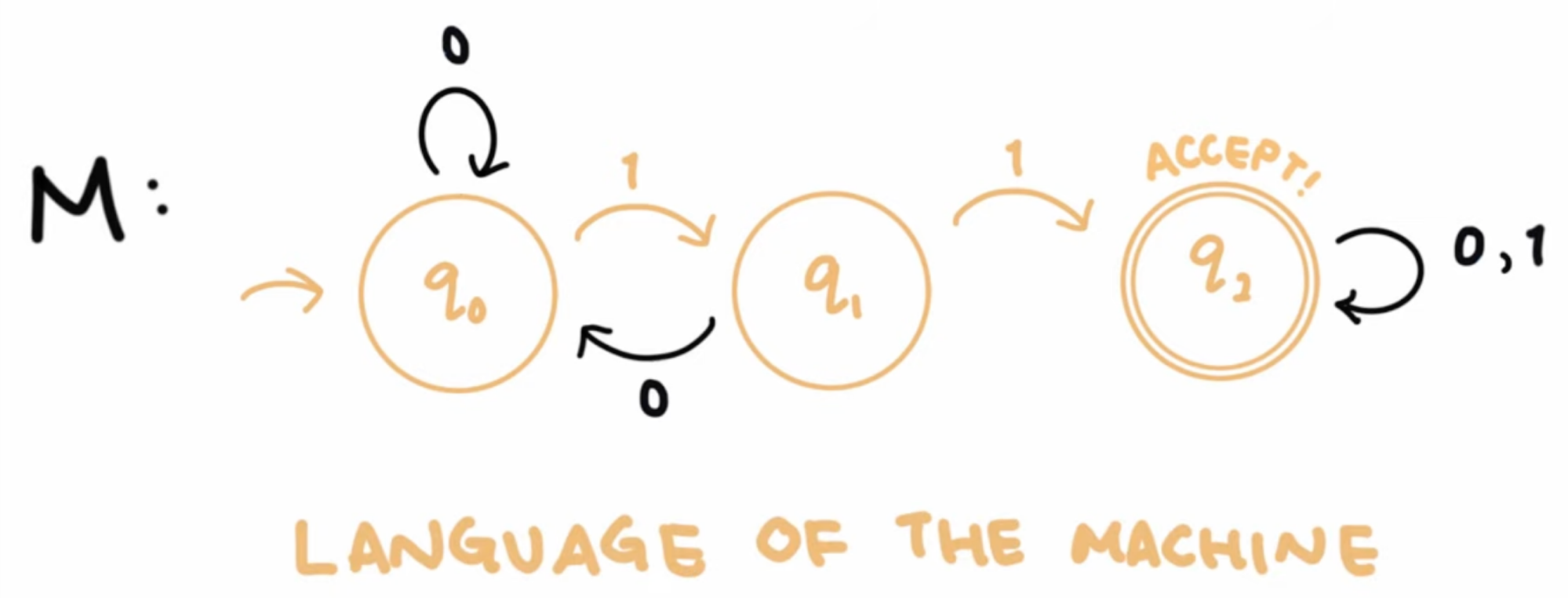
Introduction: Understanding of DFA



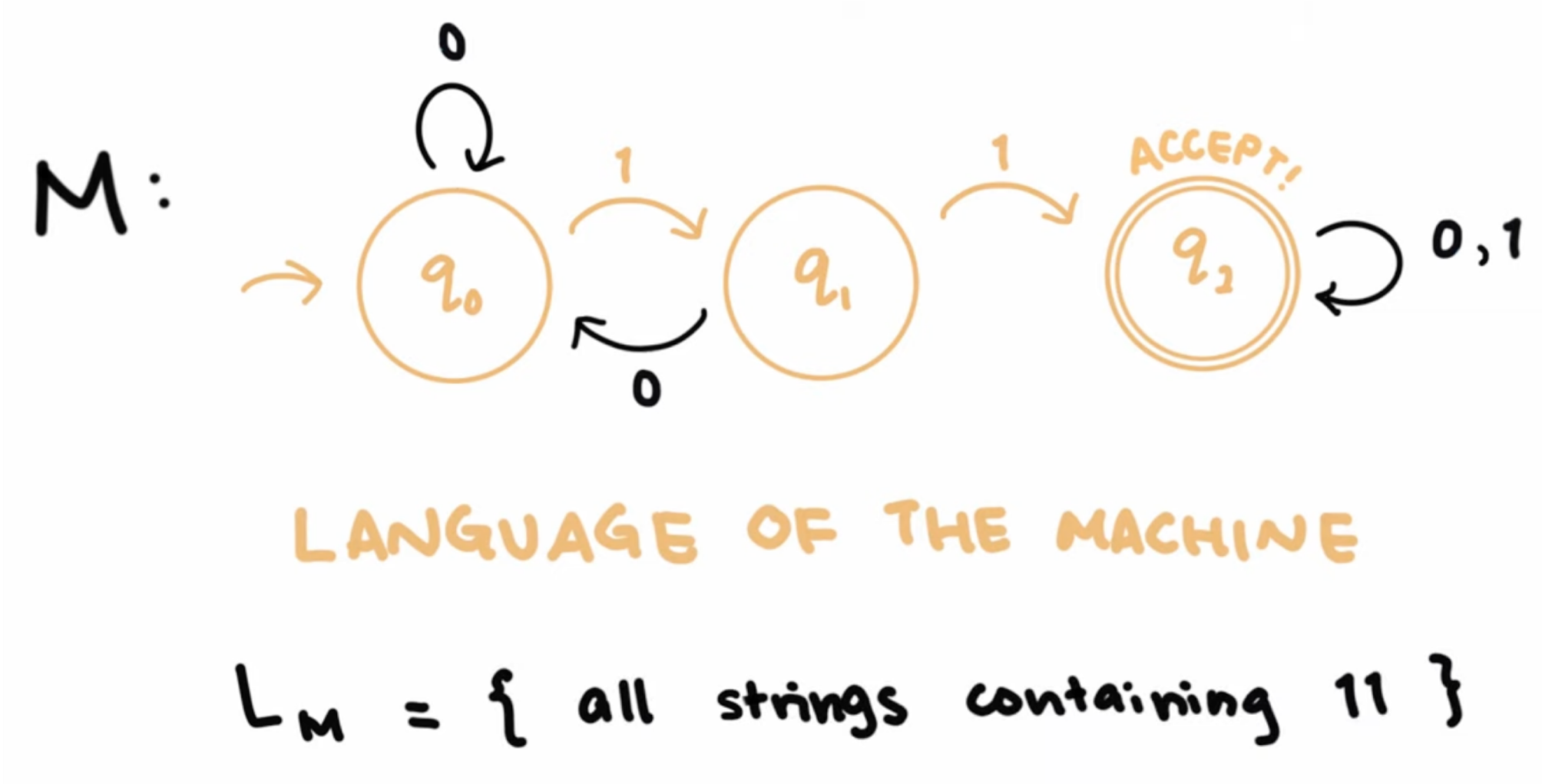
Introduction: Understanding of DFA



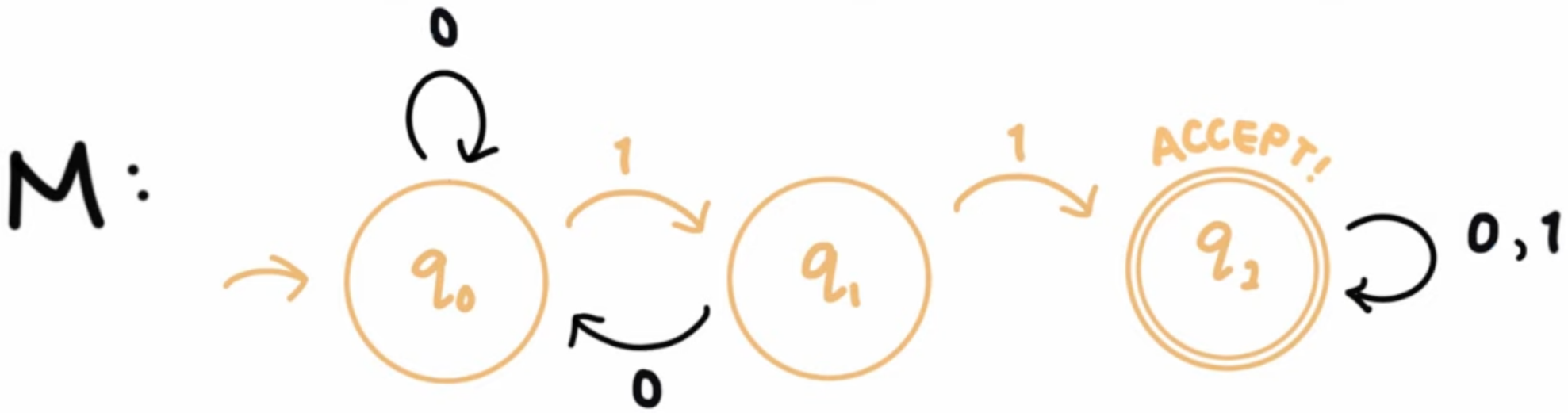
Introduction: Understanding of DFA



Introduction: Understanding of DFA



Introduction: Understanding of DFA

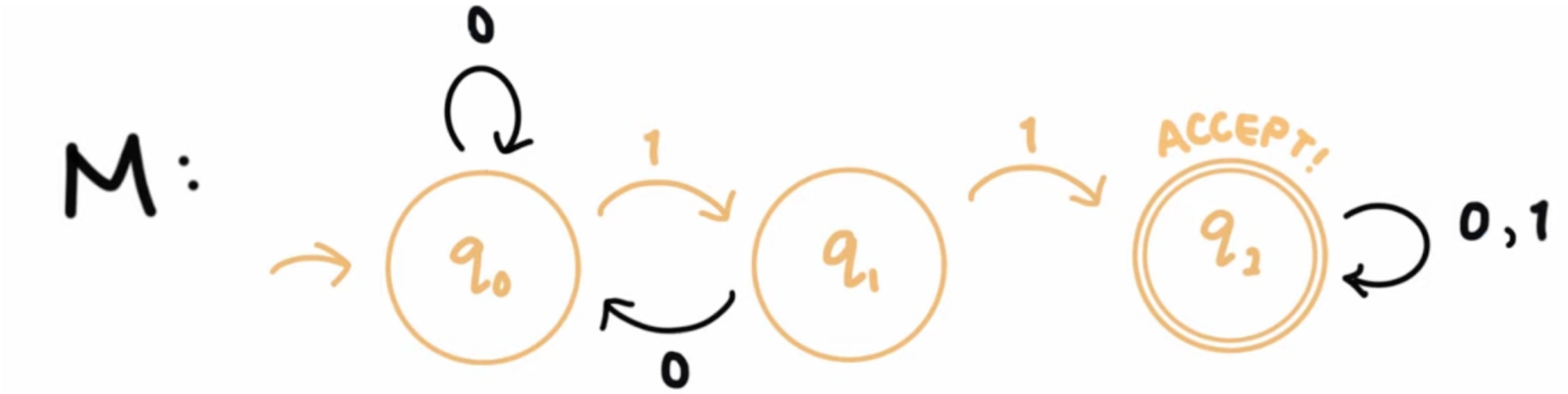


LANGUAGE OF THE MACHINE

L_M = { all strings containing 11 }

M recognizes L_M

Introduction: Understanding of DFA



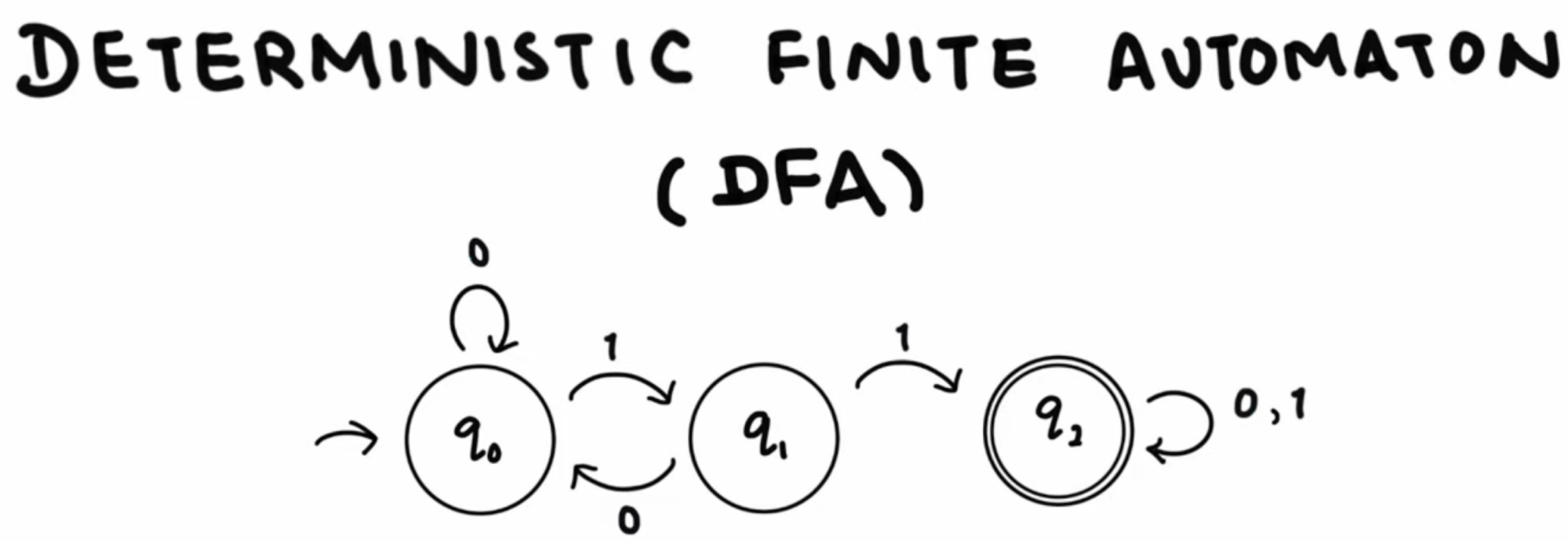
L_M = { all strings containing 11 }

M accepts 11

M accepts 110

M accepts 0110

Introduction: Understanding of DFA



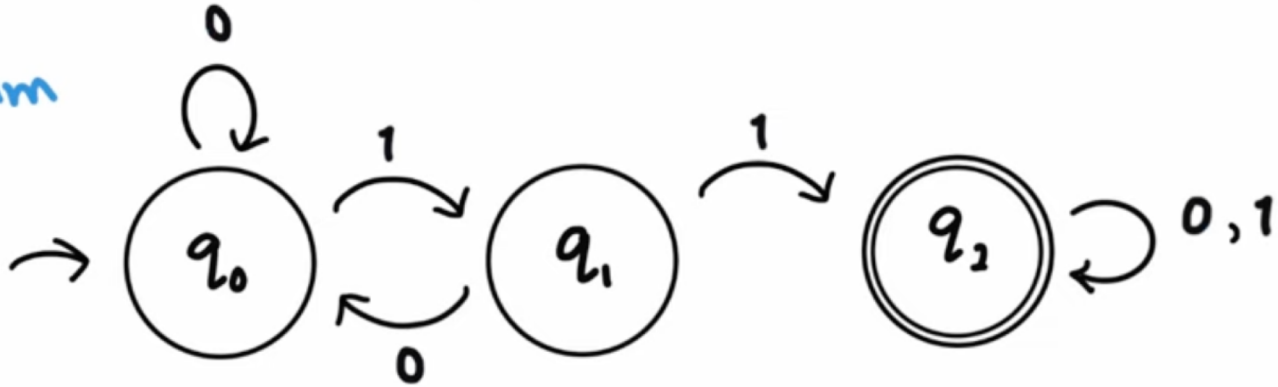
Introduction: Understanding of DFA

innovate achieve lead

DETERMINISTIC FINITE AUTOMATON

(DFA)

State diagram

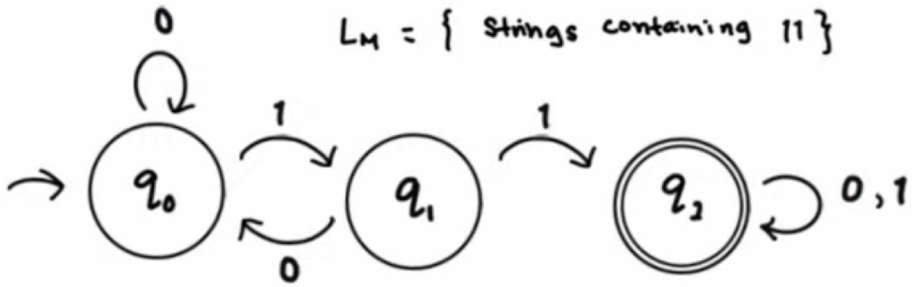


Formal definition

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

Introduction: Understanding of DFA

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

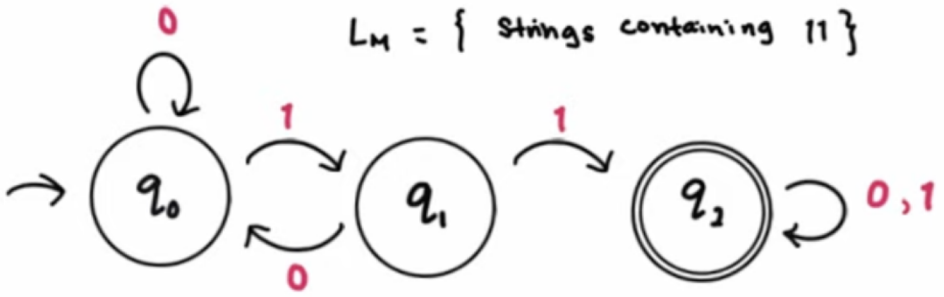


Q is the set of states

$Q = \{ q_0, q_1, q_2 \}$

Introduction: Understanding of DFA

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



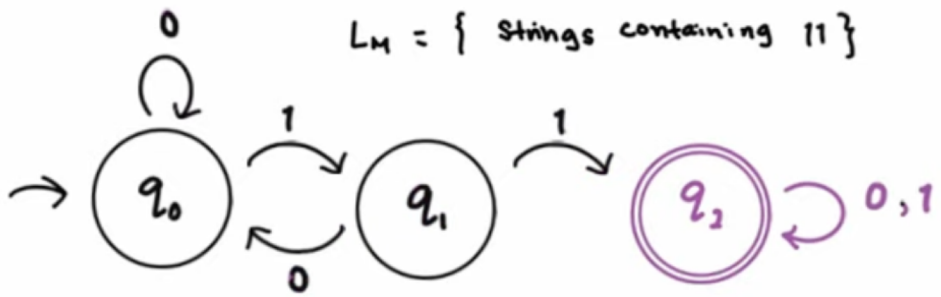
Q is the set of states

$Q = \{ q_0, q_1, q_2 \}$

Σ is the alphabet

$\Sigma = \{1, 0\}$

Introduction: Understanding of DFA



L_M = { Strings containing 11 }

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

Q is the set of states

Σ is the alphabet

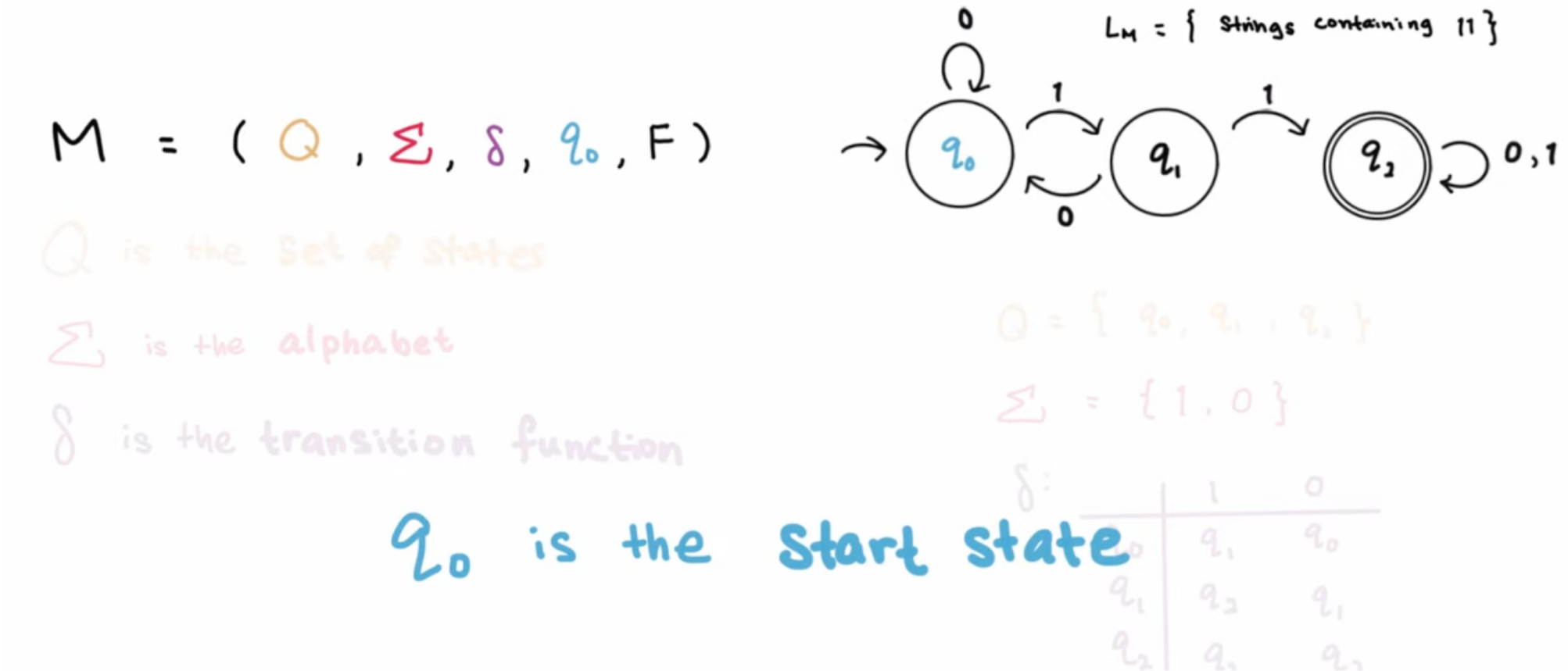
δ is the transition function

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{1, 0\}$

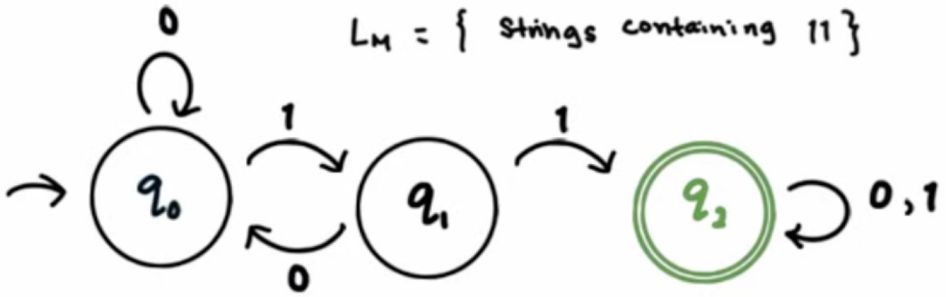
	1	0
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_2	q_2

Introduction: Understanding of DFA



Introduction: Understanding of DFA

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



F is the set of accept/final states

$F = \{ q_2 \}$

Introduction: Understanding of DFA

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

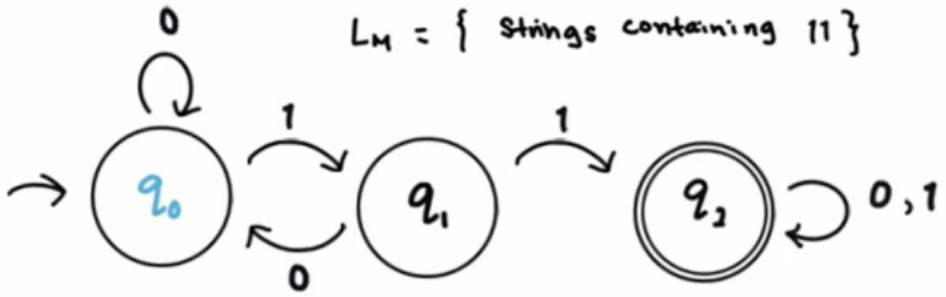
Q is the set of states

Σ is the alphabet

δ is the transition function

q_0 is the start state

F is the set of accept/final states



$Q = \{ q_0, q_1, q_2 \}$

$\Sigma = \{ 1, 0 \}$

δ:

	1	0
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_2	q_2

$F = \{ q_2 \}$

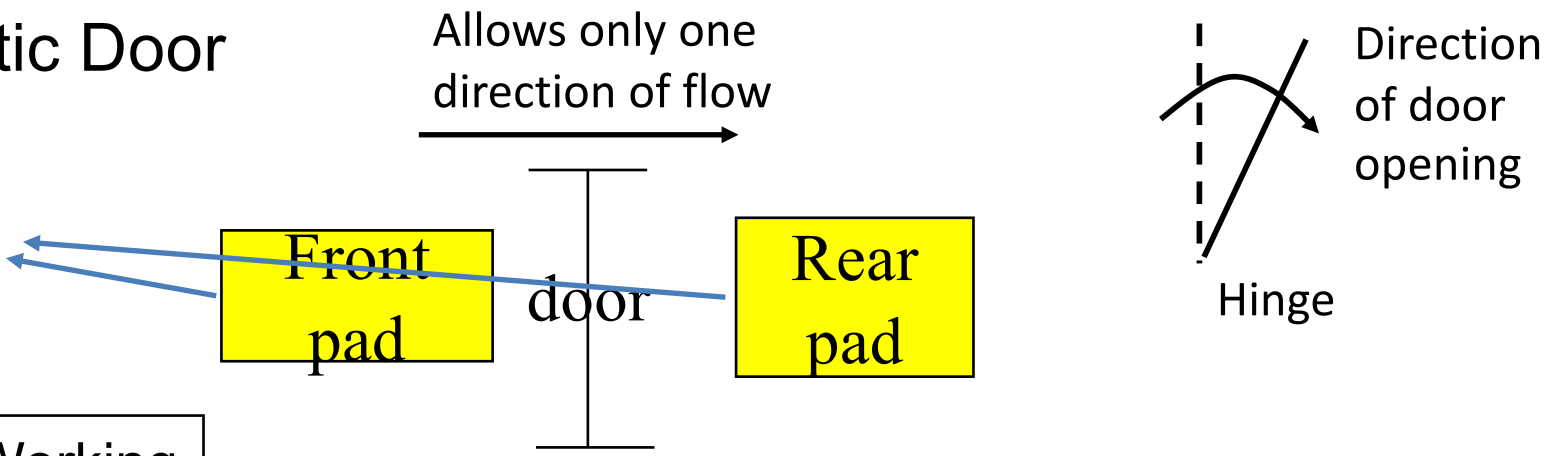
Deterministic Finite Automata

- Motiving Example
- DFA Definitions, Computation, Examples
- Regular Languages ⇔DFA
- Operations on Regular Languages
- Closure Properties of Regular Languages.

Formal Language, chapter 1, slide 5

Motivation for FA (Deterministic)

Eg. Automatic Door



- Expected Working

- Opens when someone approaches from the front
- **Holds** open until person clears (crosses)
- Doesn't open when someone is standing behind the door

- Subtle observations

- What should happen if the door is open and some one is at the rear pad?
  - Its should not close (for safety reasons) as it would hit the person.
- What should happen if the door is closed and some body is there at both front and rear?

Motivation for FA (Deterministic)

Eg. Automatic Door

- Expected Working

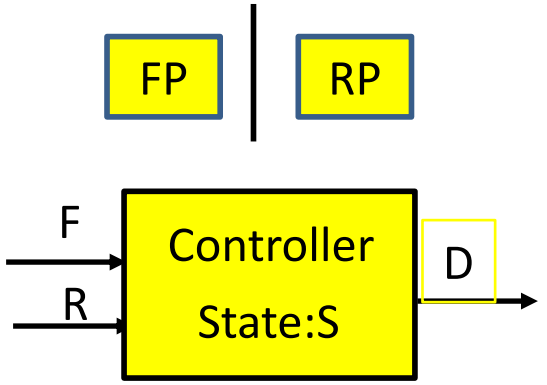
- Opens when someone approaches from the front
- **Holds** open until person clears (crosses)
- Doesn't open when someone is standing behind the door
- What should happen if the door is open and some one is at the rear pad?
  - Its should not close (for safety reasons) as it would hit the person.
- What should happen if the door is closed and some body is there at both front and rear?

- We need to "**remember**" whether the door was open/closed..
- Eg.
  - if the door were closed and some one is at the rear, then it should remain closed even if some one arrived at the front (Why?? Safety concerns)
  - if the door were open and some one is at the rear, then it should remain opened. (Why??)

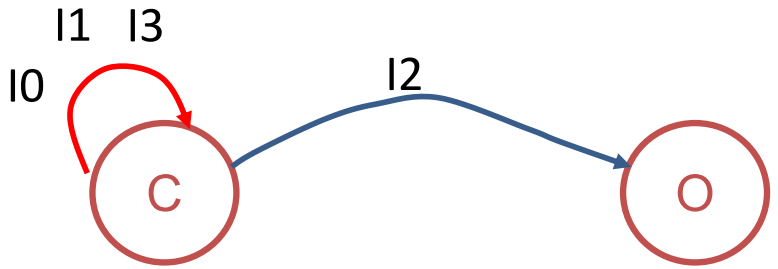
- What happens **next** (open/close) is dependent on whether the door is **currently opened / closed and current inputs**

Motivation for FA (Deterministic)

- The door has 2 sensors (Front (F), Rear (R))
- 4 possible input combinations.
  - I0: F = 0: R = 0; no one on either pad
  - I1: F = 0; R = 1; someone on the REAR pad
  - I2: F = 1; R = 0; some is on the FRONT pad
  - I3: F = 1; R = 1; someone on both pads



- "Remember" whether the door is open or closed -> **State** memorises

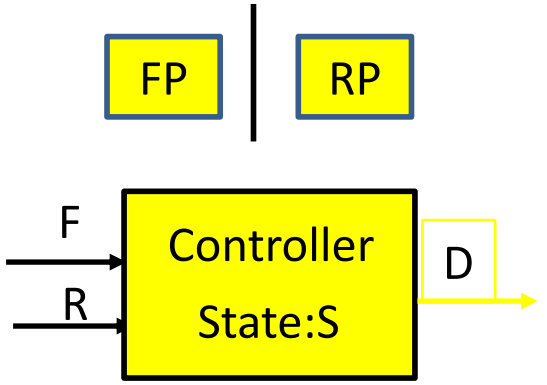


- Current State is **Closed**, What should the next state be

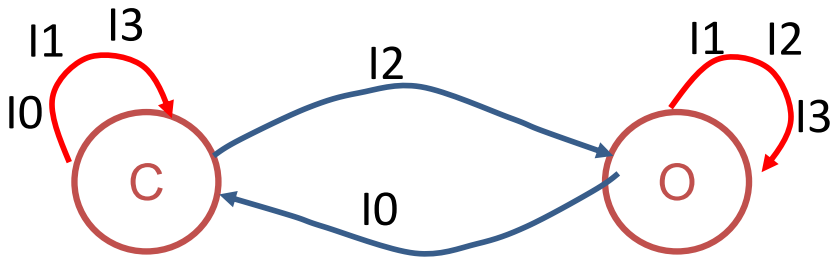
•I0 : no one on both pads,	•so remain closed, (i.e next state = C)
•I1 : some one on rear pad,	•so remain closed, (why ?) (i.e next state = C)
•I2 : some one on front pad	• open door (i.e next state = O)
•I3 : some one on both pads	•so remain closed, (why ?) (i.e next state = C)

Motivation for FA (Deterministic)

- The door has 2 sensors (Front (F), Rear (R))
- 4 possible input combinations.
  - I0: F = 0: R = 0; no one on either pad
  - I1: F = 0; R = 1; someone on the REAR pad
  - I2: F = 1; R = 0; some is on the FRONT pad
  - I3: F = 1; R = 1; someone on both pads



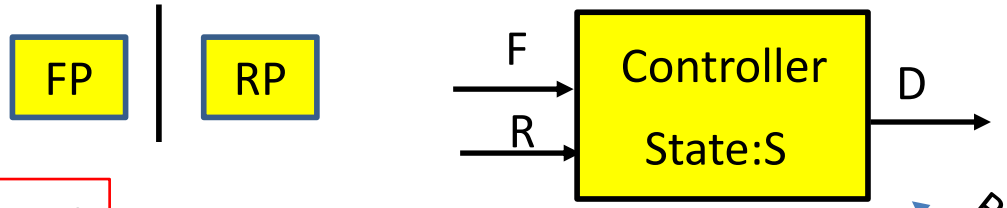
- "Remember" whether the door is open or closed -> **State** memorises

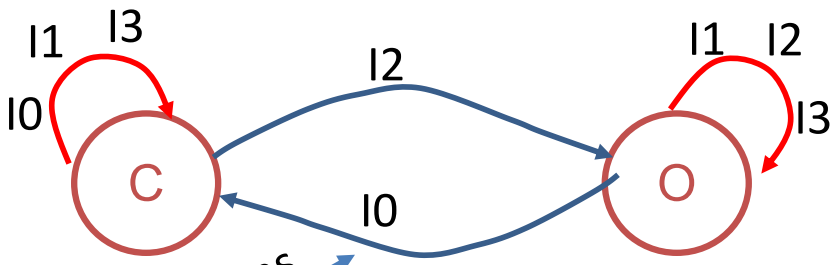


- Current State is **Open**, What should the next state be?
  - I0 : no one on both pads,
  - I1 : some one on rear pad,
  - I2 : some one on front pad
  - I3 : some one on both pads

  - Close the door, (i.e next state = C)
  - Remain open, (why ?) (i.e next state = O)
  - Remain open, (why ?) (i.e next state = O)
  - Remain open, (why ?) (i.e next state = C)

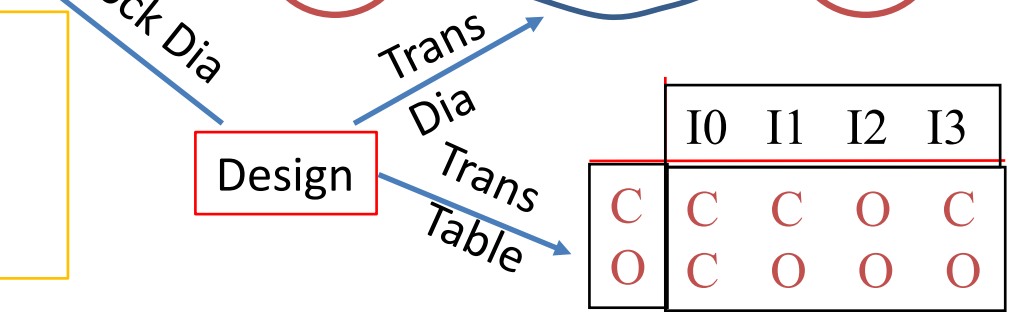
Motivation for FA (Deterministic)





Input

- I0: F = 0: R = 0; no one on either pad
- I1: F = 0; R = 1; someone on the REAR pad
- I2: F = 1; R = 0; some is on the FRONT pad
- I3: F = 1; R = 1; someone on both pads



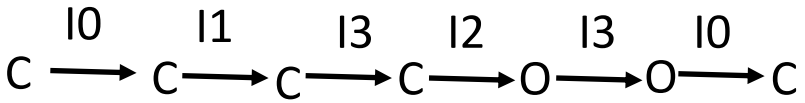
Implementation

```
def auto_door(cur_state, inp):
    trans = {  ('C','I0'):'C', ('C','I1'):'C', ('C', 'I2'):'O', ('C', 'I3'):'C',
               ('O','I0'):'C', ('O','I1'):'O', ('O', 'I2'):'O', ('O', 'I3'):'O'}

    next_state = trans[(cur_state,inp)]
    print ((cur_state,inp), next_state)
    return next_state

cur_state = 'C'
for i in ['I0', 'I1', 'I3', 'I2', 'I1', 'I0']:
    next_state = auto_door(cur_state, i)
    print(next_state, ' ')
    cur_state = next_state
```

Execution



Deterministic Finite Automata

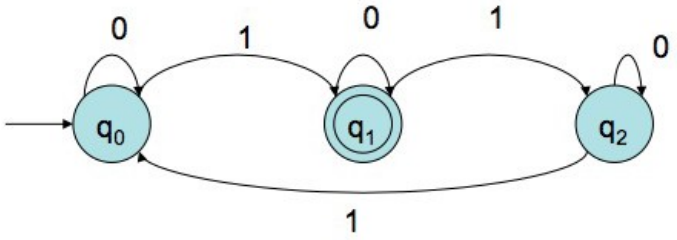
- A DFA starts in a start state and is presented with an input string.
- It moves from state to state, reading the input string one symbol at a time.
- Next state of the DFA (ie, the state the DFA moves next) depends on
  - the current state,
  - current input symbol

- When the last input symbol is read, the DFA decides whether it should accept the input string

- States are shown with circles. We usually have labels on the states.
  - One designated state is the start state, (State q_0 here).
  - States with double circles denote the accepting or final states (State q_1 here)

- Directed and labeled arrows between states denote state transitions.

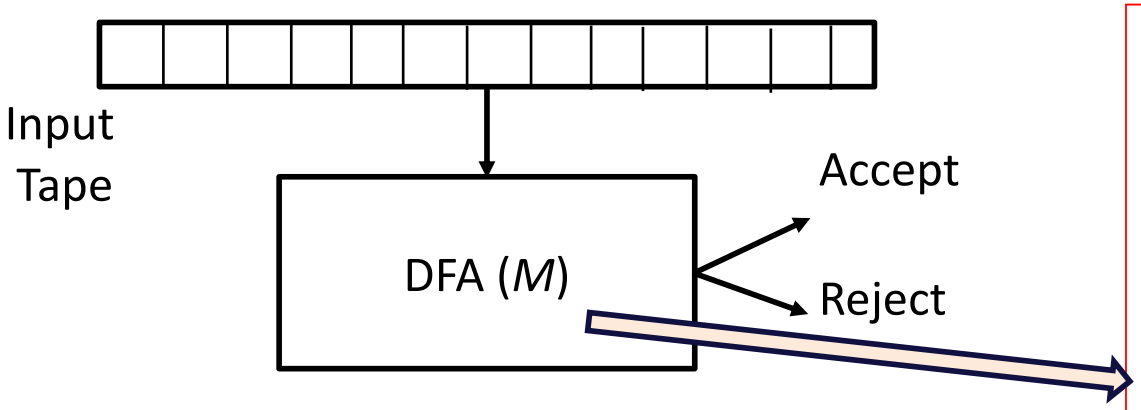
Deterministic Finite Automata

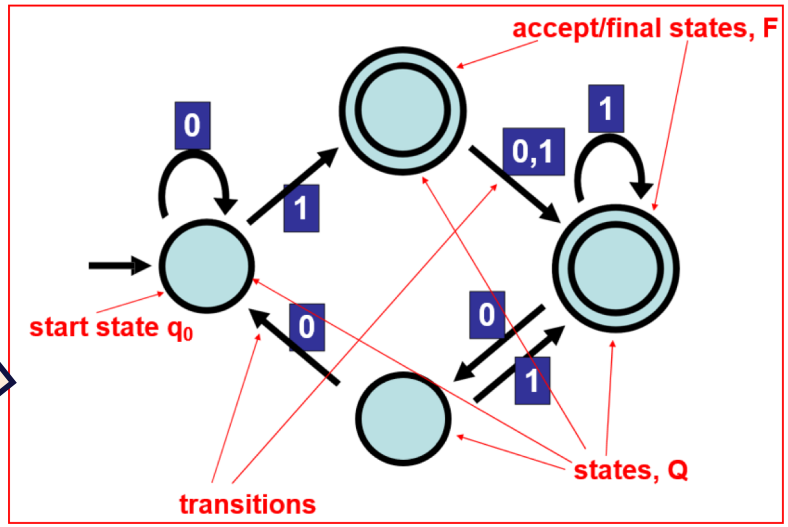


- This DFA stays in the same state when the next input symbol is a 0.
- In state q_0, an input of 1 moves the DFA to state q_1.
- In state q_1, an input of 1 moves the DFA to state q_2.
- In state q_2, an input of 1 moves the DFA back to state q_0.
- If the DFA is in state q_1 when the input is finished, the DFA accepts the input string.
- What kinds of strings does this DFA accept?
  - It accepts $w =$ 00010000, 00010011001,1
  - It rejects $w =$ 0110000, 111001100
- What is pattern observed??
- It accepts all strings $w \in \{0,1\}^*$ such that $n_1(w) = 1 \pmod 3$

Deterministic Finite Automata

- Model of a computer having a **finite** amount of memory.
- This type of machine is known as a **finite-state machine** or **finite automaton**.
- Basic idea how a finite automaton works:
  - It is presented an input string *w* over an alphabet Σ; i.e., $w \in \Sigma^*$.
  - It reads in the symbols of *w* from left to right, one at a time.
  - After reading the last symbol, it indicates if it **accepts** or **rejects** the string.
- These machines are useful for string matching, compilers, etc.



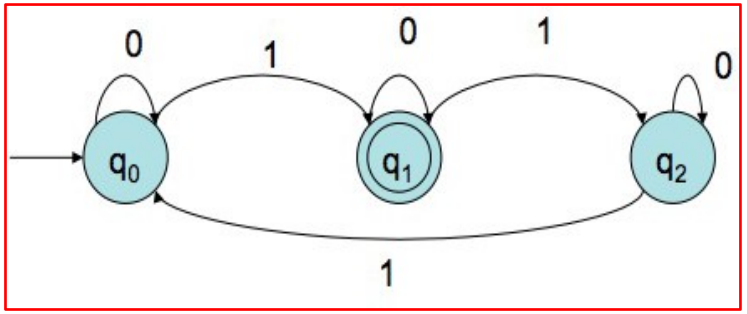


- States are the only mechanism for a FA to "remember" what it has seen of input string so far

Formal Definition of DFA

- A Deterministic Finite State Automaton (DFA) is defined as a 5-tuple

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



Q is a finite set of states $\quad Q = \{q_0, q_1, q_2\}$

Σ is a finite set of symbols – the alphabet $\quad \Sigma = \{0, 1\}$

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_0

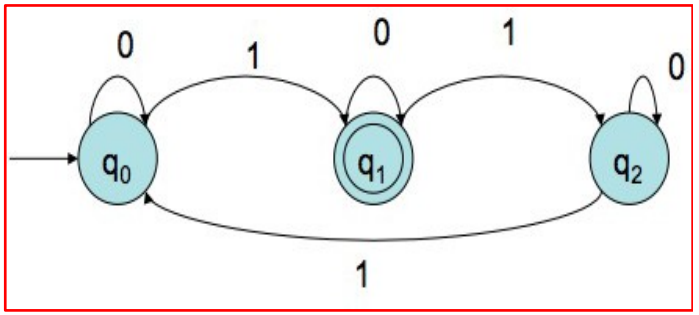
$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the (label of the) start state $\quad q_0 = q_0$

$F \subseteq Q$ is the set of final (accepting) states $\quad F = \{q_1\}$

Transition Function of DFA

Transition function $\delta : Q \times \Sigma \rightarrow Q$ works as follows:



- For each state and for each symbol of the input alphabet, the function δ tells which (one) state to go to next.

- Specifically, if $r \in Q$ and $a \in \Sigma$, then $\delta(r, a)$ is the state that the DFA goes to when it is in state r and reads in a, e.g., $\delta(q_2, 1) = q_3$.

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_0

- For each pair of state $r \in Q$ and symbol $a \in \Sigma$, there is exactly one arc leaving r labeled with a.

- Thus, there is no choice in how to process a string. So the machine is deterministic.

THE EXTENDED STATE TRANSITION FUNCTION

$\delta : Q \times \Sigma \rightarrow Q$ is the state transition function.
The input is a symbol.

$\delta^* : Q \times \Sigma^* \rightarrow Q$ is the extended state transition function.

$\delta^*(q, \epsilon) = q$

$\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, where $a \in \Sigma$ and $w \in \Sigma^*$

First, go (sort of recursively) where w (a string) takes you,
$(\delta^*(q, w) = q')$

Then, make a single transition with symbol a $(\delta(q', a))$

Formal Definition of DFA Computation

innovate achieve lead

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- String $w = w_1 w_2 \cdots w_n \in \Sigma^*$, where each $w_i \in \Sigma$ and $n \geq 0$.
- Then M accepts w if there exists a sequence of states

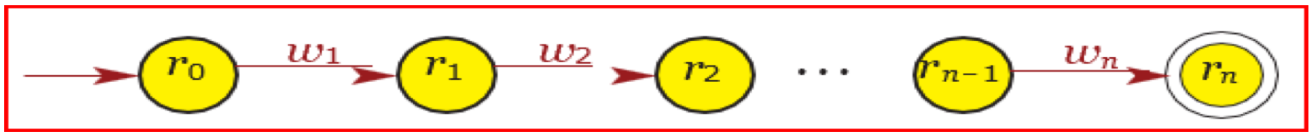
$r_0, r_1, r_2, \ldots, r_n \in Q$ such that

$1. r_0 = q_0$ i.e first state r_0 in the sequence is the start state of DFA;

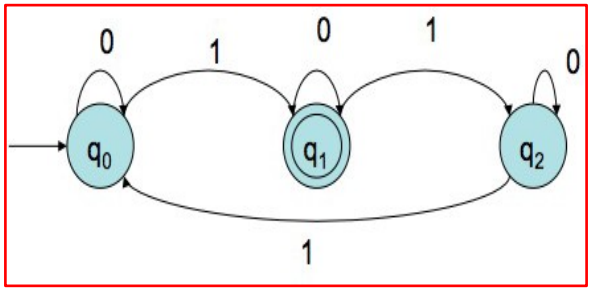
$2. r_n \in F$ $i.e.$ last state r_n in the sequence is an accept state;

$3. r_{i+1} = \delta(r_i, w_{i+1})$ for each $i = 0, 1, 2, \ldots, n-1$

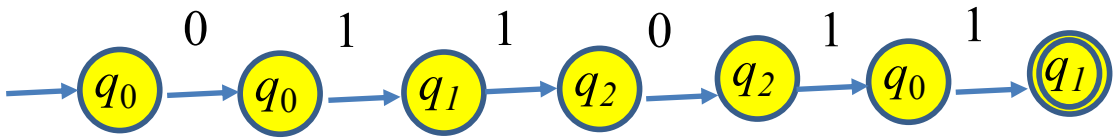
Sequence of states corresponds to valid transitions for string w.



Eg. continuation..



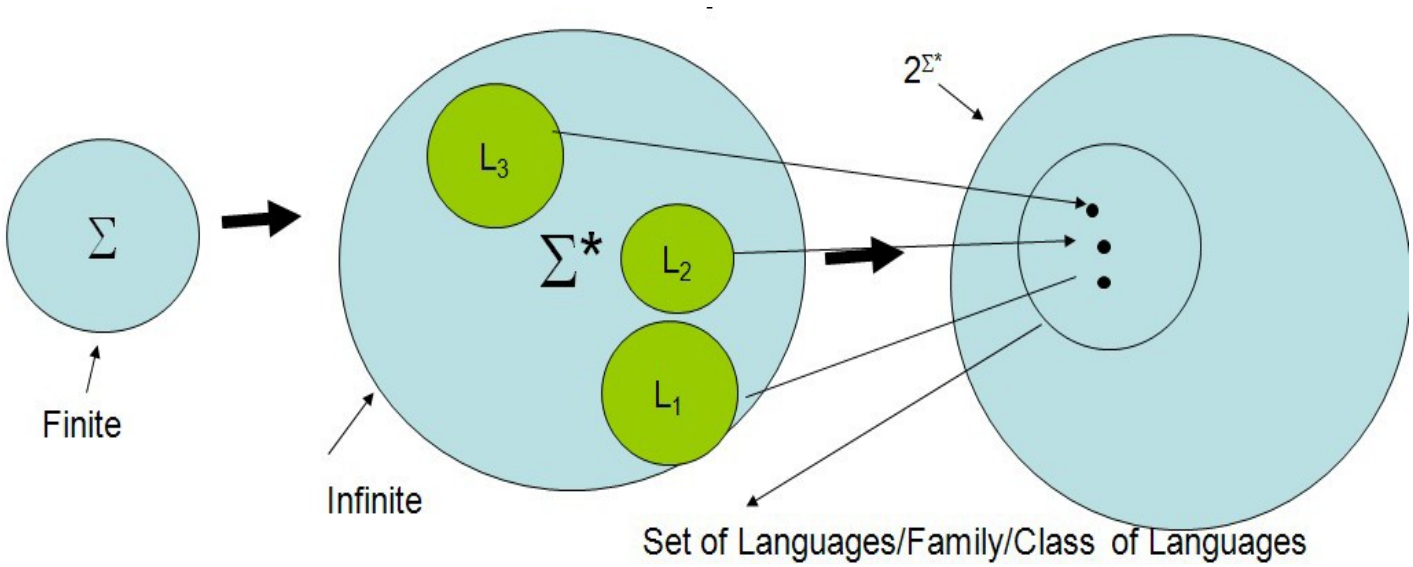
Eg. is the string 011011 accepted? Yes?



- What is the pattern of strings accepted by the automaton?
  - $n(1)_w \ (mod\ 3) = 1$

Formal Languages = Set { }

- Alphabet Σ,
- Set of all Strings, Σ^*, Language $L \subseteq \Sigma^*$
- Set of all languages 2^{Σ^*}
- L = { $w \in \Sigma^*$ | w has some property }
- Eg. S = { w $\in \{0,1\}^*$ | w contains 101 }



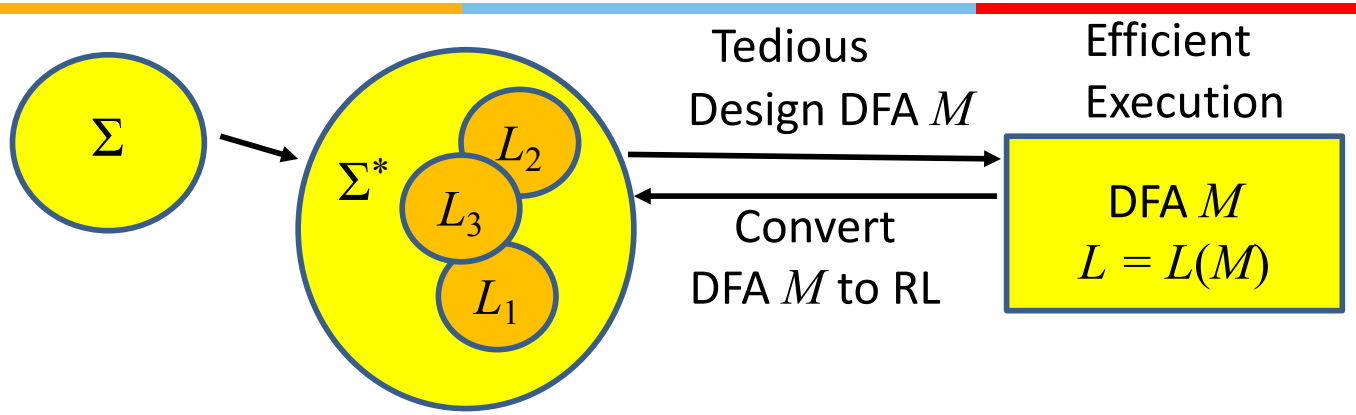
Language of a Machine (L <-> DFA)

- **Definition**: If L is the set of all strings that machine M accepts, then we say $L = L(M)$ is the language of machine M, and M recognizes L.
  - $L(M) = \{ w | w \in \Sigma^* \text{ and } \delta^*(q_0, w) \in F \}$

- A machine **may accept several strings**, but it **always recognizes** only **one language**

- If a machine **accepts no strings**, it **still recognizes one language**, namely the **empty language ø**

RL ⇔ DFA



- $L(M) = L$.

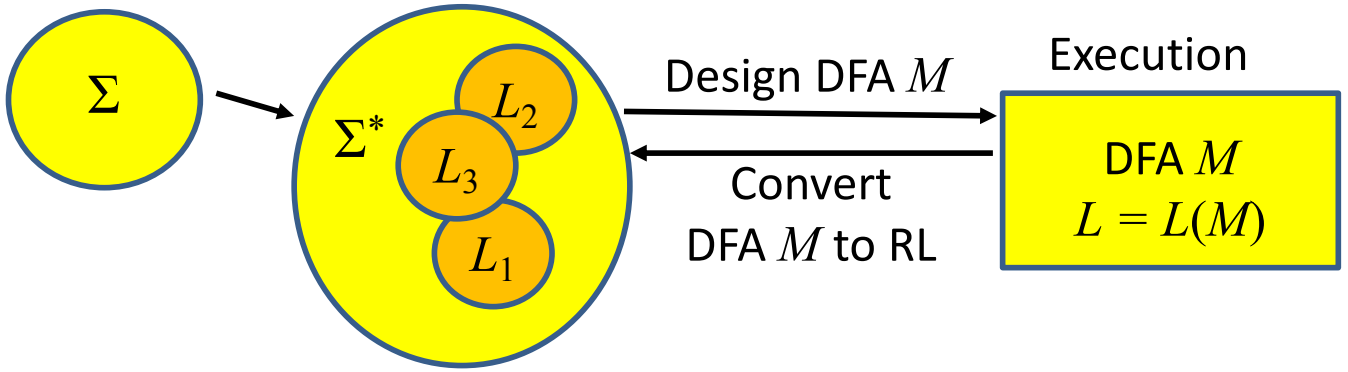
- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L_2$
- $L(M) = L$
- $M = M_1 \times M_2$
- Tedious

- **Definition**: A language L is **regular** if it is recognized by some **DFA** M.
  $L(M) = L$.

- Language recognition can also be thought as a **decision** problem..

- A decision problem is **DFA-decidable** if it can be solved by an algorithm that uses a **constant amount of memory** and that runs in **exactly n steps** on input strings of **length n**.

Operations on Languages



- $L(M) = L$.

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L_2$
- $L(M) = L$
- $M = M_1 \times M_2$

- **Definition**: A language L is **regular** if it is recognized by some **DFA** M.
  $L(M) = L$.

- Languages are usually combined used usual set operations (Why?)
  - Union, Intersection, Difference

- $L = L_1 \cup L_2$ (eg Union)
- Can we have a DFA(M) such that DFA(M) will accept i.e $L(M) = L_1 \cup L_2$
- Is it possible to construct DFA(M). if so how?
- Yes, Cross Product of M_1, M_2
- Regular Languages are closed under union

Closure Properties of Regular Lang

- Let the two DFAs be M_1 and M_2 accepting regular languages L_1 and L_2

$$M_1 = (Q_1, \Sigma, \delta_1, q^1, F_1)$$

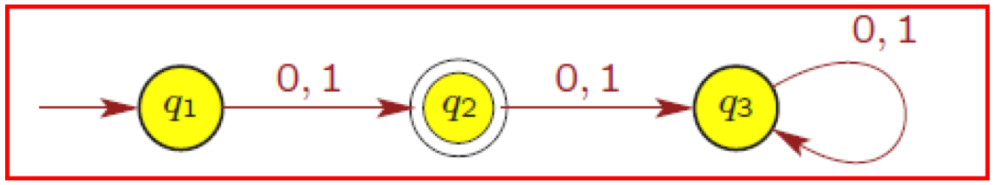
$$M_2 = (Q_2, \Sigma, \delta_2, q^2, F_2)$$

- We want to construct DFAs $M = (Q, \Sigma, \delta, q_0, F)$ that recognize Let L_1 and L_2 be regular languages. Then, the following languages are regular.
  - Complement: $L_1 = \{x \mid x \in \Sigma^* \text{ and } x \notin L_1\}$
  - Union: $L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$
  - Intersection: $L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$
  - Concatenation: $L_1 \circ L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$
  - Star: $L^*_1 = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in L_1\}$.
- Regular languages are closed under these operations.
- Operations on regular languages yields another regular languages

Eg Complement of a DFA

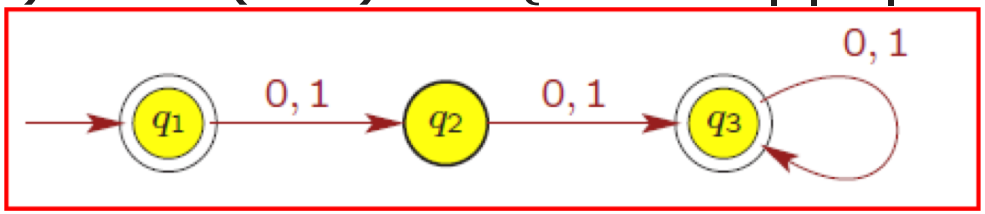
- $L(M_2)$ is language of strings over Σ that have length 1, i.e.,

$$L(M_2) = \{ w \in \Sigma^* \mid |w| = 1 \}$$



- Complement of $L(M_2)$, is the set of strings over Σ not in $L(M_2)$, i.e. $\overline{L(M_2)} = \Sigma^* - L(M_2)$.

- $L(M_3)$ is the language of strings over Σ that do not have length 1, i.e. $L(M_3) = \overline{L(M_2)} = \{w \in \Sigma \mid |w| \neq 1\}$



Accepting States of M2 -> Reject States of M3; Rejecting States of M1 -> Accept States of M2

Complement of a DFA

innovate achieve lead

- In general, given a DFA M for language L, we can make a DFA M for L from M by
  - changing all accept states in M into non-accept states in M
  - changing all non-accept states in M into accept states in M ,

- More formally, suppose language L over alphabet Σ has a DFA

$$M = (Q, \Sigma, \delta, q_1, F).$$

- Then, a DFA for the complementary language L is

$$M = (Q, \Sigma, \delta, q_1, Q - F).$$

where $Q, \Sigma, \delta, q_1, F$ are the same as in DFA M .

- DFA can have more than one accept state.
- Start state can also be an accept state.
- In general, a DFA accepts ε if and only if the start state is also an accept state.

Egs. Complement of a DFA

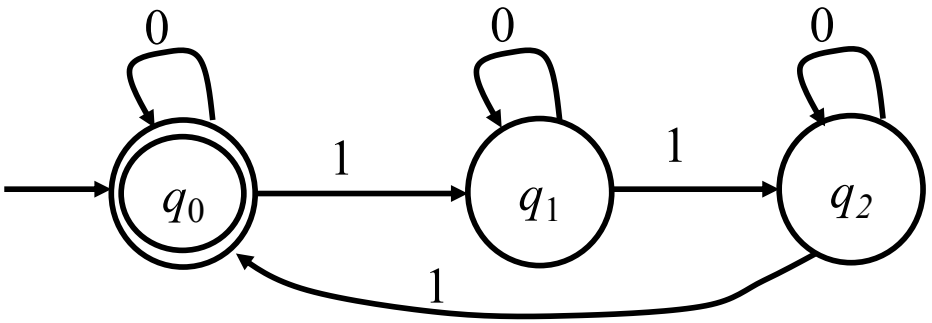
- $L_1 = \{w \in \Sigma^* \mid$ No. of 1's in w is not a multiple of 3 $\}$

Ans:

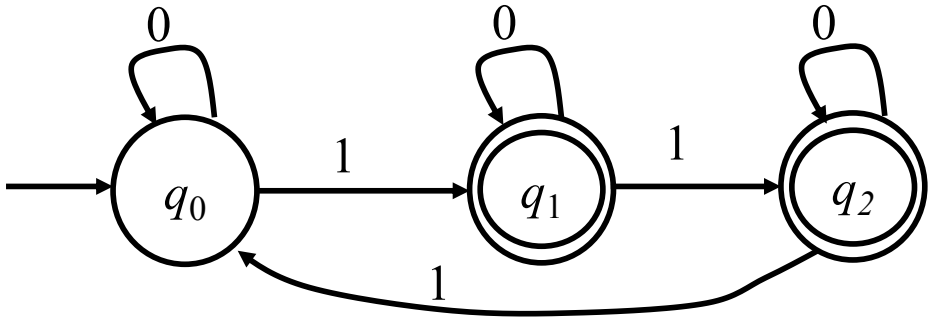
- Obtain DFA M for

$$L_2 = \{w \in \Sigma^* \mid \text{no. of 1's in w is a multiple of 3 } \}$$

$L(M_2) = L_2$.



$L_1 = \overline{L_2} = \overline{L(M_2)}$



- L_1 is regular

Closure Properties of Regular Lang

- *Given:* $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- We need to construct $M_3 = (Q_3, \Sigma, \delta_3, q_3, F_3)$ from M_1 , M_2
- Q_3 = pairs of states, one from M_1 and one from M_2
- $Q_3 = \{(x, y) | x \in Q_1 \text{ and } y \in Q_2\}$ i.e. $Q_3 = Q_1 \times Q_2$
- $q_3 = (q_1, q_2)$ (Start State of M_3)
- $\delta_3((x, y), z) = (\delta_1(x, z), \delta_2(y, z))\ z \in \Sigma$ (Transition Func of M_3)
- Union: $F_3 = \{(x, y) | x \in F_1 \text{ or } y \in F_2\}$

  $= [F_1 \times Q_2] \cup [Q_1 \times F_2]$
- Intersection: $F = \{(q_1, q_2) | q_1 \in F_1 \text{ and } q_2 \in F_2\}$

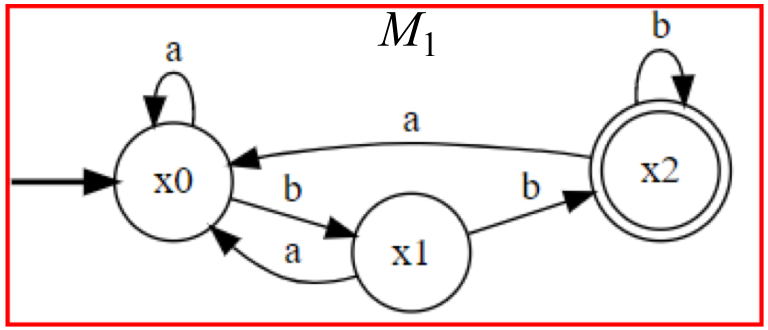
  $= F_1 \times F_2$
- Max no. of state in Q_3 is $|Q_3| = |Q_1| \text{ x } |Q_2|$

Regular Langs are closed under ∪

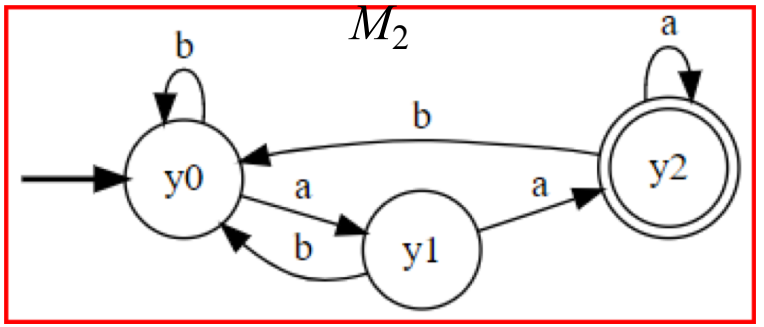
- $L = L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$
- Given two DFAs, it is possible to construct a DFA for the Union of the two languages

- Example:
  - $L_1 = \{wbb \mid w \in \{a,b\}^*\}$ = strings that ends with bb
  - $L_2 = \{waa \mid w \in \{0,1\}^*\}$ = strings that ends with aa
  - $L = L_1 \cup L_2 = \{x \in \{0,1\}^* \mid x \text{ ends with } bb \text{ or } aa\}$

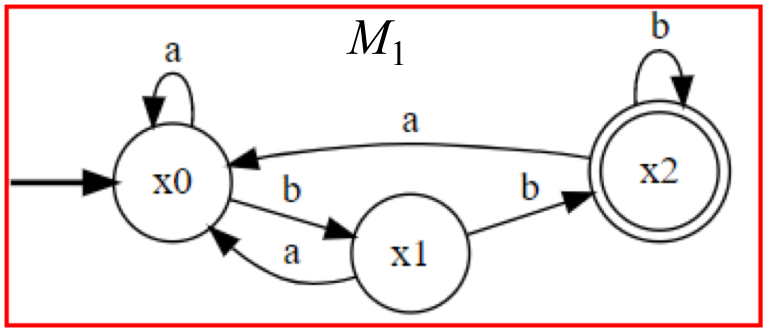
$L_1 = L(M_1)$

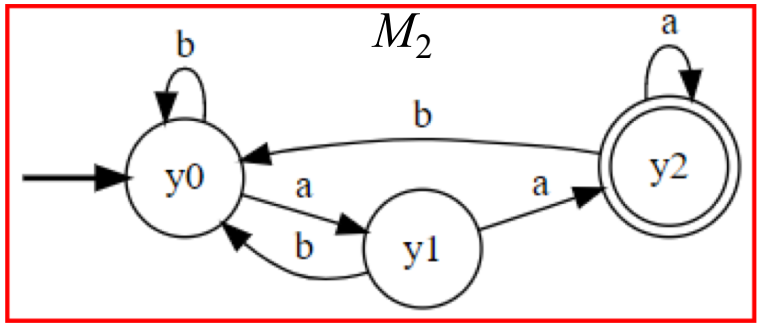


$L_2 = L(M_2)$

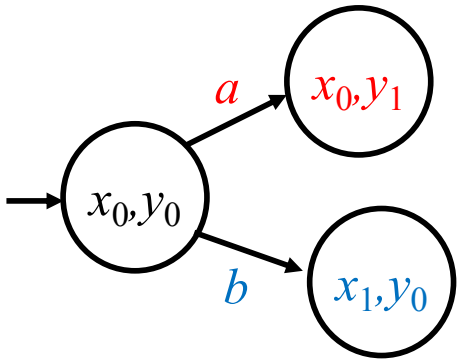


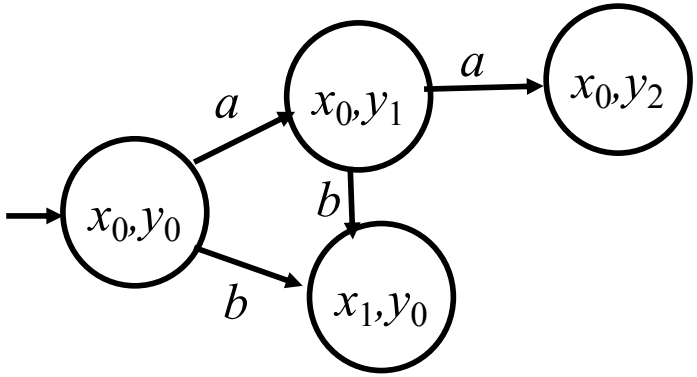
Cross Product of DFAs

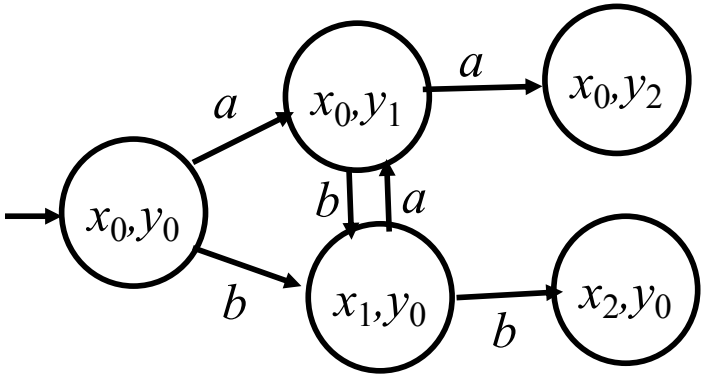




Starting state of M: $q_0 = (x_0,y_0)$



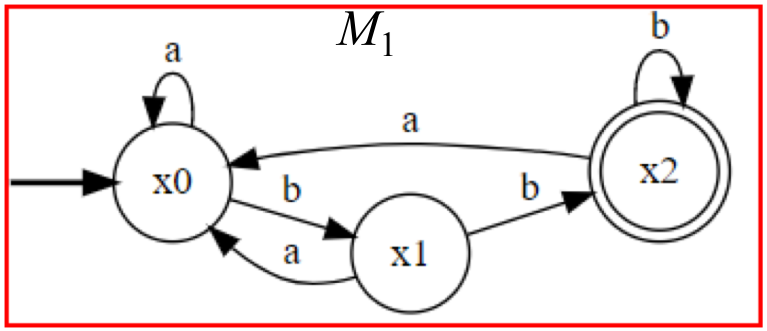


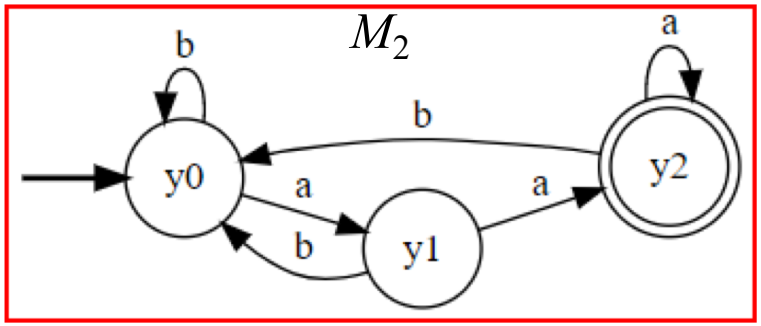


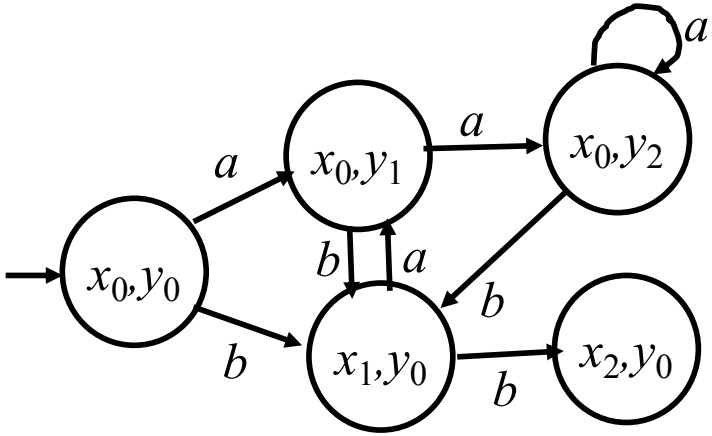
$\delta((x_0,y_0),\ a\ ) = (\delta_1(x_0,\ a\ ),\ \delta_2(y_0,\ a\ )) = (x_0,y_1)$

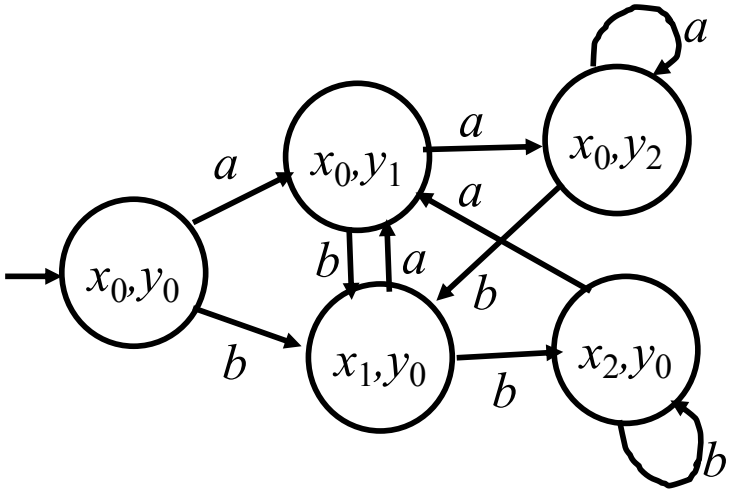
$\delta((x_0,y_0),\ b\ ) = (\delta_1(x_0,\ a\ ),\ \delta_2(y_0,\ a\ )) = (x_1,y_0)$

Cross Product of DFAs

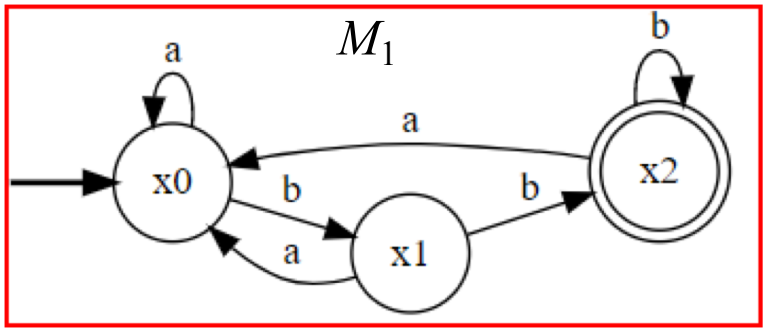


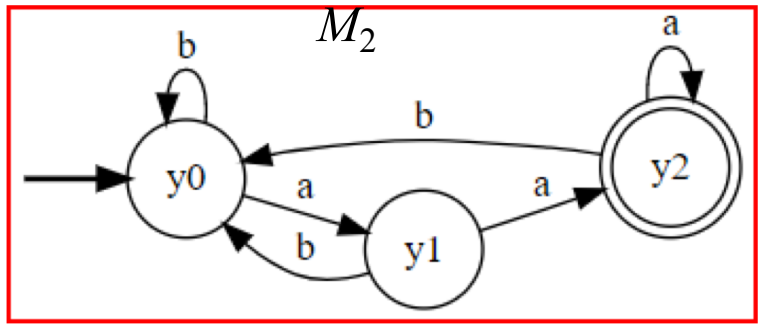


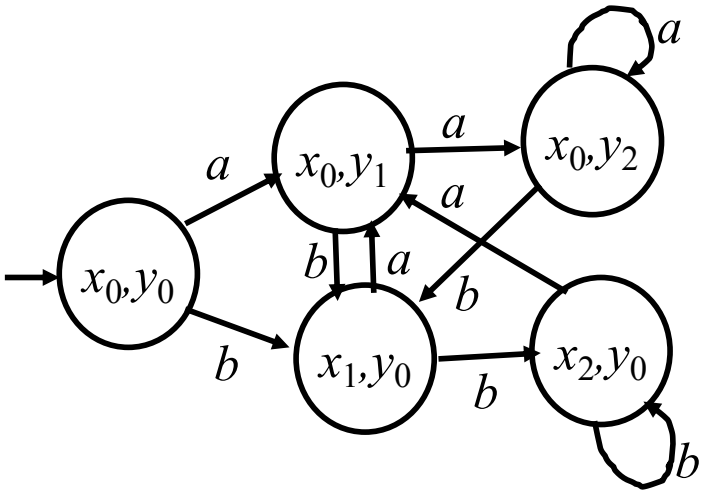


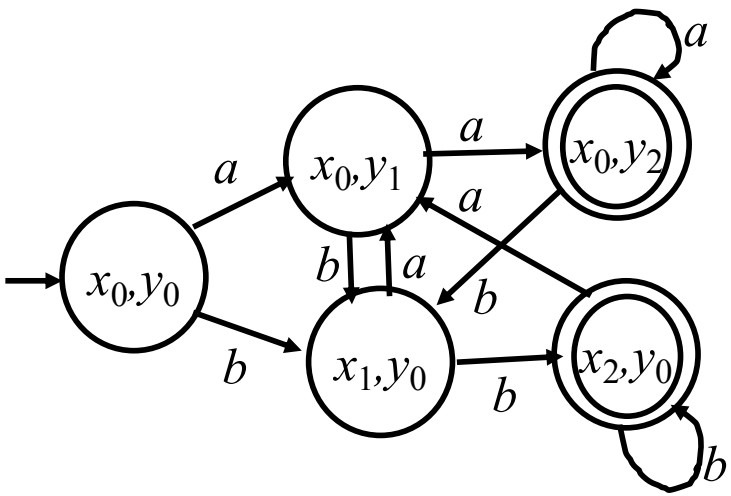


Cross Product of DFAs









δ	a	b
(x_0,y_0)	(x_0,y_1)	(x_1,y_0)
(x_0,y_1)	(x_0,y_2)	(x_1,y_0)
(x_1,y_0)	(x_0,y_1)	(x_2,y_0)
(x_0,y_2)	(x_0,y_2)	(x_1,y_0)
(x_2,y_0)	(x_0,y_1)	(x_2,y_0)

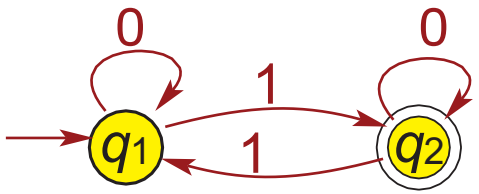
$$F_3 = \{ (x_0,y_2), (x_2,y_0) \}$$

Thank You!

Tutorial – II

Eg. DFA for odd no. of 1's

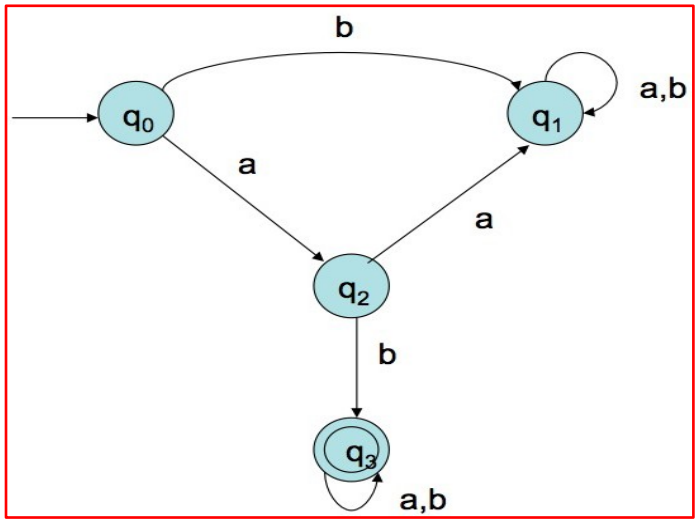
Example: Design a DFA M_1 with alphabet Σ= {0, 1} for recognizing odd no of 1's



- 010110 is accepted, but 0101 is rejected.
- $L(M_1)$ is the language of strings over Σ in which the total number of 1's is odd.

- Design a DFA that recognizes the language of strings over Σ having an even number of 1's ?

Eg. DFA – Accept strings starting with *ab*



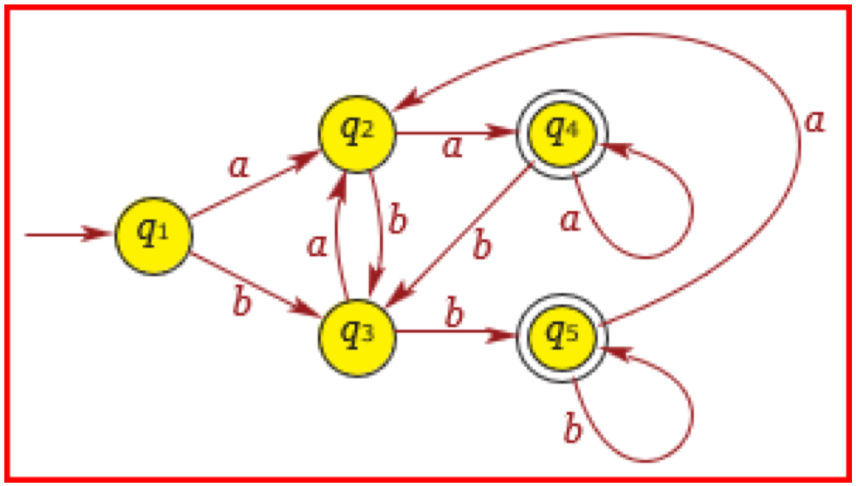
- This DFA accepts strings that start with *ab* $\Sigma = \{a, b\}$
- Once the string starts with *ab* the rest is ignored!
- The state q_1 is known as a sink/dead/trap state.
  - Once a machine enters a sink state, there is no getting out!
  - The input string is rejected.

Eg. DFA

Eg. Strings which end in aa or bb $\Sigma = \{a, b\}$

$L(M) = \{ w \in \Sigma^* \mid w = saa$ or $w = sbb$ for some string $s \in \Sigma^* \}$.

Note that $abbb \in L(M)$ and $bba \notin L(M)$.



$Q = \{q_1, q_1, q_2, q_3, q_4, q_5\}$

$\Sigma = \{a, b\}$

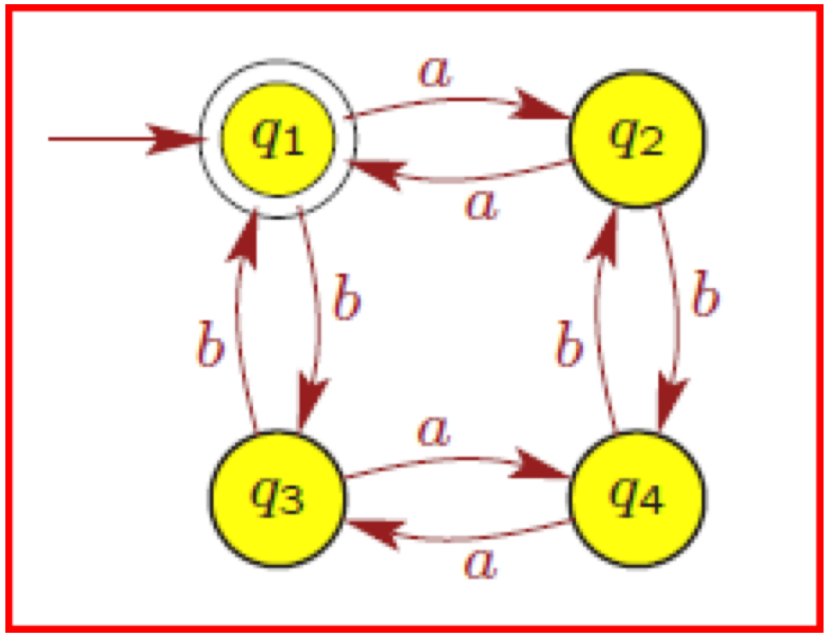
δ	a	b
q_1	q_2	q_3
q_2	q_4	q_3
q_3	q_2	q_5
q_4	q_4	q_3
q_5	q_2	q_5

$q_0 = q_1$

$F = \{q_4, q_5\}$

Eg. DFA

- This DFA recognizes the language of strings over $\Sigma = \{a, b\}$ having even number of a's and even number of b's.
- Note that $ababaa \in L(M)$ and $bba \notin L(M)$.



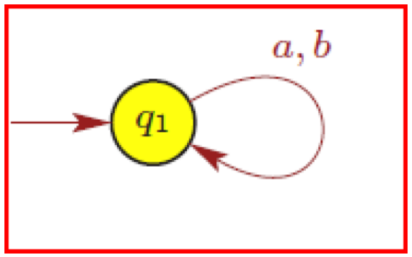
Eg DFA

Eg. A DFA accepts no strings over Σ,

i.e. $L(M) = \emptyset$.

A DFA may have no accept states, i.e., $F = \emptyset \subseteq Q$.

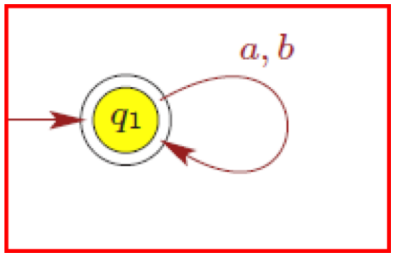
Such a DFA recognizes the language $\emptyset$.



- A DFA accepts all strings over Σ.

  i.e. $L(M) = \Sigma^*$.

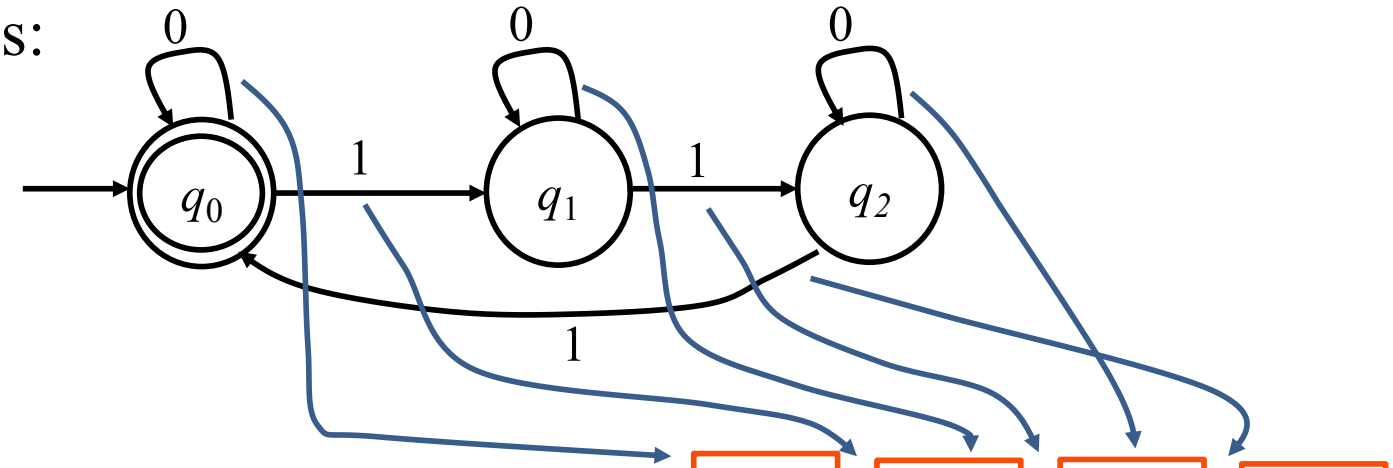
- Any DFA in which all states are accept states recognizes the language Σ^*.



Egs. DFA

- $L_1 = \{w \in \Sigma^* \mid n(1)_w$ is a multiple of 3 $\}$, Σ= {0, 1}
- $n(1)_w$ is the no. of 1's in w

Ans:



- Pattern to be recognized: 0…0 1 0…0 1 0…0 1 0…0

- Strings containing 0, 3, 6, 9 … etc ones(1)'s.
- q_i is the state where the machines has received $n(1)_w$ (mod 3) ones's

- q0 is **accepting** state (0, 3, 6, 9 … etc ones(1)'s)
- we can have three 1's inter-spread with zero or more 0's
- And this pattern can repeat..

Egs. DFA

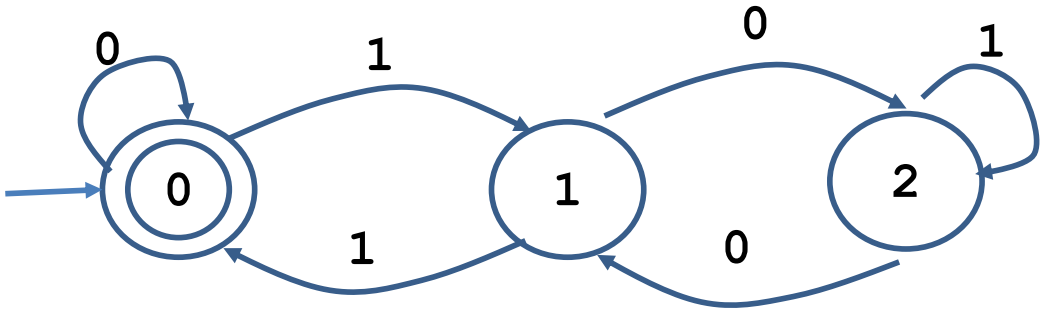
$L_1 = \{w \in \{0,1\}^* \mid \text{val}(w) \text{ is a multiple of 3 } \}$

Detect binary pattern whose value is divisible by 3

- When a no. is divide by 3, the reminder is 0,1,2.
- The remainder of the bits (mod 3) received so far is represented by the states (0, 1, 2).
- Each time a bit arrives, it progressively move to the next state which is the remainder (of the bits received so far) as per the table.

$0100\ 1 : (0100)_2 * 2 + 1 = 9 \ \% \ 3 = 0$
$1010\ 0 : (1010)_2 * 2 + 0 = 20 \ \% \ 3 = 2$

Cur Rem	Inp Bit	Value	New Rem
0	0	0*2+0=0 (mod 3)	0
0	1	0*2+1=1 (mod 3)	1
1	0	1*2+0=2 (mod 3)	2
1	1	1*2+1=3 (mod 3)	0
2	0	2*2+0=4 (mod 3)	1
2	1	2*2+1=5 (mod 3)	2



How to modify this so that M accepts no % 3 = 1

Tutorial – III

DFA's

1. Suppose that X and Y are regular sets. Then L(M) = X and L(N) = Y for some DFA's M and N with sets of states' S and T and start states s_o and t_o, respectively.
   $\sum = \{a,b\}$
   $X = \{x \in \sum^* : \#a(x) \text{ is even}\}$
   $Y = \{x \in \sum^* : \#b(x) \text{ is odd}\}$
   Construct the new DFA:
   - accepts $X \cup Y$
   - accepts x just when $x \in X \cup Y$
   - accepts x just when $x \in X \vee x \in Y$
   - accepts x just when M accepts x $\vee$ N accepts X

2. Suppose that X and Y are regular sets. Then L(M) = X and L(N) = Y for some DFA's M and N with sets of states' A and B and start states A_o and B_o, respectively.
   $X = \{w \mid w \in \sum^*$:set of all strings containing 00 over input alphabets $\sum = \{0, 1\}\}$
   $Y = \{w \mid w \in \sum^*$: set of all strings ending with 01 over input alphabets $\sum = \{0, 1\}\}$

   Draw a DFA which accepts set of all strings
   a. either containing 00 or ending with 01
   b. containing 00 or ending with 01