THEORY OF COMPUTATION

CSC-251

UNIT 1

DWIT

Sailesh.bajracharya@gmail.com

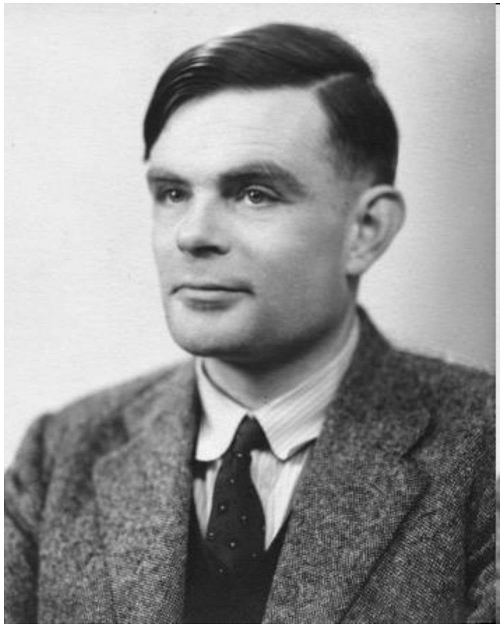
9841594548

Outline

- Finite Automata: Deterministic and Non-deterministic Finite Automata
- Equivalence of Deterministic and Non-deterministic Finite Automata with Epsilon-Transition

Automata Theory

- Study of abstract computing devices or machines
- Automaton = an abstract computing device



ALAN TURING(1912-1954)

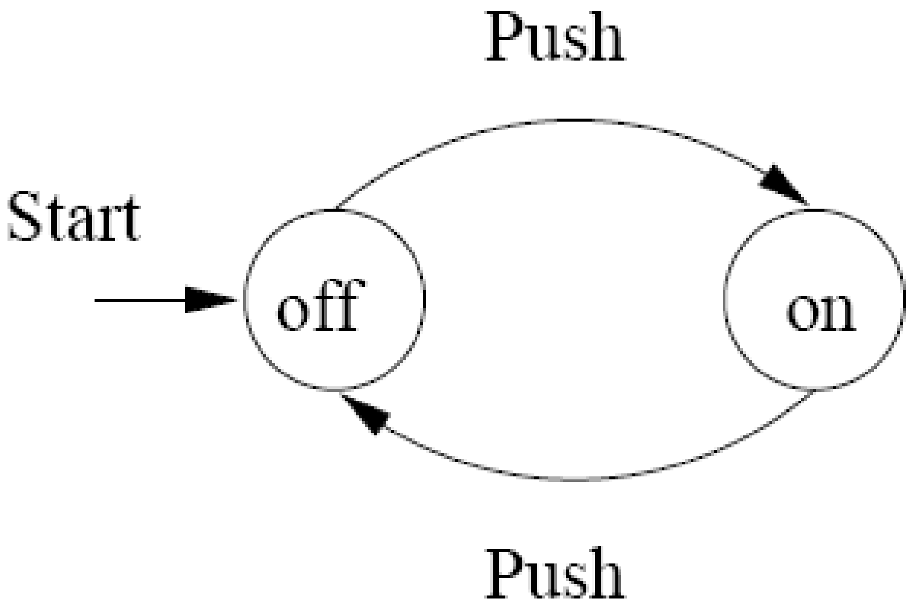
- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called

Turing machines even before computers existed

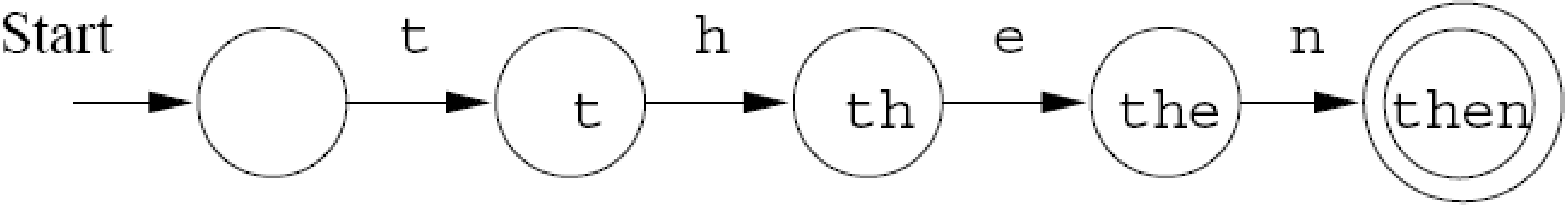
- Turing test?

Finite Automata is used as a model for

- Software for designing digital cicuits
- Lexical analyzer of a compiler(characters into tokens)
- Software for verifying finite state systems, such as communication protocols.
- Example: Finite Automaton modelling an on/off switch



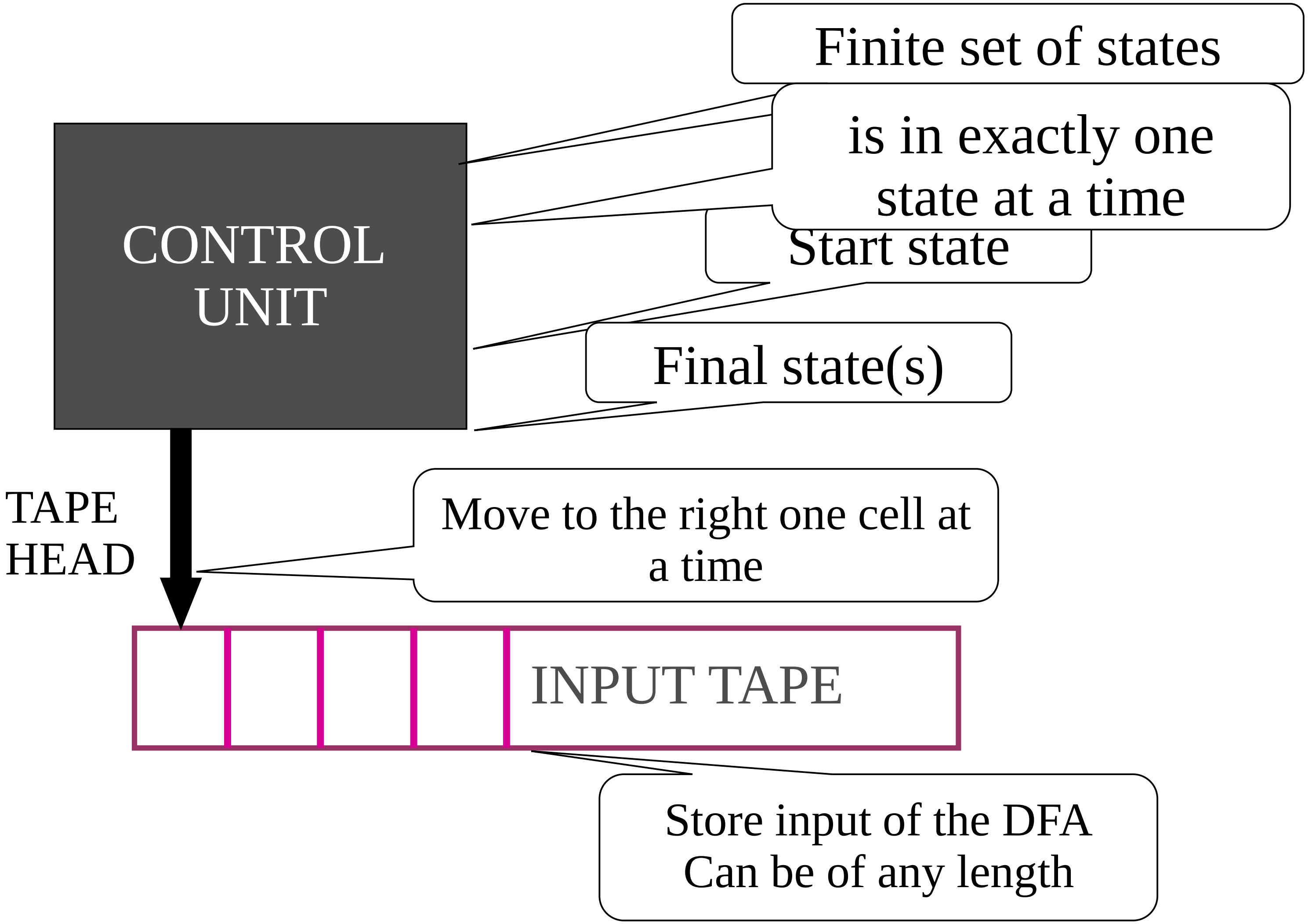
FA recognizing the string” then”



Finite Automata (FA)

- Finite collections of states with transition rules that take you from one state to another
- Recognizer for “Regular Languages”
- *Deterministic Finite Automata (DFA)*
  - The machine can exist in only one state at any given time
- *Non-deterministic Finite Automata (NFA)*
  - The machine can exist in multiple states at the same time

Finite Automata



Finite set of states

is in exactly one state at a time

Start state

Final state(s)

Move to the right one cell at a time

Store input of the DFA
Can be of any length

Alphabets

- finite set of symbols.
- {0,1}
- {a,b,c}

Strings

- The set of strings over an alphabet Σ is the set of lists, each element of which is a member of Σ
- Example: abc
- Σ* denotes this set of strings
- {0,1}* = {ε, 0, 1, 00, 01, 10, 11, 000,001, . . . }

Languages

- A language is a subset of Σ^* for some alphabet Σ
- The set of strings of 0's and 1's with no two consecutive 1's.
- L = {ε, 0, 1, 00, 01, 10, 000, 001, 010,100, 101, 0000, 0001, 0010, 0100,0101, 1000, 1001, 1010, . . . }
- Let L(A) be a language *recognized* by a DFA A.
  - Then L(A) is called a "*Regular Language*".

Deterministic Finite Automata(DFA)

- States ---->Determined
- **Quintuple {Q, Σ, δ, qo, F}**
  - A finite set of states (Q, typically).
  - An input alphabet (Σ, typically).
  - A transition function (δ, typically).
    - δ is a function $Q \times \Sigma \rightarrow Q$
  - A start state (q0, in Q, typically).
  - A set of final states (F ⊆ Q, typically).

Graph Representation of DFA’s

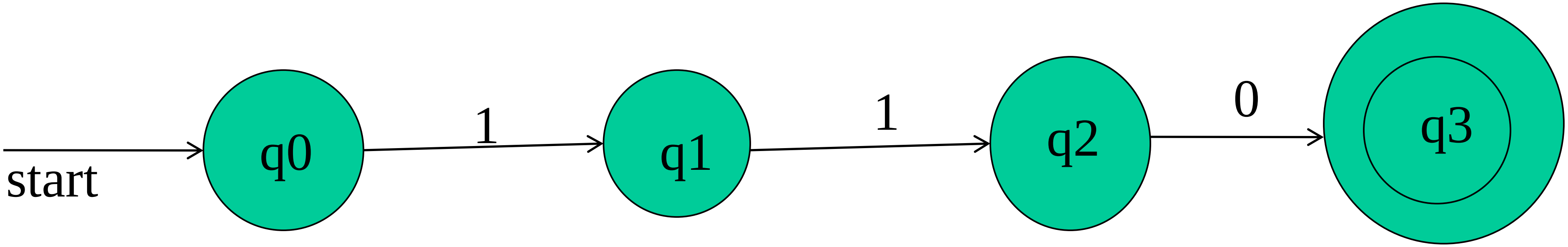
- Nodes = states.
- Arcs represent transition function.
- Arrow labeled “Start” to the start state.
- Final states indicated by **double circles**

The Transition Function

- Takes two arguments:
  - a state and
  - an input symbol.
- δ(q, a) = the state that the DFA goes to when it is in state q and input a is received.

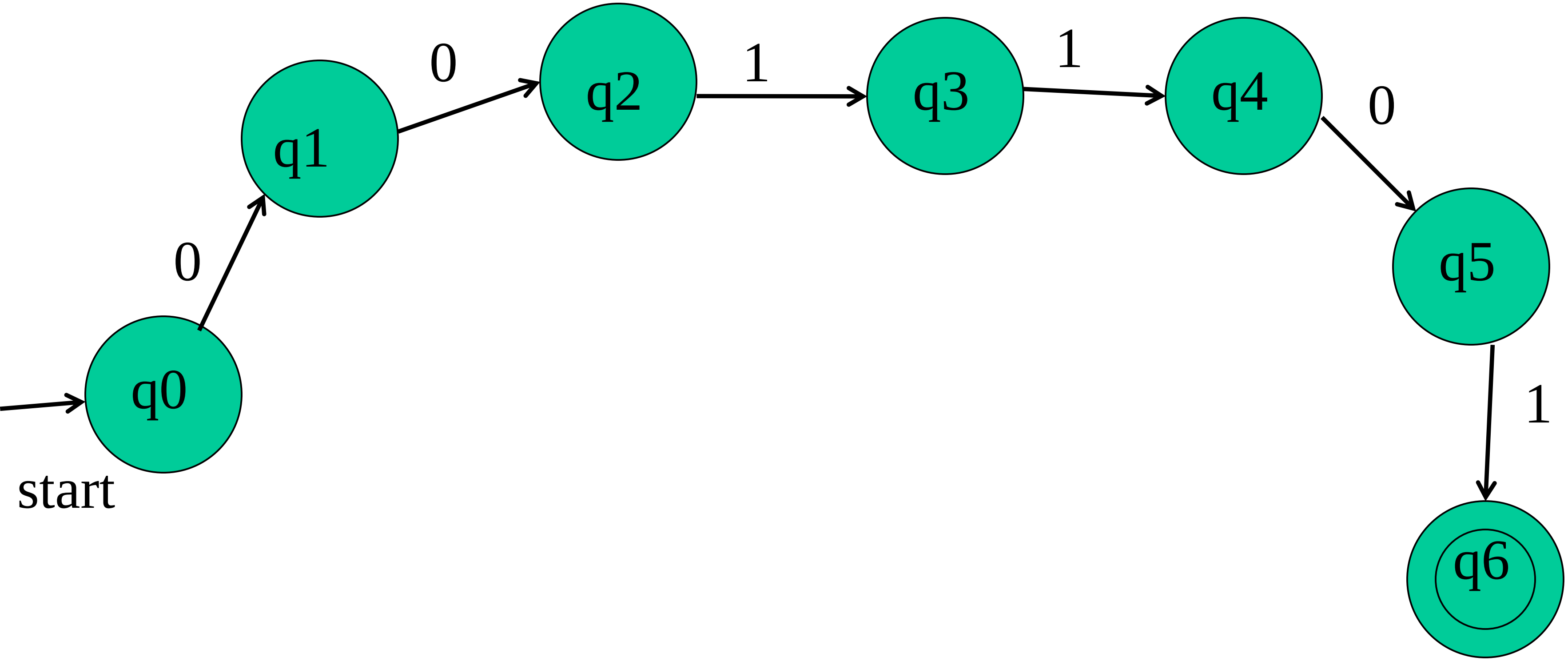
Construct a DFA that accepts **110**

Construct a DFA that accepts 110



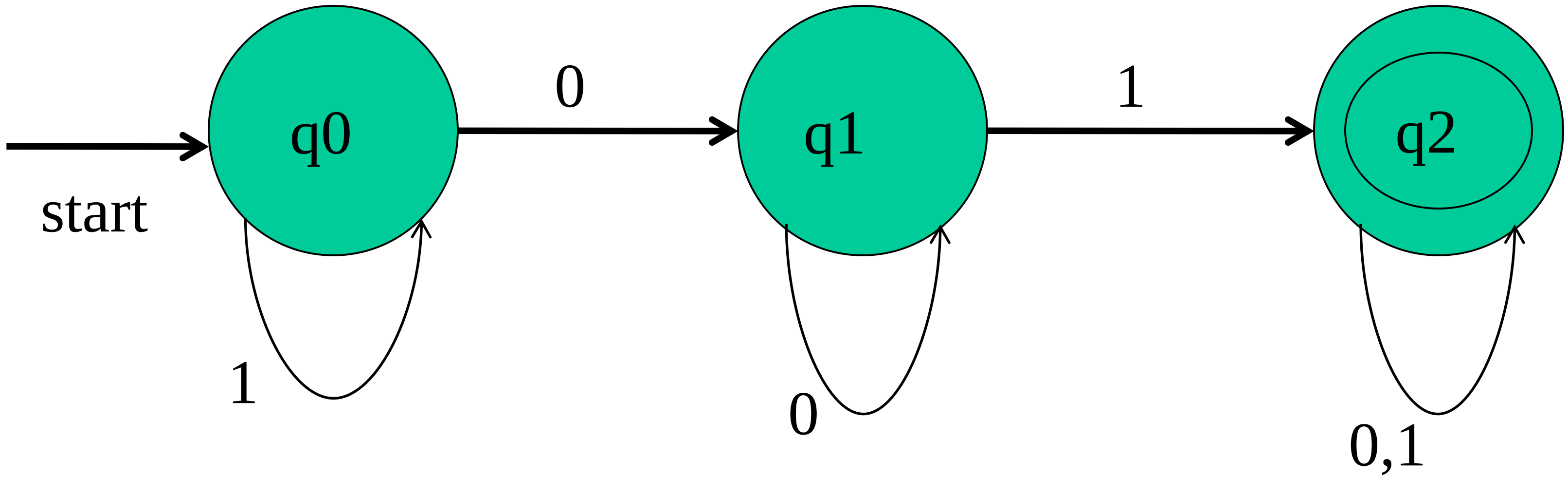
Construct a DFA that accepts 001101

Construct a DFA that accepts 001101



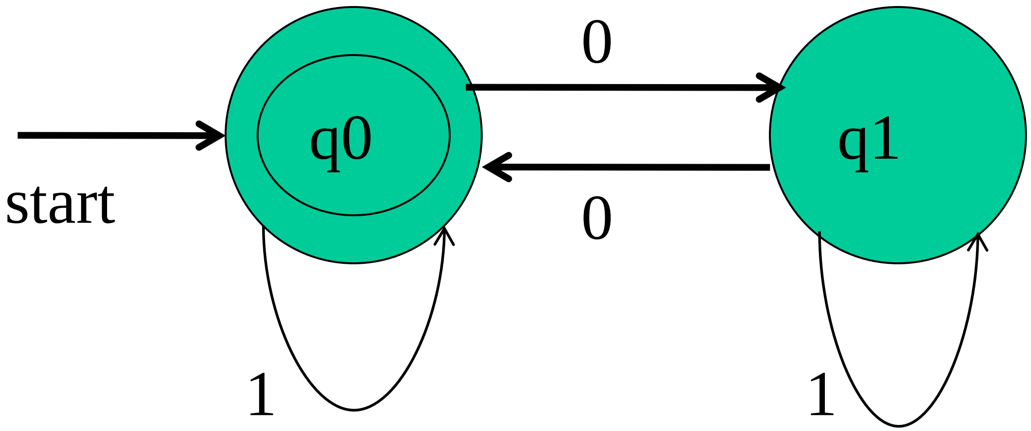
Construct a DFA to accept string containing a 0 followed by a 1

Construct a DFA to accept string containing a 0 followed by a 1



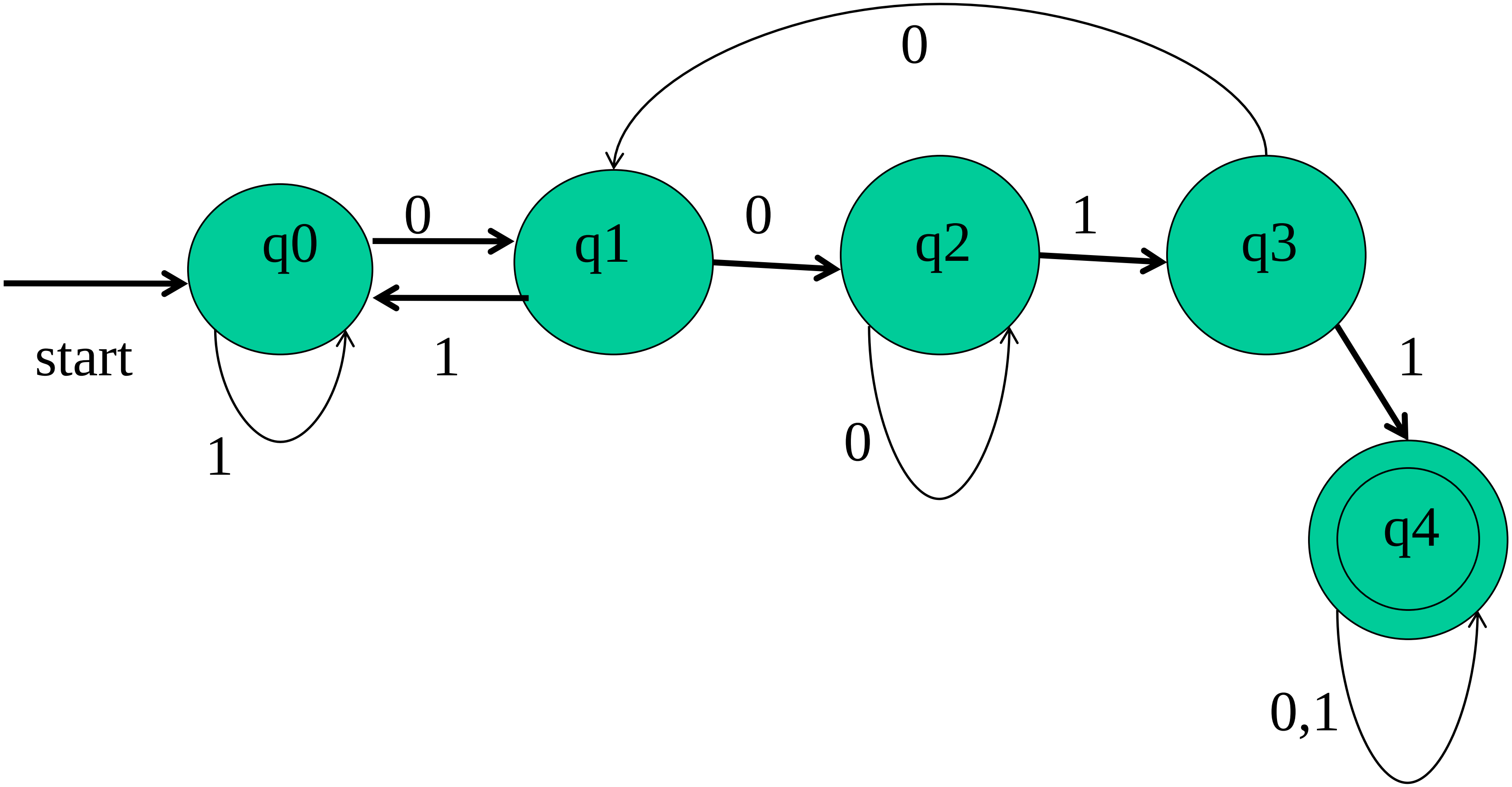
Construct a DFA to accept string containing even no. of 0’s and any no. of 1’s

Construct a DFA to accept string containing even no. of 0’s and any no. of 1’s



Construct a DFA to accept string containing two consecutive 0’s followed by two consecutive 1’s

Construct a DFA to accept string containing two consecutive 0's followed by two consecutive 1's



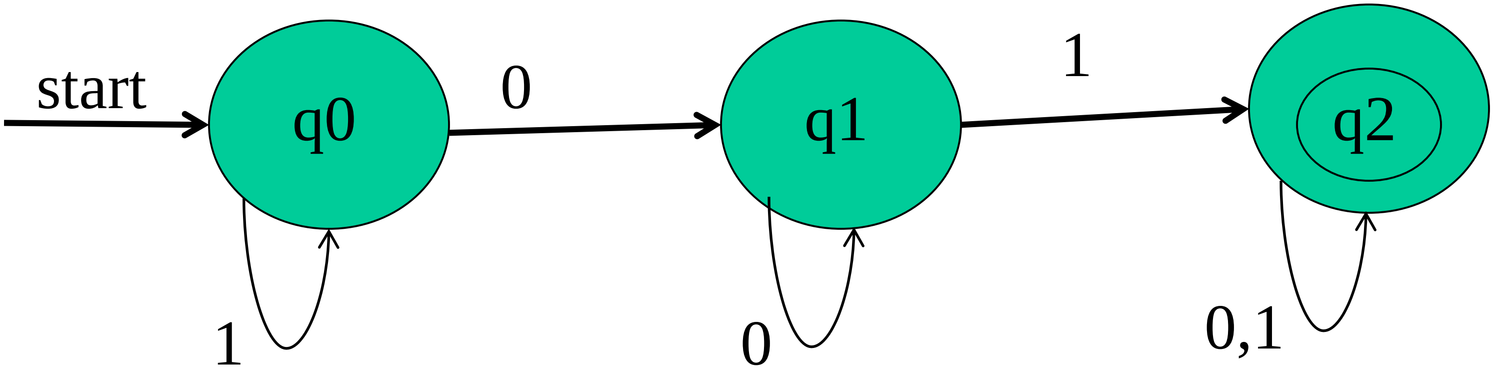
Construct a DFA to accept string that ends in 1101

An automaton A that accepts

$$L = \{x01y : x, y \in \{0,1\}^*\}$$

An automaton A that accepts

$$L = \{x01y : x, y \in \{0,1\}^*\}$$

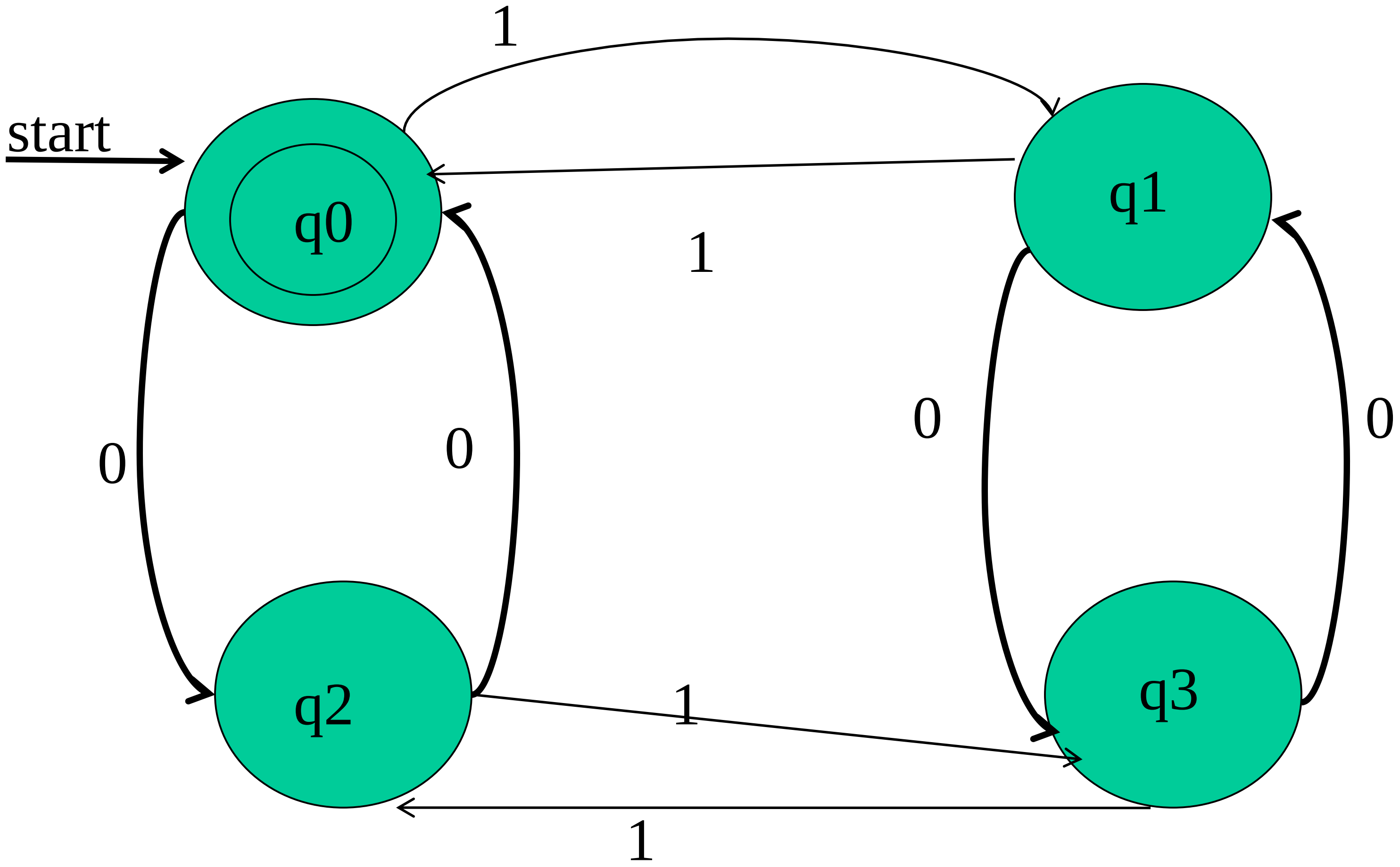


01,11010,1100011

111000 not accepted..why

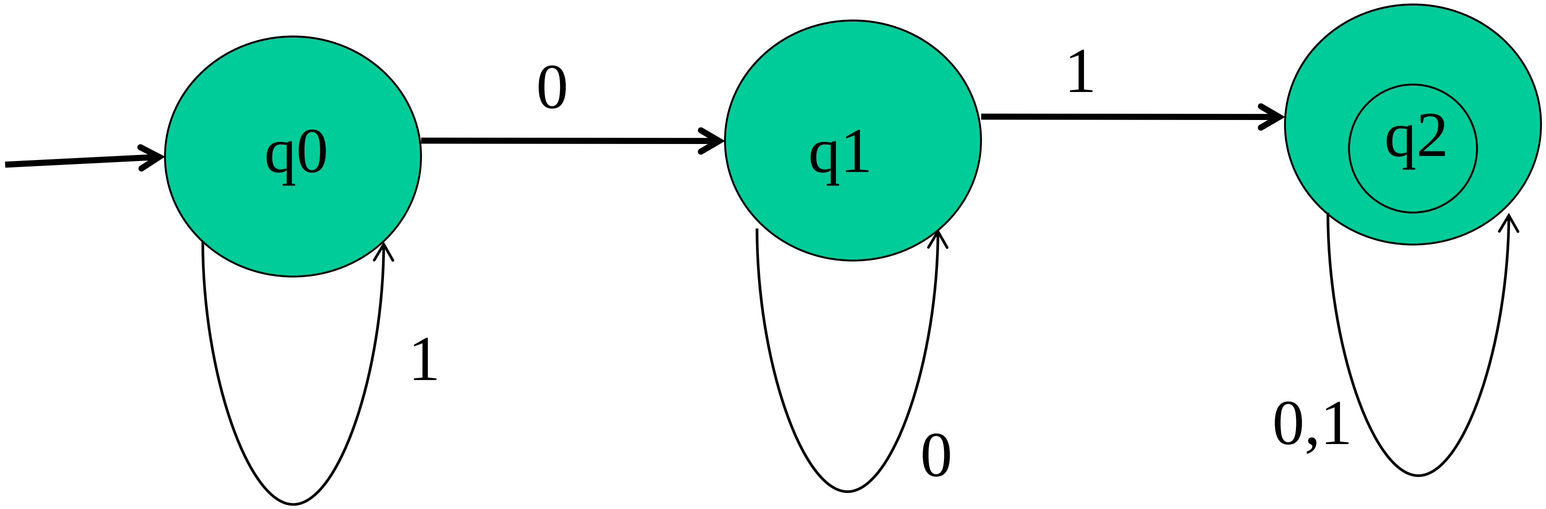
DFA accepting all and only strings with an even number of 0's and an even number of 1's

DFA accepting all and only strings with an even number of 0's and an even number of 1's



L={w|w has both an even no of 0’s and even no of 1’s

DFA accepting all strings with a substring 01



Construct a DFA

- Set of all strings ending in 00
- Set of all strings with three consecutive 0’s
- Set of strings with 011 as substring

Transition Function

$\delta(q1,101) = \delta(\,\delta(q1,1)\,,01)$

$= \delta(q1,01)$

$= \delta(\,\delta(q1,0)\,,1)$

$= \delta(q2,1)$

$= \delta(\,\delta(q2,0)\,,\varepsilon)$

$= \delta(q2,\varepsilon)$

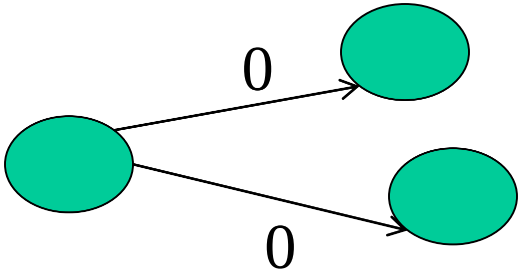
$= q2$

Nondeterministic Finite Automata

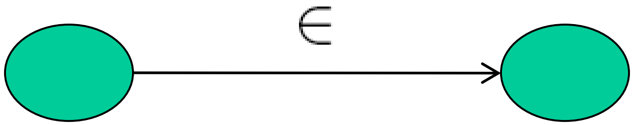
- The machine can exist in multiple states at the same time
- Transitions could be non-deterministic
- Power to be in several states at once
- **Guess** which state to go to next
- Easier to design than DFA
- Each transition function therefore maps to a set of states
- Differs with DFA in the transition function
    - Multiples states

Nondeterministic Finite Automata

- Nondeterministic move
  - On reading an input symbol, the automaton can choose to make a transition to one of selected states.



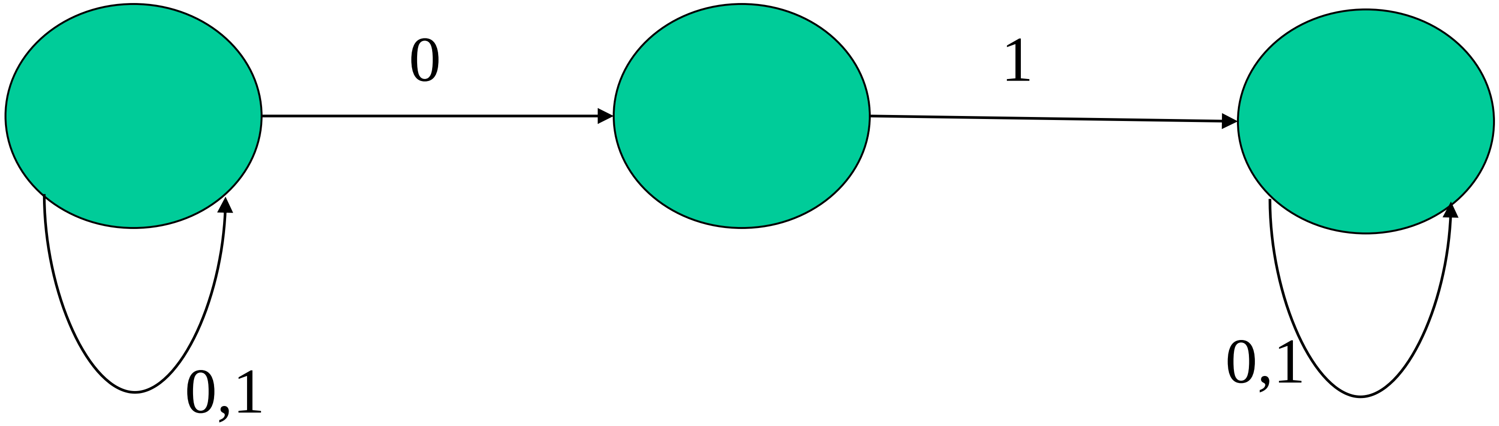
  - Without reading any symbol, the automaton can choose to make a transition to one of selected states or not.



Nondeterministic Finite Automata

- NFA is a quintuple
  - **{Q, Σ, δ, qo, F}**
- Q is a finite set of states
- **Σ** is a finite alphabet
- **δ** is a transition function from QX **Σ** to the powerset of Q
- A start state (q0, in Q, typically).
- A set of final states (F ⊆ Q, typically).

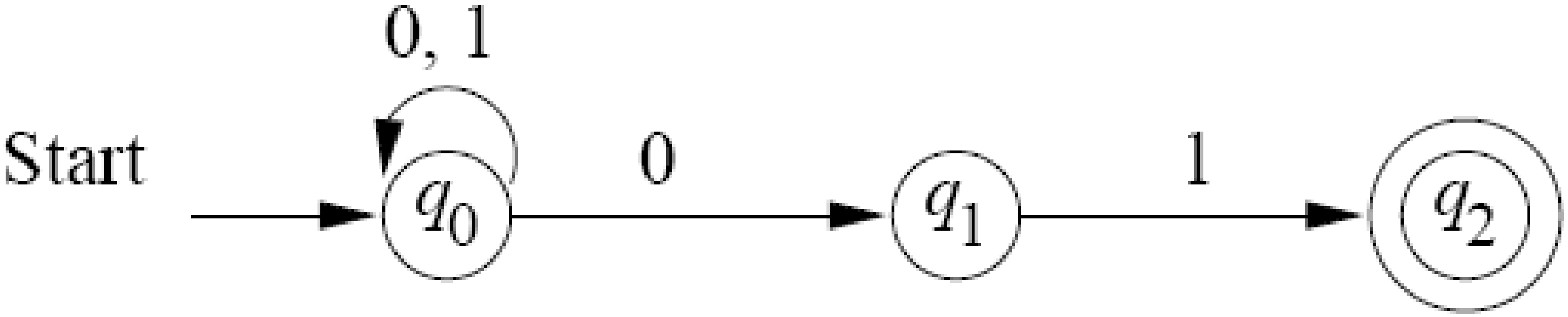
NFA for strings containing 01 or Regular expression: (0+1)*01(0+1)*



Transition function ?????

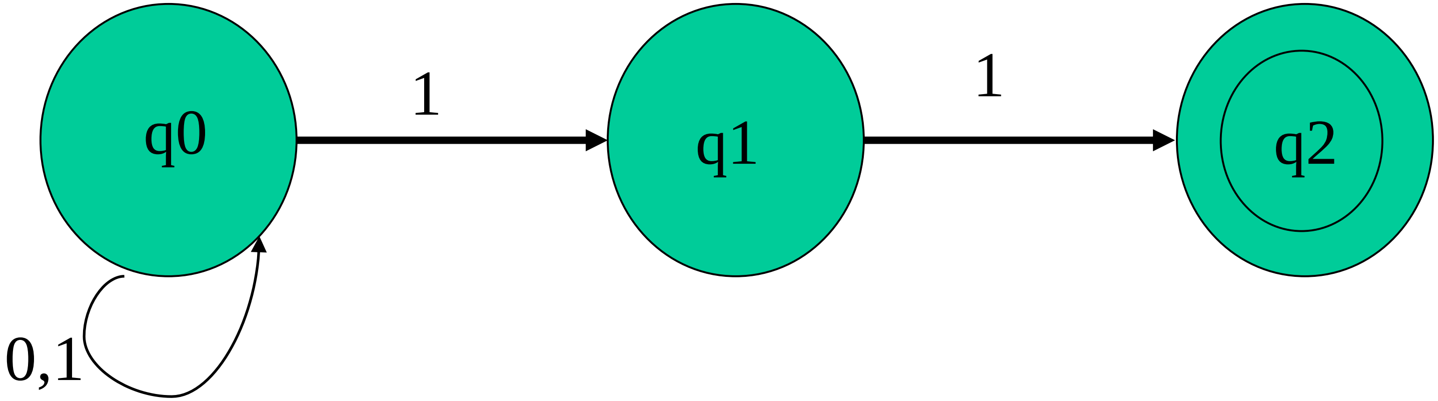
NFA: An automaton that accepts all and only strings ending in 01

NFA: An automaton that accepts all and only strings ending in 01



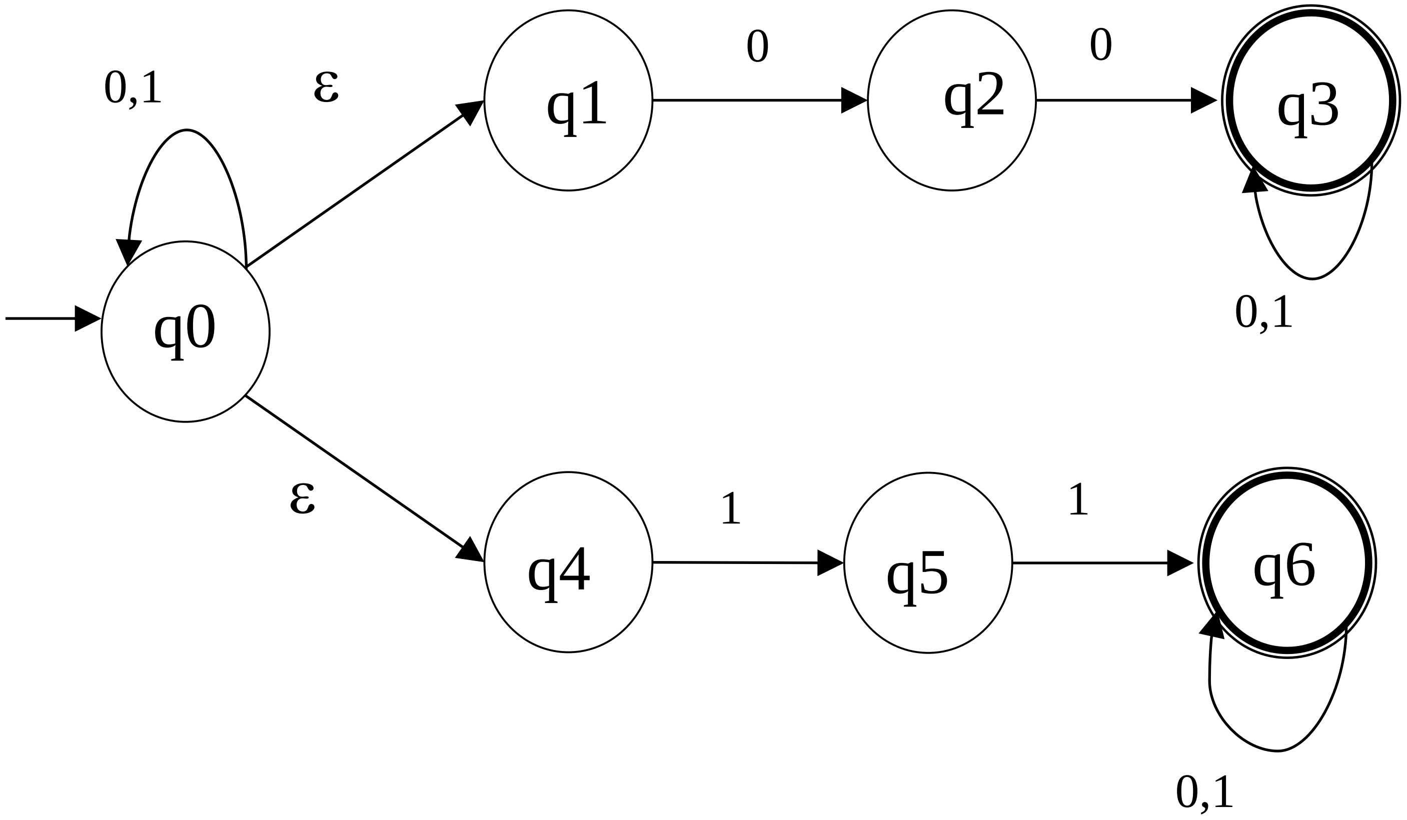
δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

An NFA accepting {$w \in \{0,1\}^*$ |w ends with 11}

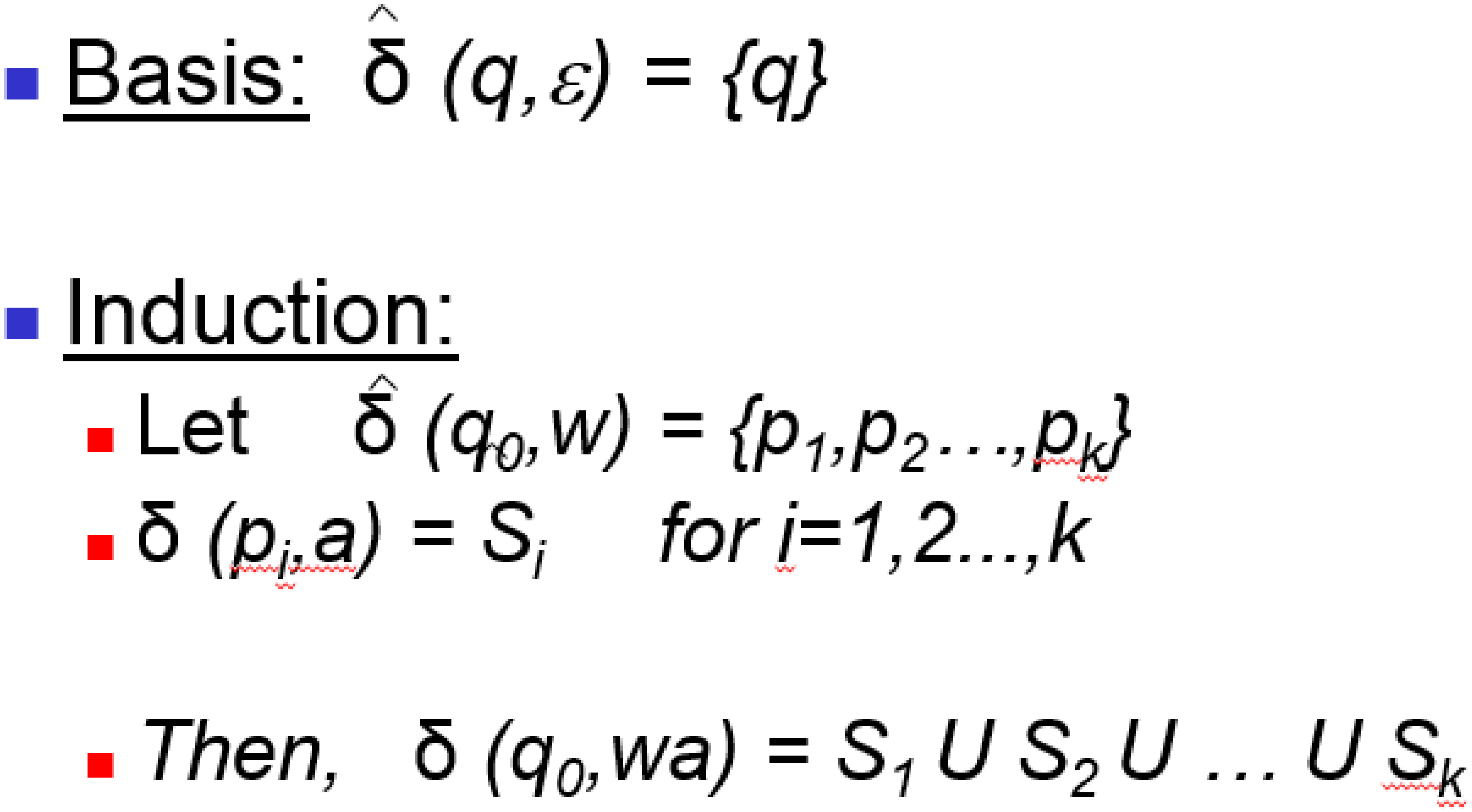


$\{w \in \{0,1\}^* \mid w$ has either 00 or 11 as substring$\}$

{*w* ∈ {0,1}* | *w* has either 00 or 11 as substring}



Extended Transition Function i.e. Extension of δ to NFA Paths



- <u>Basis:</u> $\hat{\delta}(q,\varepsilon) = \{q\}$
- <u>Induction:</u>
  - Let $\hat{\delta}(q_0,w) = \{p_1,p_2\ldots,p_k\}$
  - $\delta(p_i,a) = S_i$ *for i=1,2...,k*
  - *Then,* $\delta(q_0,wa) = S_1 \cup S_2 \cup \ldots \cup S_k$

Language of an NFA

- An NFA accepts *w* if *there exists at least one* path from the start state to an accepting (or final) state that is labeled by *w*
- $L(N) = \{ w \mid \hat{\delta}(q_0,w) \cap F \neq \Phi \}$

Equivalence of DFA & NFA: Subset construction

- Subset construction starts from and NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
- Describe a DFA

  $$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

  such that L(D)=L(N)
- If-part: A language L is accepted by a DFA if it is accepted by an NFA
- Given any NFA N, we can construct a DFA D such that L(N)=L(D)
- How to convert an NFA into a DFA?
  - Observation: In an NFA, each transition maps to a *subset* of states
  - Idea: Represent:

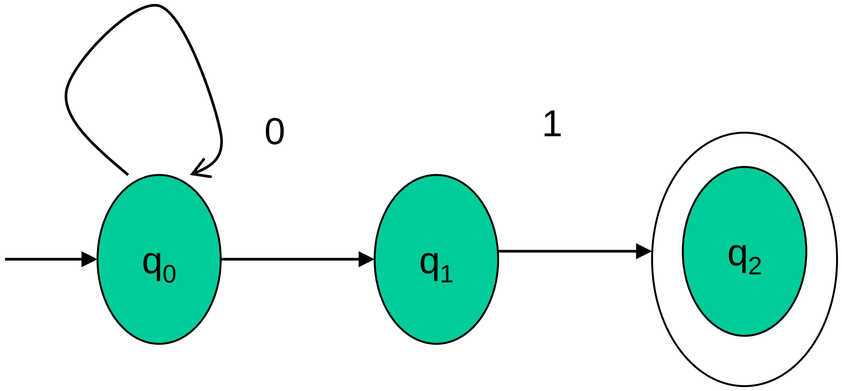
    each “subset of NFA_states” ➔ a single “DFA_state”

NFA to DFA by subset construction

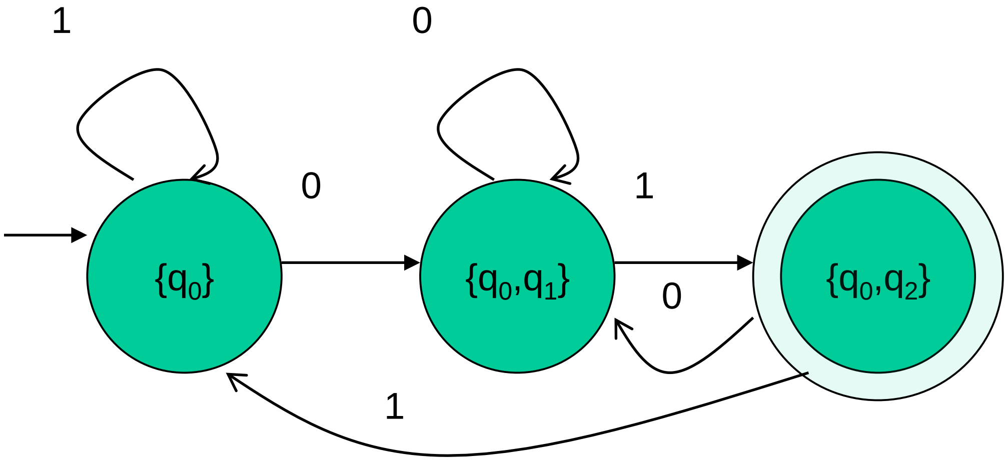
- Let N = $\{Q_N, \sum, \delta_N, q_0, F_N\}$
- Goal: Build D=$\{Q_D, \sum, \delta_D, \{q_0\}, F_D\}$ s.t. L(D)=L(N)
- Construction:

1. Q_D= all subsets of Q_N (i.e., power set)
2. F_D=set of subsets S of Q_N s.t. $S \cap F_N \neq \Phi$
3. δ_D: for each subset S of Q_N and for each input symbol a in $\sum$:
   - $\delta_D(S,a) = \cup\ \delta_N(p,a)$

$L = \{w \mid w \text{ ends in } 01\}$



NFA



DFA

FA with ε-Transitions

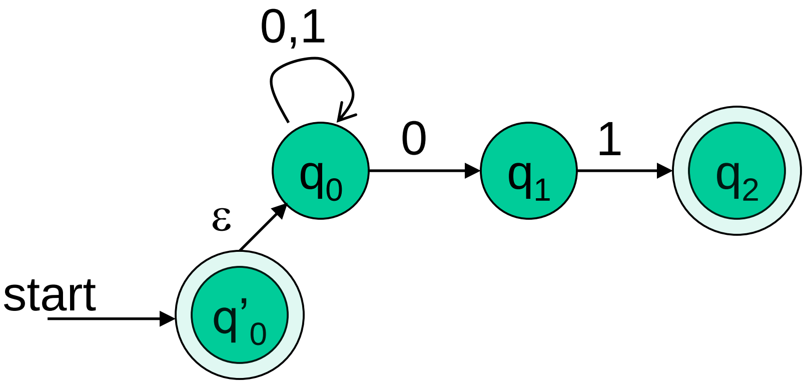
- We can allow explicit ε-transitions in finite automata
  - i.e., a transition from one state to another state without consuming any additional input symbol
  - Makes it easier sometimes to construct NFAs

***Definition:* ε -NFAs are those NFAs with at least one explicit ε-transition defined.**

- ε -NFAs have one more column in their transition table

Example of an ε-NFA

L = {w | w is empty, or if non-empty will end in 01}



δ_E	0	1	ε
→ *q'$_0$	Ø	Ø	{q'$_0$,q$_0$}
q$_0$	{q$_0$,q$_1$}	{q$_0$}	{q$_0$}
q$_1$	Ø	{q$_2$}	{q$_1$}
*q$_2$	Ø	Ø	{q$_2$}

ECLOSE(q'$_0$)

ECLOSE(q$_0$)

ECLOSE(q$_1$)

ECLOSE(q$_2$)

- ε-closure of a state q, ***ECLOSE(q)***,
- If state p is in ECLOSE(q) and there is a transition from state p to state r labelled ε, then r is in ECLOSE(q).