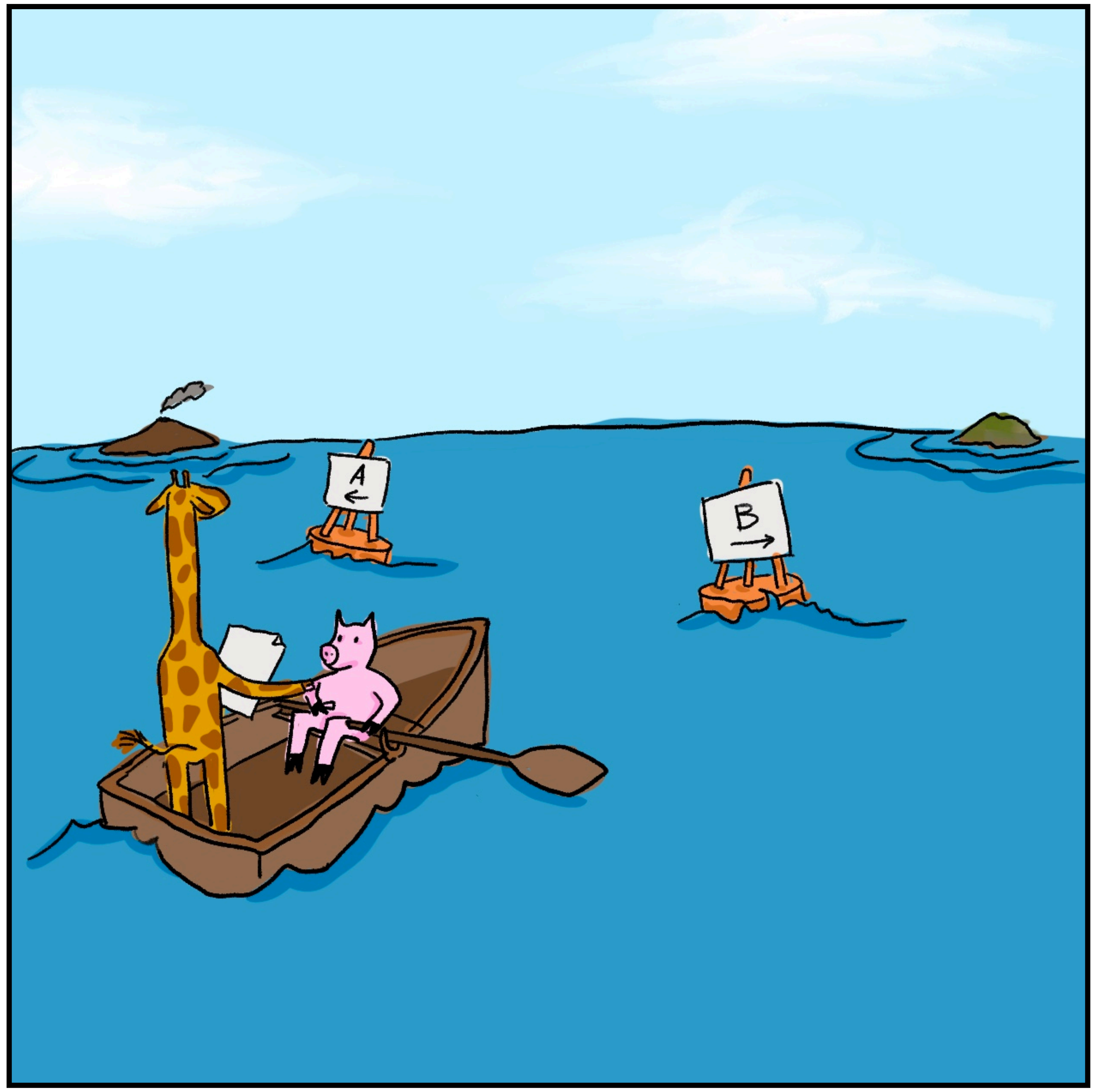
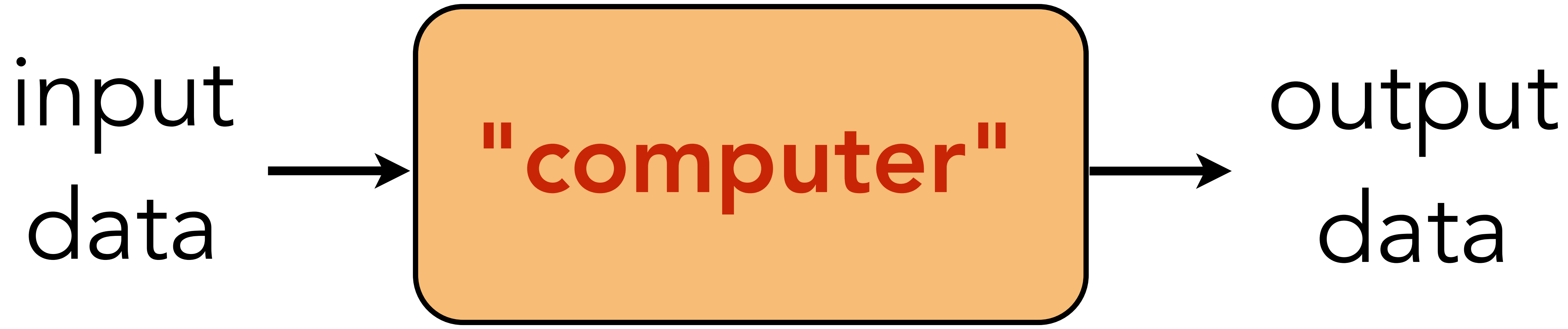
CS251

Great Ideas in *Theoretical* Computer Science



Deterministic Finite Automata 1

This Chapter and Next Chapter



What is **computation**?

What is an **algorithm**?

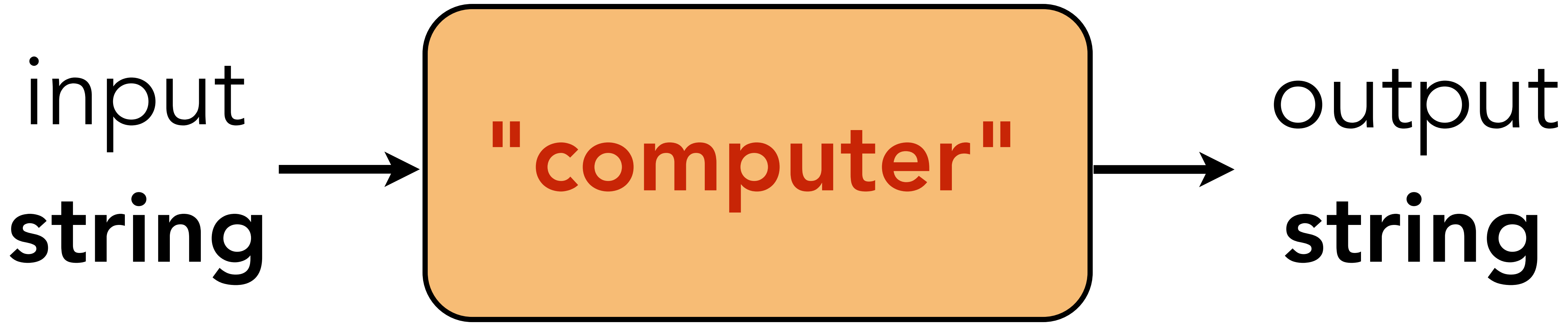
How can we mathematically define them?

Can encode/represent data

(*numbers*, *text*, *pairs of numbers*, *graphs*, *images*,…)

with a ***finite-length*** **(binary) string**.

This Chapter and Next Chapter



What is **computation**?

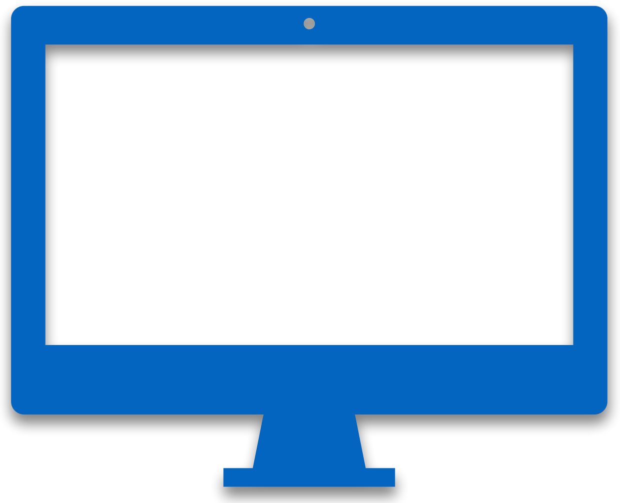
What is an **algorithm**?

How can we mathematically define them?

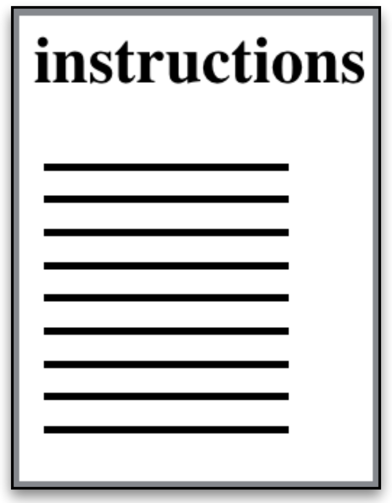
Terminology:

Computational Model — Allowed rules for information processing.

Machine = Computer = Program = Algorithm — An instantiation of the computational model.
(a specific sequence of information processing rules)



physical realization



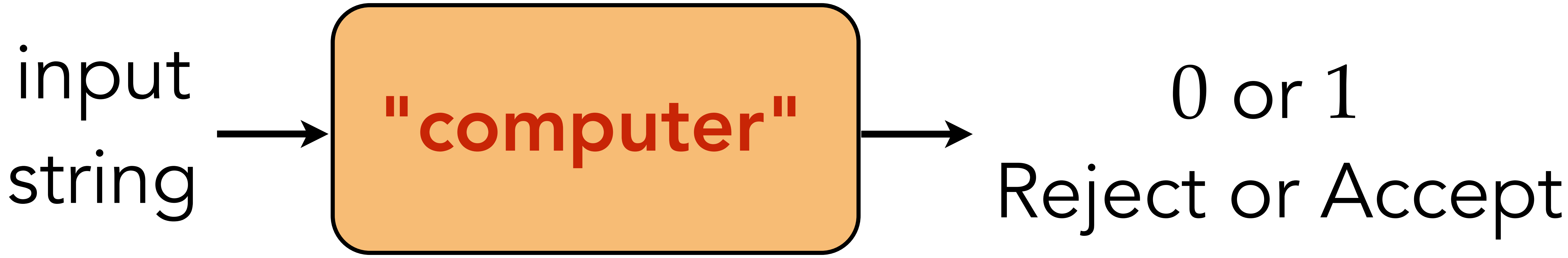
mathematical representation

2 Assumptions:

1. No "universal machines".

2. We only care about decision problems.

This Chapter and Next Chapter



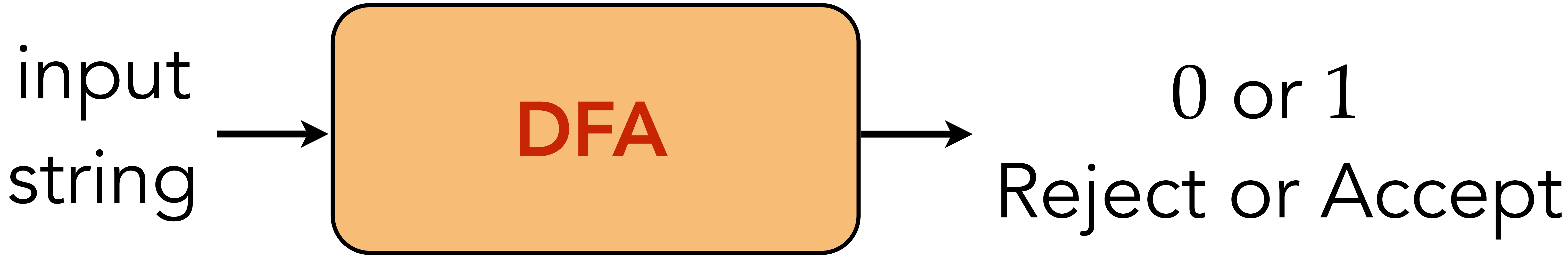
What is **computation**?

What is an **algorithm**?

How can we mathematically define them?

This Chapter

Deterministic Finite Automata (DFA)

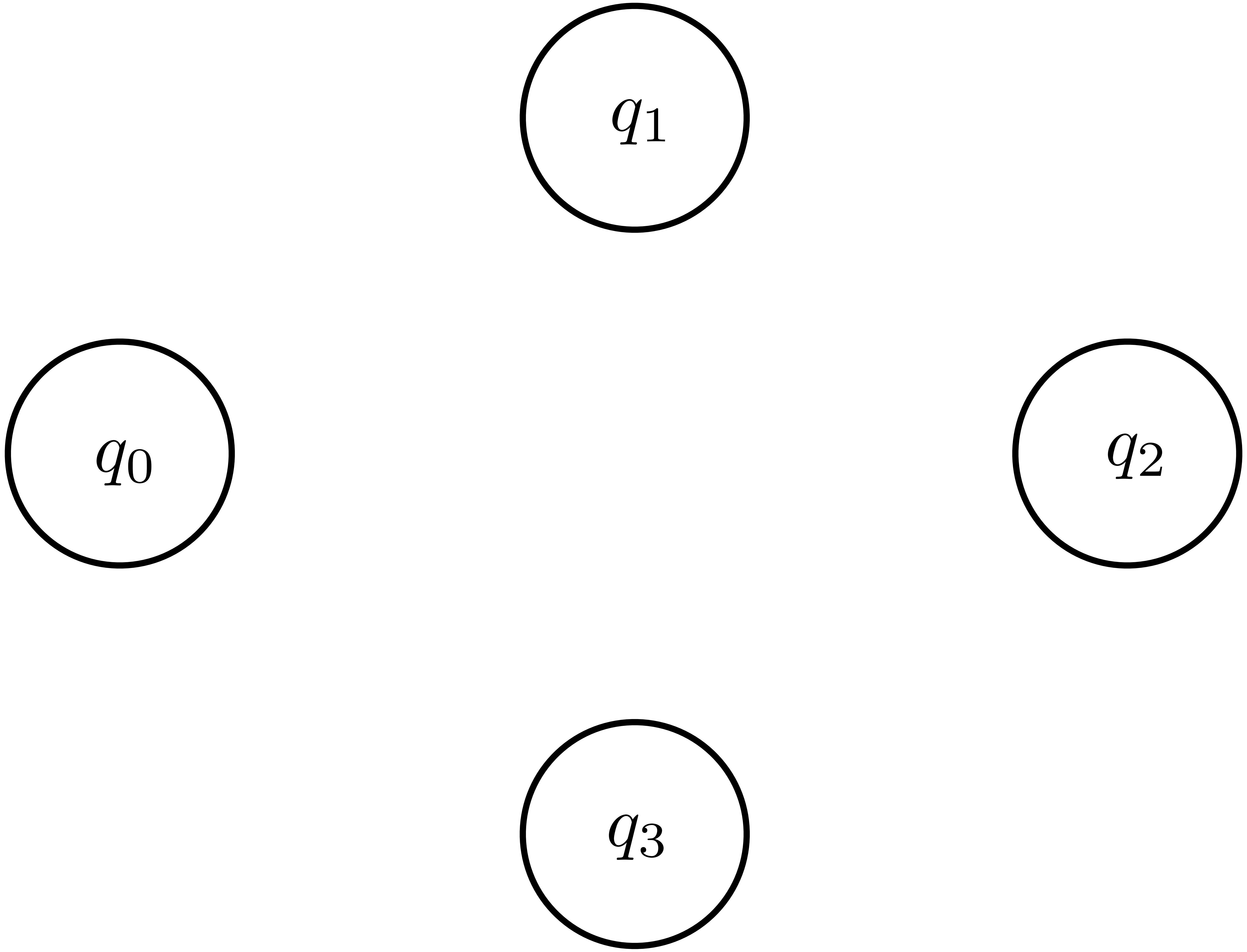


A restricted model of computation:

- *limited memory*
- *reads input from left to right, and accepts or rejects.*

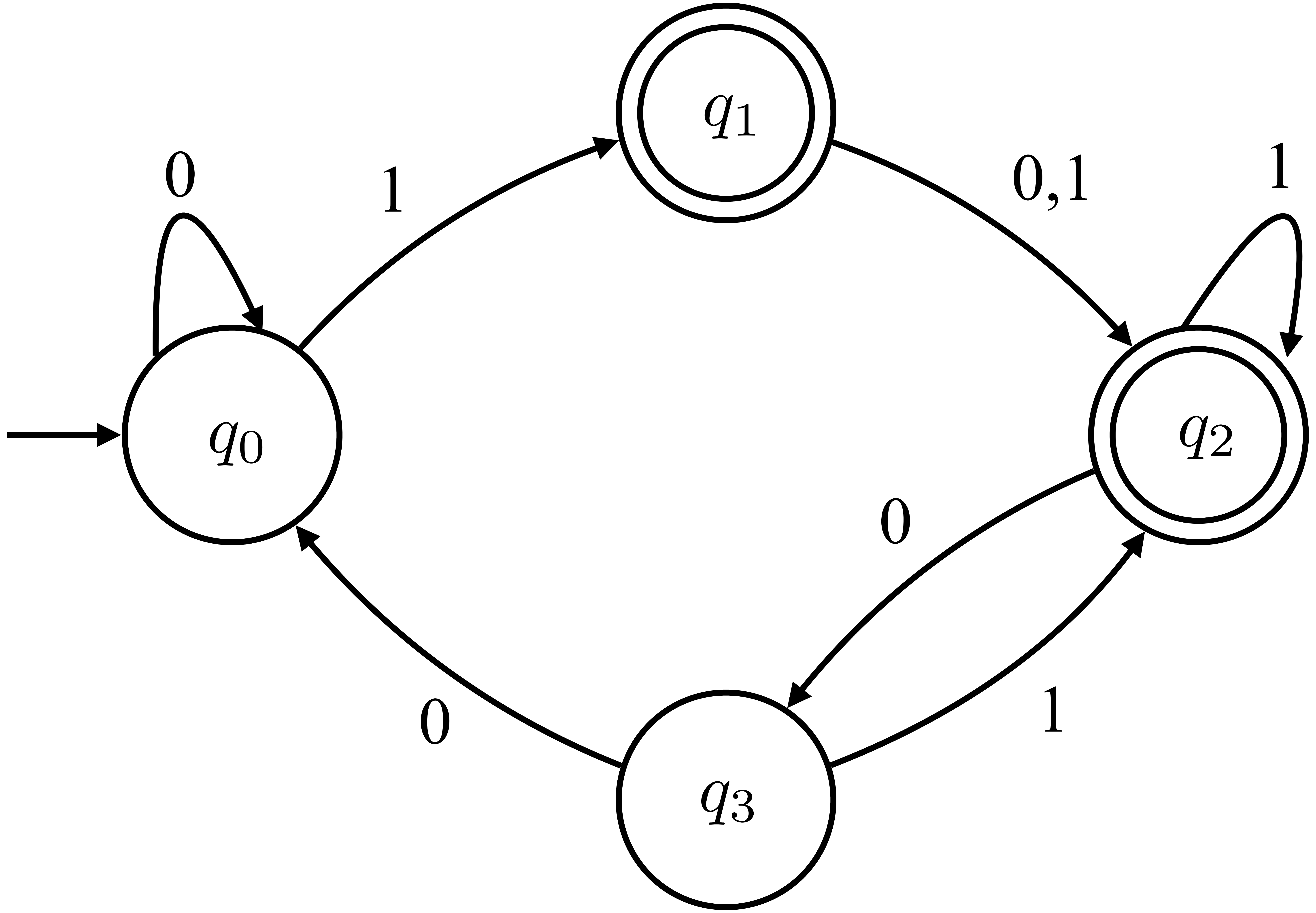
State diagram of a DFA

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



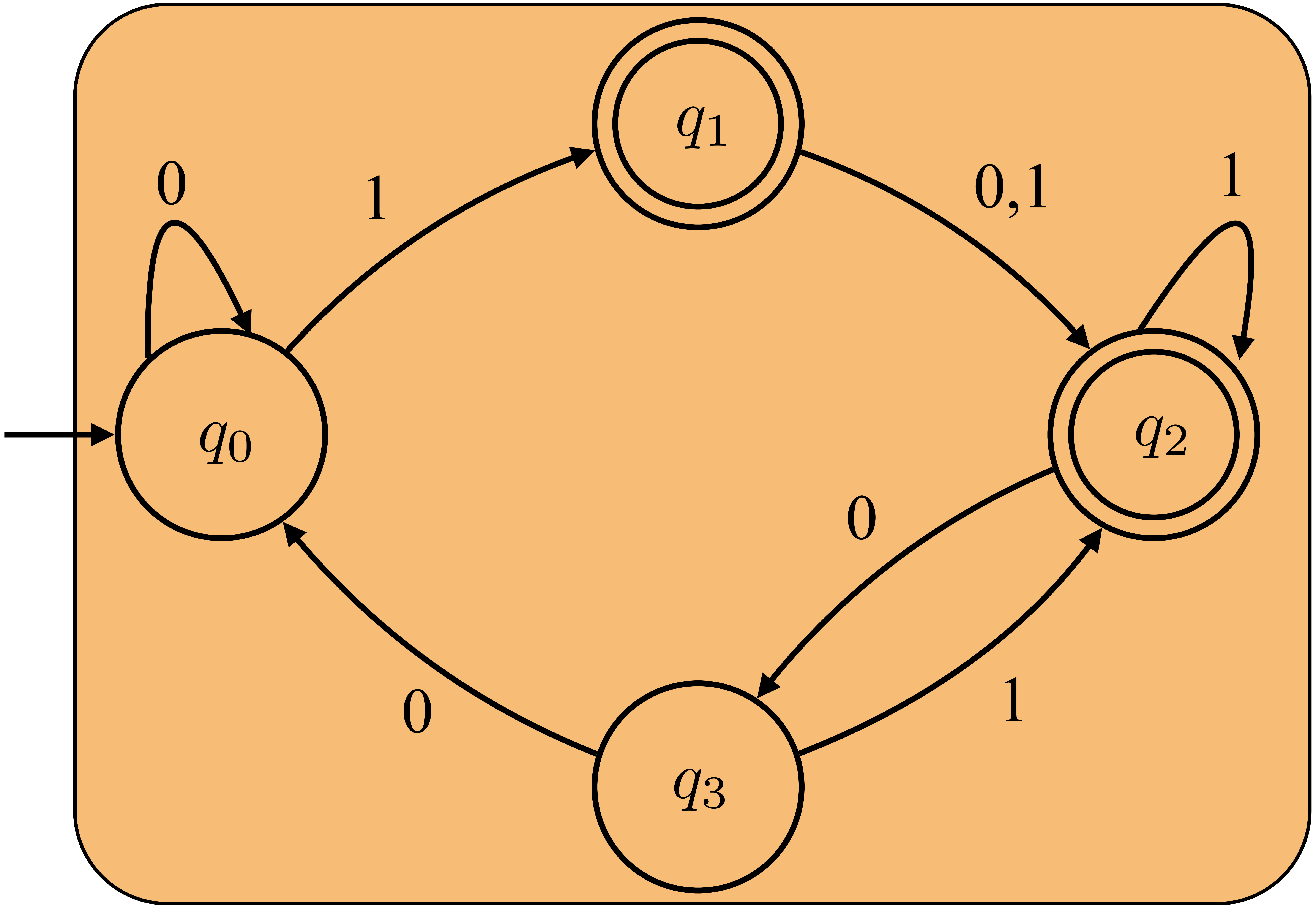
State diagram of a DFA

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



State diagram of a DFA

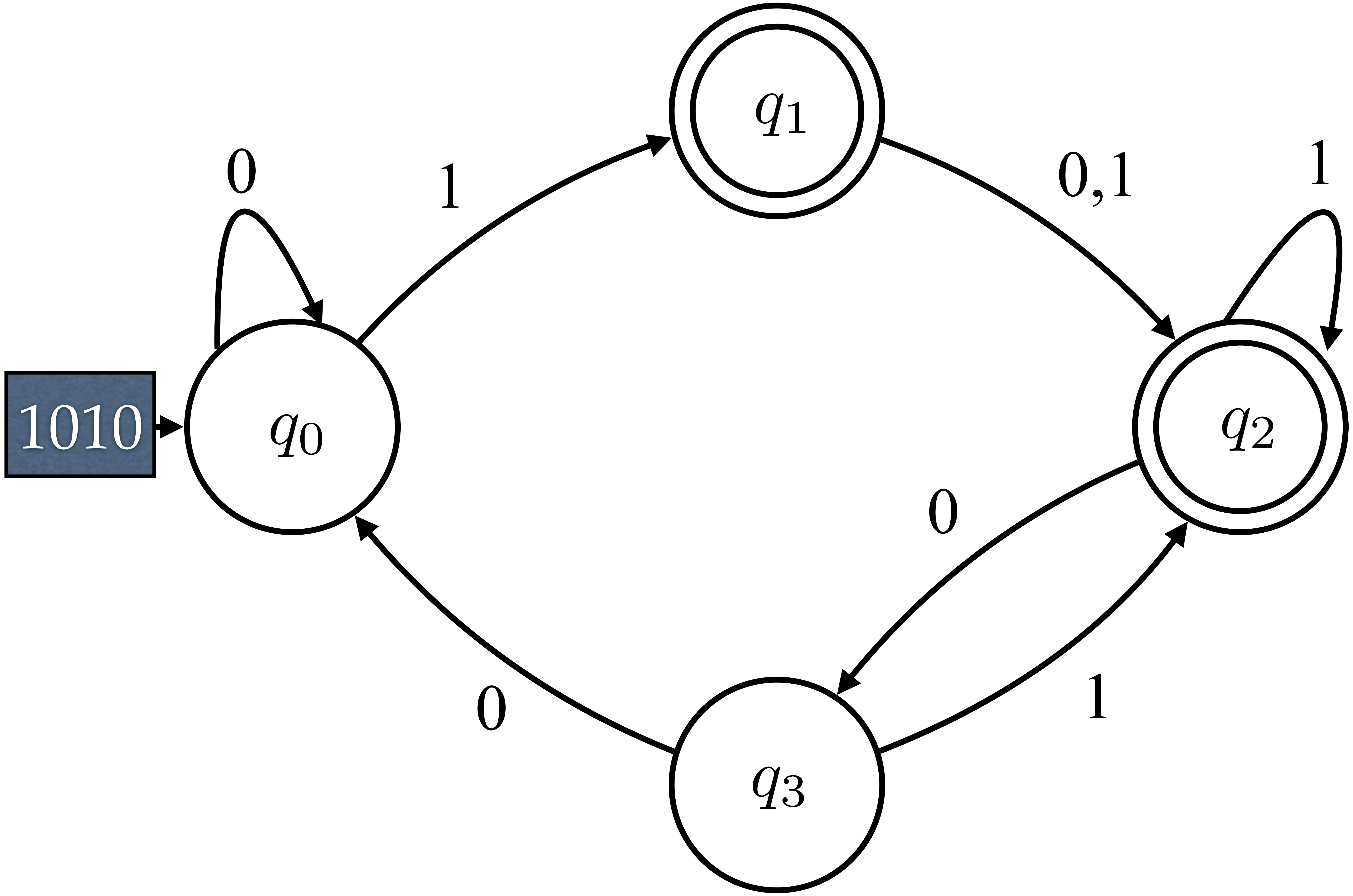
$\Sigma = \{0,1\}$



State diagram of a DFA

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

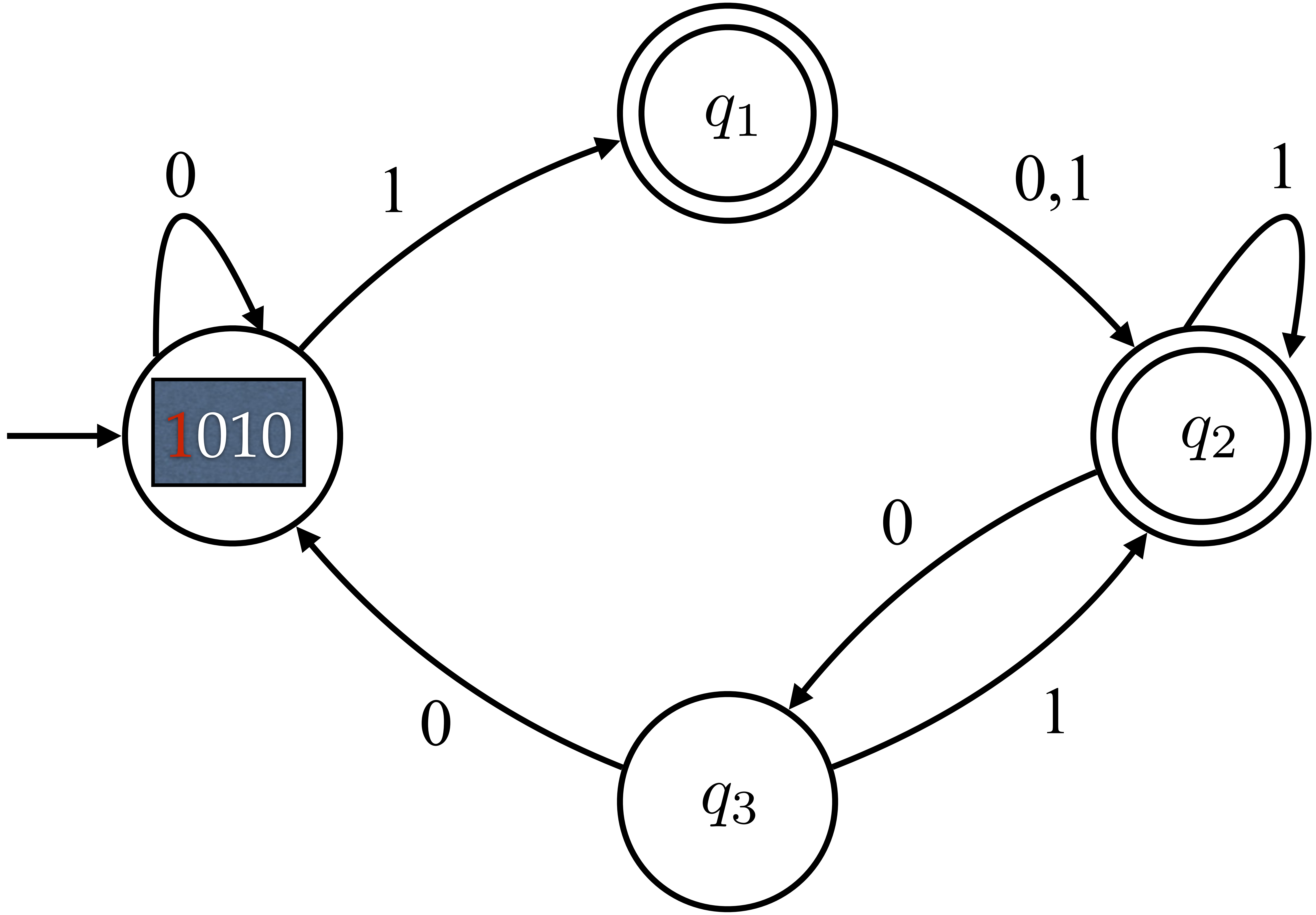
Input: 1010



State diagram of a DFA

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

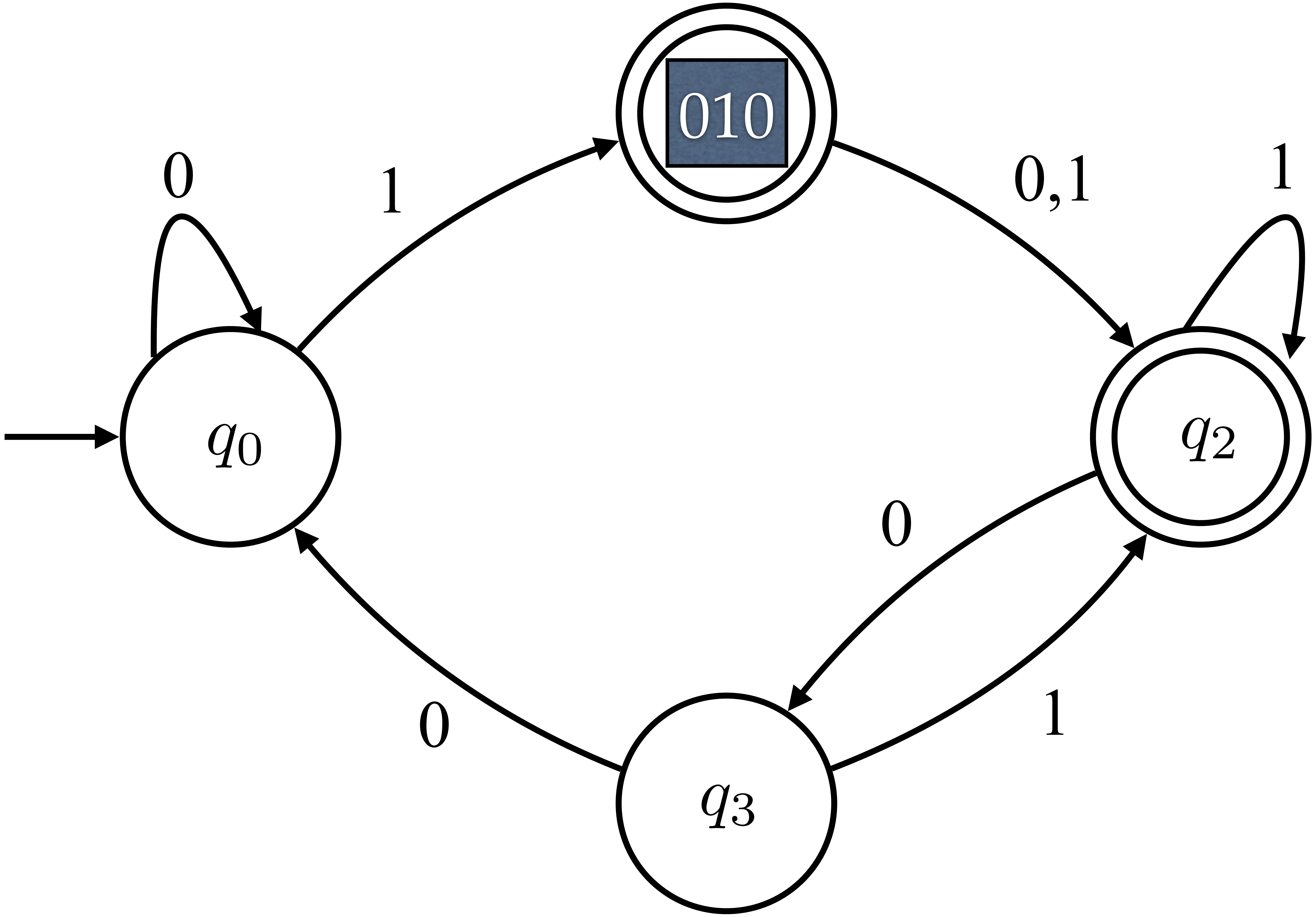
Input: 1010



State diagram of a DFA

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

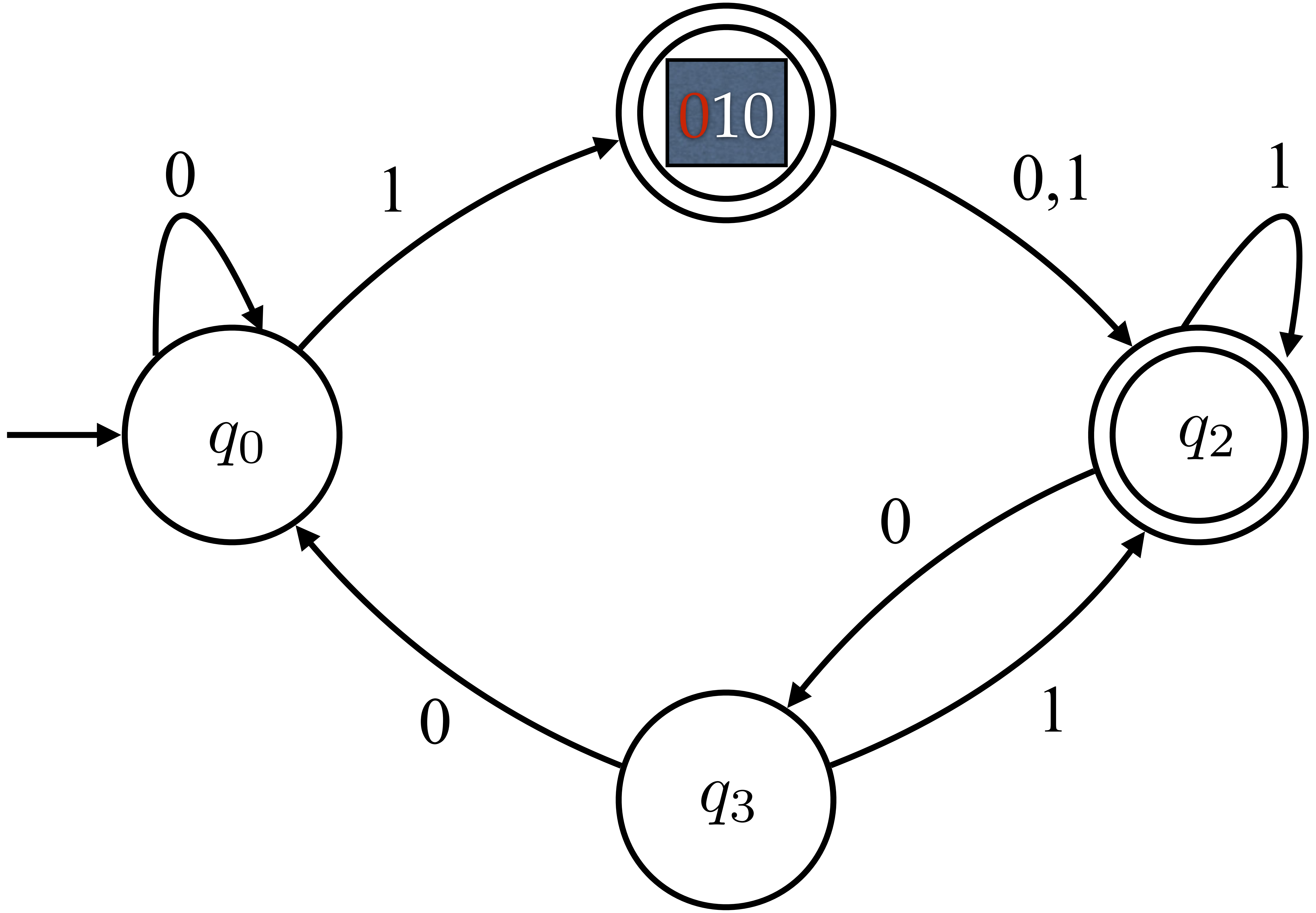
Input: 1010



State diagram of a DFA

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

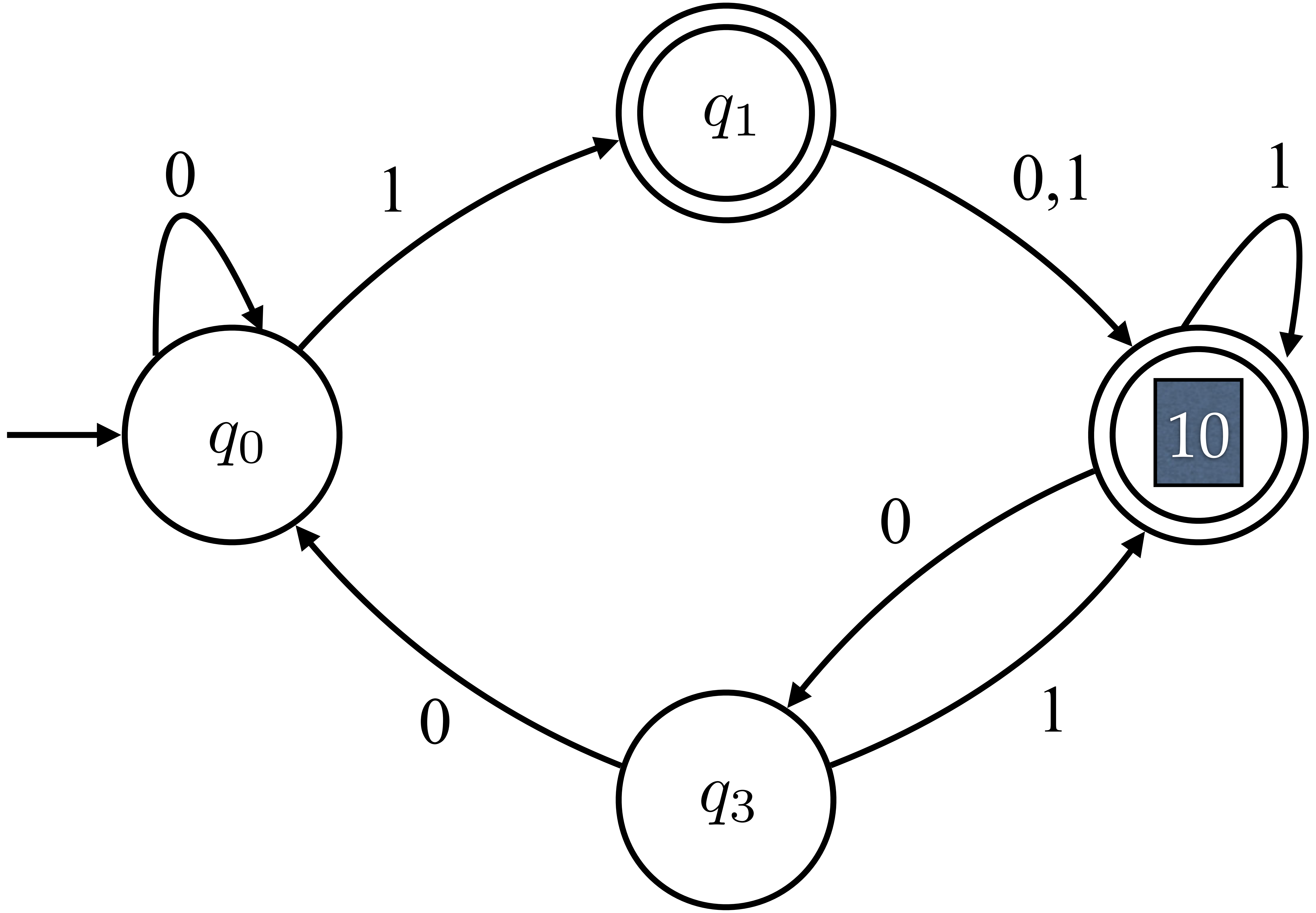
Input: 1010



State diagram of a DFA

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

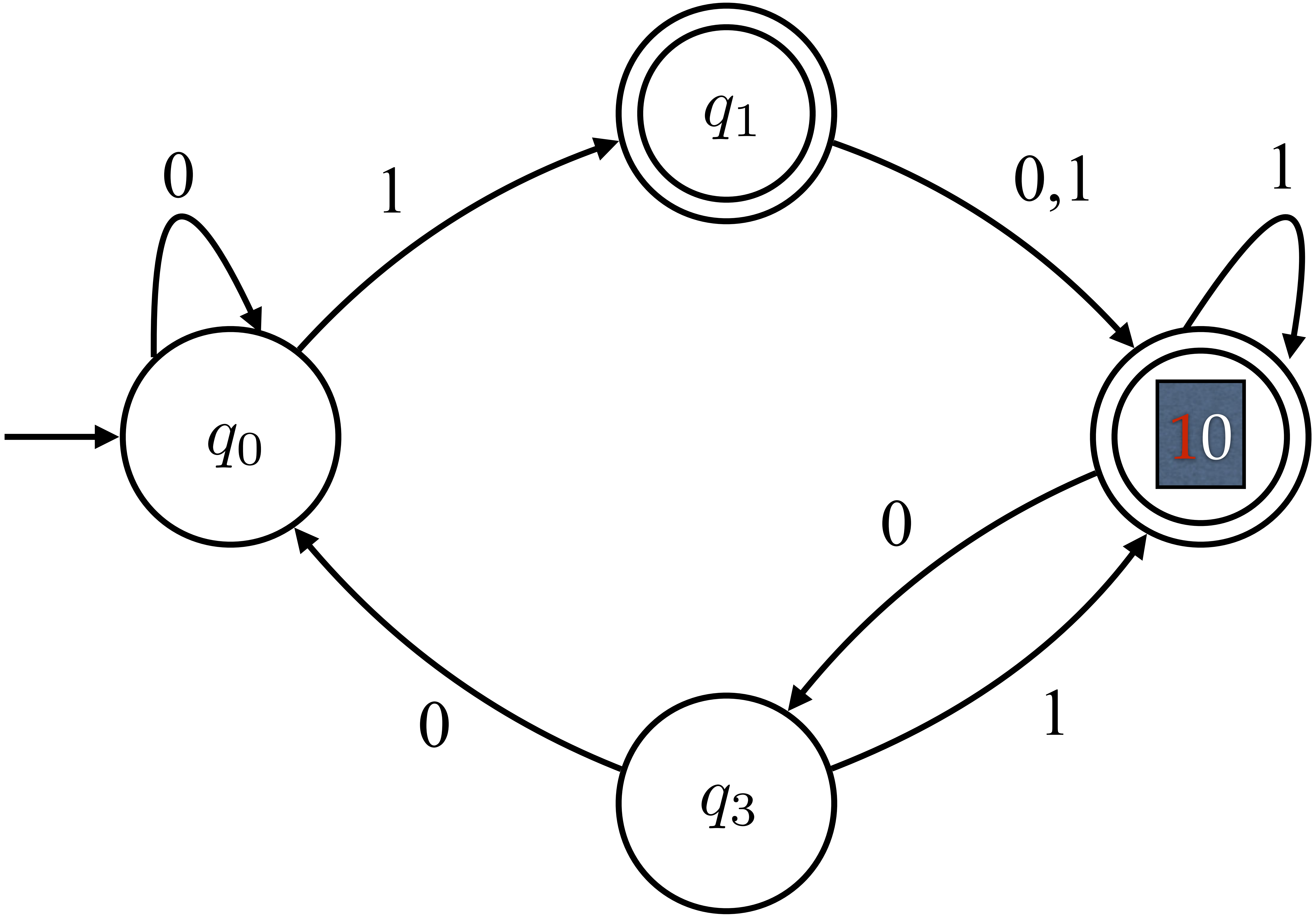
Input: 1010



State diagram of a DFA

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

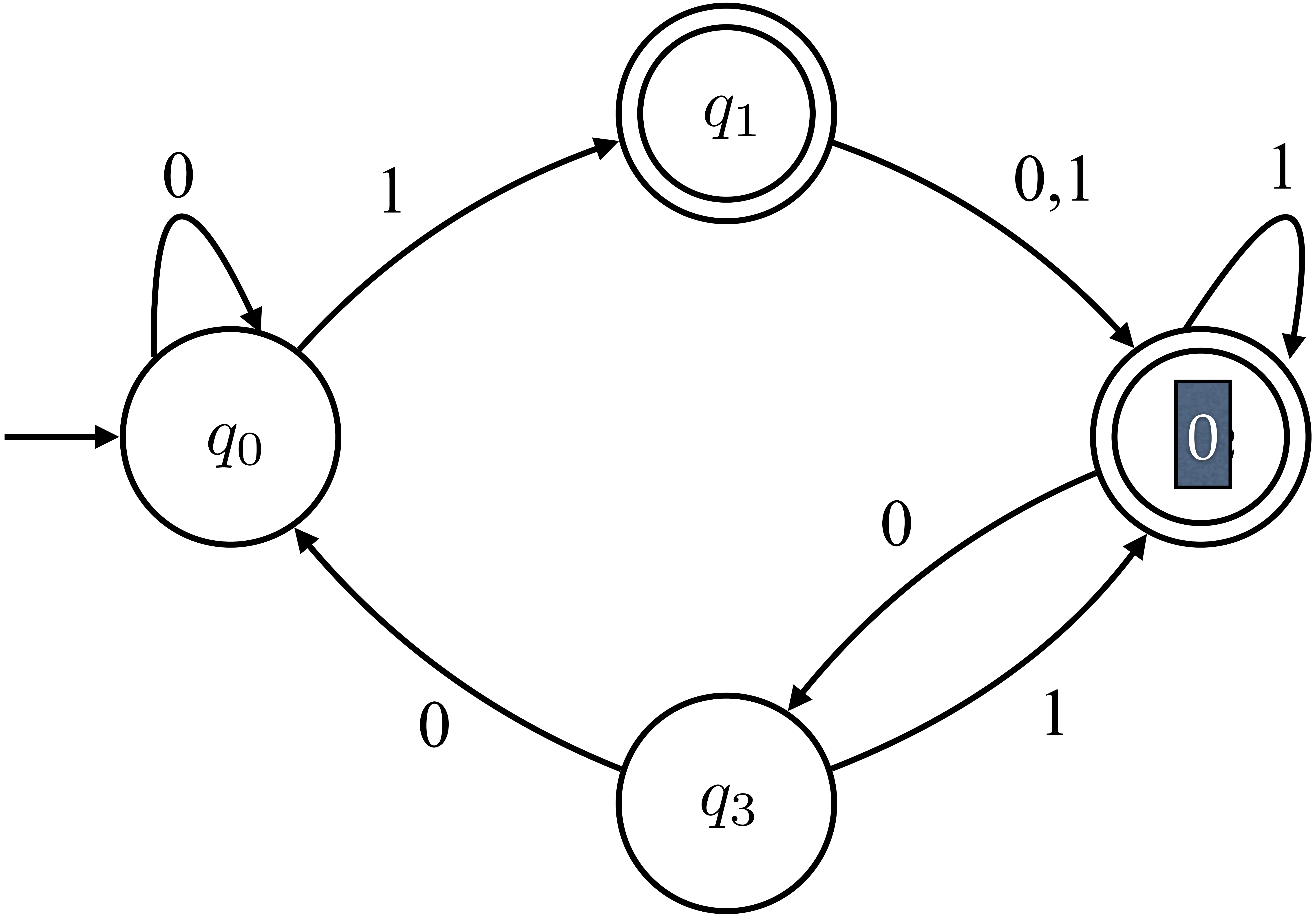
Input: 1010



State diagram of a DFA

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

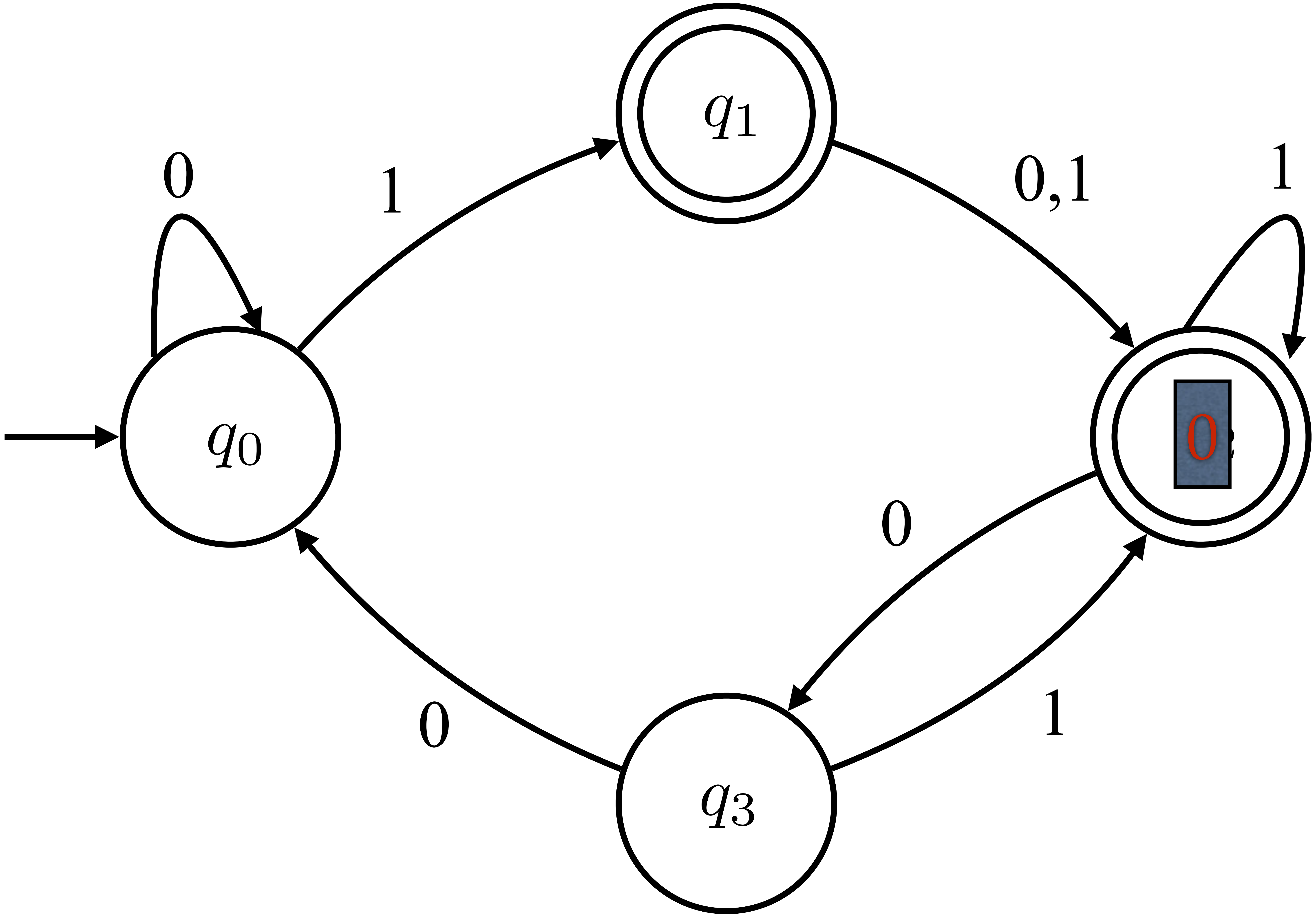
Input: 1010



State diagram of a DFA

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

Input: 1010

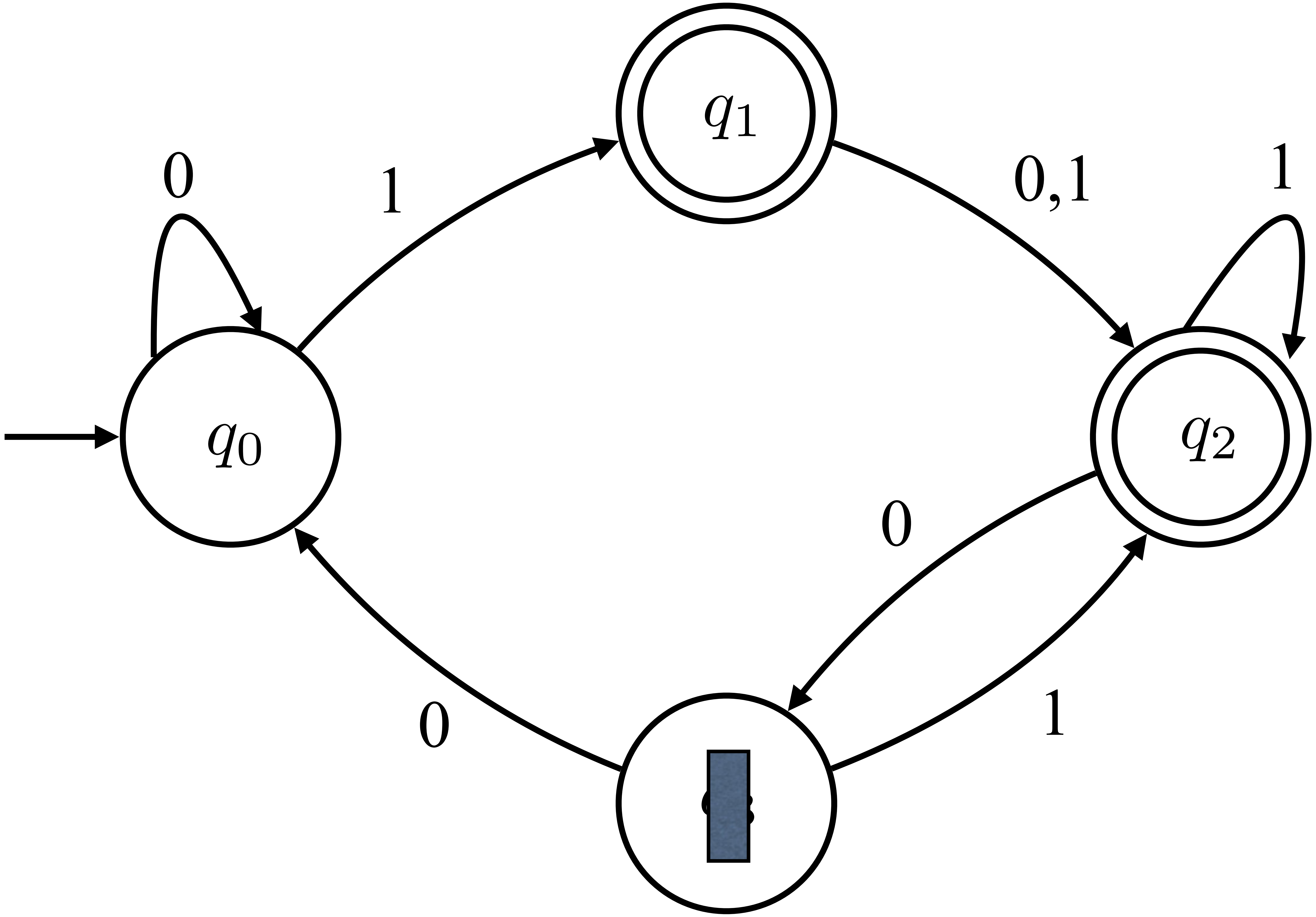


State diagram of a DFA

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

Input: 1010

Decision: REJECT

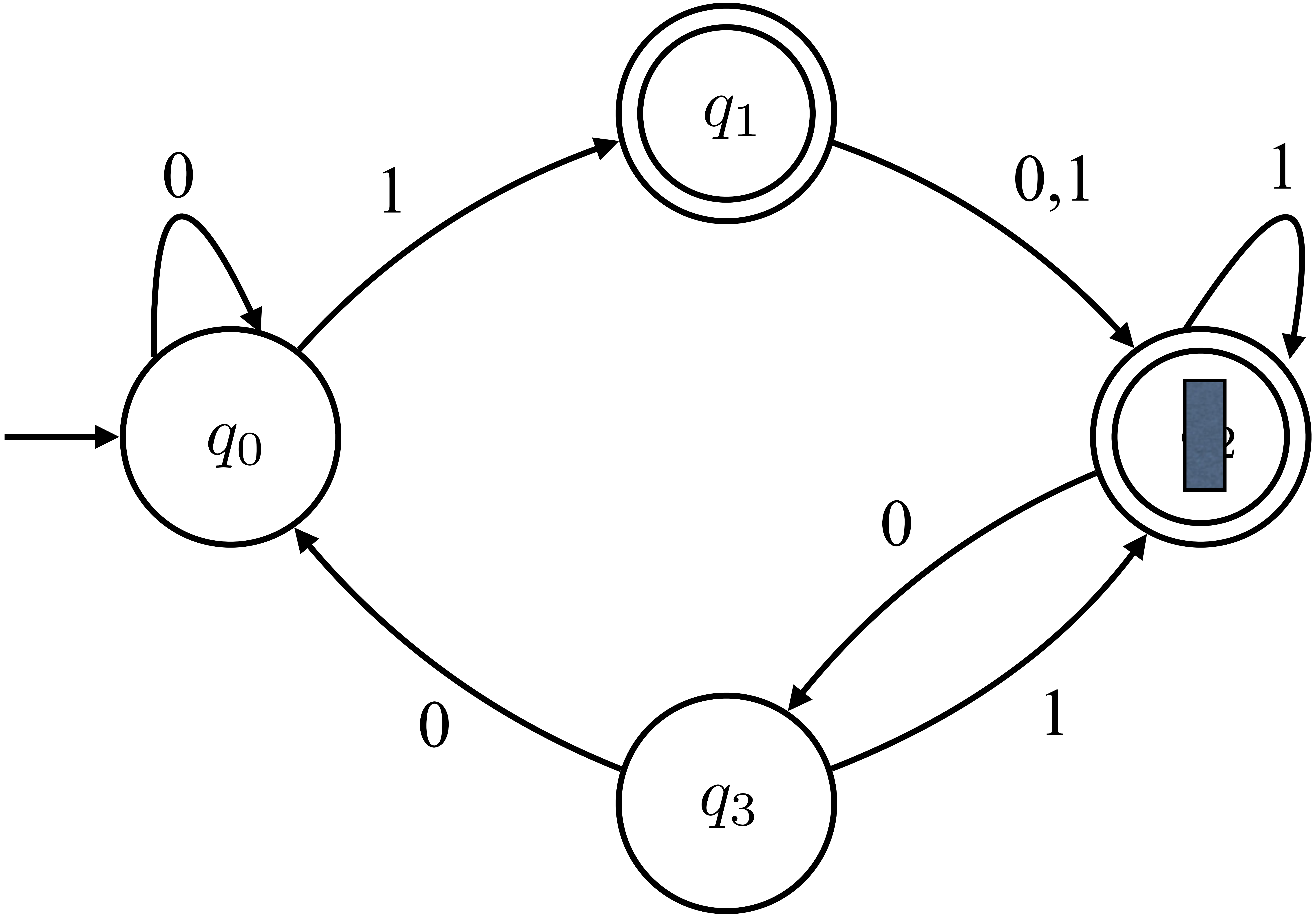


State diagram of a DFA

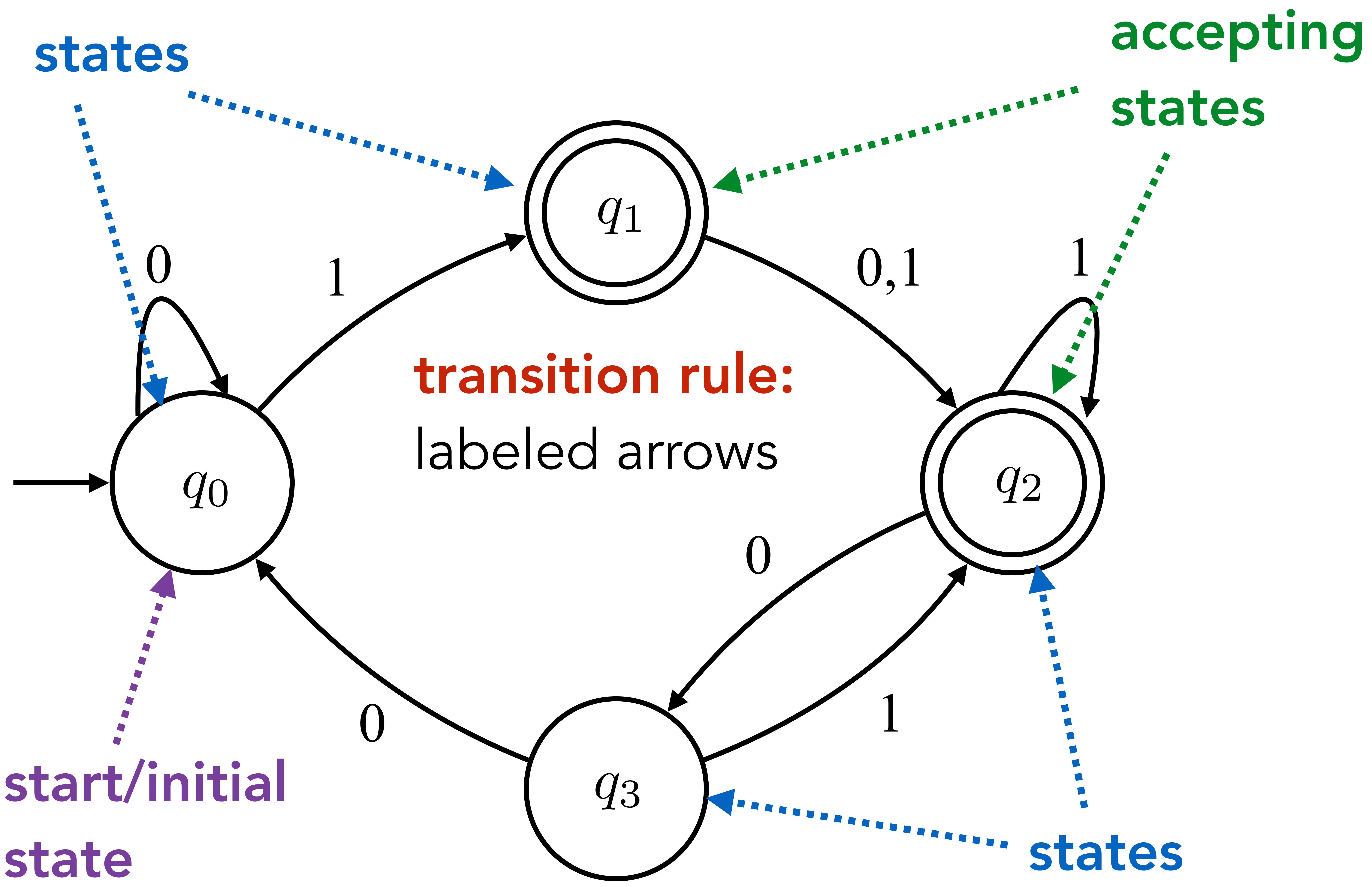
$\Sigma = \{0,1\}$

Input: 10101

Decision: ACCEPT



Anatomy of a DFA



Definition: Language solved by a DFA

Definition: Let M be a DFA and $L \subseteq \Sigma^*$ a language.

We say that M ***solves*** L if the following holds:

- if $w \in L$, M accepts w;
- if $w \notin L$, M rejects w.

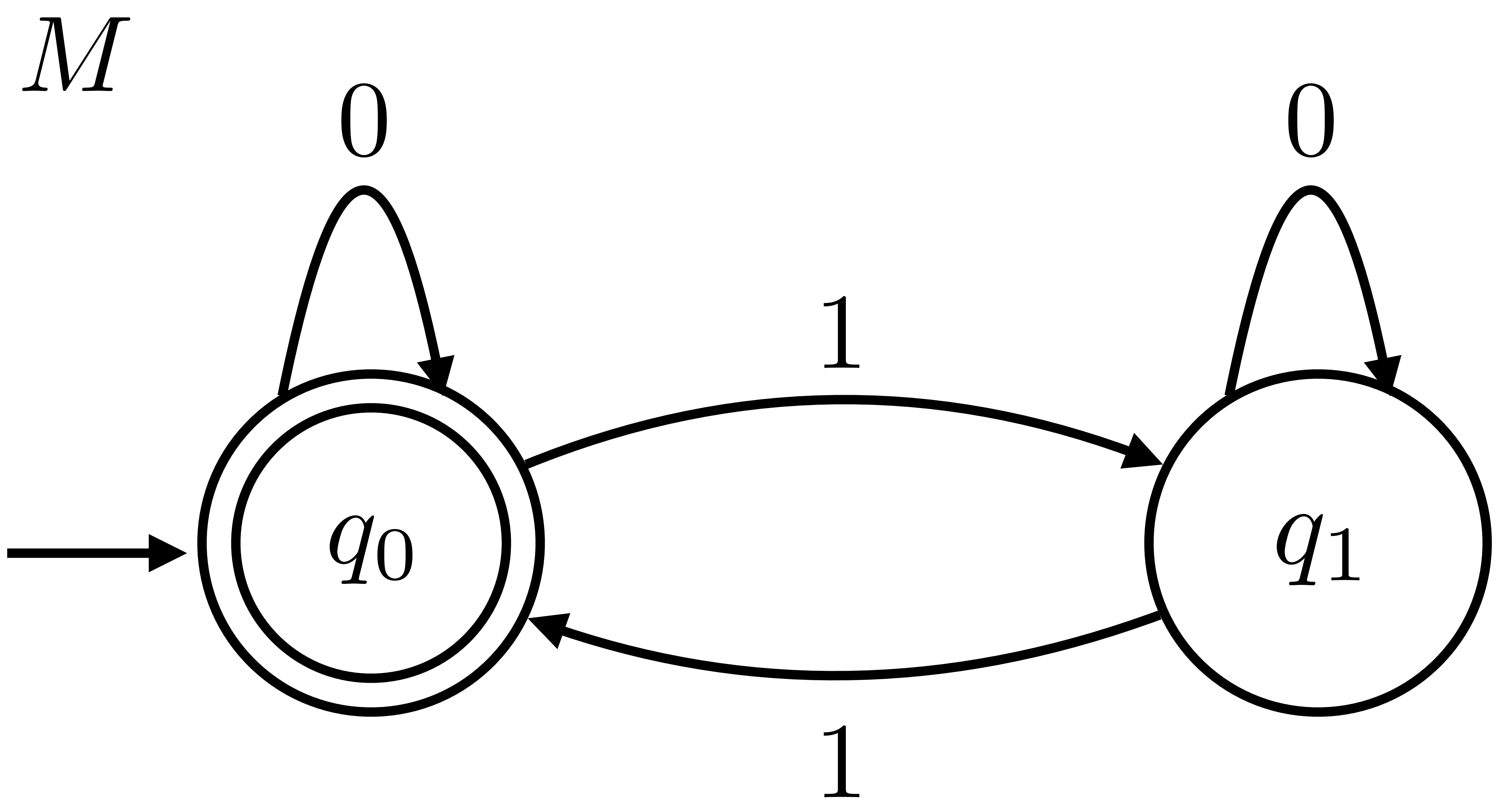
decides
computes

Useful Notation:

$L(M) =$ set of strings that M accepts.

$L(M)$ is ***the*** language that M solves/decides/computes.

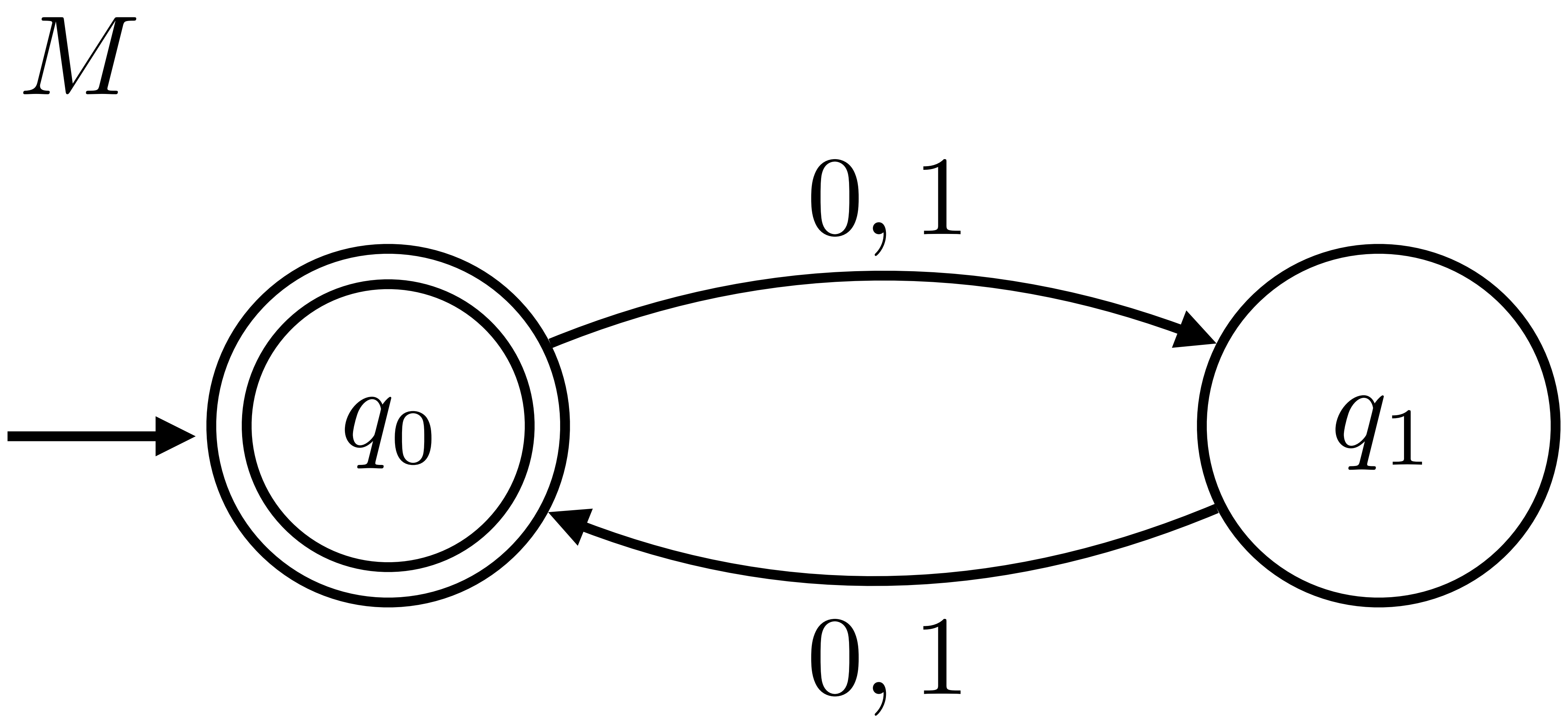
DFA Examples



$L(M) =$ all binary strings with even number of 1's

$= \{x \in \{0,1\}^* :$ x has an even number of 1's$\}$

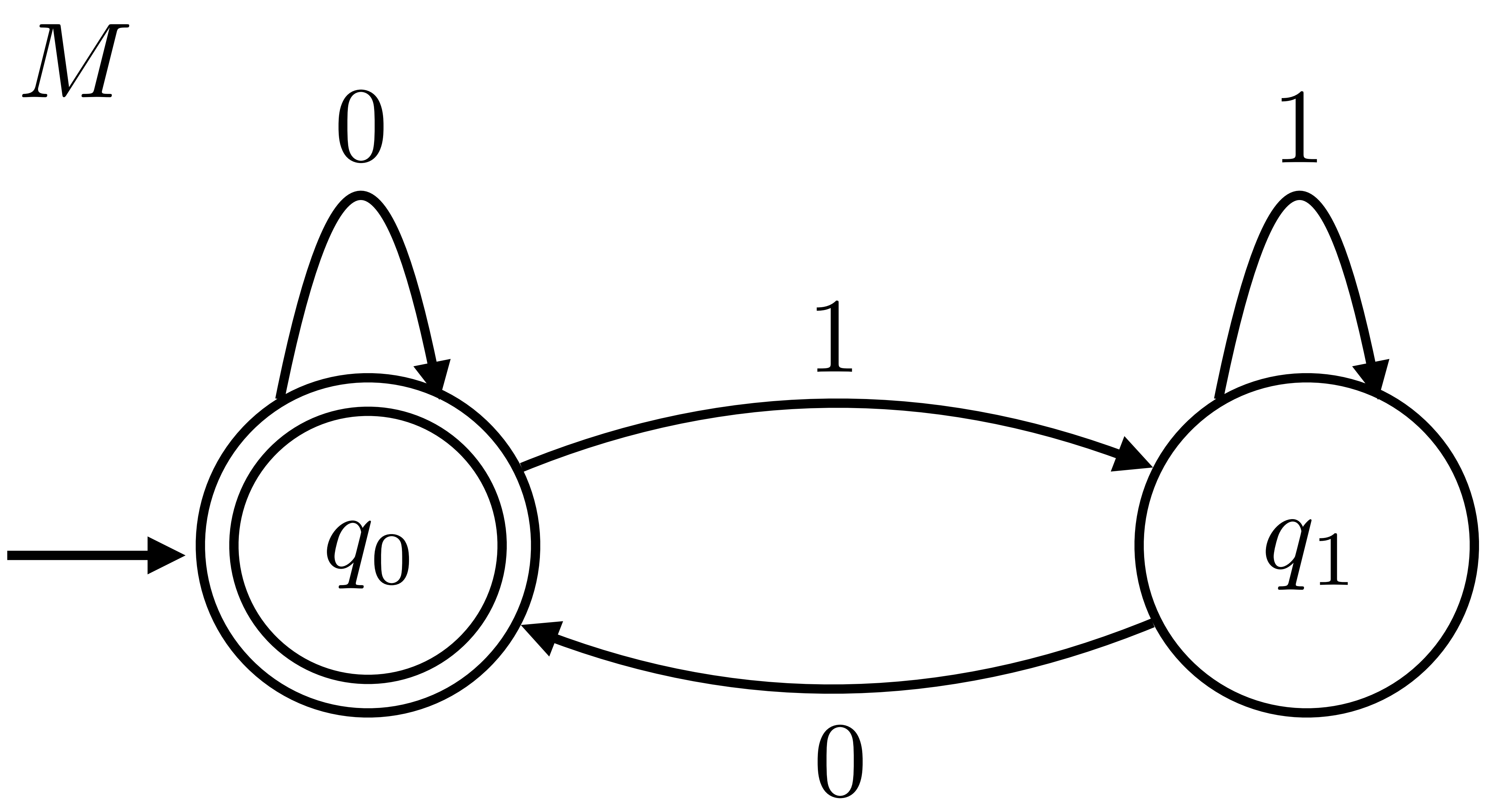
DFA Examples



$L(M) =$ all binary strings with even length

$= \{x \in \{0,1\}^* : \ |x| \text{ is even}\}$

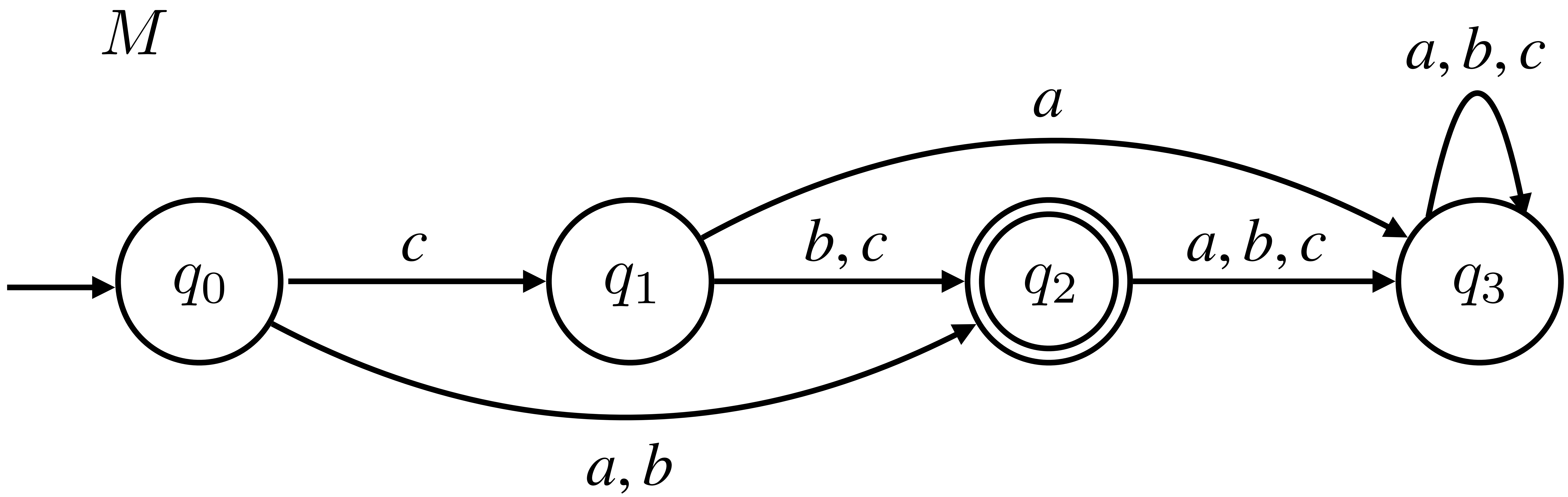
DFA Examples



$L(M) = \{x \in \{0,1\}^* : x \text{ ends with a } 0\} \cup \{\epsilon\}$

DFA Examples

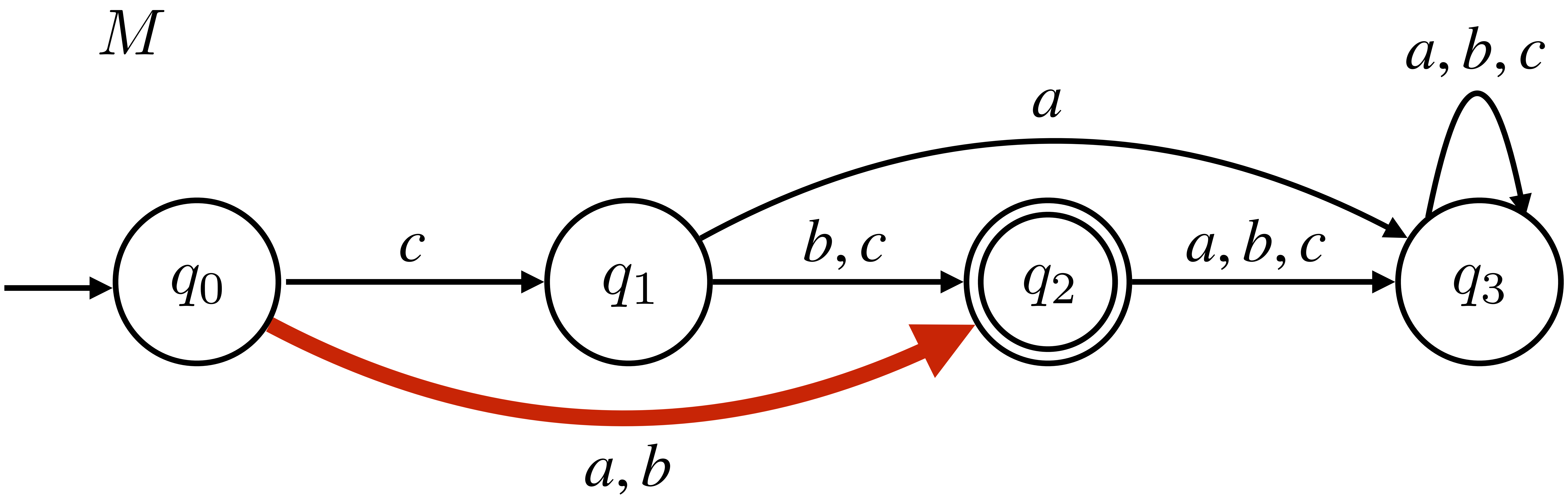
$$\Sigma = \{a, b, c\}$$



$L(M) = \{$

DFA Examples

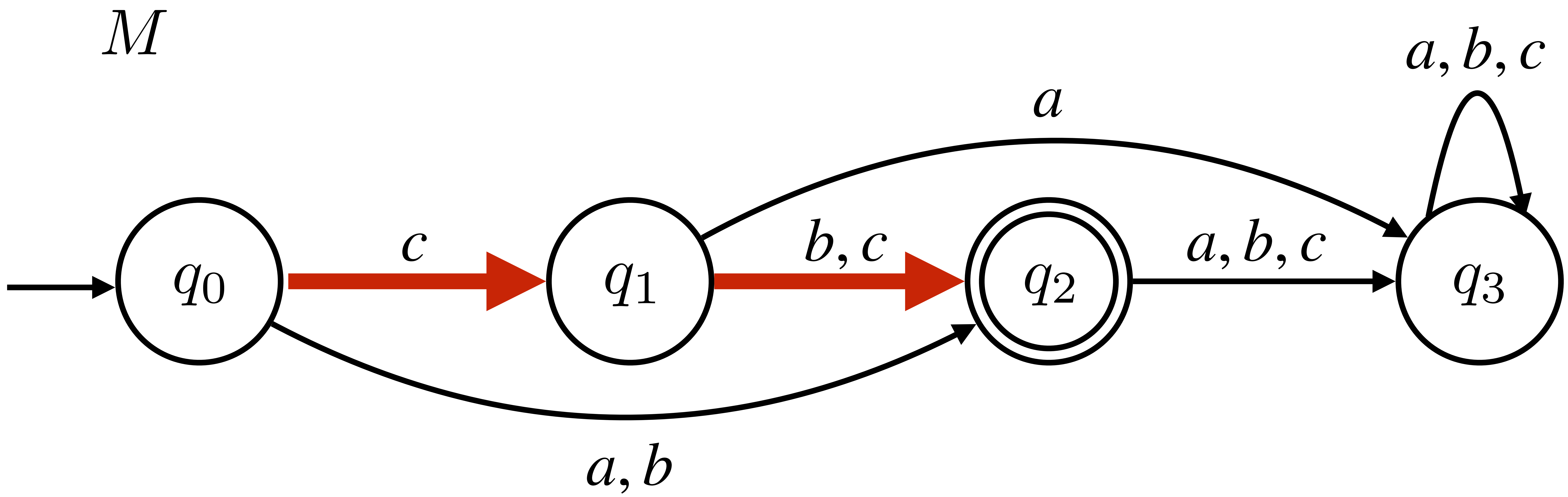
$\Sigma = \{a, b, c\}$



$L(M) = \{a, b$

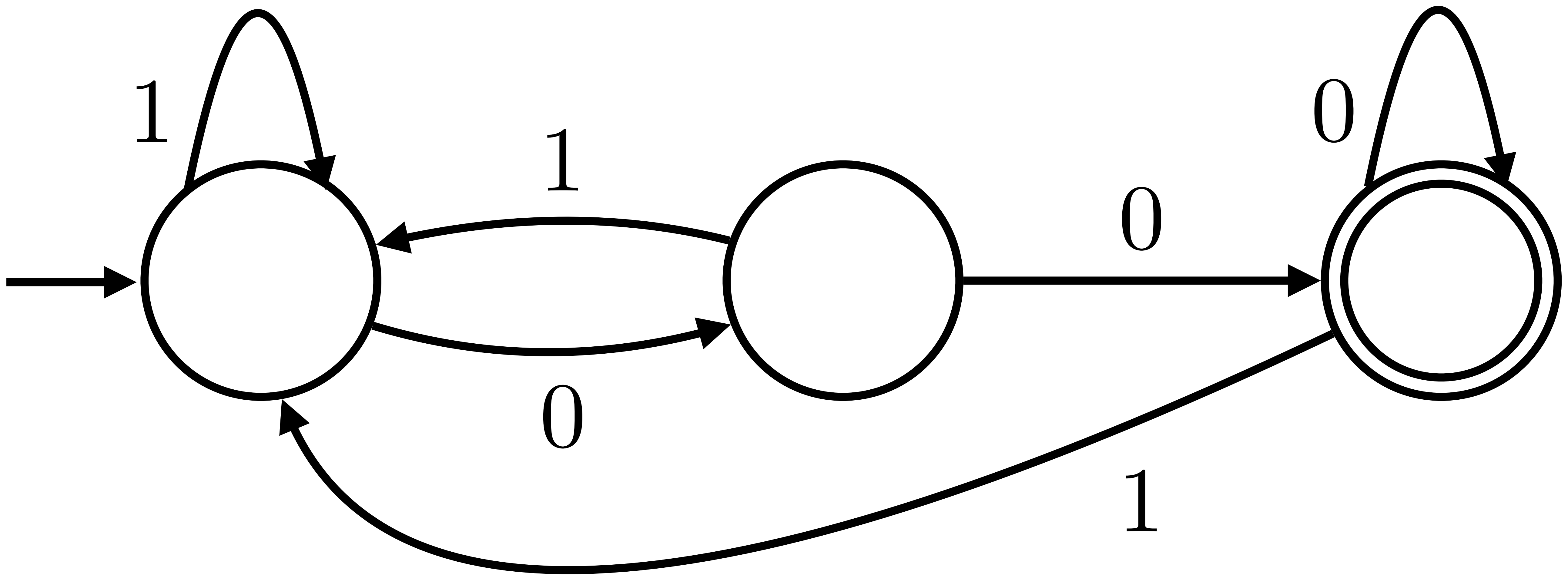
DFA Examples

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



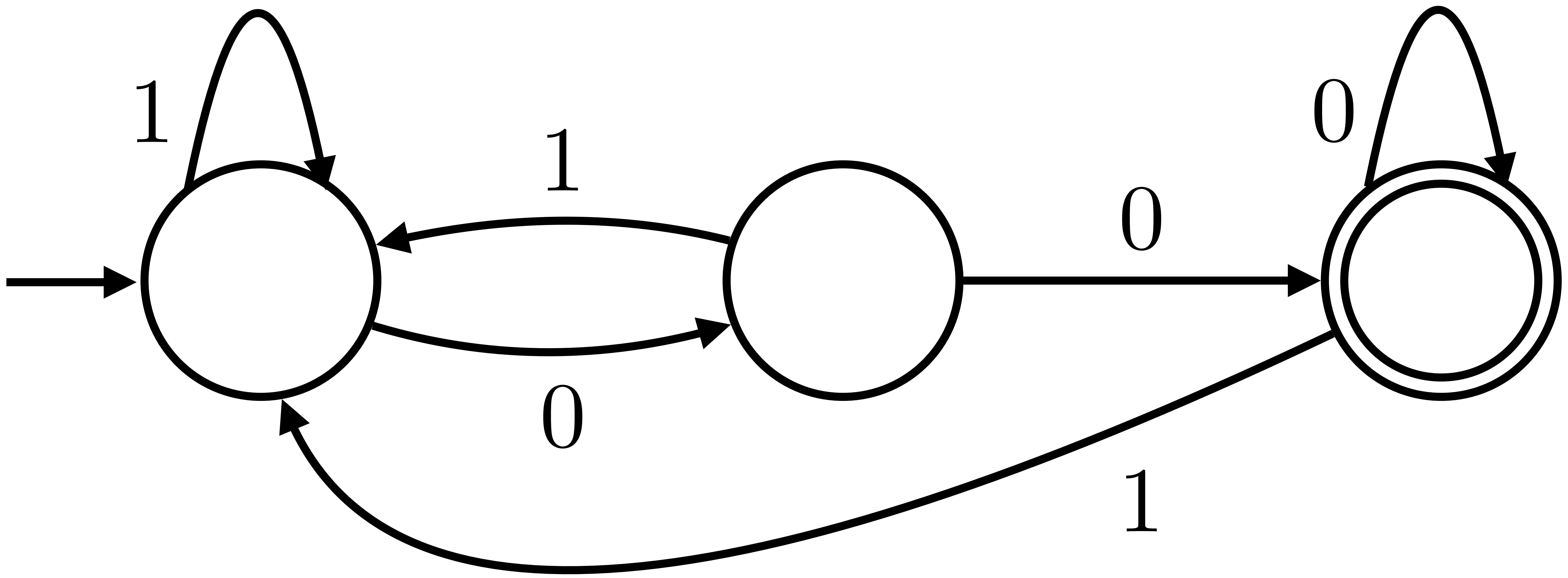
$L(M) = \{a, b, cb, cc\}$

poll.cs251.com



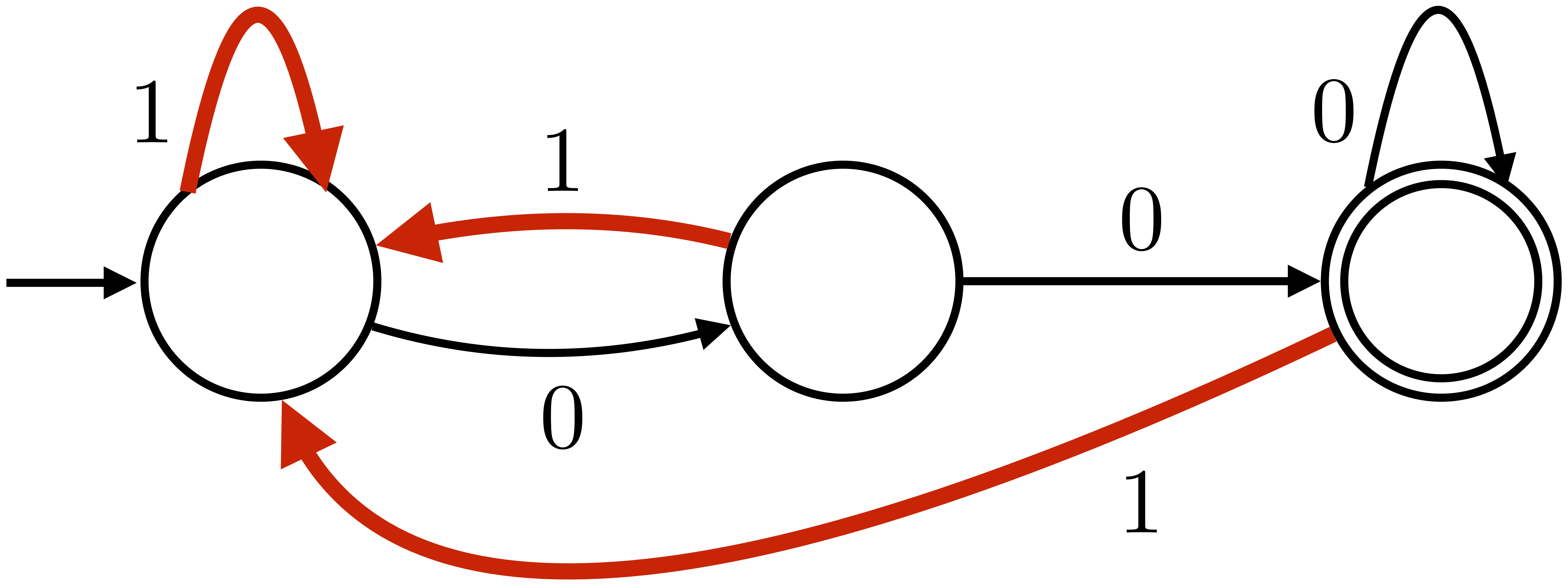
$L(M) =$

Poll - Answer



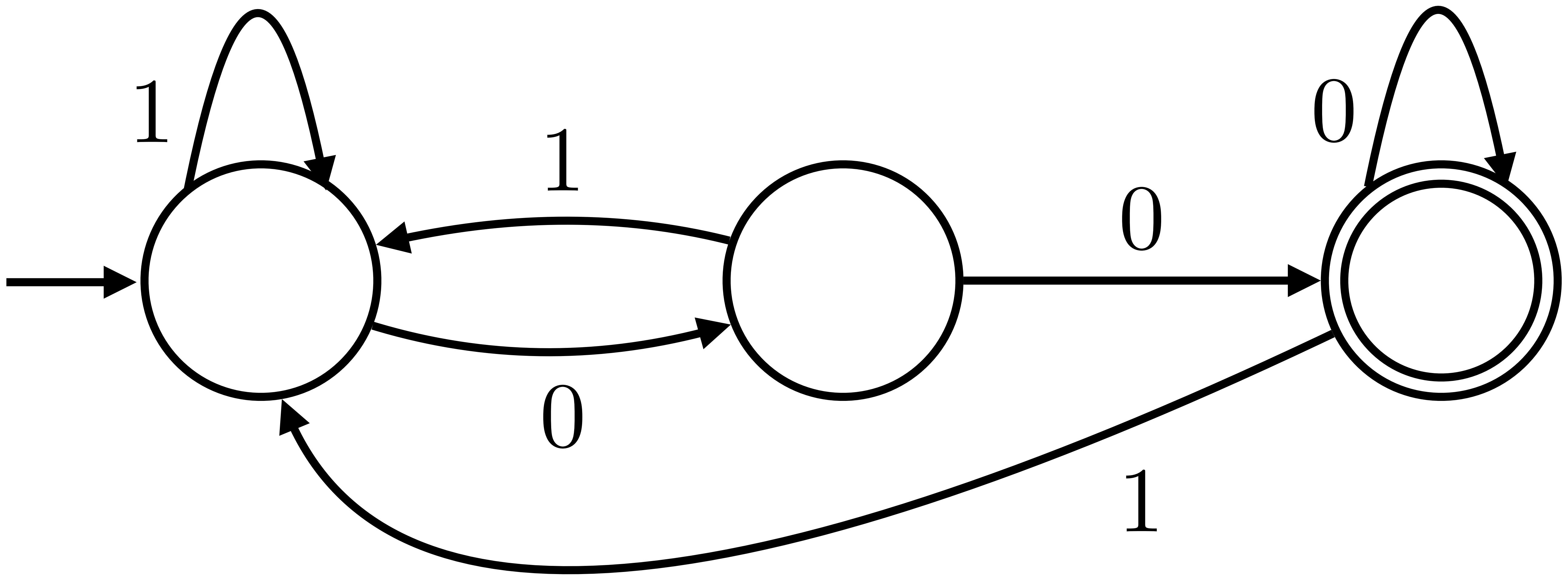
$L(M) =$

Poll - Answer



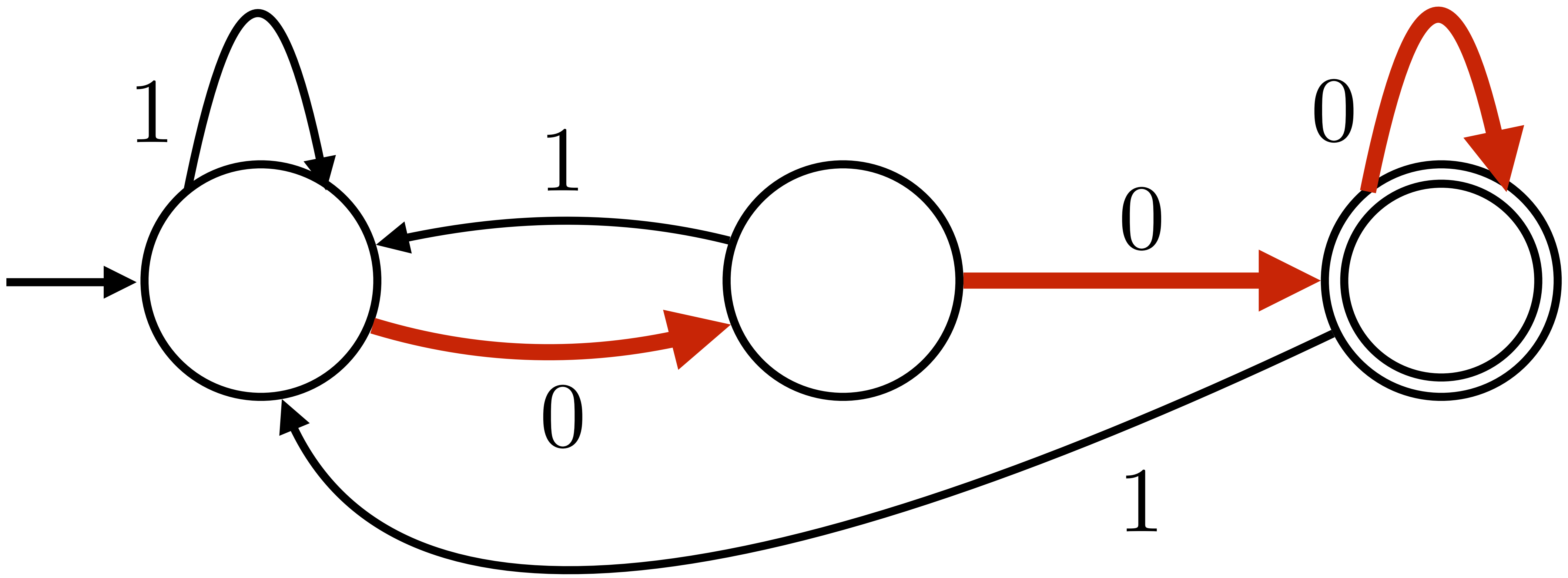
$L(M) =$

Poll - Answer



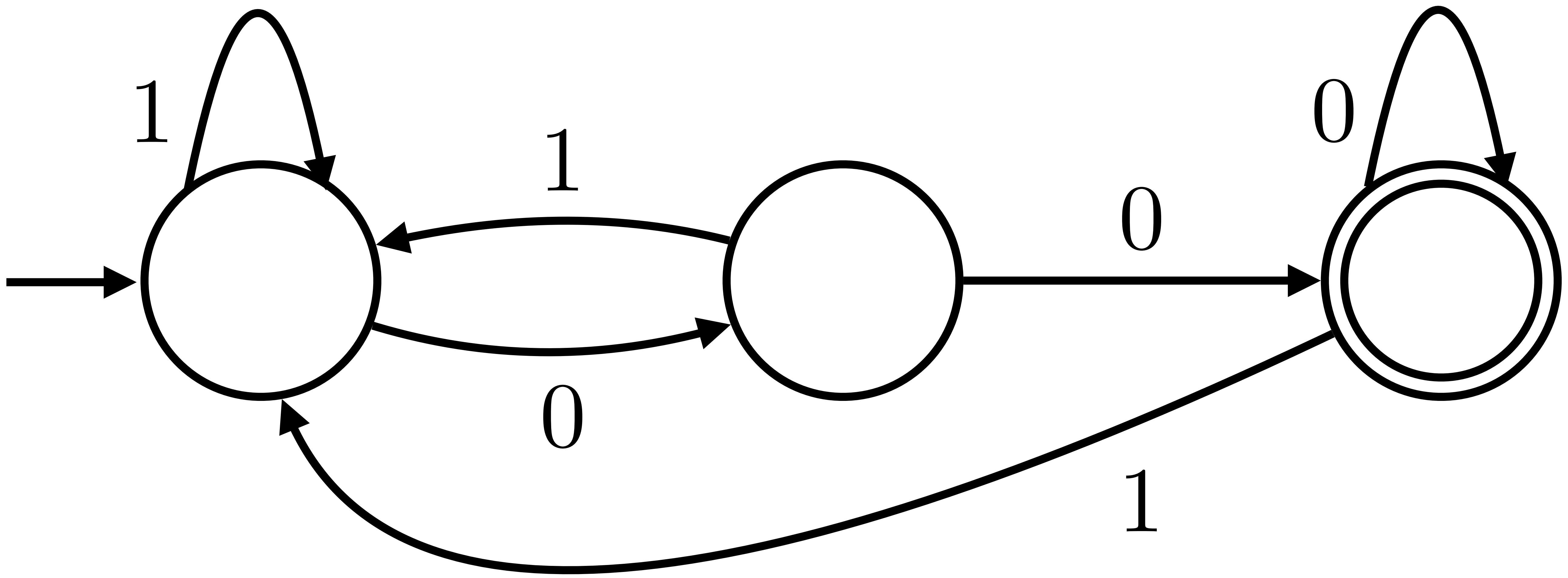
$L(M) =$

Poll - Answer



$L(M) =$

Poll - Answer



$L(M) =$ set of all strings ending in 00

If w ends with 00, M accepts.

If w does not end with 00, M rejects.

DFA construction practice

$L = \varnothing$

$L = \Sigma^*$

$L = \{110, 101\}$

$L = \{0,1\}^* \setminus \{110, 101\}$

$L = \{x \in \{0,1\}^* :$ x starts and ends with same bit$\}$

$L = \{x \in \{0,1\}^* :$ $|x|$ is divisible by 2 or 3$\}$

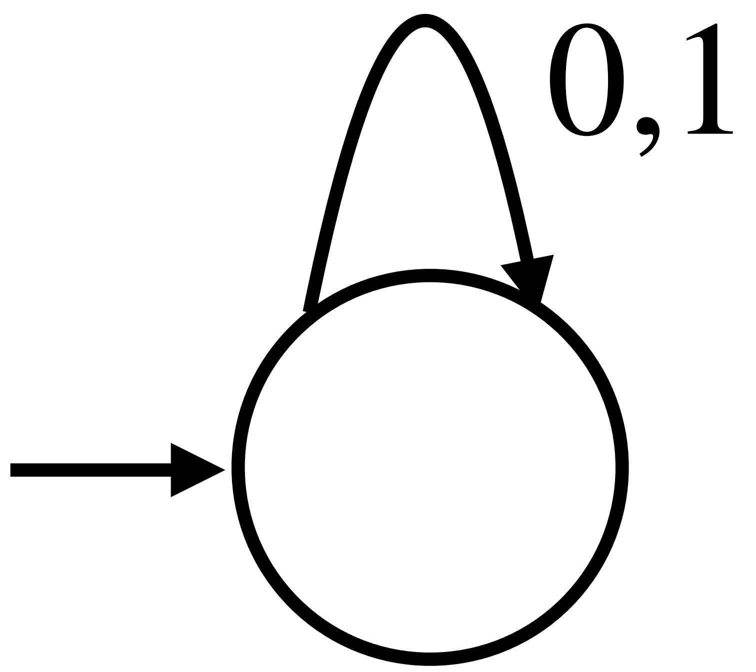
$L = \{\epsilon, 110, 110110, 110110110, \ldots\}$

$L = \{x \in \{0,1\}^* :$ x contains the substring 110$\}$

$L = \{x \in \{0,1\}^* :$ 10 and 01 occur equally often in $x\}$

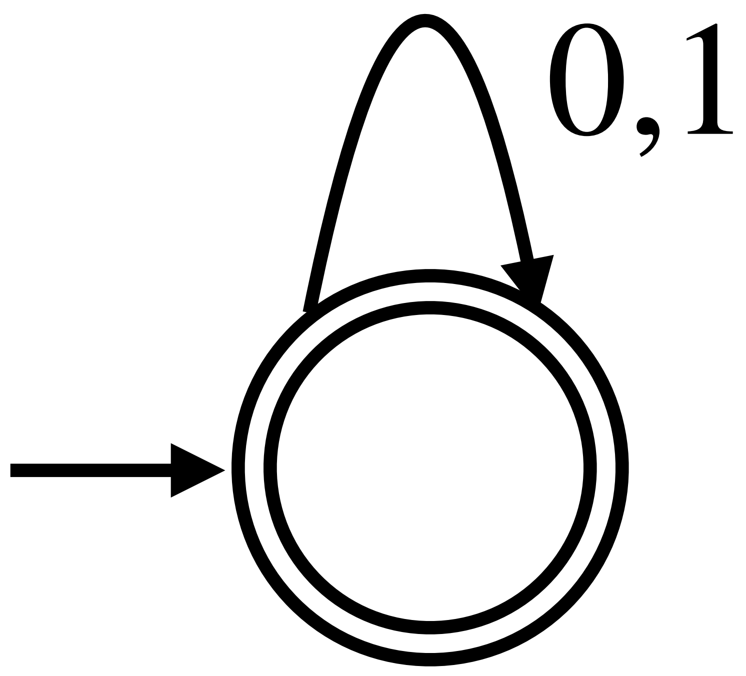
DFA construction practice

$L = \varnothing$



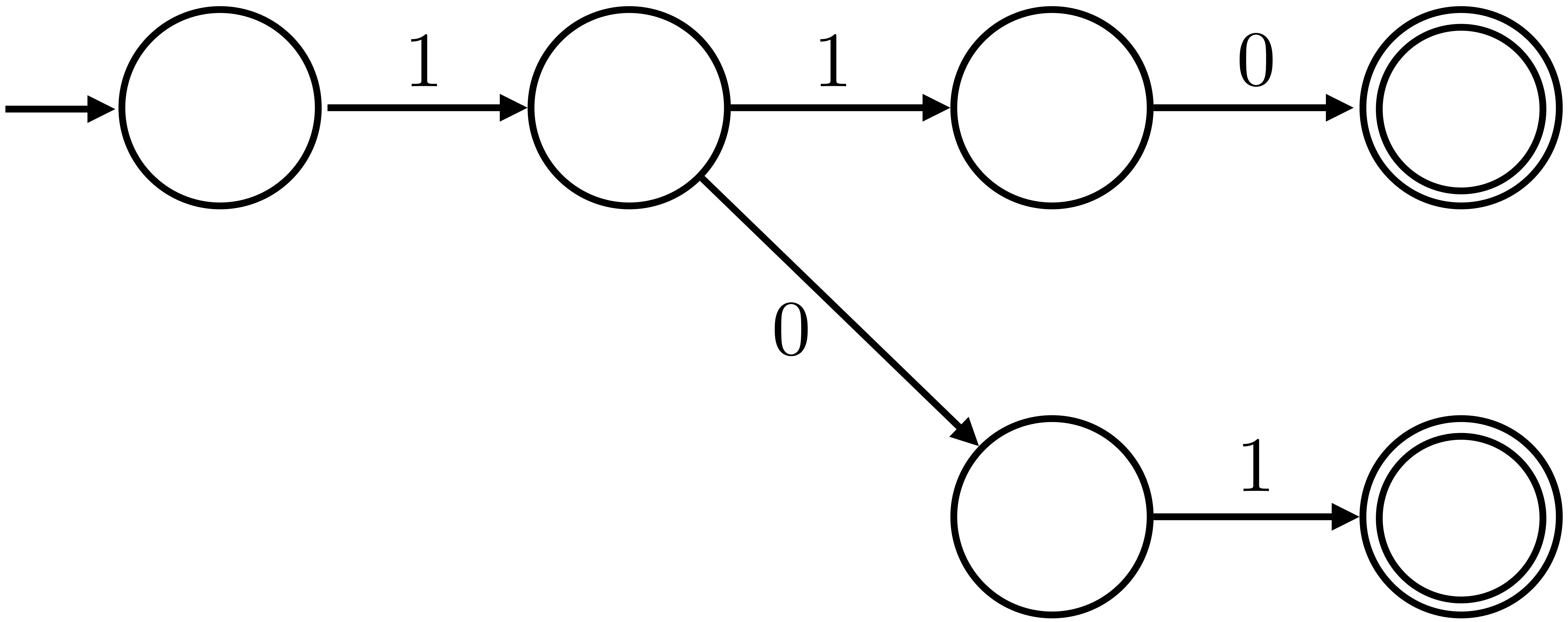
DFA construction practice

$L = \Sigma^*$



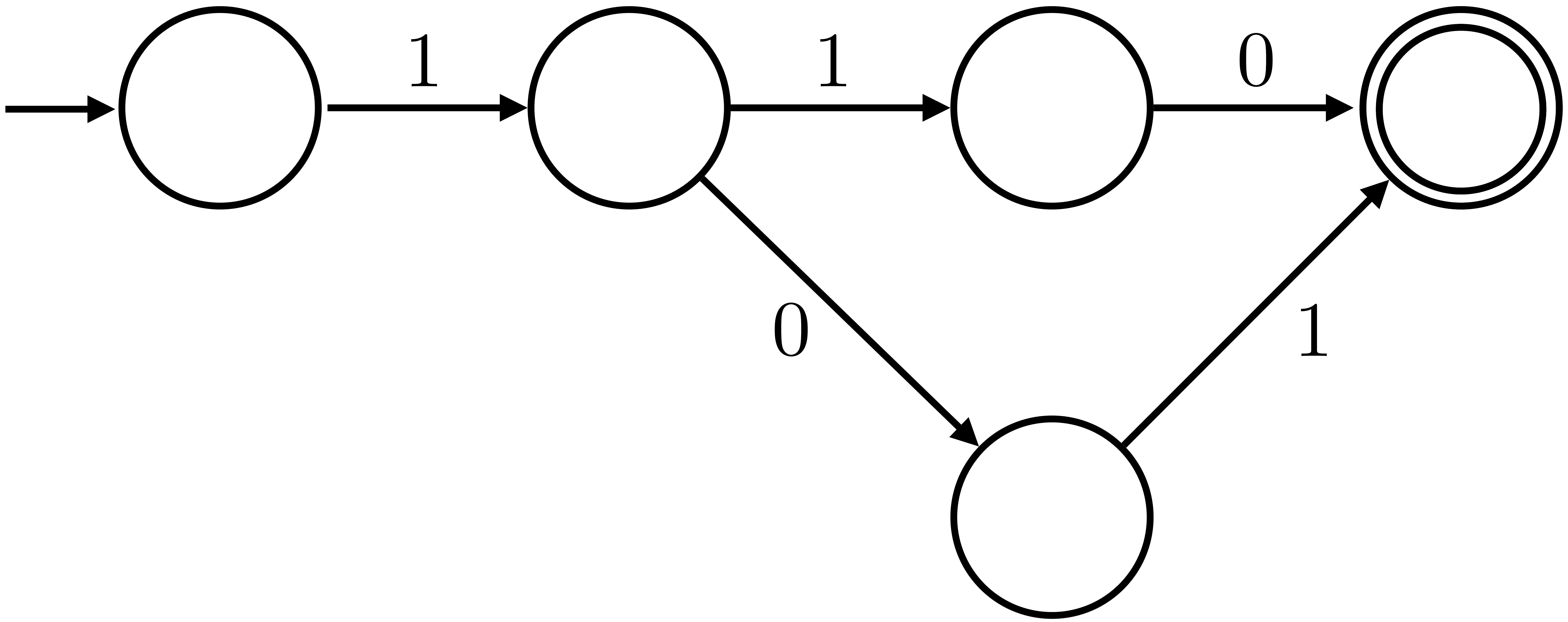
DFA construction practice

$L = \{110, 101\}$



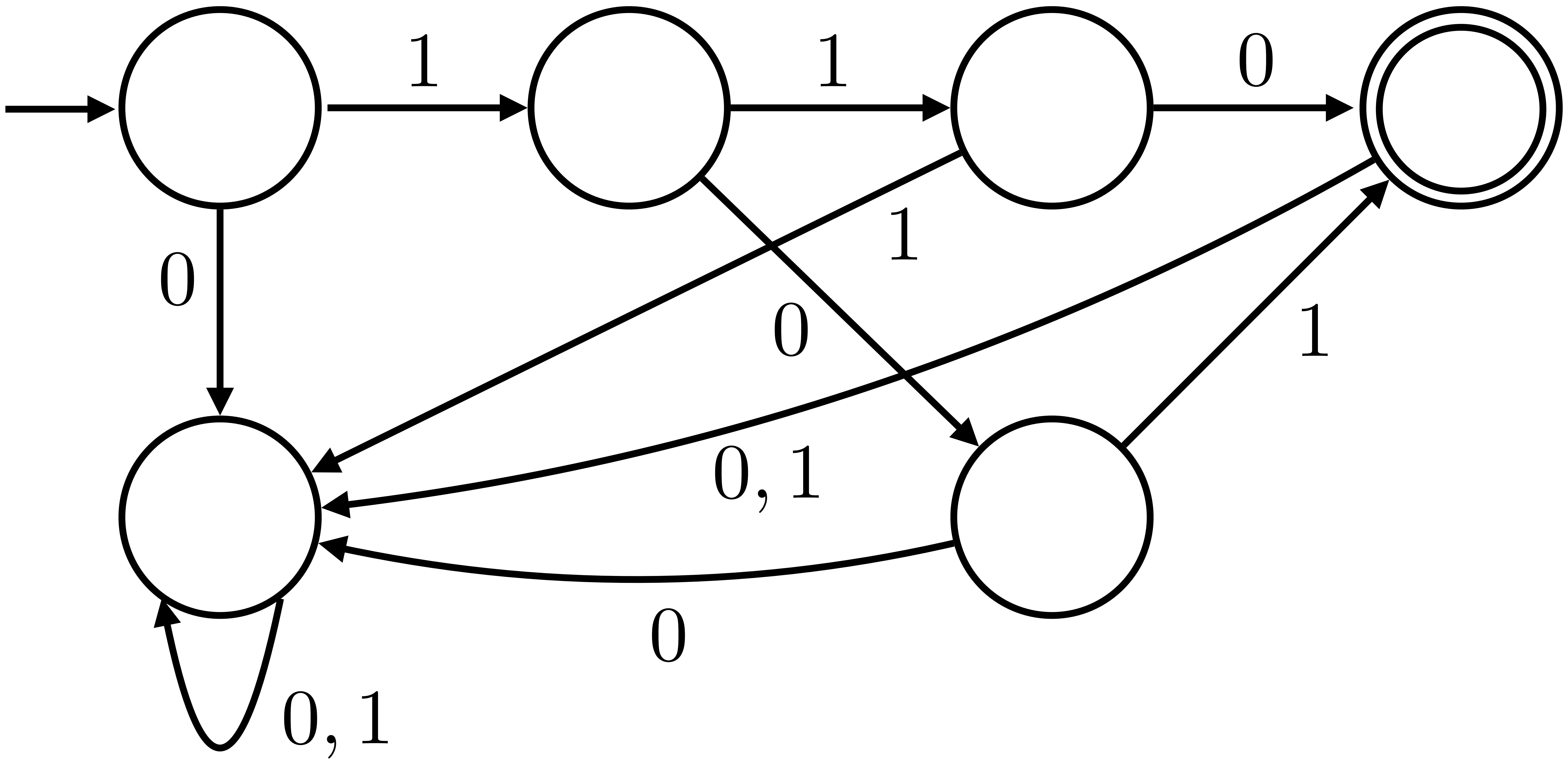
DFA construction practice

$L = \{110, 101\}$



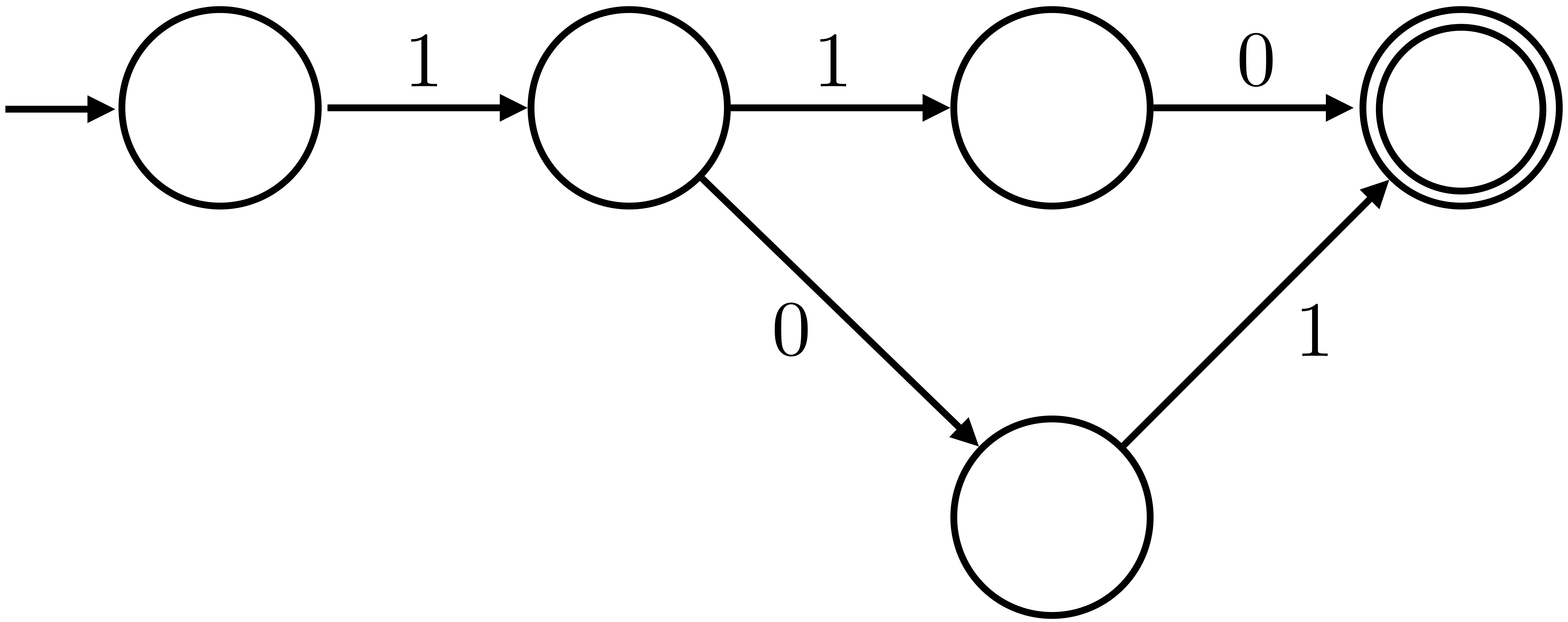
DFA construction practice

$L = \{110, 101\}$



DFA construction practice

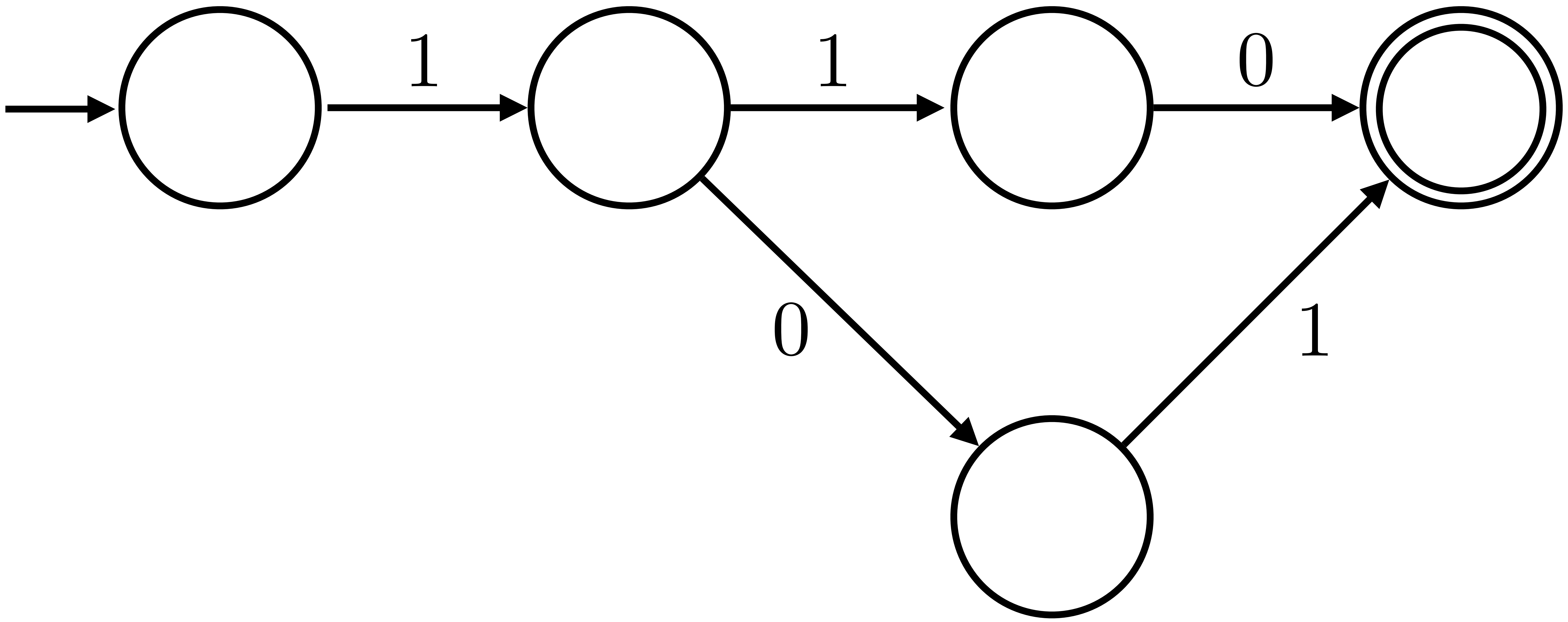
$L = \{110, 101\}$



All missing transitions go to a rejecting sink state.

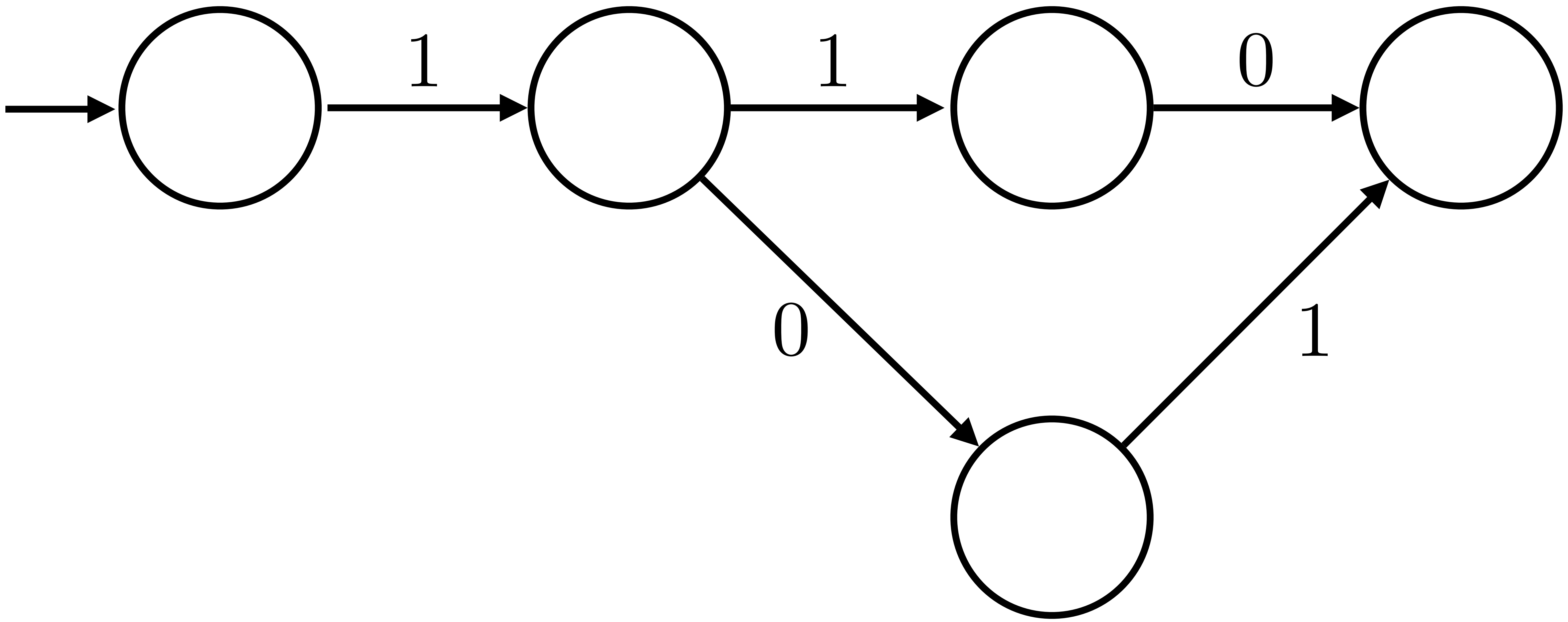
DFA construction practice

$L = \{0,1\}^* \setminus \{110,101\}$



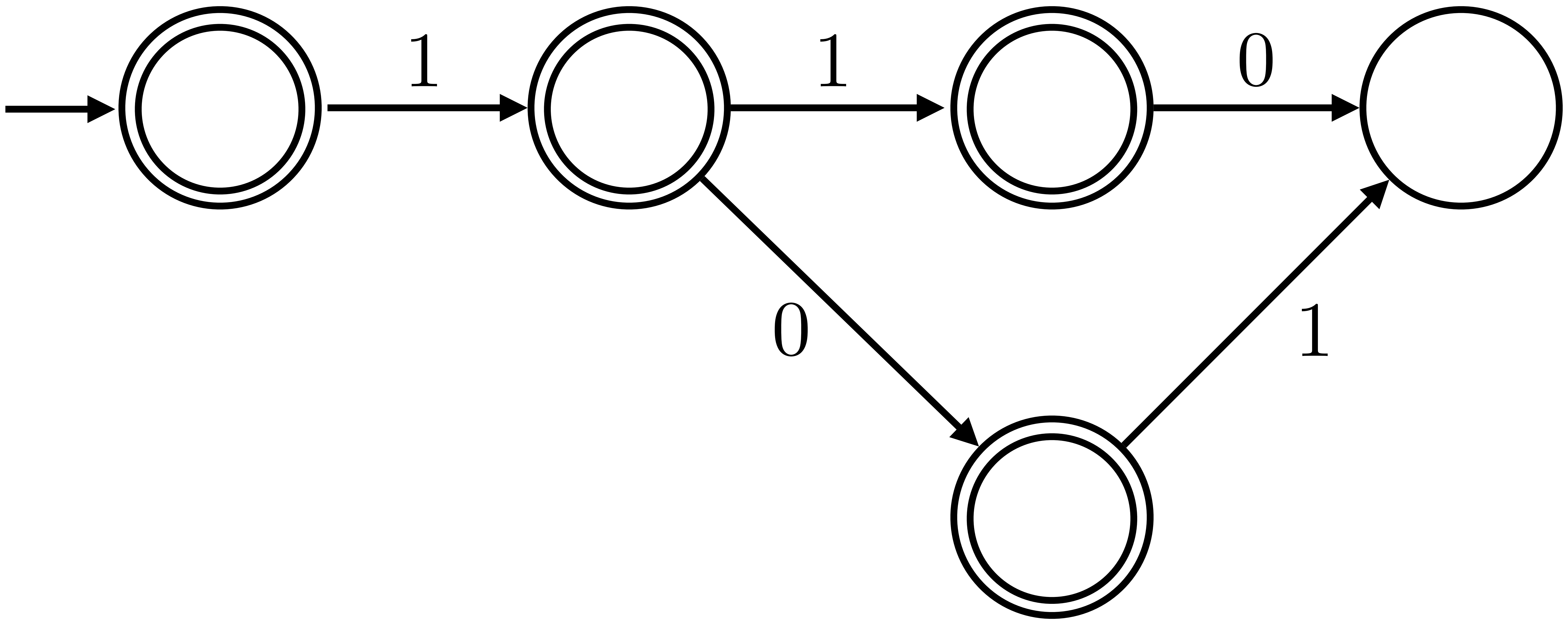
DFA construction practice

$L = \{0,1\}^* \setminus \{110,101\}$



DFA construction practice

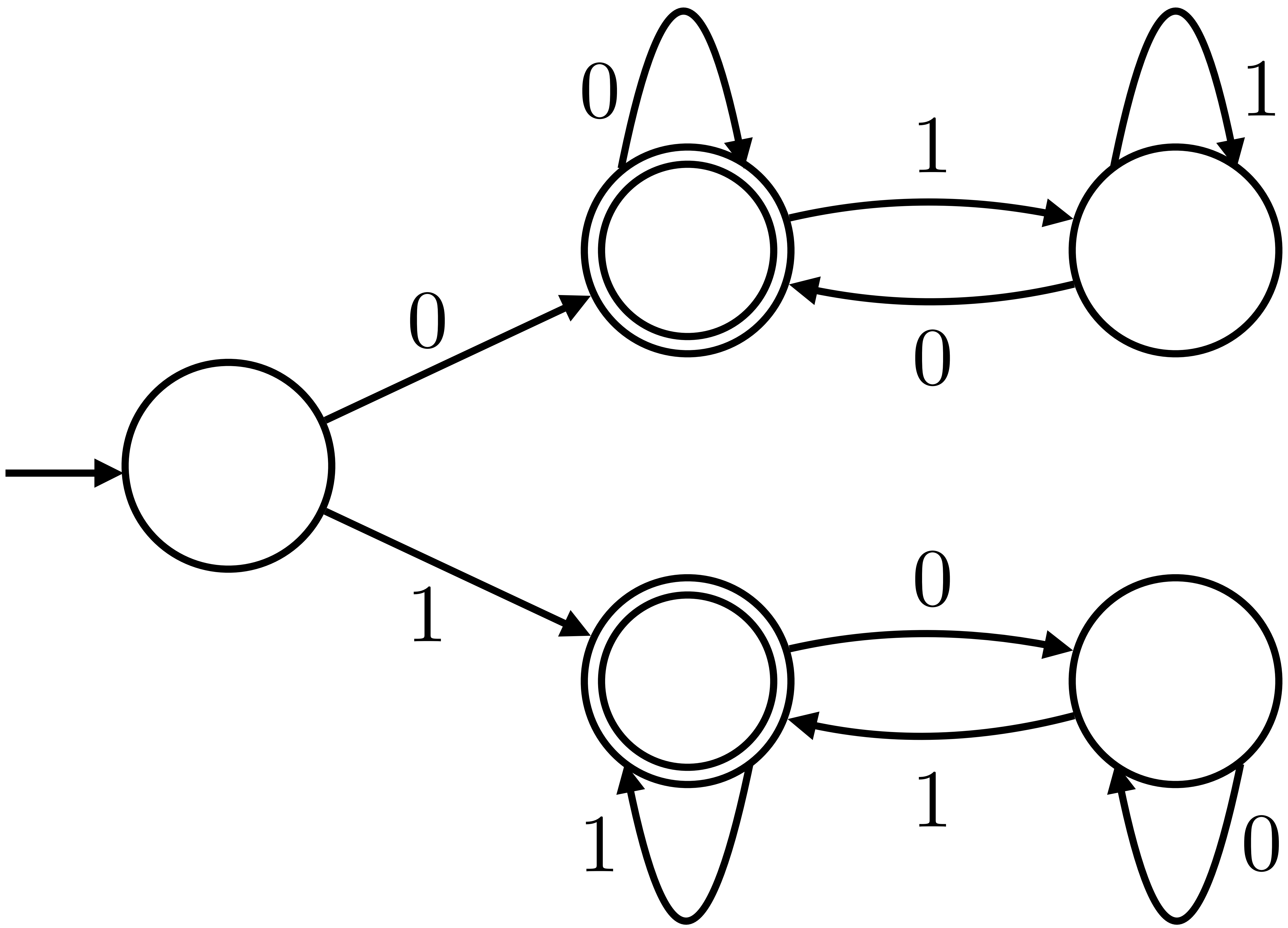
$L = \{0,1\}^* \setminus \{110,101\}$



All missing transitions go to an accepting sink state.

DFA construction practice

$L = \{x \in \{0,1\}^* :$ x starts and ends with same bit$\}$



Terminology:

Computational Model — Allowed rules for information processing.

The Deterministic Finite Automaton computational model

Machine = Computer = Program = Algorithm — An instantiation of the computational model.
(a specific sequence of information processing rules)

A Deterministic Finite Automaton (DFA)

DFA as a programming language

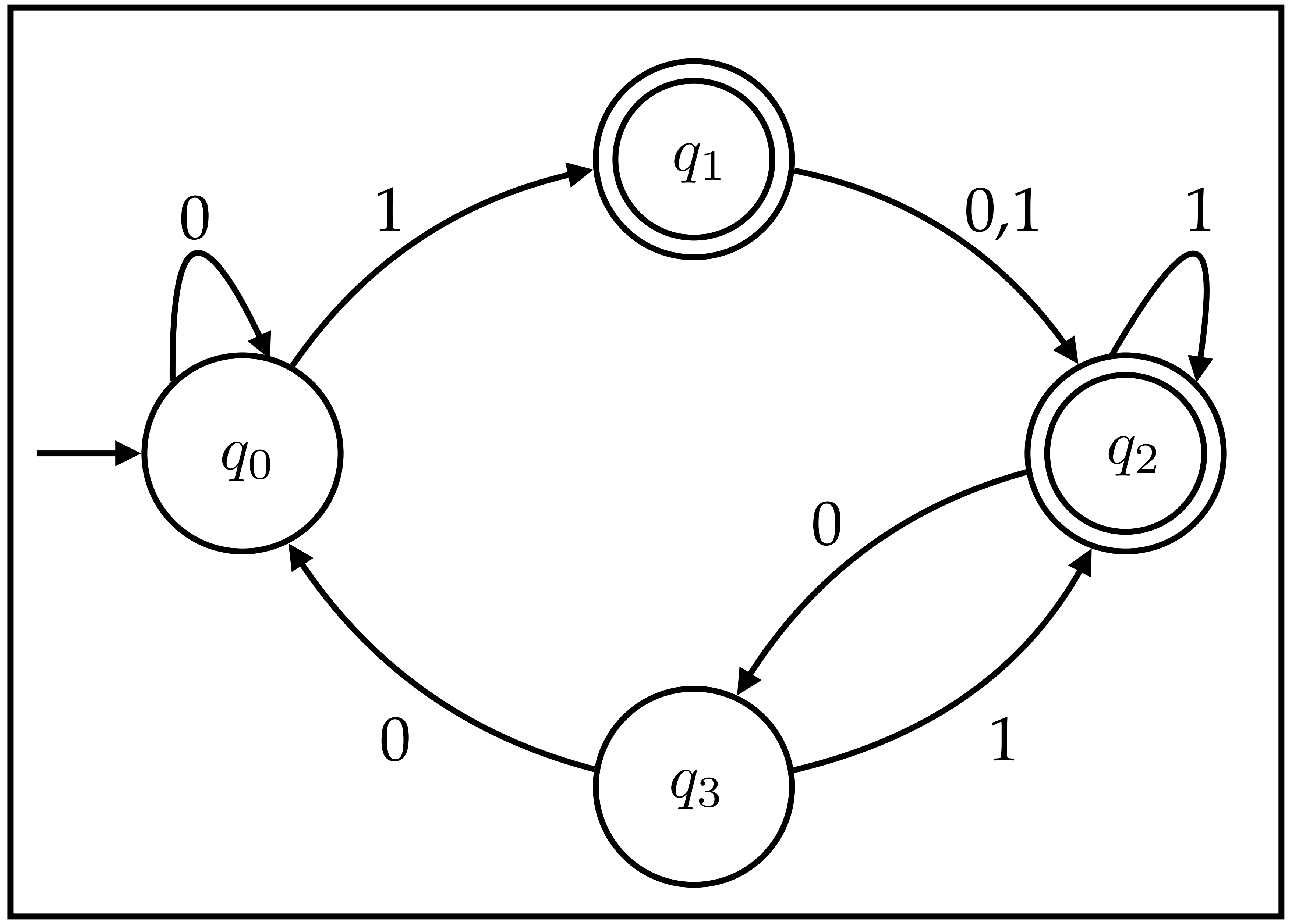
```
def foo(input):
    i = 0;
    STATE 0:
        if (i == input.length): return False;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 0;
          case '1':  go to STATE 1;

    STATE 1:
        if (i == input.length): return True;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 2;
          case '1':  go to STATE 2;

  ...
```

input =

0	1	1	1	1



DFA as a programming language

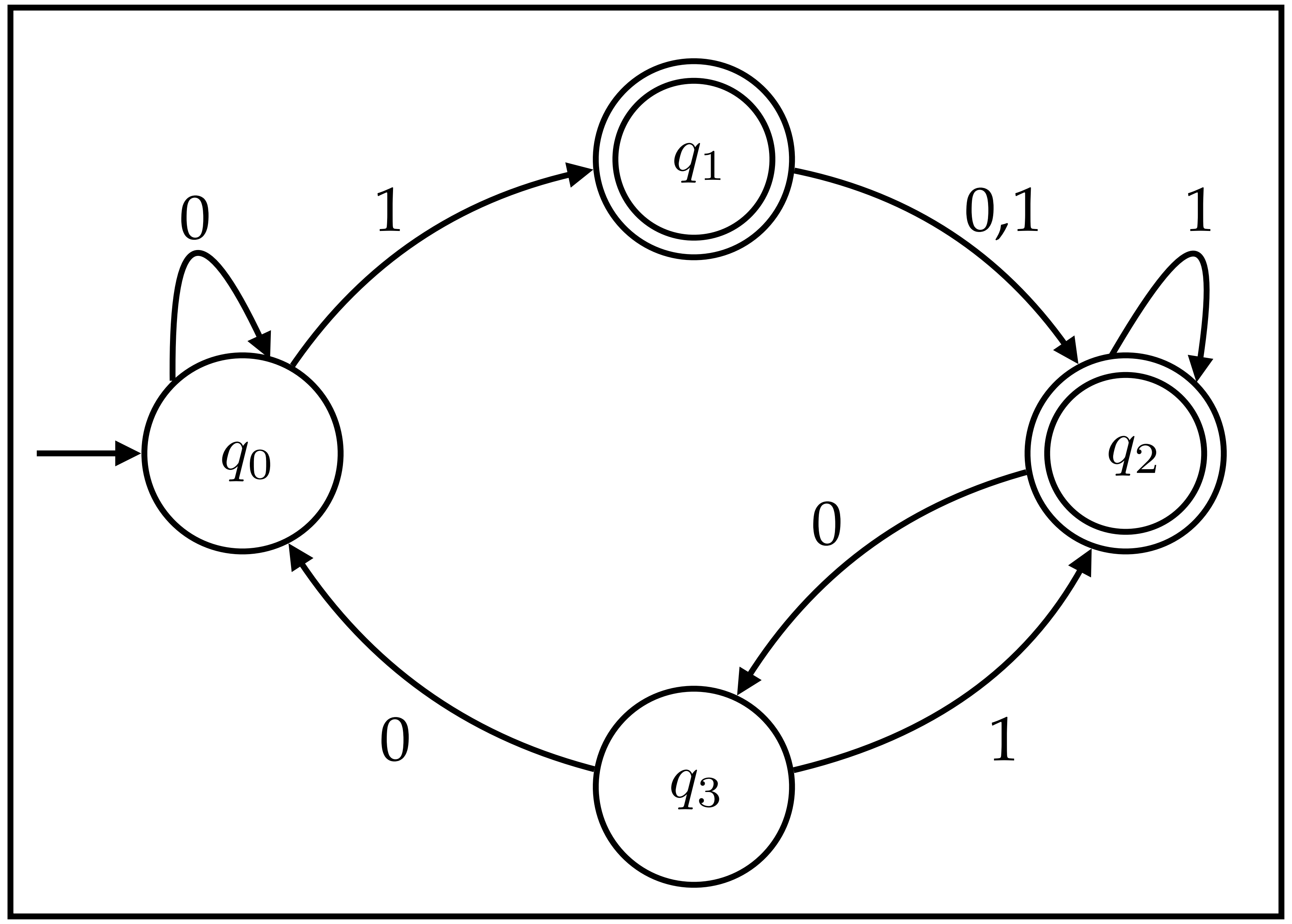
```
def foo(input):
    i = 0;
    STATE 0:
        if (i == input.length): return False;
        letter = input[i];
        i++;
        switch(letter):
            case '0':  go to STATE 0;
            case '1':  go to STATE 1;

    STATE 1:
        if (i == input.length): return True;
        letter = input[i];
        i++;
        switch(letter):
            case '0':  go to STATE 2;
            case '1':  go to STATE 2;

    ...
```

input =

0	1	1	1	1



DFA as a programming language

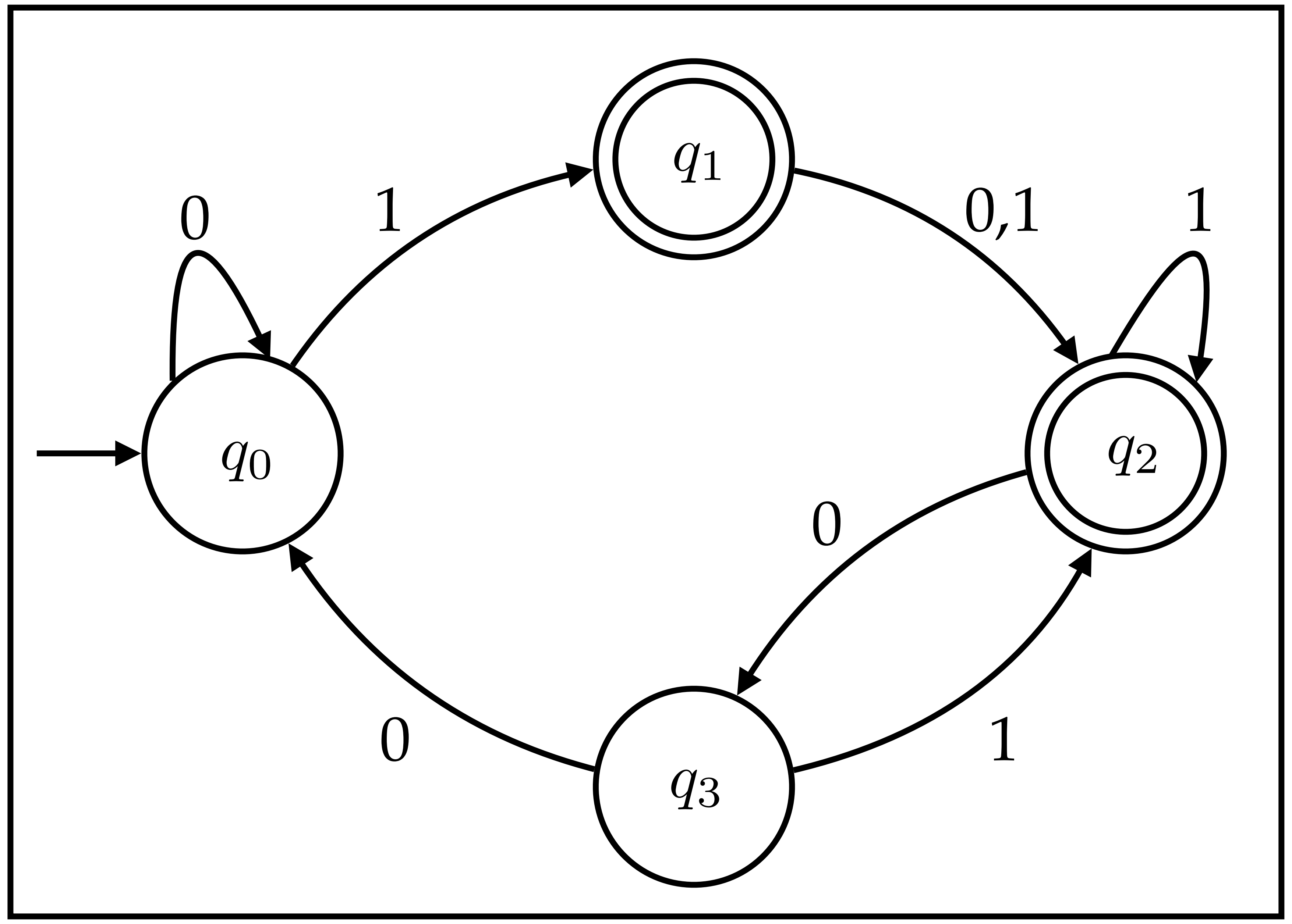
```
def foo(input):
  i = 0;
  STATE 0:
    if (i == input.length): return False;
    letter = input[i];
    i++;
    switch(letter):
      case '0':  go to STATE 0;
      case '1':  go to STATE 1;

  STATE 1:
    if (i == input.length): return True;
    letter = input[i];
    i++;
    switch(letter):
      case '0':  go to STATE 2;
      case '1':  go to STATE 2;

  ...
```

input =

0	1	1	1	1



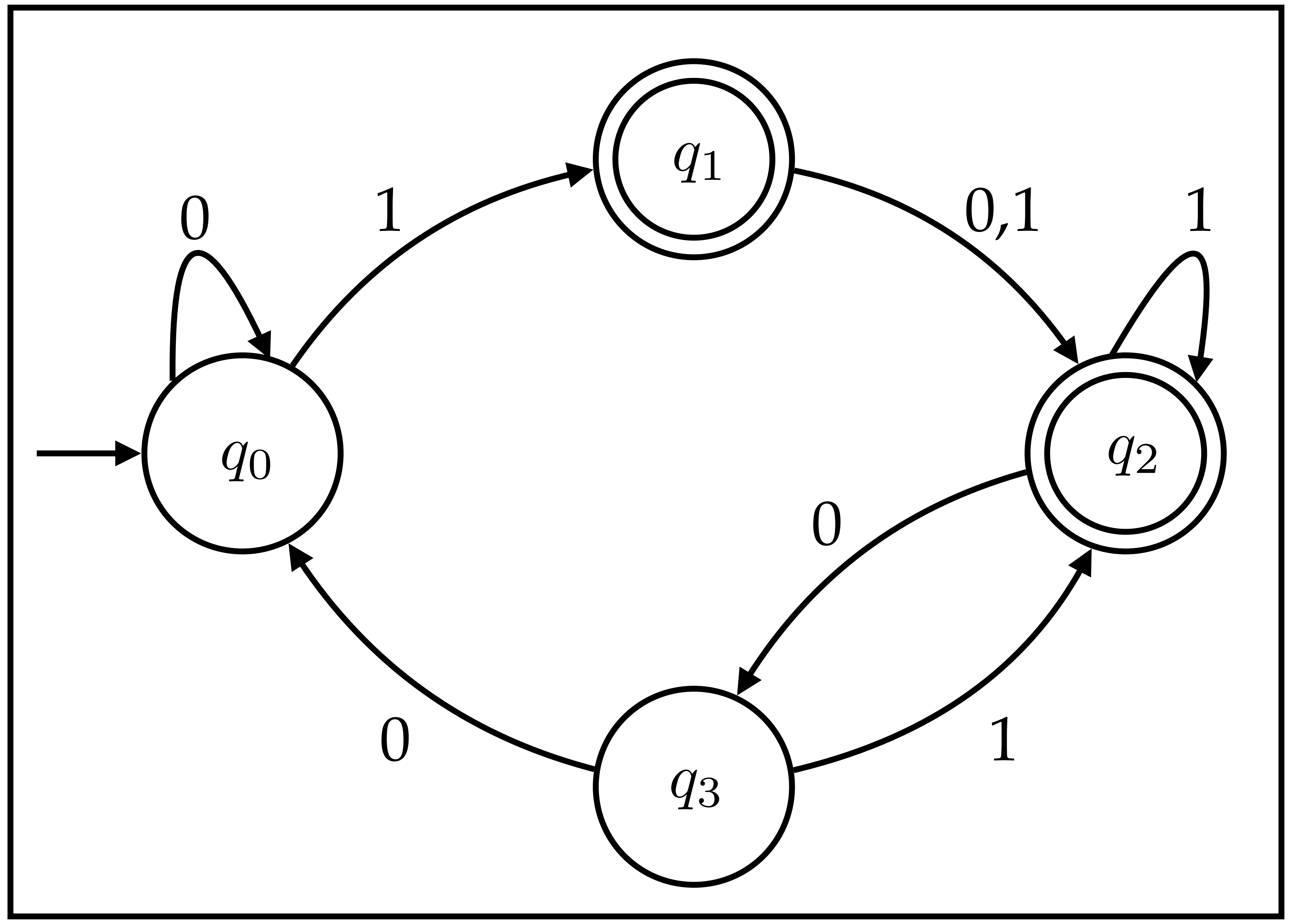
DFA as a programming language

```
def foo(input):
    i = 0;
    STATE 0:
        if (i == input.length): return False;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 0;
          case '1':  go to STATE 1;

    STATE 1:
        if (i == input.length): return True;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 2;
          case '1':  go to STATE 2;

   ...
```

input = | 0 | 1 | 1 | 1 | 1 |



DFA as a programming language

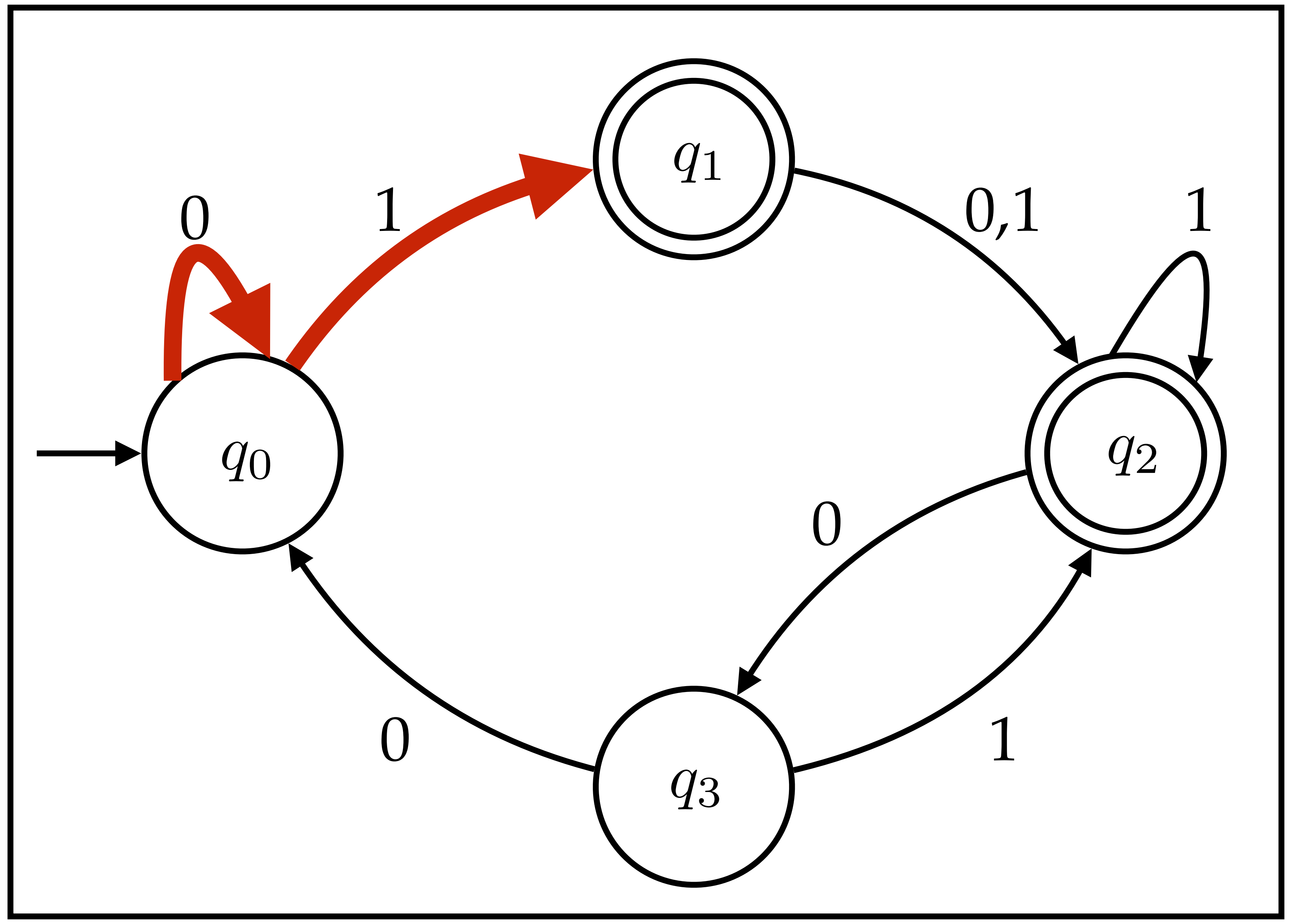
```
def foo(input):
    i = 0;
    STATE 0:
        if (i == input.length): return False;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 0;
          case '1':  go to STATE 1;

    STATE 1:
        if (i == input.length): return True;
        letter = input[i];
        i++;
        switch(letter):
          case '0':  go to STATE 2;
          case '1':  go to STATE 2;

    ...
```

input =

0	1	1	1	1



DFA as a programming language

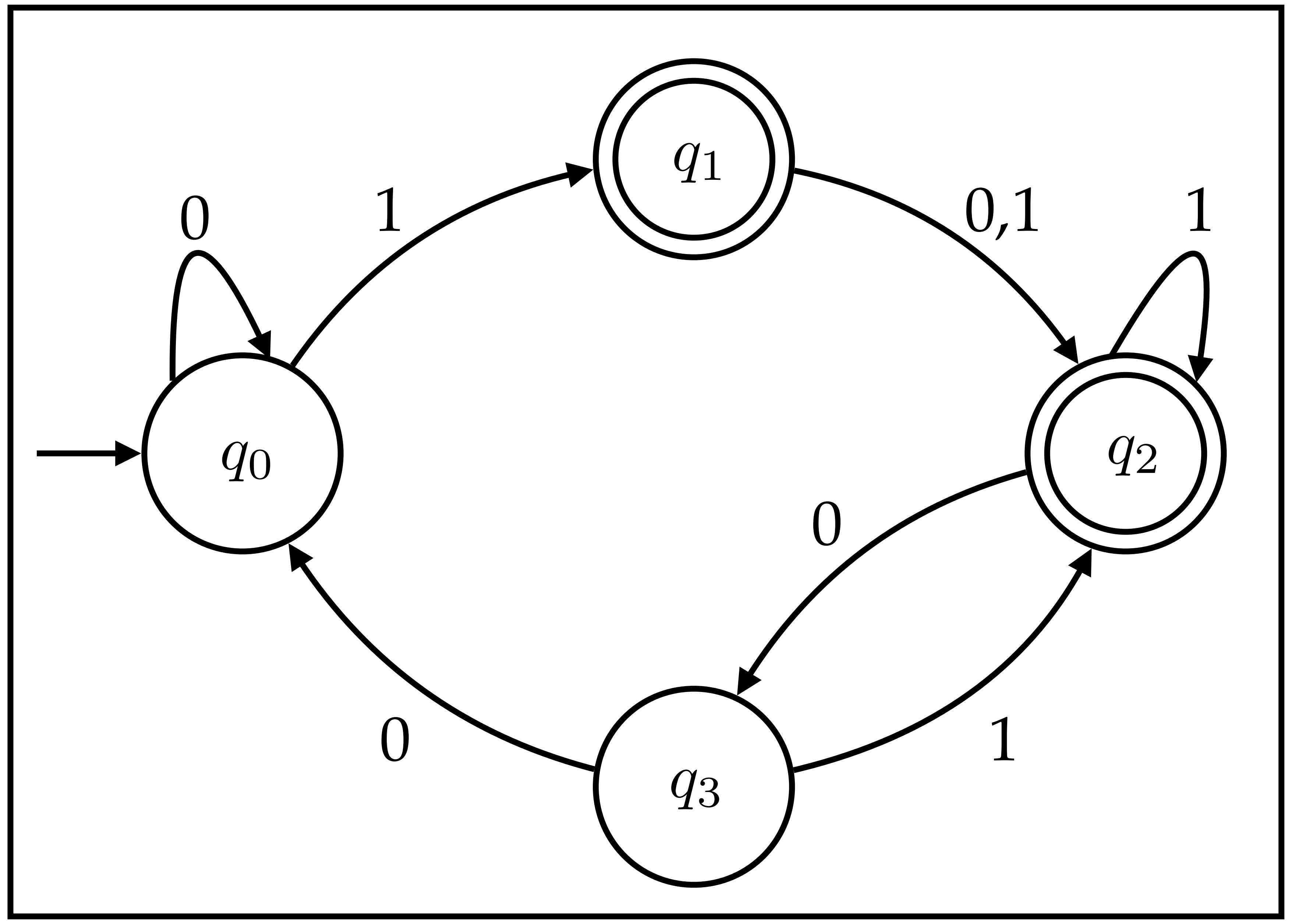
```
def foo(input):
  i = 0;
  STATE 0:
    if (i == input.length): return False;
    letter = input[i];
    i++;
    switch(letter):
      case ‘0’:  go to STATE 0;
      case ‘1’:  go to STATE 1;

  STATE 1:
    if (i == input.length): return True;
    letter = input[i];
    i++;
    switch(letter):
      case ‘0’:  go to STATE 2;
      case ‘1’:  go to STATE 2;

  ...
```

input =

0	1	1	1	1



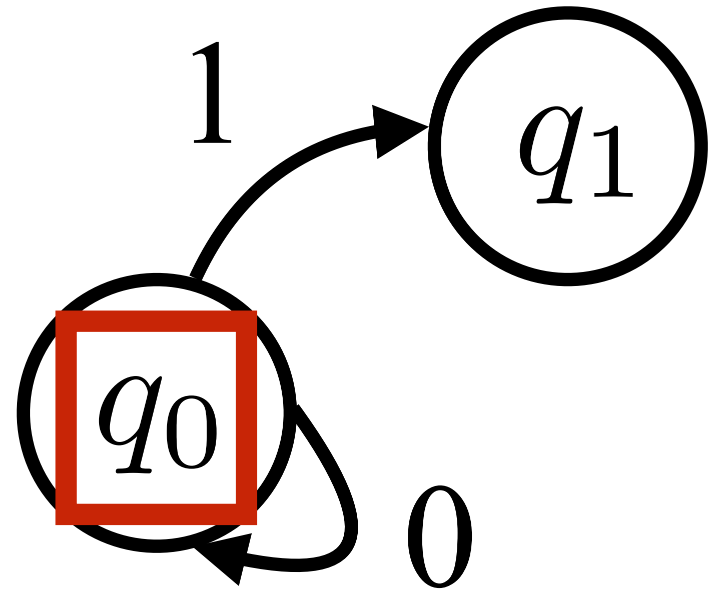
Formal Definition

Formal definition: DFA

Definition: A ***deterministic finite automaton (DFA)***

is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

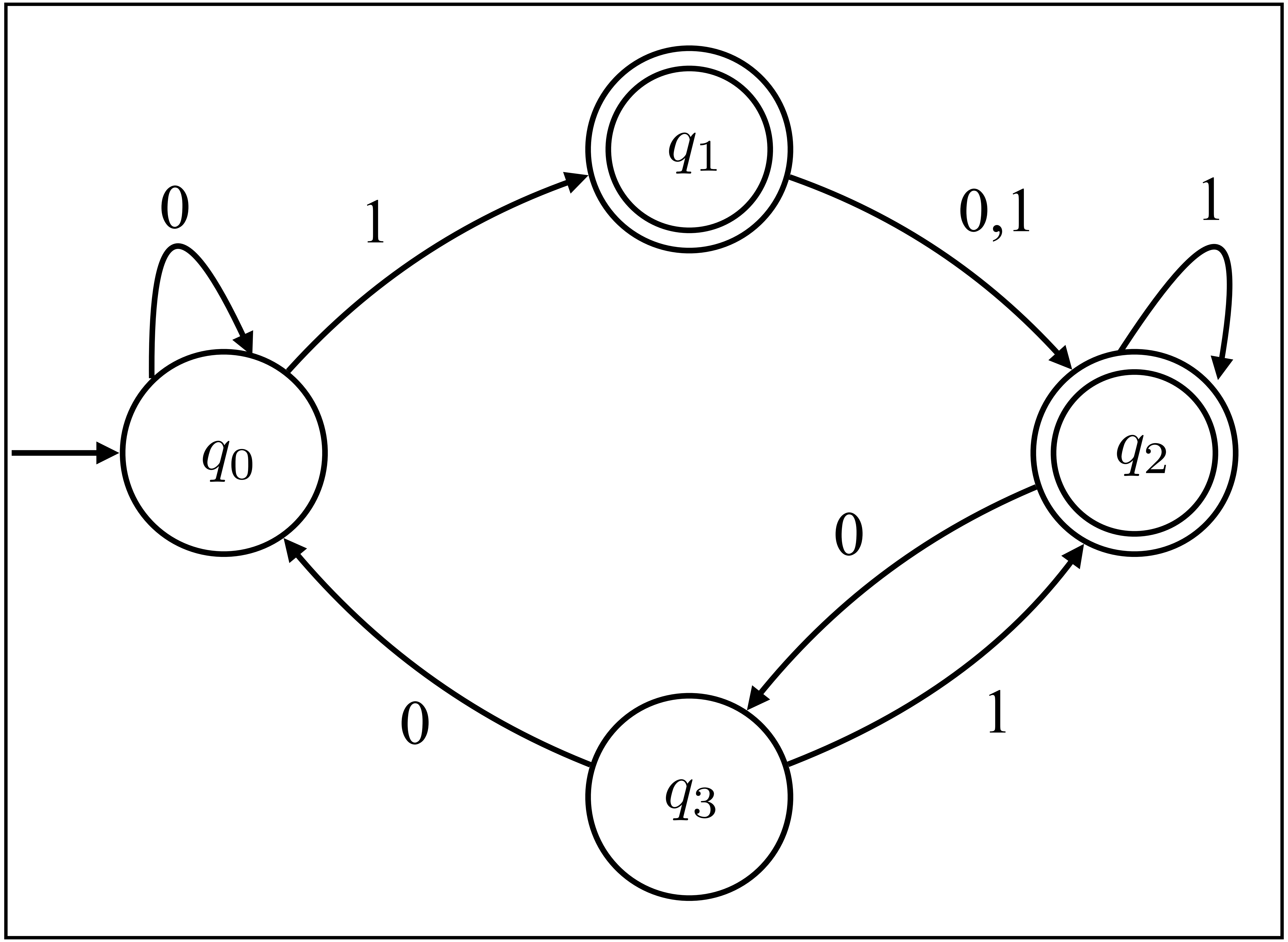
- Q is a finite, non-empty set (called the ***set of states***);
- Σ is a finite, non-empty set (called the ***alphabet***);
- δ is a function of the form $\delta : Q \times \Sigma \rightarrow Q$;
  (called the ***transition function***);
- q_0 is an element of Q (called the ***start state***);
- F is a subset of Q (called the ***set of accepting states***).



$$\delta(q_0, 1) = q_1$$

$$\delta(q_0, 0) = q_0$$

Formal definition: DFA



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

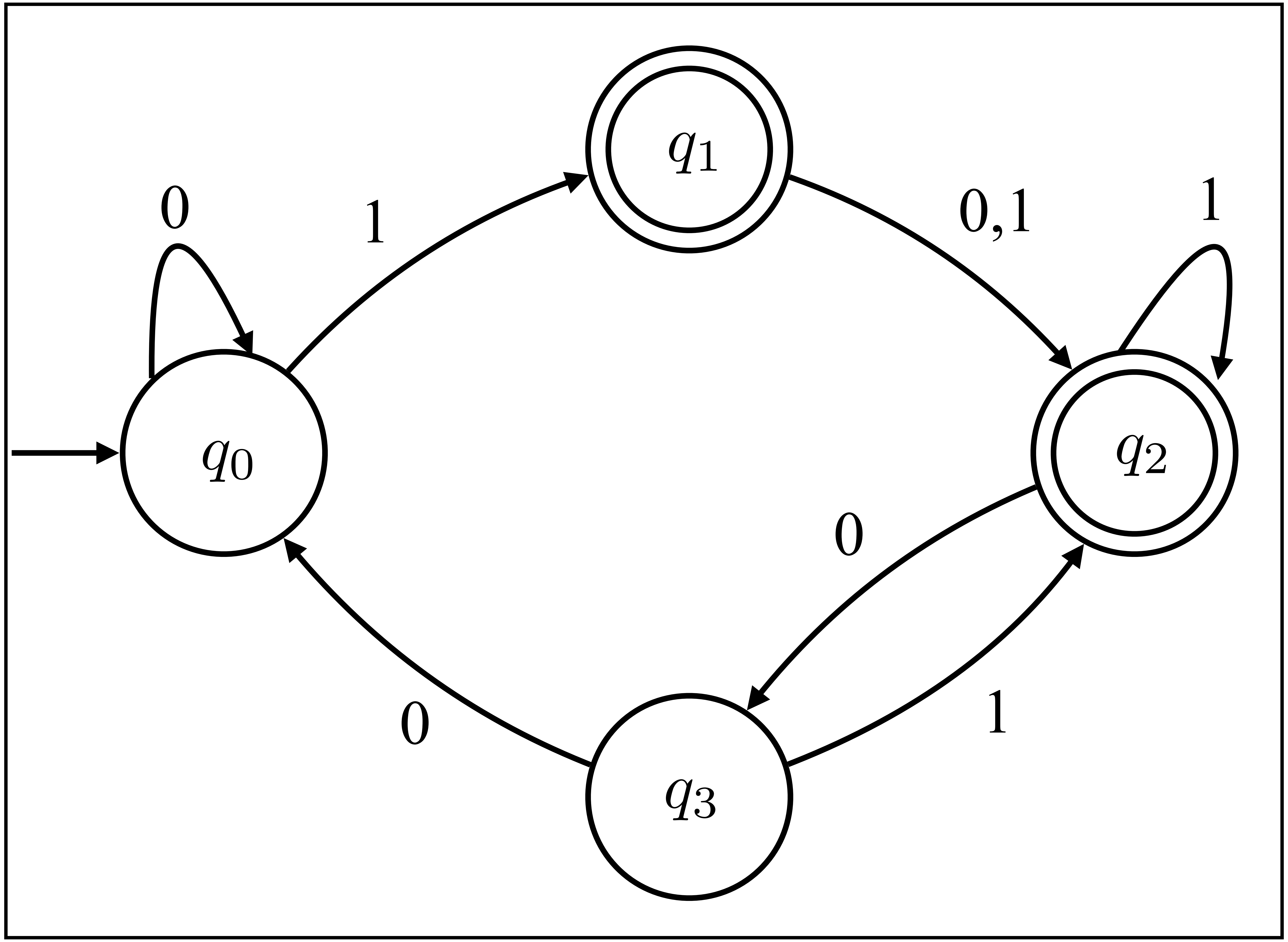
$$Q = \{q_0, q_1, q_2, q_3\}$$

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

$$\delta : Q \times \Sigma \to Q$$

δ	0	1
q_0	q_0	
q_1		
q_2		
q_3		

Formal definition: DFA



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

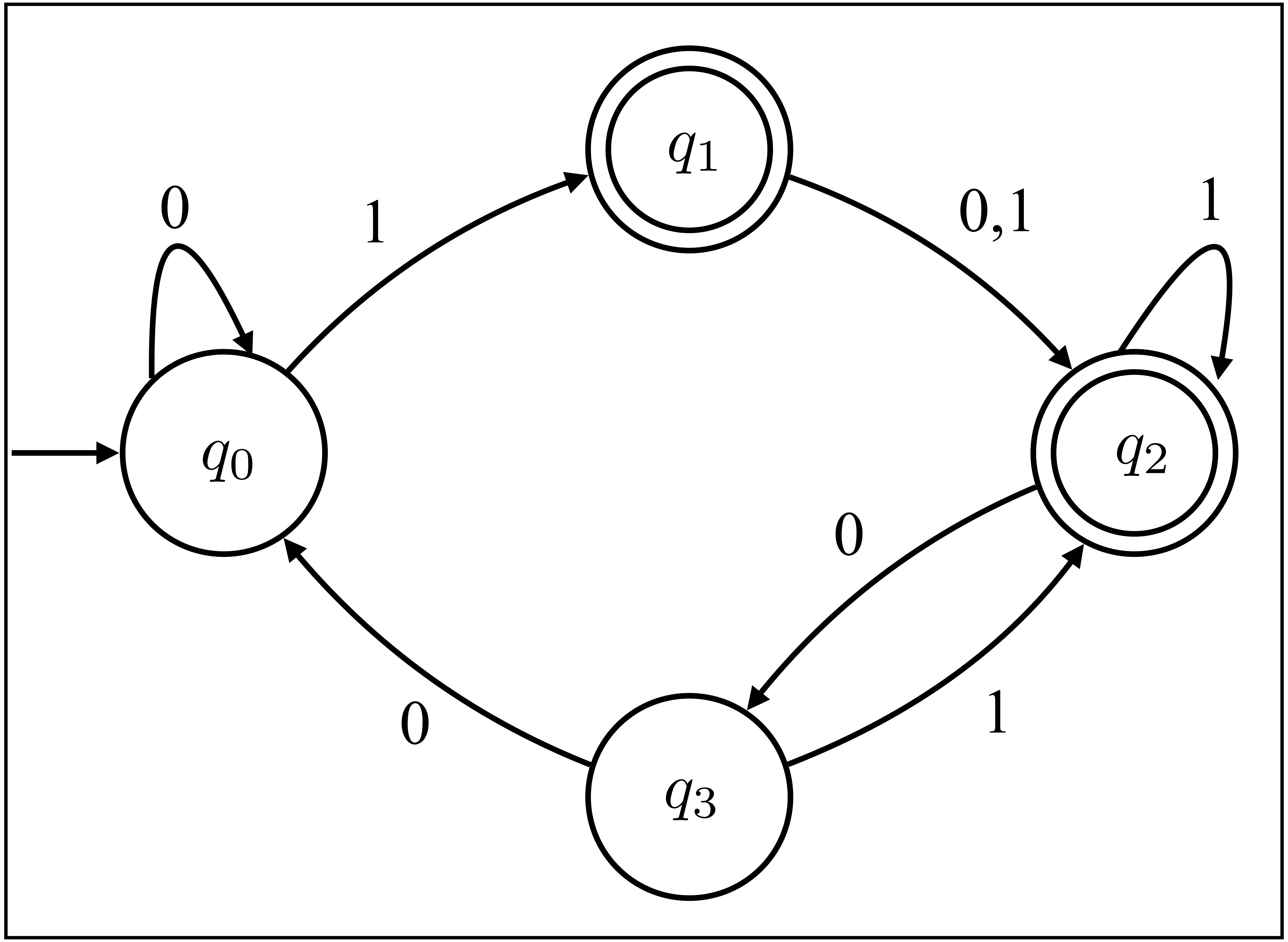
$Q = \{q_0, q_1, q_2, q_3\}$

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

$\delta : Q \times \Sigma \rightarrow Q$

δ	0	1
q_0	q_0	q_1
q_1		
q_2		
q_3		

Formal definition: DFA



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

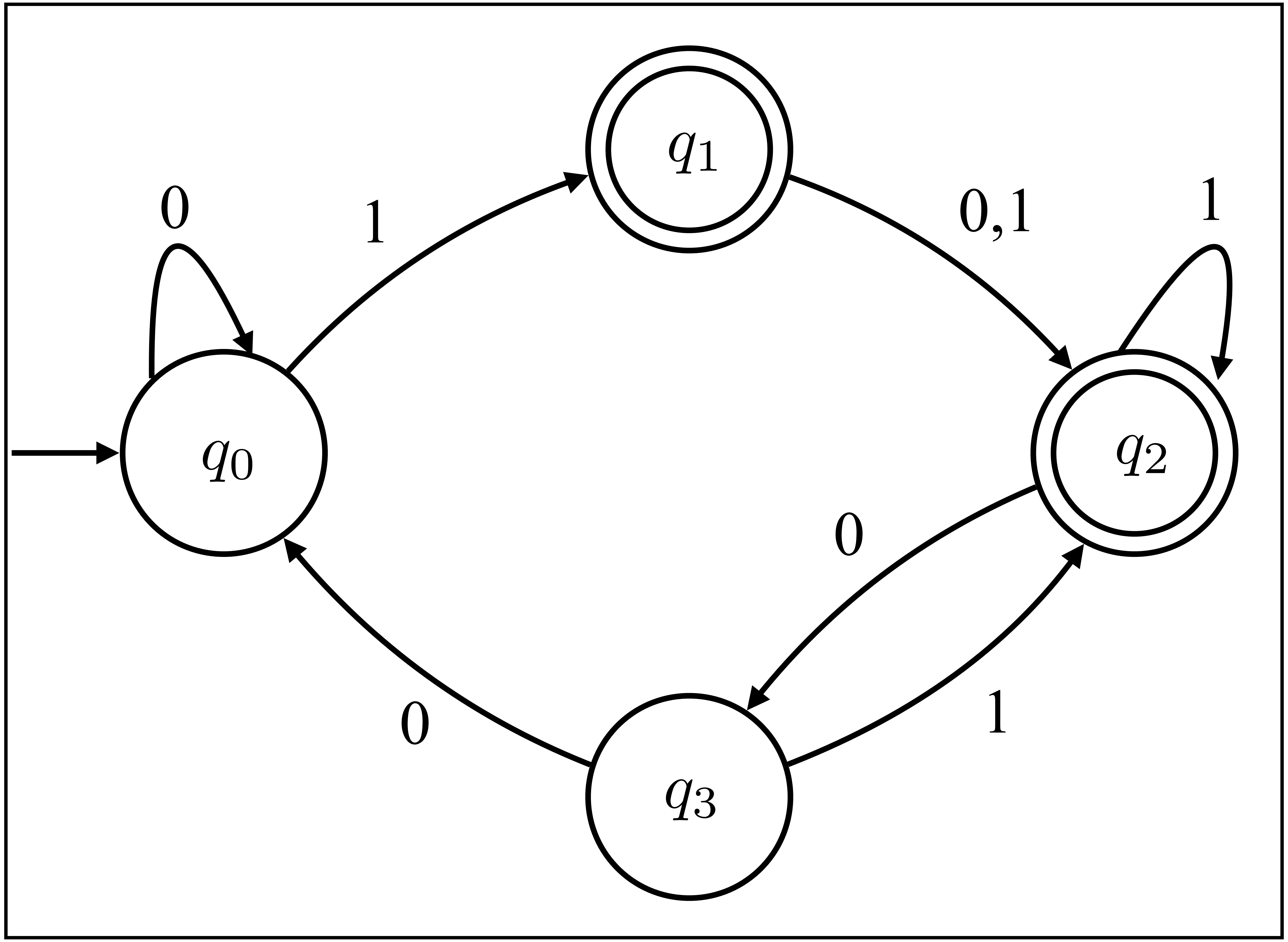
$$Q = \{q_0, q_1, q_2, q_3\}$$

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

$$\delta : Q \times \Sigma \rightarrow Q$$

δ	0	1
q_0	q_0	q_1
q_1	q_2	
q_2		
q_3		

Formal definition: DFA



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

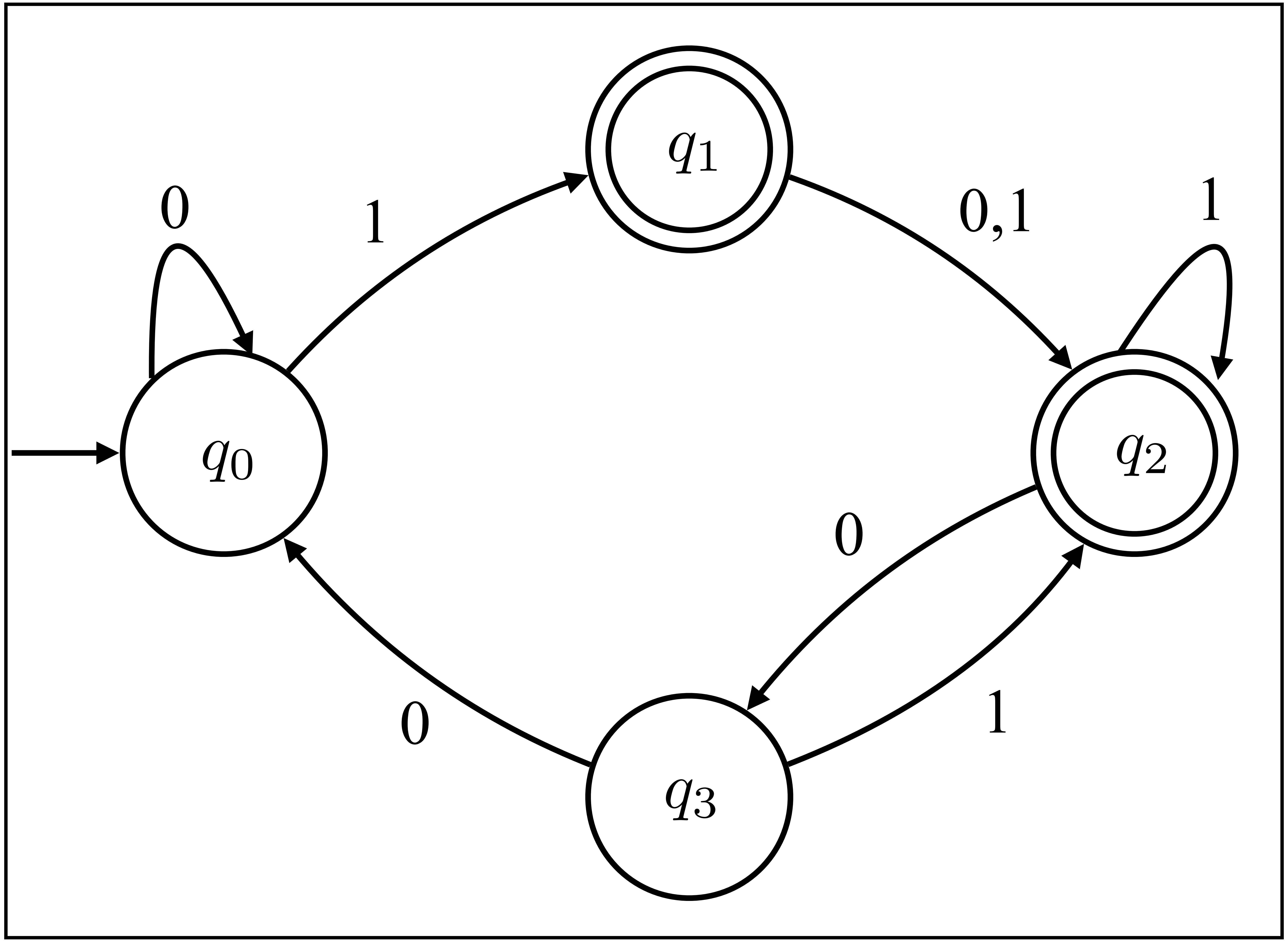
$Q = \{q_0, q_1, q_2, q_3\}$

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

$\delta : Q \times \Sigma \to Q$

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2		
q_3		

Formal definition: DFA



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

$Q = \{q_0, q_1, q_2, q_3\}$

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

$\delta : Q \times \Sigma \rightarrow Q$

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_3	q_2
q_3	q_0	q_2

q_0 is the start state

$F = \{q_1, q_2\}$

Formal definition: DFA accepting a string

Useful Notation:

For $q \in Q, w \in \Sigma^*$:

$\delta^*(q, w) =$ state we end up at when we start at q and read w.

$$= \delta(\ldots\delta(\delta(\delta(q, w_1), w_2), w_3)\ldots, w_n)\,.$$

Definition: We say DFA M ***accepts*** w if $\delta^*(q_0, w) \in F$.

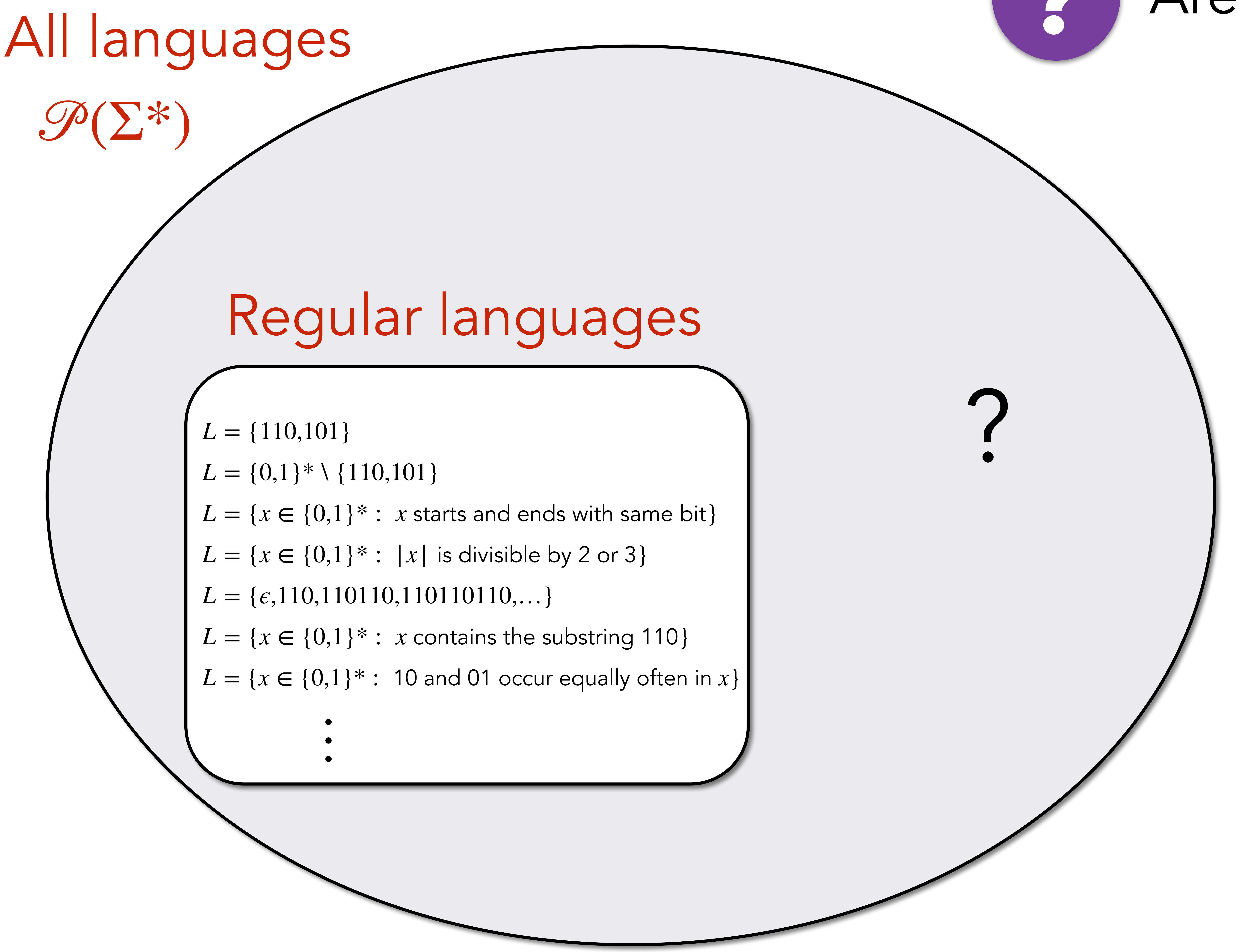
Otherwise M ***rejects*** w.

Definition: Regular languages

Definition: A language L is called ***regular*** if there is some DFA solving L.

The Big Question

Are all languages regular?



A non-regular language

How to choose a candidate non-regular language?

What are the key limitations of DFAs?

- Scans input once.
- Constant number of states.
  (constant memory)

A non-regular language

Theorem: The language consisting of all strings with an equal number of 0's and 1's is **not** regular.

A non-regular language

Theorem: The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

$$L = \{\epsilon, 01, 0011, 000111, \ \dots\}$$

A non-regular language

Theorem: The language $L = \{0^n 1^n : n \in \mathbb{N}\}$ is **not** regular.

Intuition:

Seems DFA would need to remember # 0's it sees.

But it has a constant number of states.

(and no other way of remembering things)

Careful:

$L = \{x \in \{0,1\}^* : 01$ and 10 occur equally often in $x\}$ is regular!

A non-regular language

Theorem: The language $L = \{0^n1^n : n \in \mathbb{N}\}$ is **not** regular.

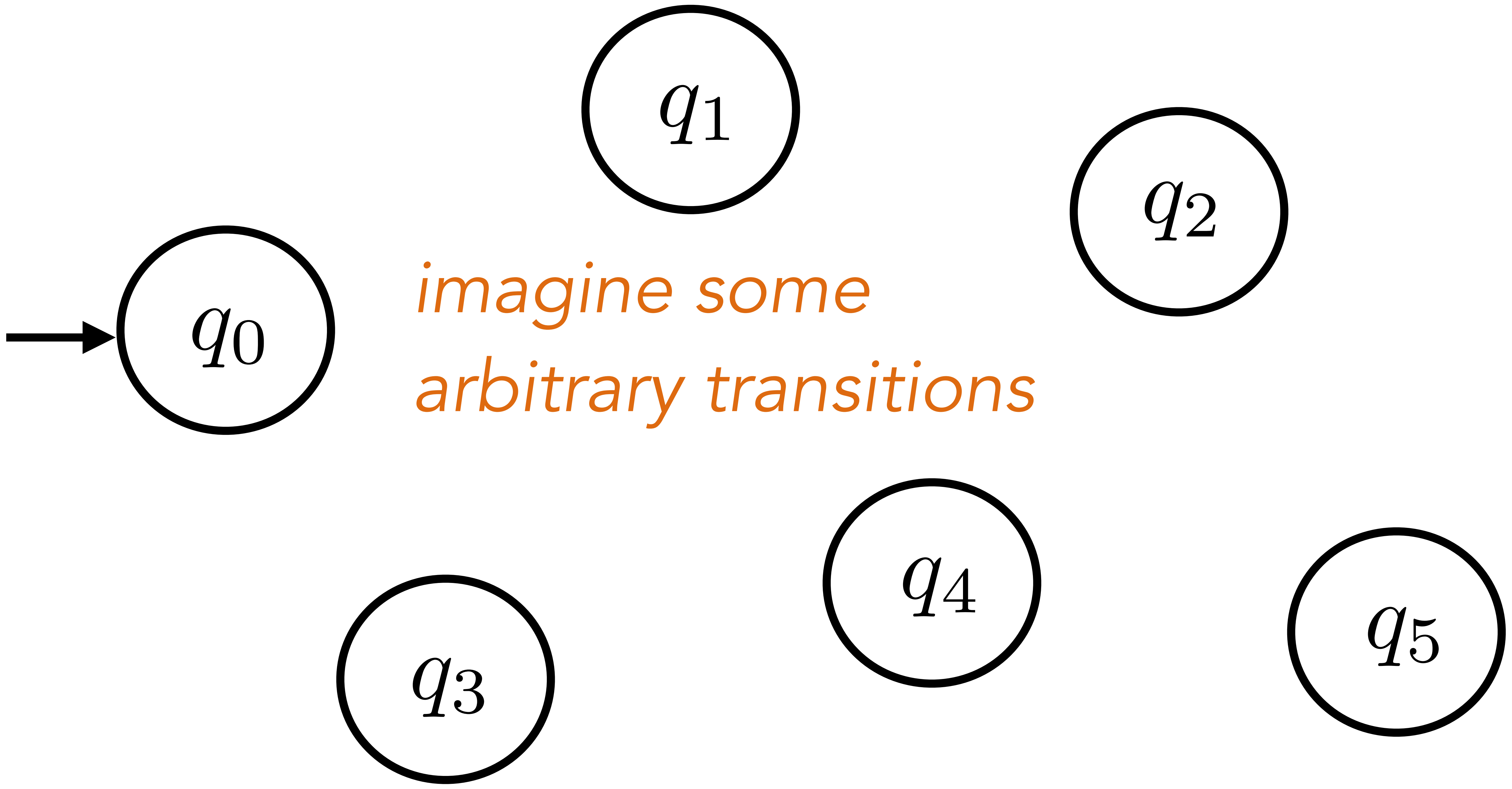
A key component of the proof:

Pigeonhole principle (PHP)

$L = \{0^n1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n1^n : n \in \mathbb{N}\}$.

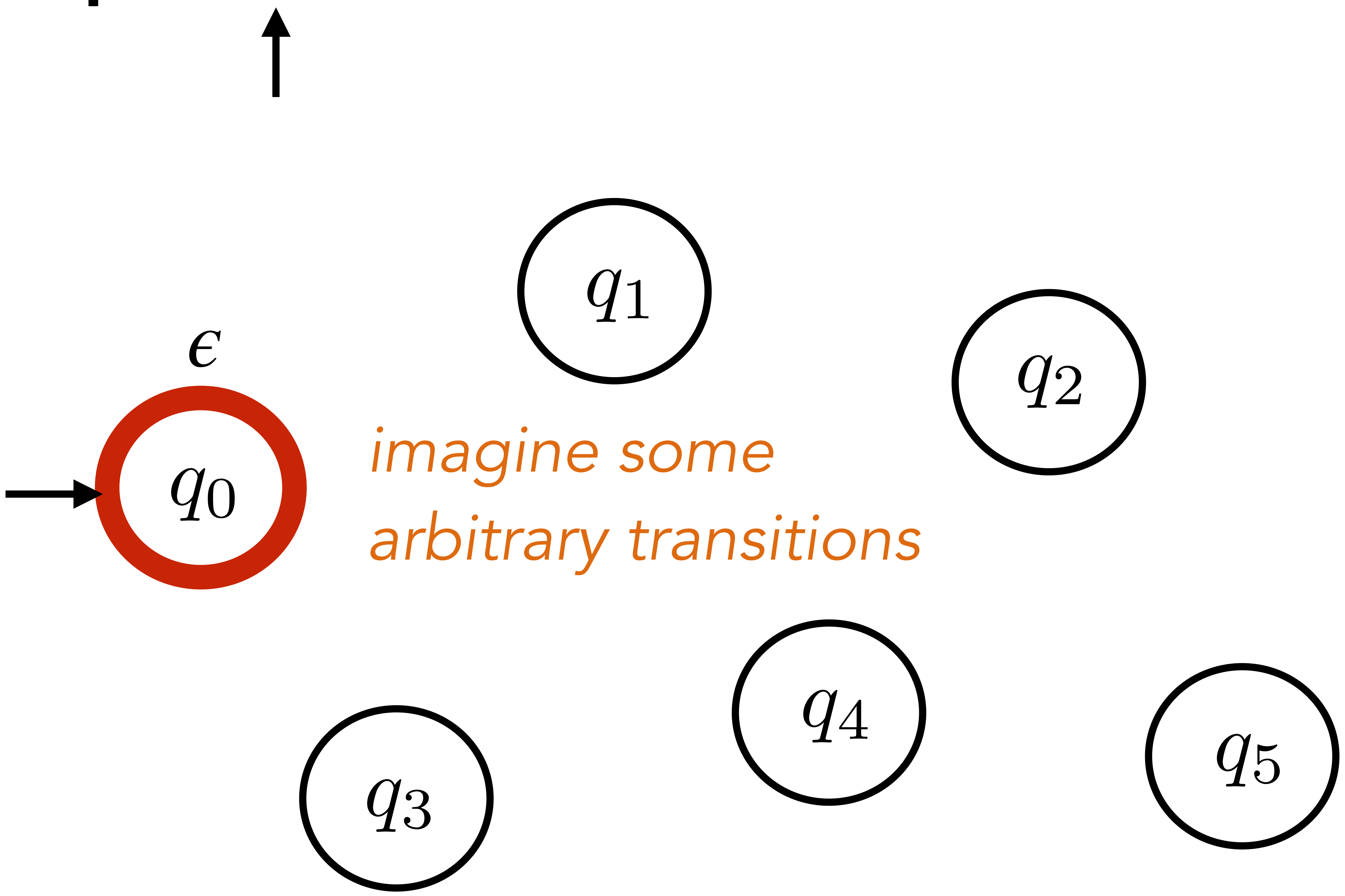
Input: 0000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

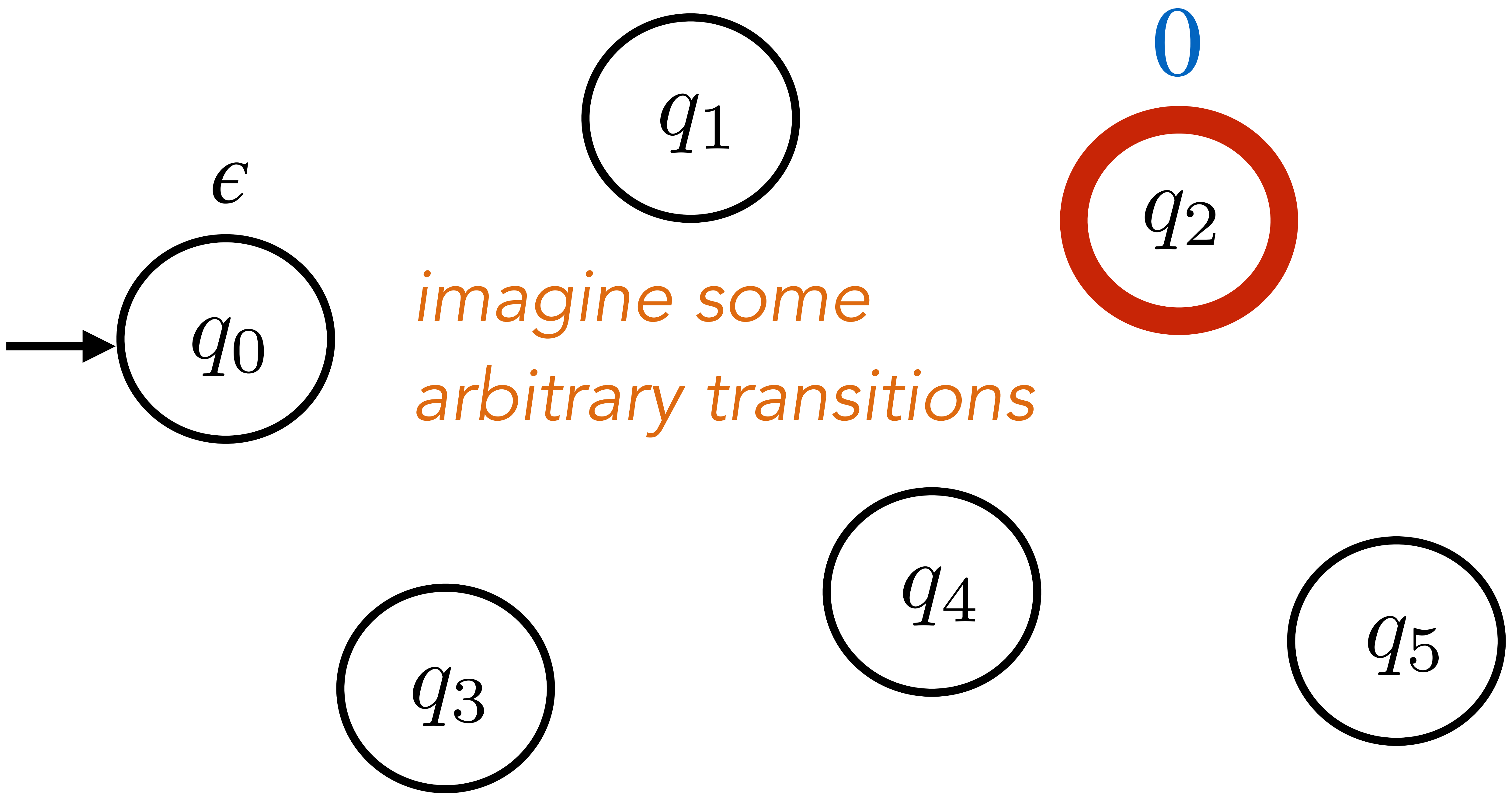
Input: 0000000000000000



$L = \{0^n1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n1^n : n \in \mathbb{N}\}$.

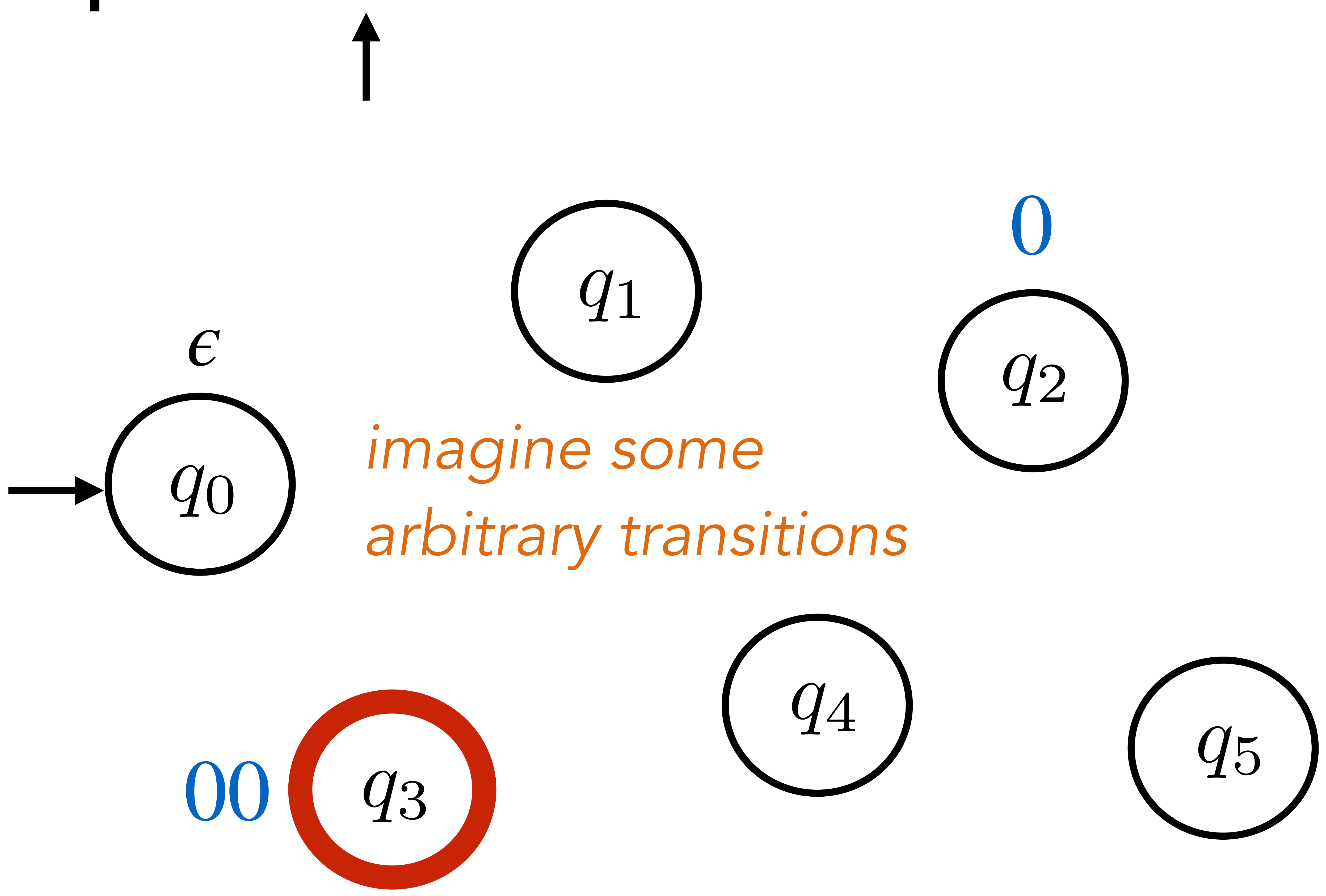
Input: 0000000000000000



$L = \{0^n1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n1^n : n \in \mathbb{N}\}$.

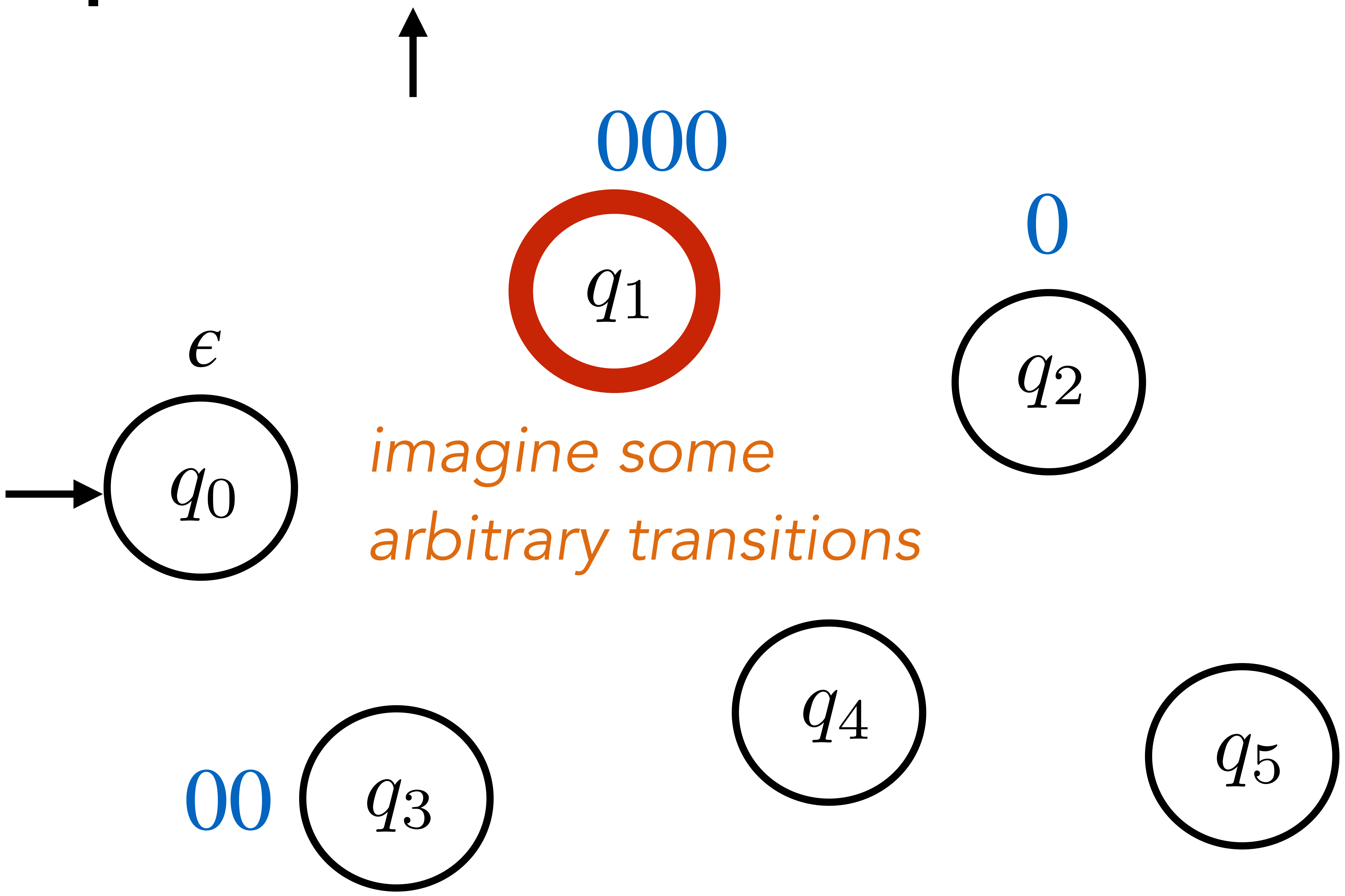
Input: 0000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

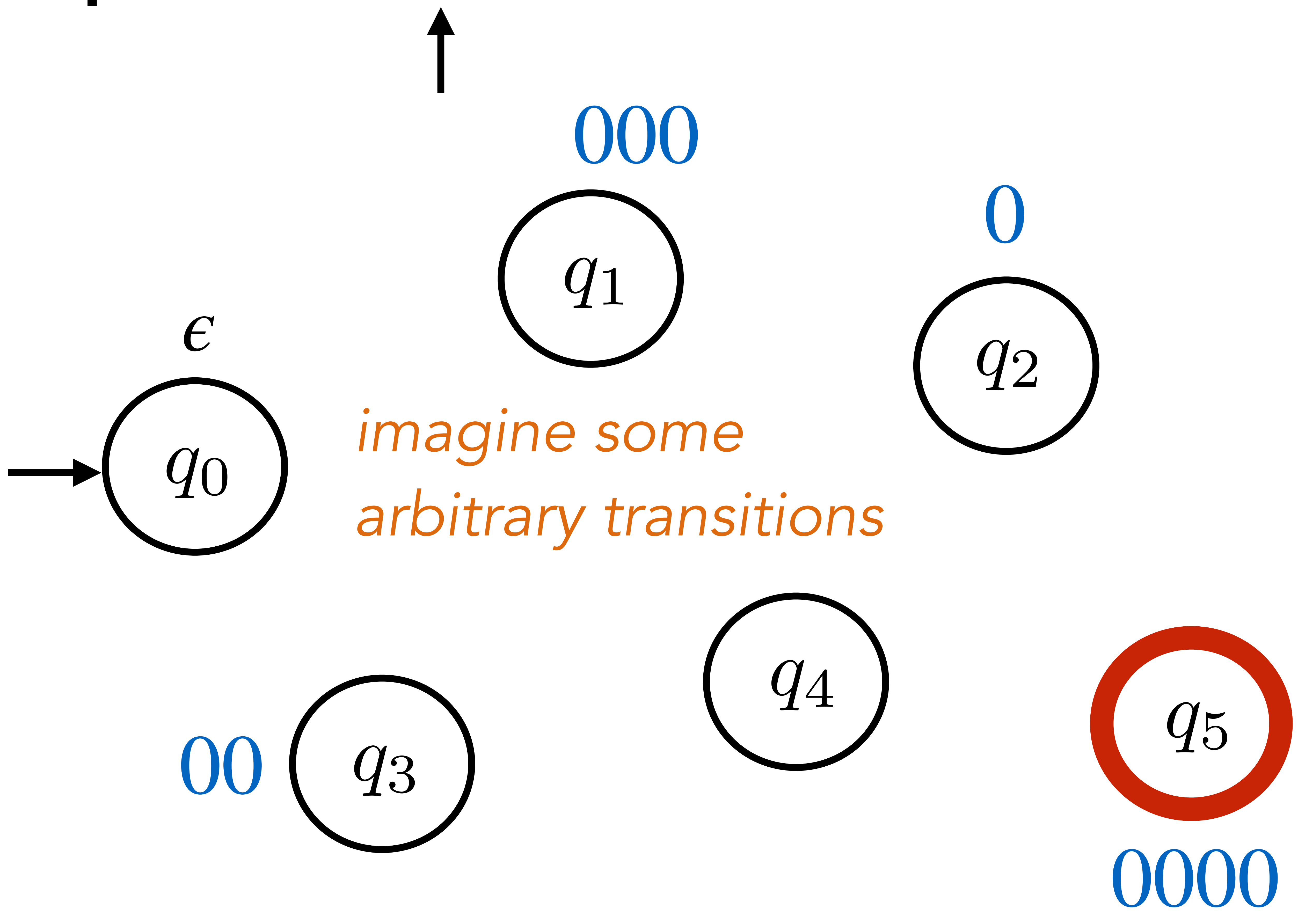
Input: 000000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

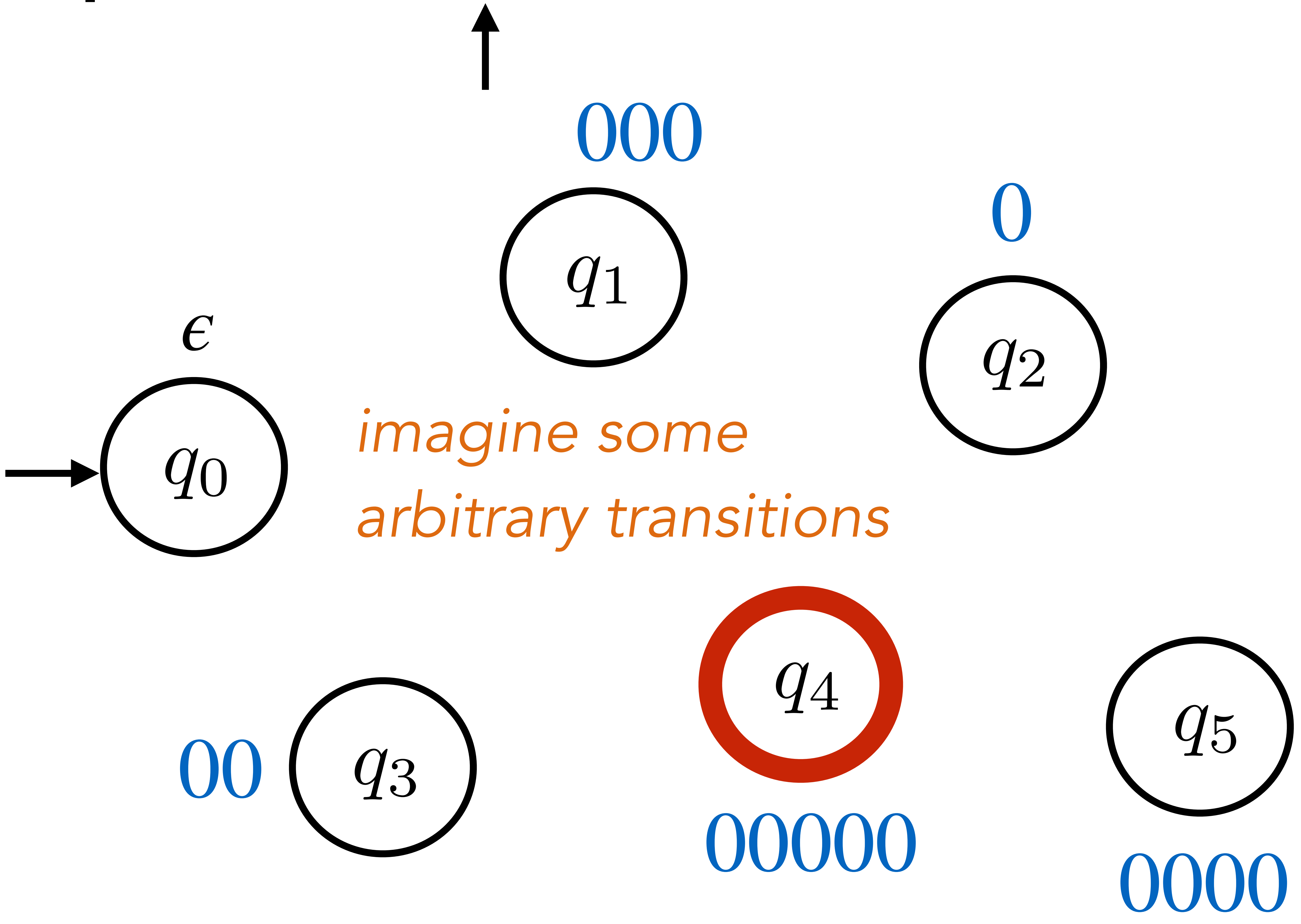
Input: 0000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

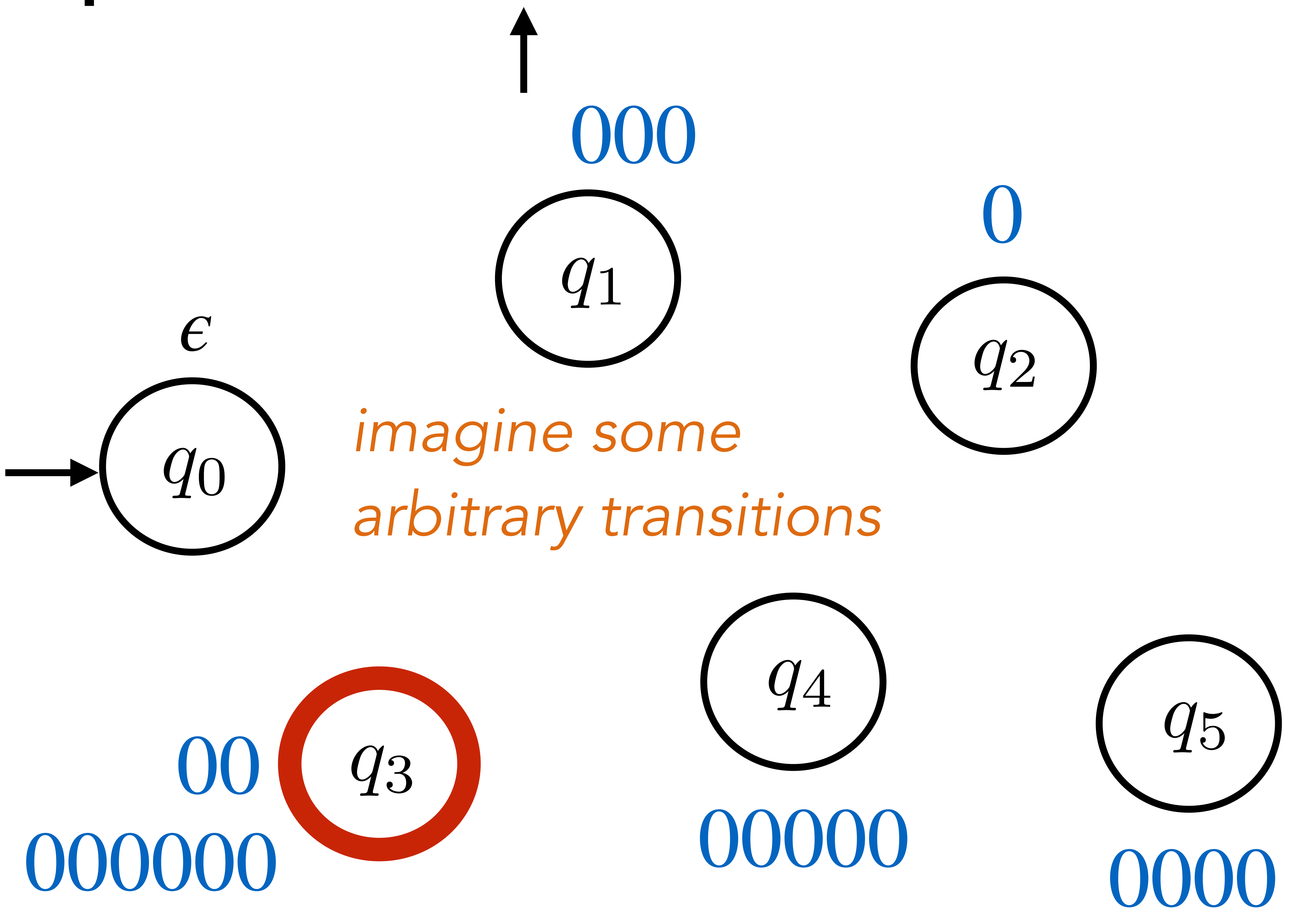
Input: 000000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

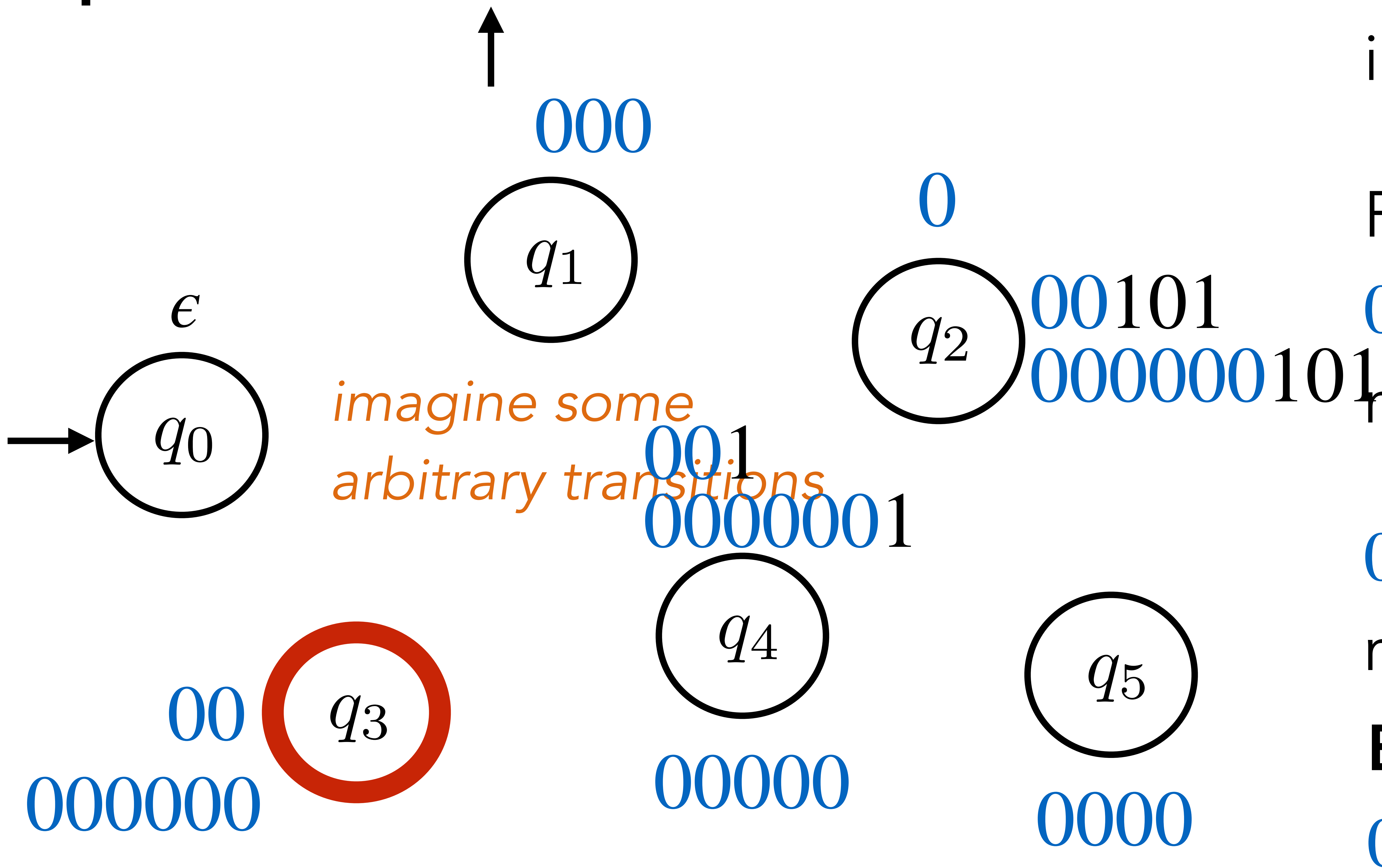
Input: 0000000000000000



$L = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n 1^n : n \in \mathbb{N}\}$.

Input: 0000000000000000



After 00 and 000000 we ended up in the same state q_3.

For *any* string z,
00z and 000000z
must end up in the same state.

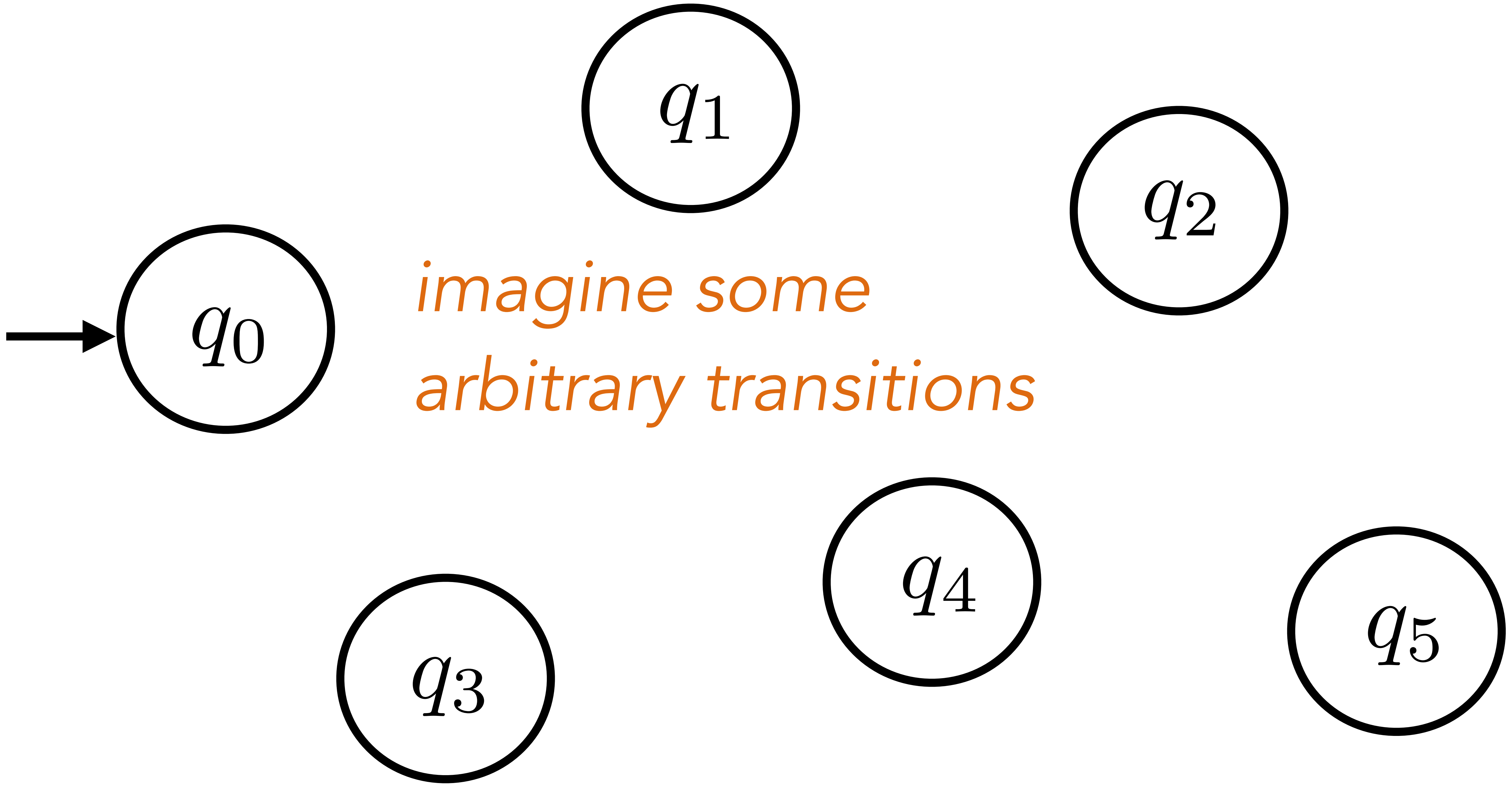
0011 and 00000011
must end up in the same state.

But: 0011 —> accepting state
00000011 —> rejecting state

$L = \{0^n1^n : n \in \mathbb{N}\}$ is not regular - Proof idea

Suppose a DFA with 6 states solves $\{0^n1^n : n \in \mathbb{N}\}$.

Input: 0000000000000000



ϵ
0
00
000
0000
00000
000000

Not enough holes
for all the pigeons.

$L = \{0^n 1^n : n \in \mathbb{N}\}$ not regular - Proof write-up

AFSOC $\exists$ a DFA solving L. Let k be the number of states of the DFA.

Consider the following set of $k+1$ strings $P = \{\epsilon, 0, 00, 000, \ldots, 0^k\}$.

By PHP, $\exists x, y \in P$ such that x and y end up in the same state.

So $\exists i, j,\ i \neq j,$ such that $x = 0^i$ and $y = 0^j$ end up in the same state.

Therefore $\forall z \in \{0,1\}^*,\ 0^i z$ and $0^j z$ end up in the same state.

However for $z = 1^i$, $0^i z = 0^i 1^i$ must end in an accepting state,

whereas since $i \neq j$, $0^j z = 0^j 1^i$ must end in a rejecting state.

This is the desired contradiction. ■

Strategy for proving a language is not regular

1. Set up a proof by contradiction:

Assume that the language **is** regular.

So a DFA with k states solves it.

2. Pick your pigeons: (Would $P = \{1, 11, 111, \ldots\}$ work in previous proof?)

Identify $k+1$ strings as the pigeons.

Two pigeons, x and y, must end up in the same state.

For **any** string z, xz and yz end up in the same state.

3. Reach a contradiction:

Find a string z such that exactly one of xz, yz is in L.

Exercise:

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

Show $L = \{ca^n b^{2n} : n \in \mathbb{N}\}$ is not regular.

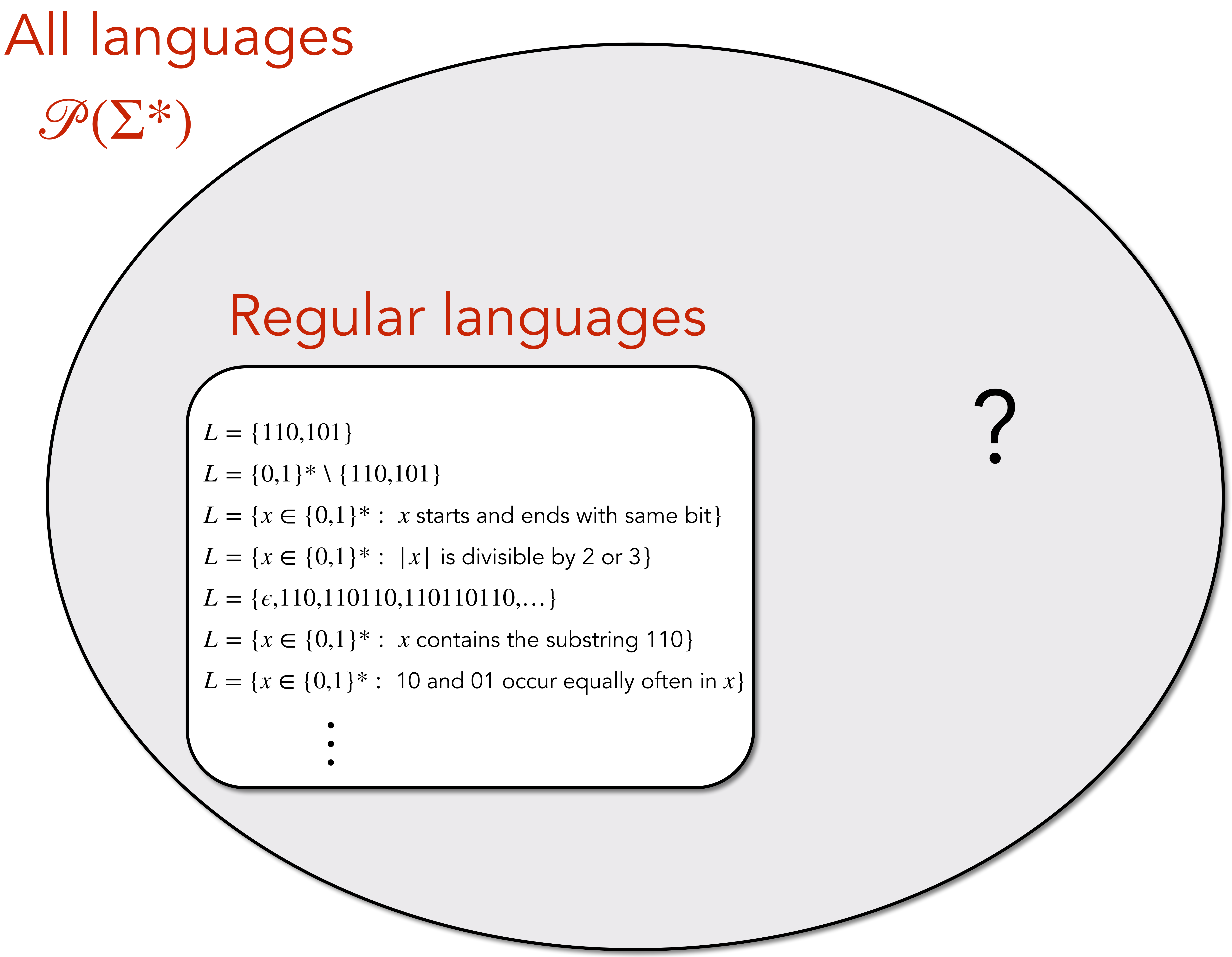
$L = \{c, cabb, caabbbb, caaabbbbbb, \ldots\}$

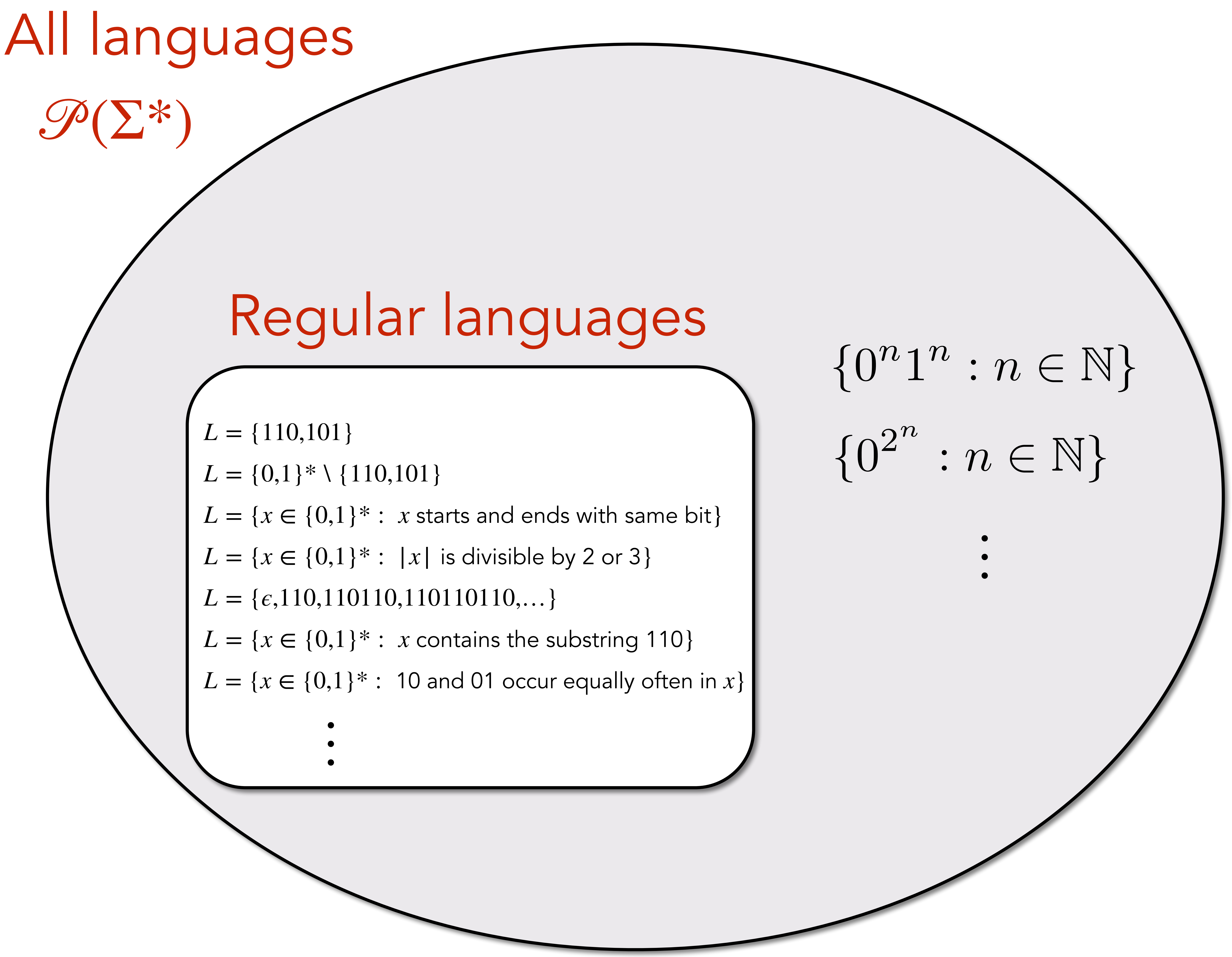
Exercise:

$\Sigma = \{0\}$

Show $L = \{0^{2^n} : n \in \mathbb{N}\}$ is not regular.

$L = \{0, 00, 0000, 00000000, \ \ldots\}$


All languages
$\mathscr{P}(\Sigma^*)$
Regular languages
$L = \{110,101\}$
$L = \{0,1\}^* \setminus \{110,101\}$
$L = \{x \in \{0,1\}^* : x$ starts and ends with same bit$\}$
$L = \{x \in \{0,1\}^* : |x|$ is divisible by 2 or 3$\}$
$L = \{\epsilon, 110, 110110, 110110110, \dots\}$
$L = \{x \in \{0,1\}^* : x$ contains the substring 110$\}$
$L = \{x \in \{0,1\}^* :$ 10 and 01 occur equally often in $x\}$
⋮
?


All languages
$\mathscr{P}(\Sigma^*)$
Regular languages
$L = \{110, 101\}$
$L = \{0,1\}^* \setminus \{110, 101\}$
$L = \{x \in \{0,1\}^* :$ x starts and ends with same bit$\}$
$L = \{x \in \{0,1\}^* :$ $|x|$ is divisible by 2 or 3$\}$
$L = \{\epsilon, 110, 110110, 110110110, \ldots\}$
$L = \{x \in \{0,1\}^* :$ x contains the substring 110$\}$
$L = \{x \in \{0,1\}^* :$ 10 and 01 occur equally often in $x\}$
$\vdots$
$\{0^n 1^n : n \in \mathbb{N}\}$
$\{0^{2^n} : n \in \mathbb{N}\}$
$\vdots$