COS210 - Theoretical Computer Science
Finite Automata and Regular Languages (Part 1)

Motivating Example: Vending Machine

Consider a simple coffee vending machine:

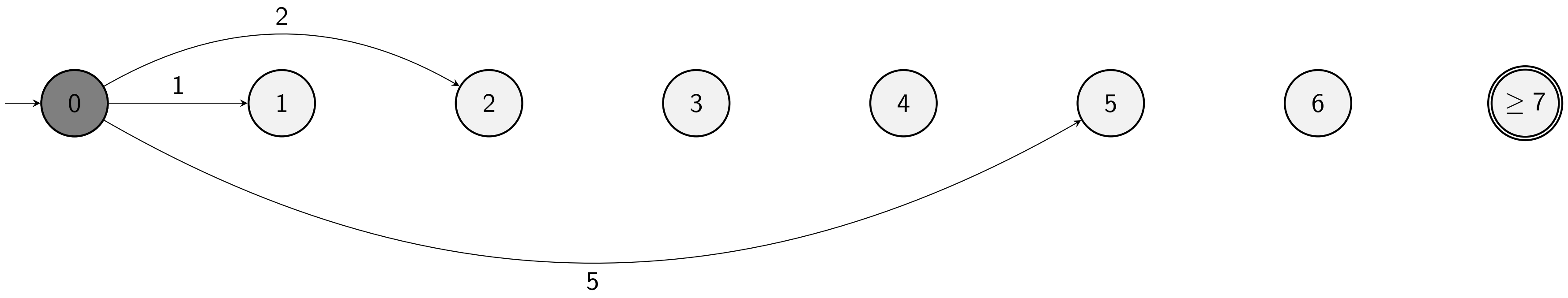
- The vending machine takes R1, R2 and R5 coins as an input
- Coffee will be released after at least R7 have been inserted
- The machine does not give change

Modelling questions:

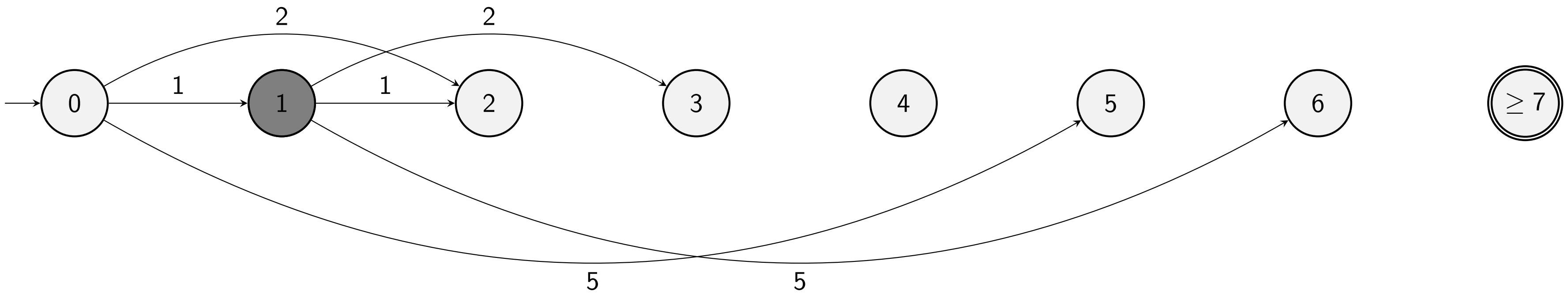
- What are the possible "states" of the machine?
- How does the insertion of coins change the "state" of the machine?

Motivating Example: Vending Machine

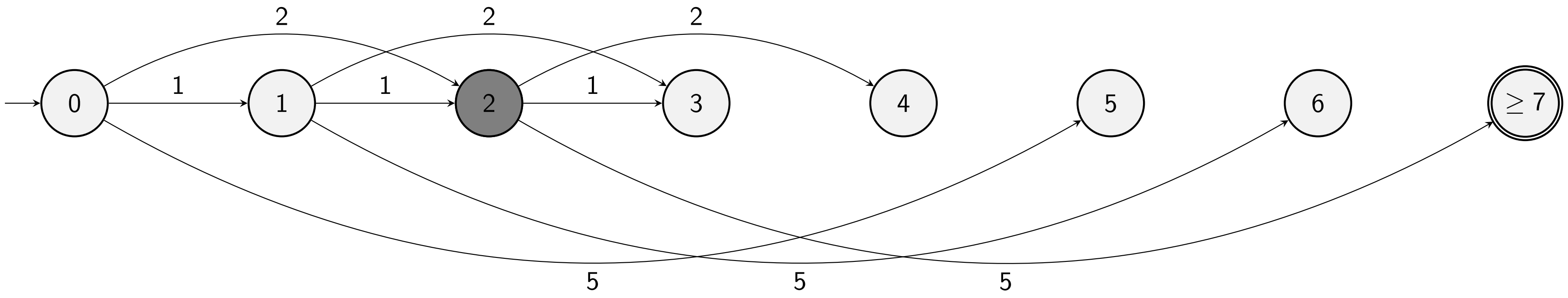
Motivating Example: Vending Machine



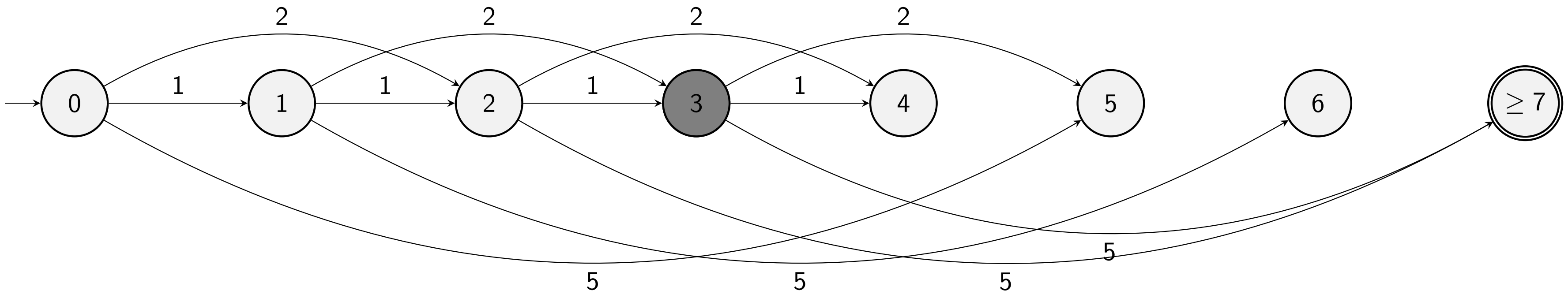
Motivating Example: Vending Machine



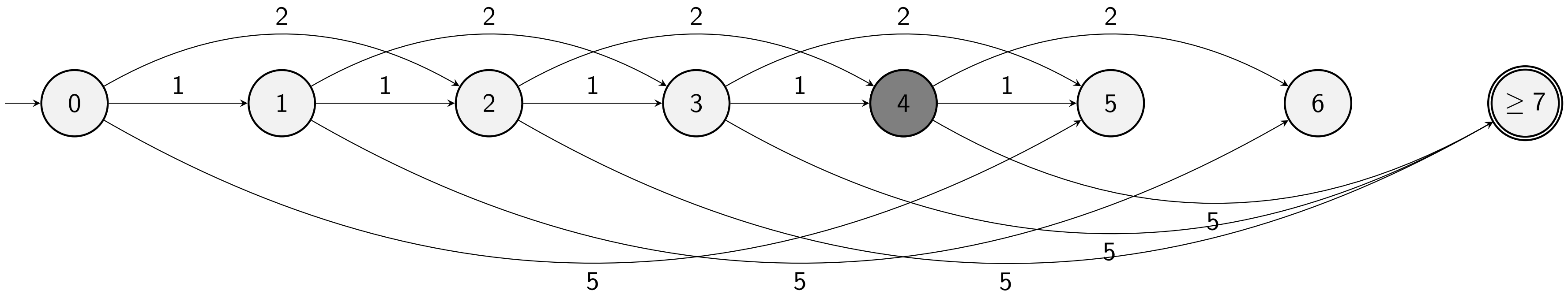
Motivating Example: Vending Machine



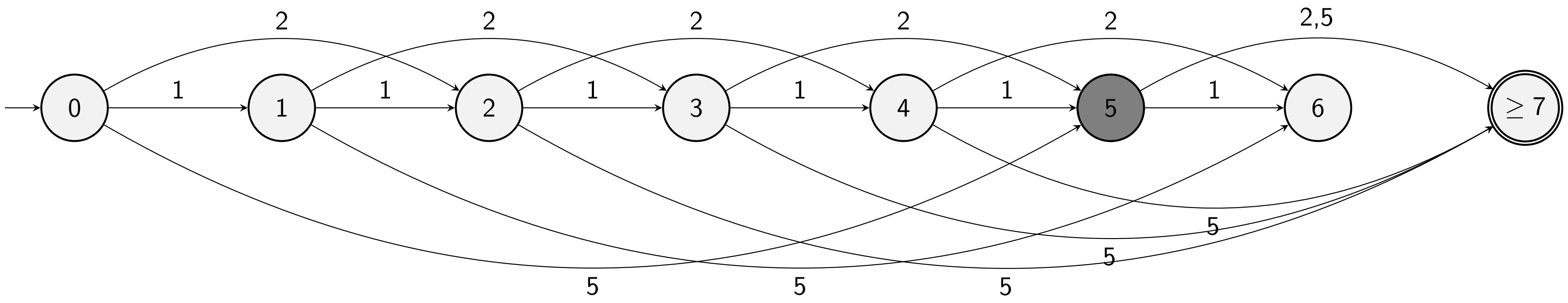
Motivating Example: Vending Machine



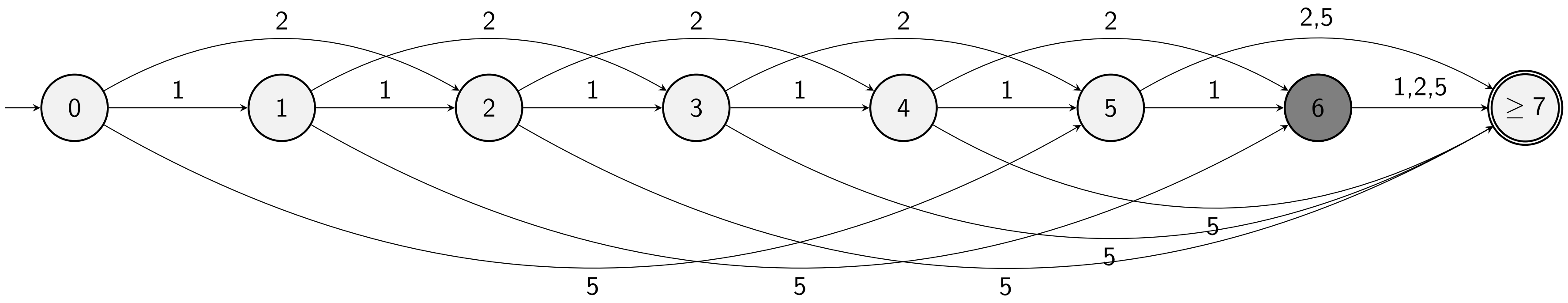
Motivating Example: Vending Machine



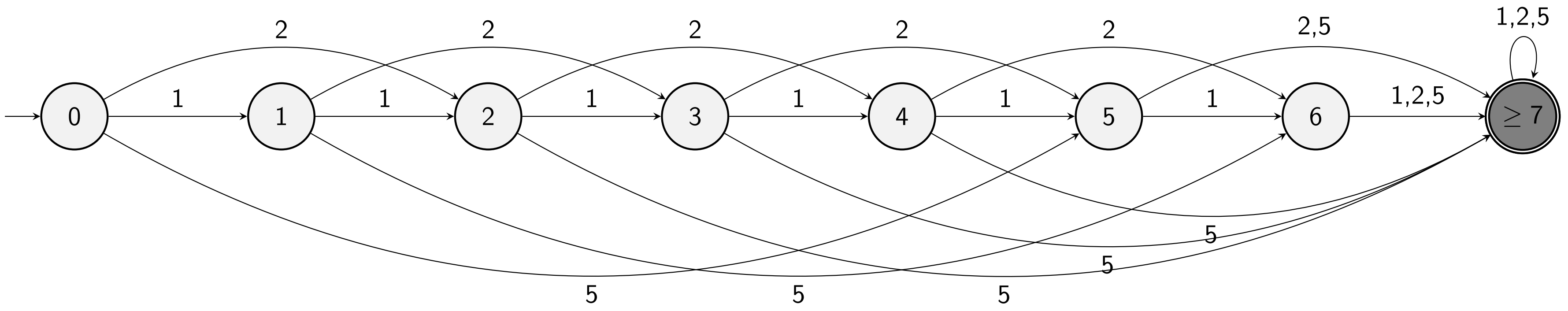
Motivating Example: Vending Machine



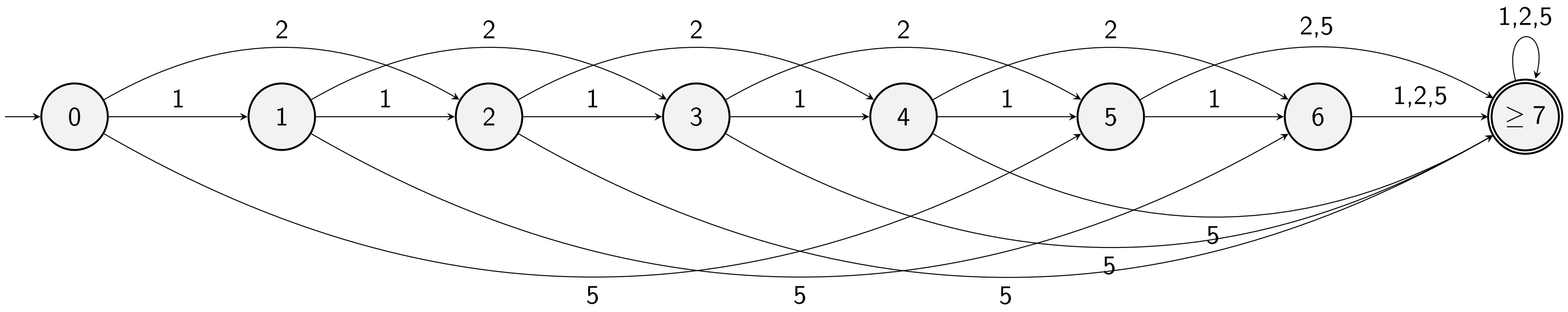
Motivating Example: Vending Machine



Motivating Example: Vending Machine



Motivating Example: Vending Machine



Example of accepted input: $1, 5, 2$

Example of rejected input: $2, 1, 2, 1$

Abstract Example

Consider the following abstract machine:

- The machine takes finite sequences of bits as an input, e.g.

$$\mathbf{0010111101010001110101100100 0}$$

  which we call an input string
- The machine only accepts input strings that end with $\mathbf{00}$

Such a machine can be modelled as a deterministic finite automaton (DFA)

Deterministic Finite Automaton for Abstract Example

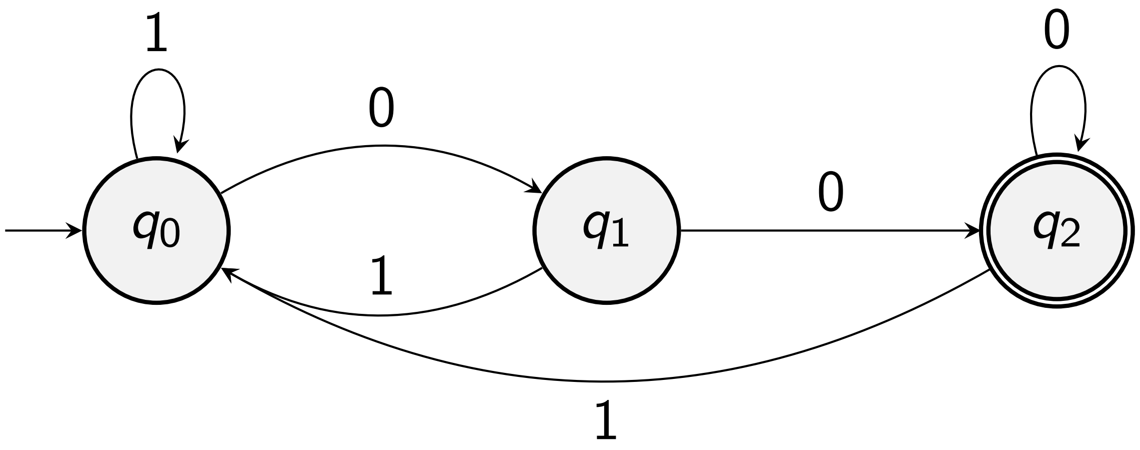


Figure: DFA that accepts only strings ending with 00

- start state: q_0, accepting state: q_2
- transitions show how an input changes the state of the DFA

Deterministic Finite Automaton for Abstract Example

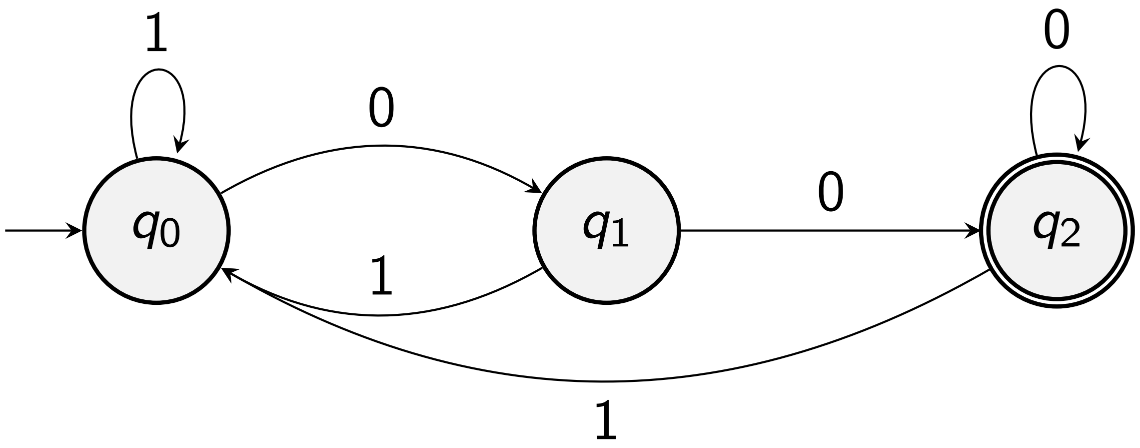


Figure: DFA that accepts only strings ending with 00

- start state: q_0, accepting state: q_2
- transitions show how an input changes the state of the DFA
- accepted inputs: 00, 0000, 110100, ...
- non-accepted inputs: 111, 1001, 00110, ...

DFA: Formal Definition

Definition

A deterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q, F)$, where

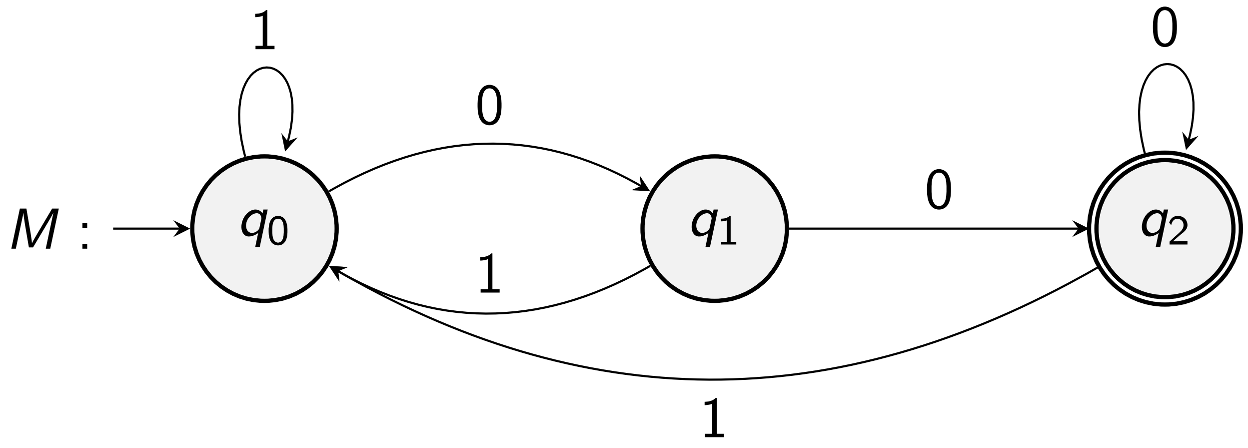
- Q is a finite set, whose elements are called *states*,
- Σ is a finite set, called the *alphabet*;
  - the elements of Σ are called *symbols*,
- $\delta : Q \times \Sigma \to Q$ is a function, called the *transition function*,
- q is an element of Q; it is called the *start state*,
- F is a subset of Q; the elements of F are called *accept states*.

DFA: Formal Definition

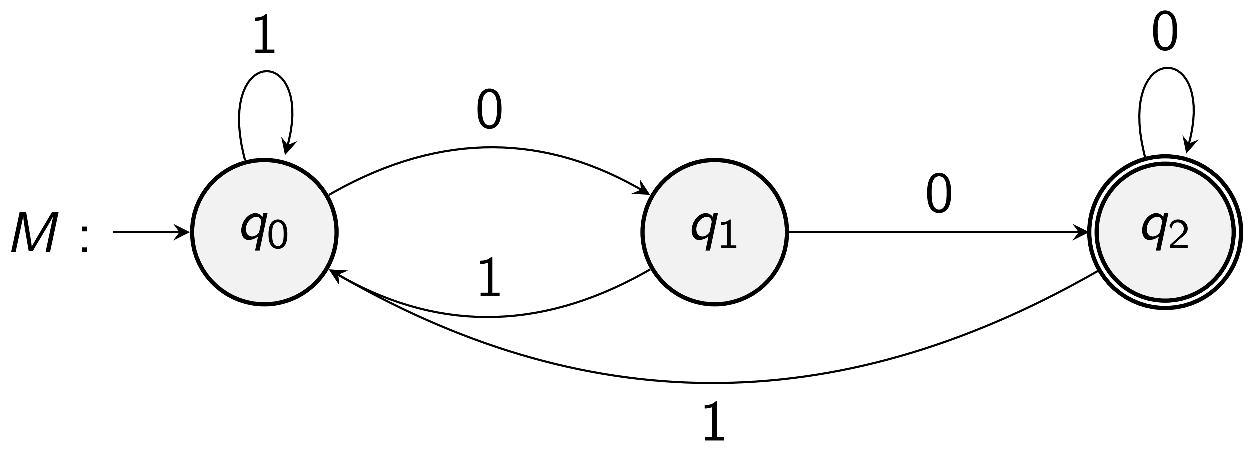
Definition

A deterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q, F)$, where

- Q is a finite set, whose elements are called *states*,
- Σ is a finite set, called the *alphabet*;
    - the elements of Σ are called *symbols*,
- $\delta : Q \times \Sigma \to Q$ is a function, called the *transition function*,
- q is an element of Q; it is called the *start state*,
- F is a subset of Q; the elements of F are called *accept states*.



DFA Graphically and Formally

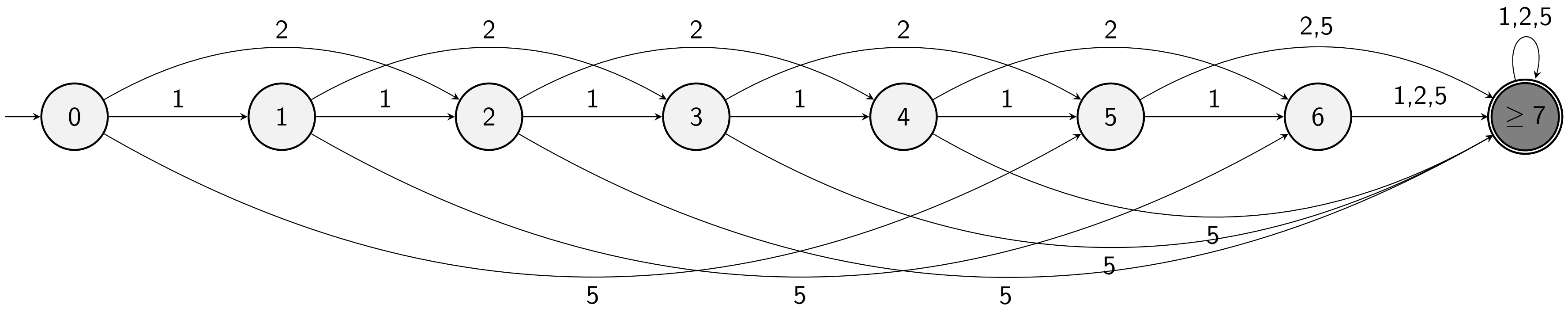


- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- $q = q_0$
- $F = \{q_2\}$

The transition function δ:

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

Motivating Example: Vending Machine



Deterministic Finite Automaton for the Vending Machine

Consider the vending machine example:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
- $\Sigma = \{R1, R2, R5\}$
- q_0 is the *start state*, $q = q_0$
- $F = \{q_7\}$
- δ is given by (complete for homework):

	R1	R2	R5
q_0	q_1	q_2	q_5
q_1	q_2	q_3	q_6
q_2	q_3	q_4	q_7
...	...	...	...

Abstract Example 2

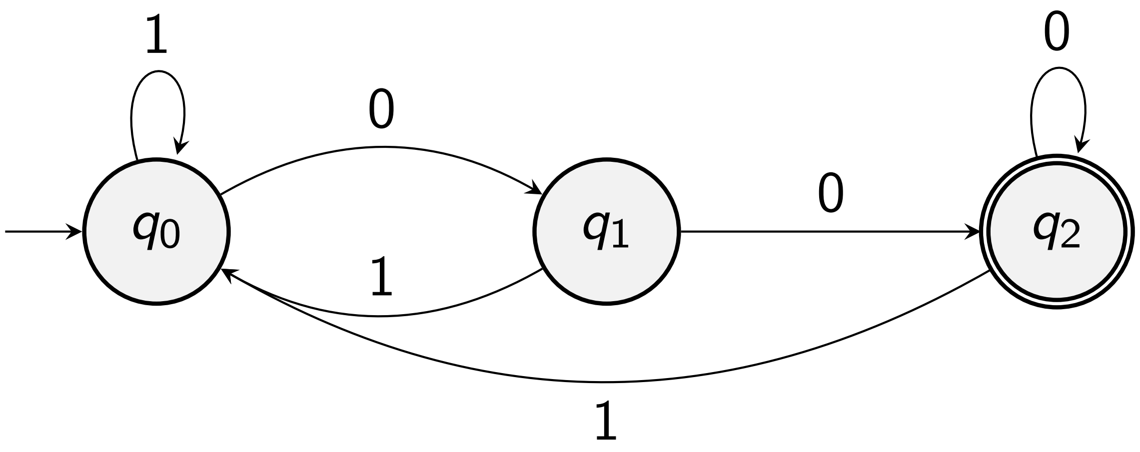
Consider the following extension of our earlier abstract example:

- The machine takes finite sequences of bits as an input
- The machine only accepts inputs that either end with $\mathbf{00}$ or with $\mathbf{11}$

Model this machine as a deterministic finite automaton

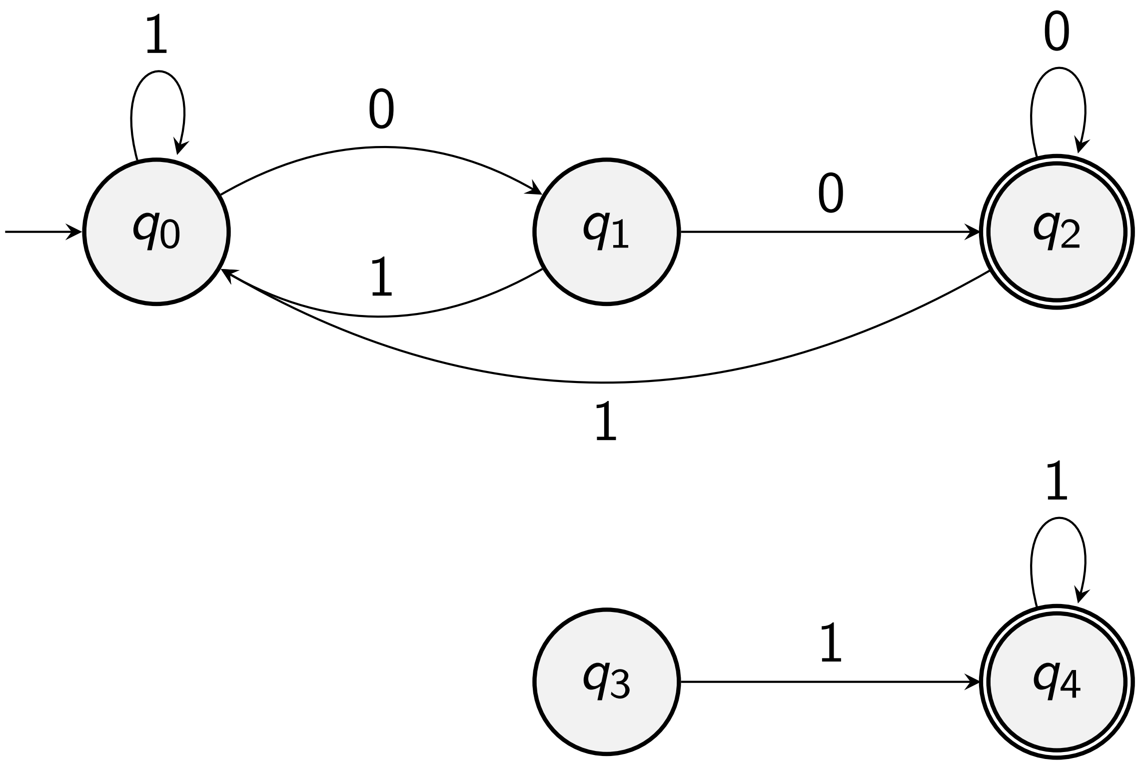
This can be done by extending and modifying the DFA for Abstract Example 1

DFA for Abstract Example 2



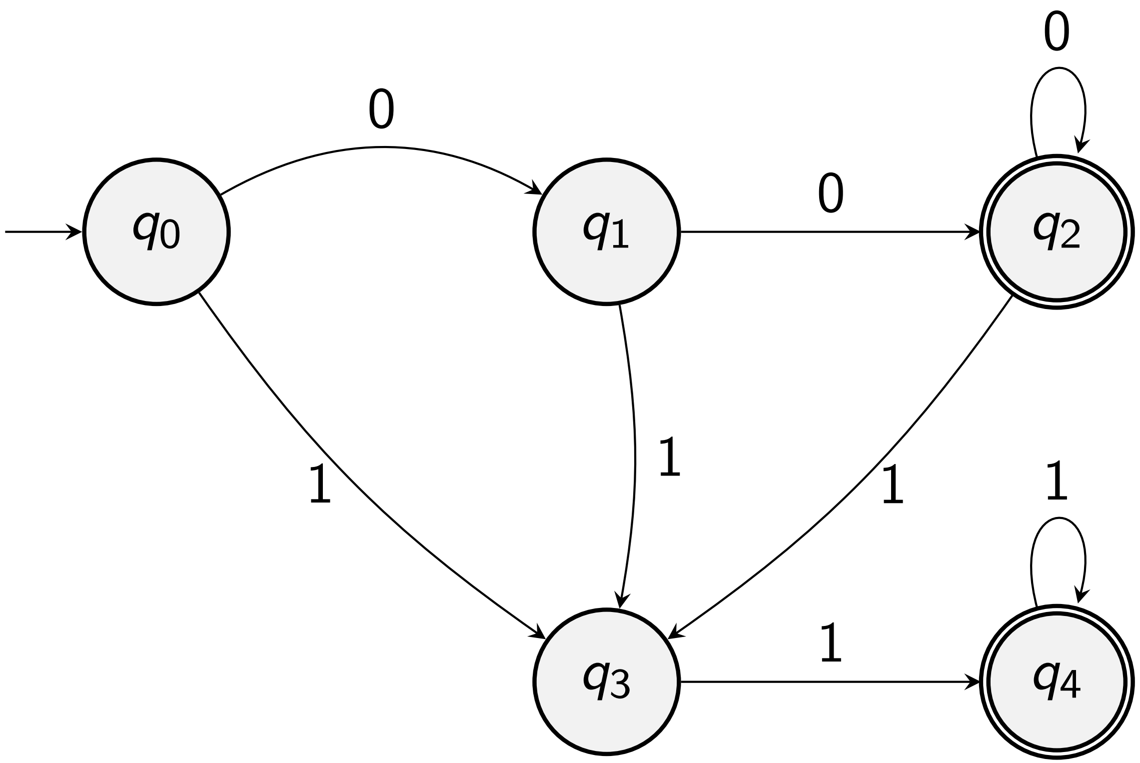
- $q0$: last processed bit was not 0
- $q1$: last processed bit was 0, but second last bit was not 0
- $q2$: last two processed bits were 0

DFA for Abstract Example 2



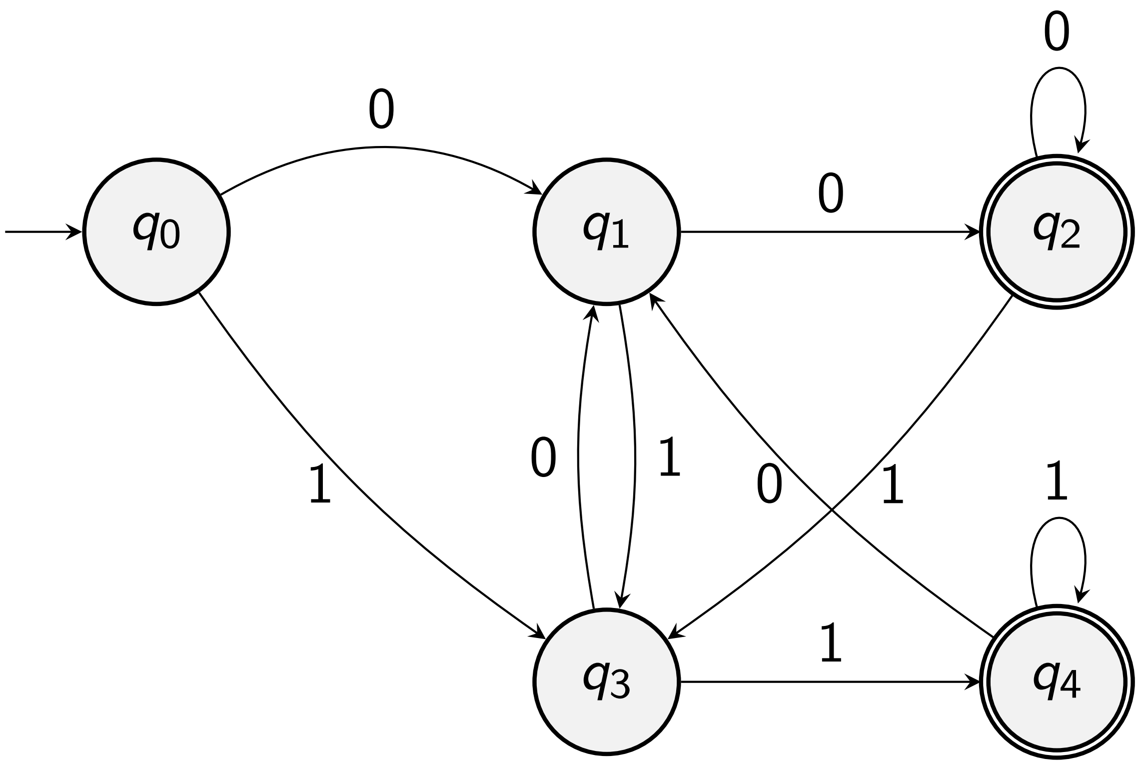
- $q0$: last processed bit was not 0
- $q1$: last processed bit was 0, but second last bit was not 0
- $q2$: last two processed bits were 0
- $q3$: last processed bit was 1, but second last bit was not 1
- $q4$: last two processed bits were 1

DFA for Abstract Example 2



- $q0$: **no bits processed yet**
- $q1$: last processed bit was 0, but second last bit was not 0
- $q2$: last two processed bits were 0
- $q3$: last processed bit was 1, but second last bit was not 1
- $q4$: last two processed bits were 1

DFA for Abstract Example 2



- $q0$: **no bits processed yet**
- $q1$: last processed bit was 0, but second last bit was not 0
- $q2$: last two processed bits were 0
- $q3$: last processed bit was 1, but second last bit was not 1
- $q4$: last two processed bits were 1

Language of an Automaton

A language is a set of strings over an alphabet.

The *language* $L(M)$ of an automaton M is the set of all input strings accepted by M.

General form:

$$L(M) = \{w : \text{string } w \text{ accepted by } M\}$$

For our abstract Example 2 we get:

$$L(M) = \{w : \ \text{string } w \text{ ends with a } 00 \text{ or } 11\}$$

Subsequently, we will formally define what *acceptance* is.

Runs and Acceptance

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton and let $w = w_1 \ldots w_n$ be a string over Σ. A **run** of M over w is a sequence of states $q_0, \ldots, q_n$ such that

- $q_0 = q$,
- $\delta(q_i, w_{i+1}) = q_{i+1}$, for all $i < n$.
  (the transitions along $q_0, \ldots, q_n$ are labelled with $w_1 \ldots w_n$)

A string w is **accepted** by M if the following holds for the run over w:

- $q_n \in F$.

Otherwise a string is **rejected** by M.

Note that a string w may be the empty string, denoted by $w = \epsilon$. In this case the run over w consists of the start state only.

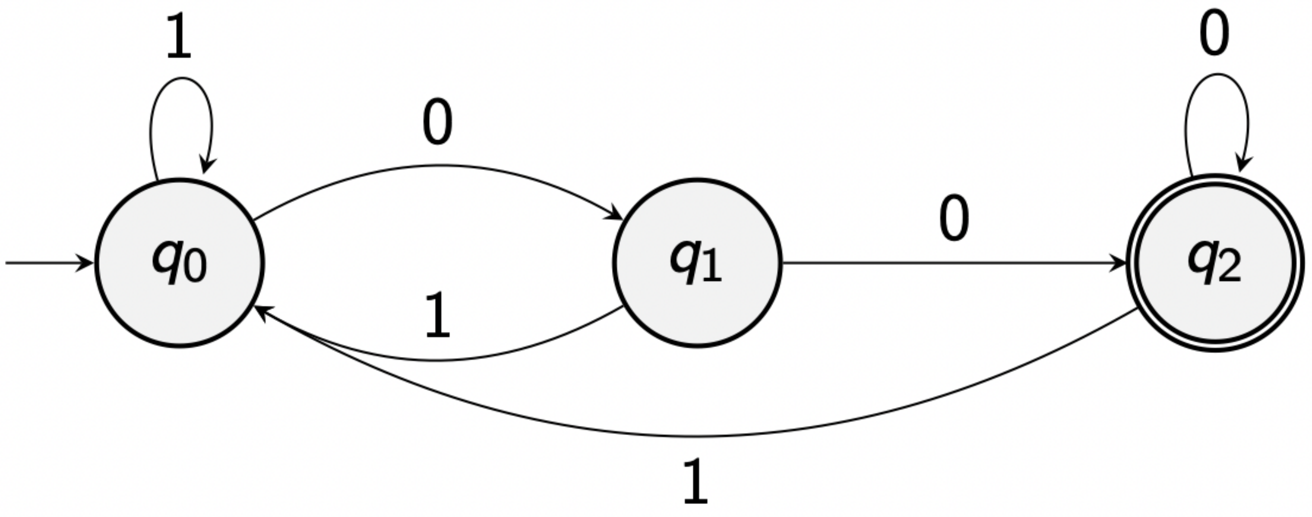
Runs and Acceptance

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton and let $w = w_1 \ldots w_n$ be a string over Σ. A **run** of M over w is a sequence of states $q_0, \ldots, q_n$ such that

- $q_0 = q$,
- $\delta(q_i, w_{i+1}) = q_{i+1}$, for all $i < n$.
  (the transitions along $q_0, \ldots, q_n$ are labelled with $w_1 \ldots w_n$)

Example:
For input w = 101100
The corresponding run is q_0, q_0, q_1, q_0, q_0, q_1, q_2



Runs and Acceptance

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton and let $w = w_1 \dots w_n$ be a string over Σ. A **run** of M over w is a sequence of states $q_0, \dots, q_n$ such that

- $q_0 = q$,
- $\delta(q_i, w_{i+1}) = q_{i+1}$, for all $i < n$.
  (the transitions along $q_0, \dots, q_n$ are labelled with $w_1 \dots w_n$)

A string w is **accepted** by M if the following holds for the run over w:

- $q_n \in F$.

Otherwise a string is **rejected** by M.

Note that a string w may be the empty string, denoted by $w = \epsilon$. In this case the run over w consists of the start state only.

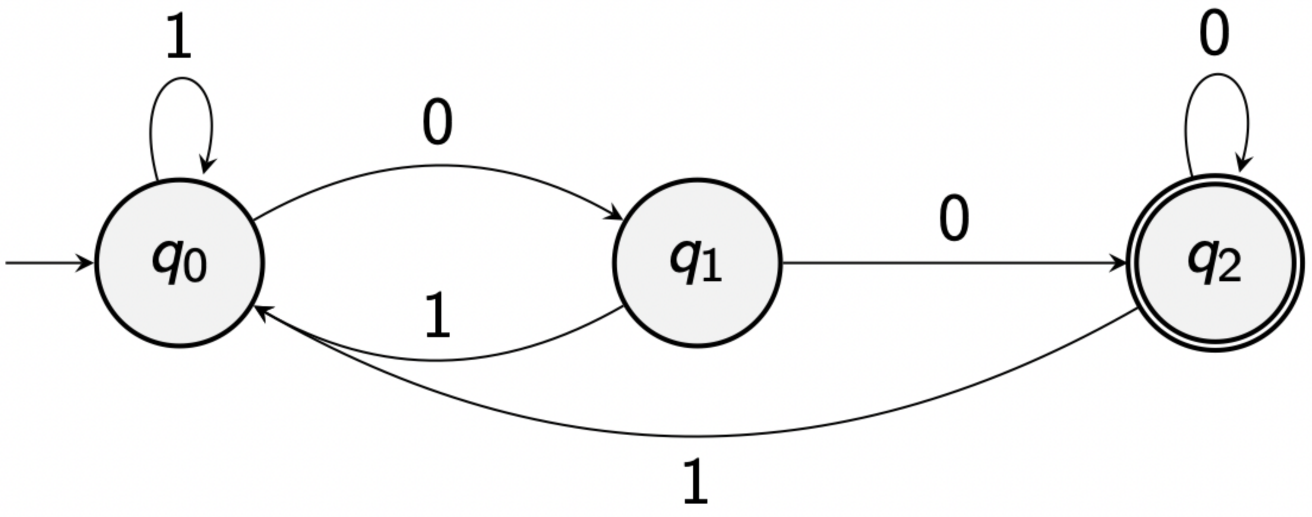
Runs and Acceptance

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton and let $w = w_1 \ldots w_n$ be a string over Σ. A **run** of M over w is a sequence of states $q_0, \ldots, q_n$ such that

- $q_0 = q$,
- $\delta(q_i, w_{i+1}) = q_{i+1}$, for all $i < n$.
  (the transitions along $q_0, \ldots, q_n$ are labelled with $w_1 \ldots w_n$)

Example:
For input w = 101100
The corresponding run is q_0, q_0, q_1, q_0, q_0, q_1, q_2



Runs and Acceptance

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton and let $w = w_1 \dots w_n$ be a string over Σ. A **run** of M over w is a sequence of states $q_0, \dots, q_n$ such that

- $q_0 = q$,
- $\delta(q_i, w_{i+1}) = q_{i+1}$, for all $i < n$.
  (the transitions along $q_0, \dots, q_n$ are labelled with $w_1 \dots w_n$)

A string w is **accepted** by M if the following holds for the run over w:

- $q_n \in F$.

Otherwise a string is **rejected** by M.

Note that a string w may be the empty string, denoted by $w = \epsilon$. In this case the run over w consists of the start state only.

Regular Languages

Definition

Let $M = (Q, \Sigma, \delta, q, F)$ be an automaton. The **language** $L(M)$ of M is the set of all strings that are accepted by M:

$$L(M) = \{w : w \text{ is a string over } \Sigma \text{ and } M \text{ accepts } w\}$$

Definition

A language L is called **regular**, if there exists a finite automaton M such that

$$L = L(M).$$

How to prove that a language L is regular?
Construct finite automaton M with $L(M) = L$ (proof by construction).

Example

Is the following language regular? Is there an M with $L(M) = L$?

$$L = \{w \text{ over } \{0,1\} : \text{the third last symbol of } w \text{ is } 1\}$$

Example

Is the following language regular? Is there an M with $L(M) = L$?

$$L = \{w \text{ over } \{0,1\} : \text{the third last symbol of } w \text{ is } 1\}$$

Idea for the construction of M with $L(M) = L$:

- we can observe that each string accepted by M must be of the form $w = \ldots \mathbf{1}\ \mathbf{y}\ \mathbf{z}$ where $y, z \in \{0, 1\}$

Example

Is the following language regular? Is there an M with $L(M) = L$?

$$L = \{w \text{ over } \{0,1\} : \text{the third last symbol of } w \text{ is } 1\}$$

Idea for the construction of M with $L(M) = L$:

- we can observe that each string accepted by M must be of the form $w = \ldots \mathbf{1}\ \mathbf{y}\ \mathbf{z}$ where $y, z \in \{0,1\}$
- we use the binary numbered states $q_{000}, \ldots, q_{111}$ for M
- the digits shall represent the last three symbols along a run of M:

$$\mathbf{q_{xyz}}$$

the last symbol was **z**, the second last was **y**, the third last was **x**

Example

Is the following language regular? Is there an M with $L(M) = L$?

$$L = \{w \text{ over } \{0,1\} : \text{the third last symbol of } w \text{ is } 1\}$$

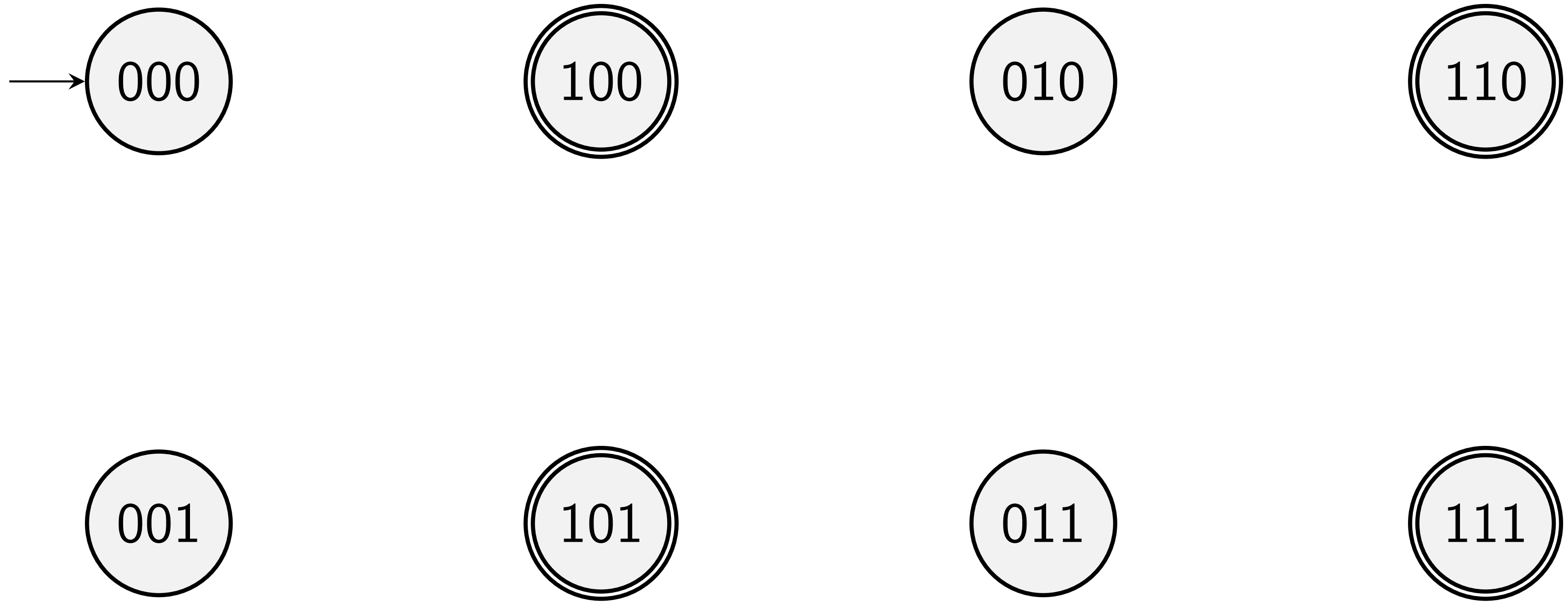
Idea for the construction of M with $L(M) = L$:

- we can observe that each string accepted by M must be of the form $w = \ldots \mathbf{1}\,\mathbf{y}\,\mathbf{z}$ where $y, z \in \{0,1\}$
- we use the binary numbered states $q_{000}, \ldots, q_{111}$ for M
- the digits shall represent the last three symbols along a run of M:

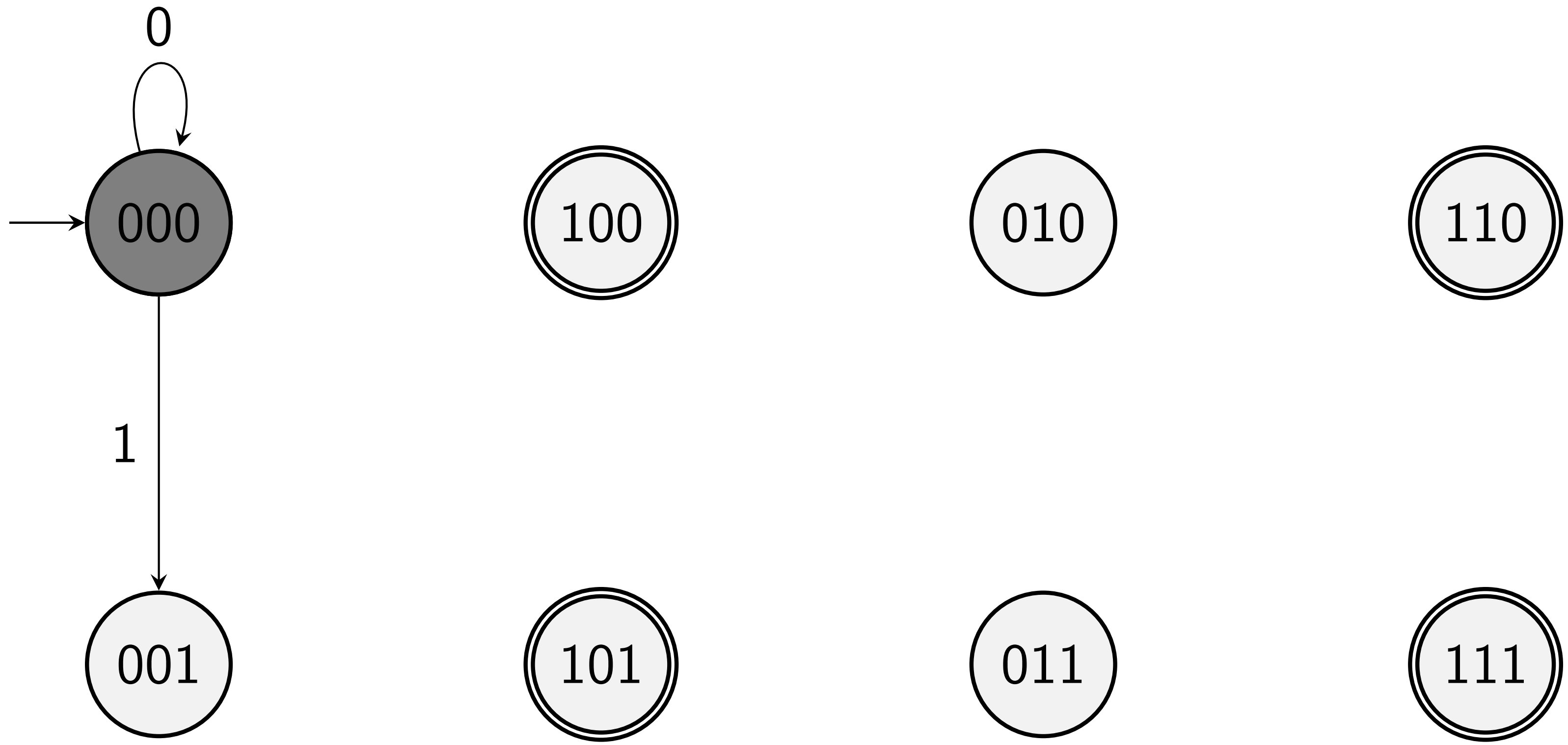
  $$\mathbf{q_{xyz}}$$

  the last symbol was **z**, the second last was **y**, the third last was **x**
- we start in q_{000} (no 1's encountered so far)
- the accepting states are $q_{100}, q_{101}, q_{110}, q_{111}$

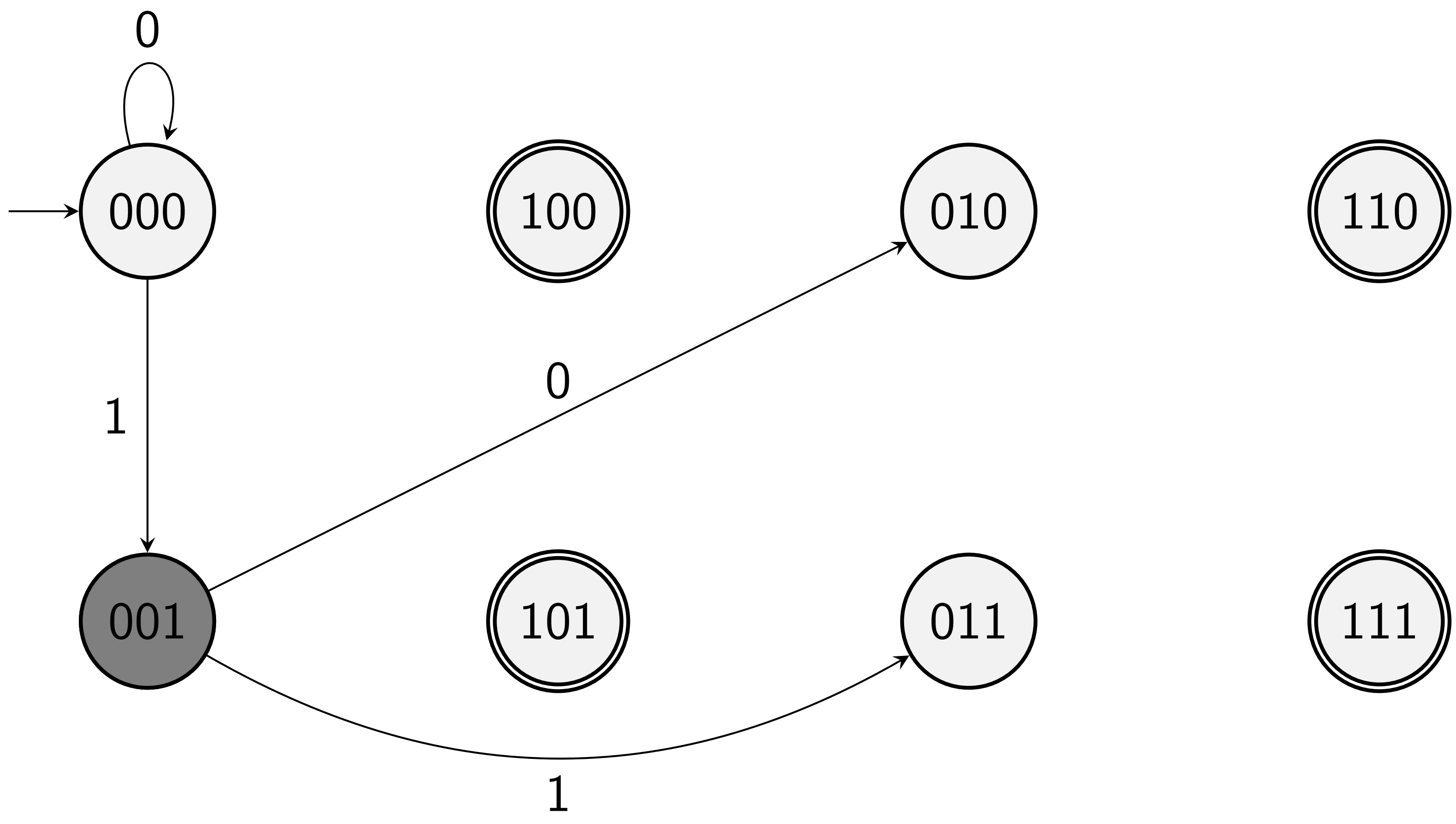
Example



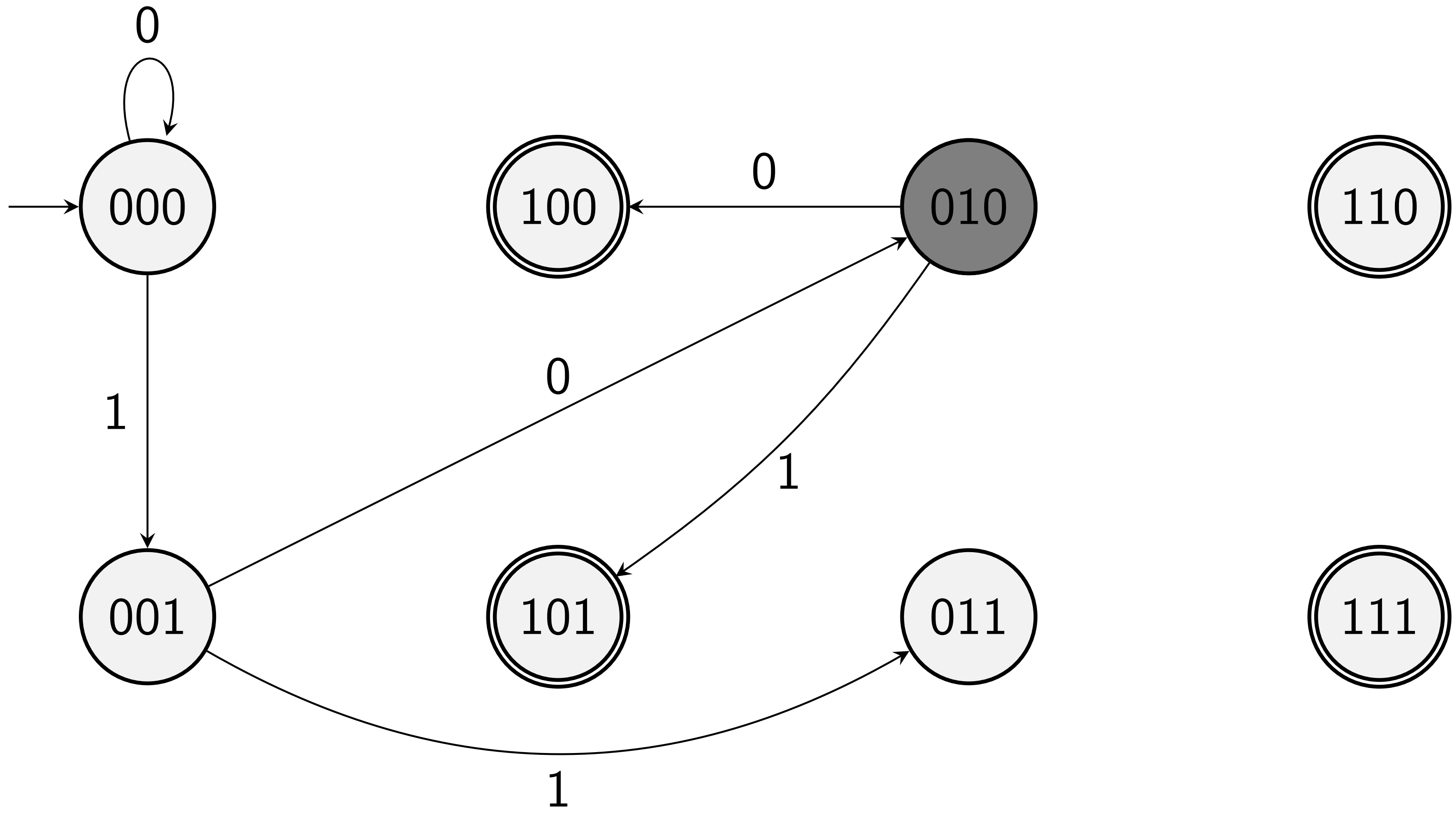
Example



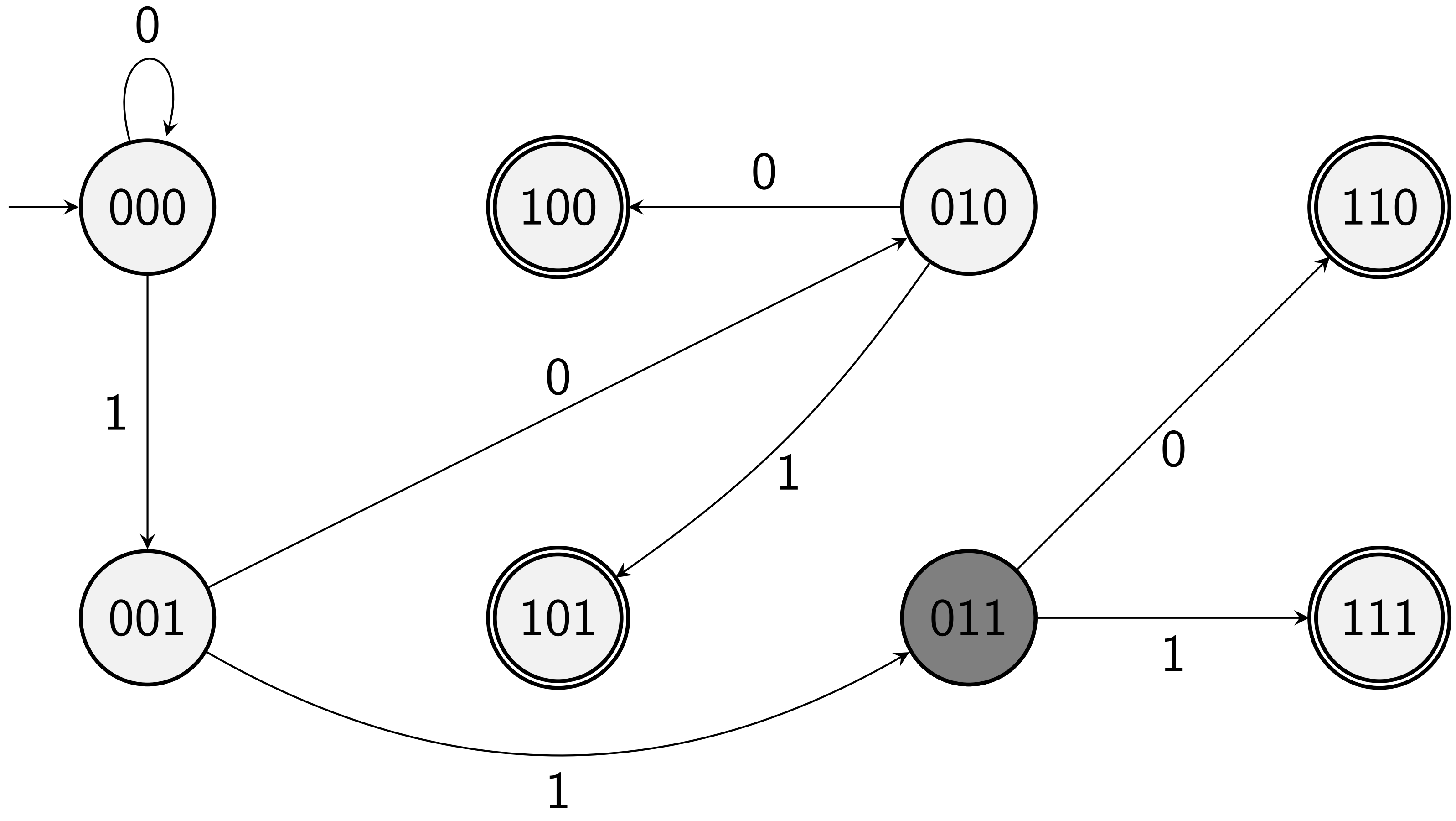
Example



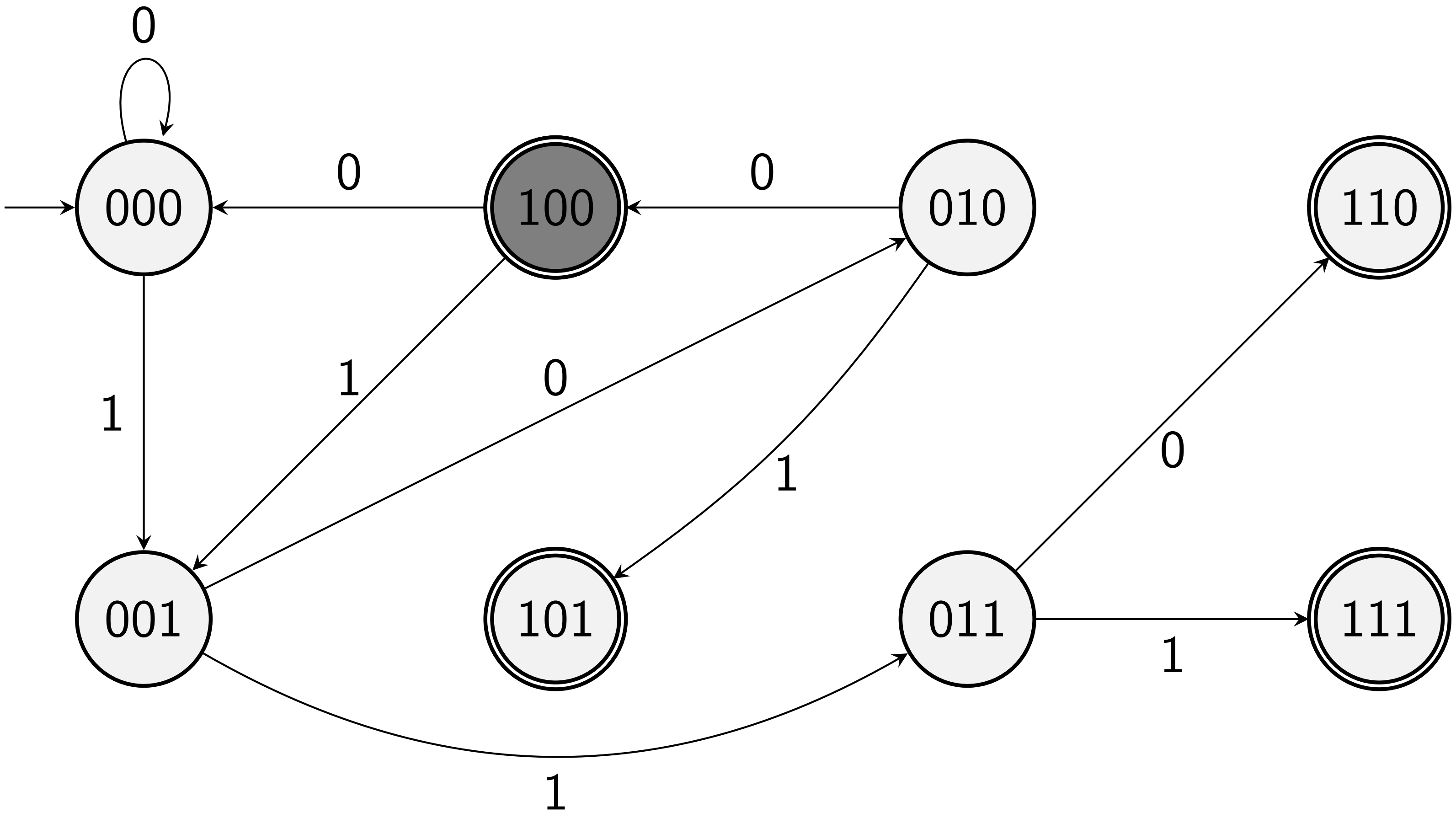
Example



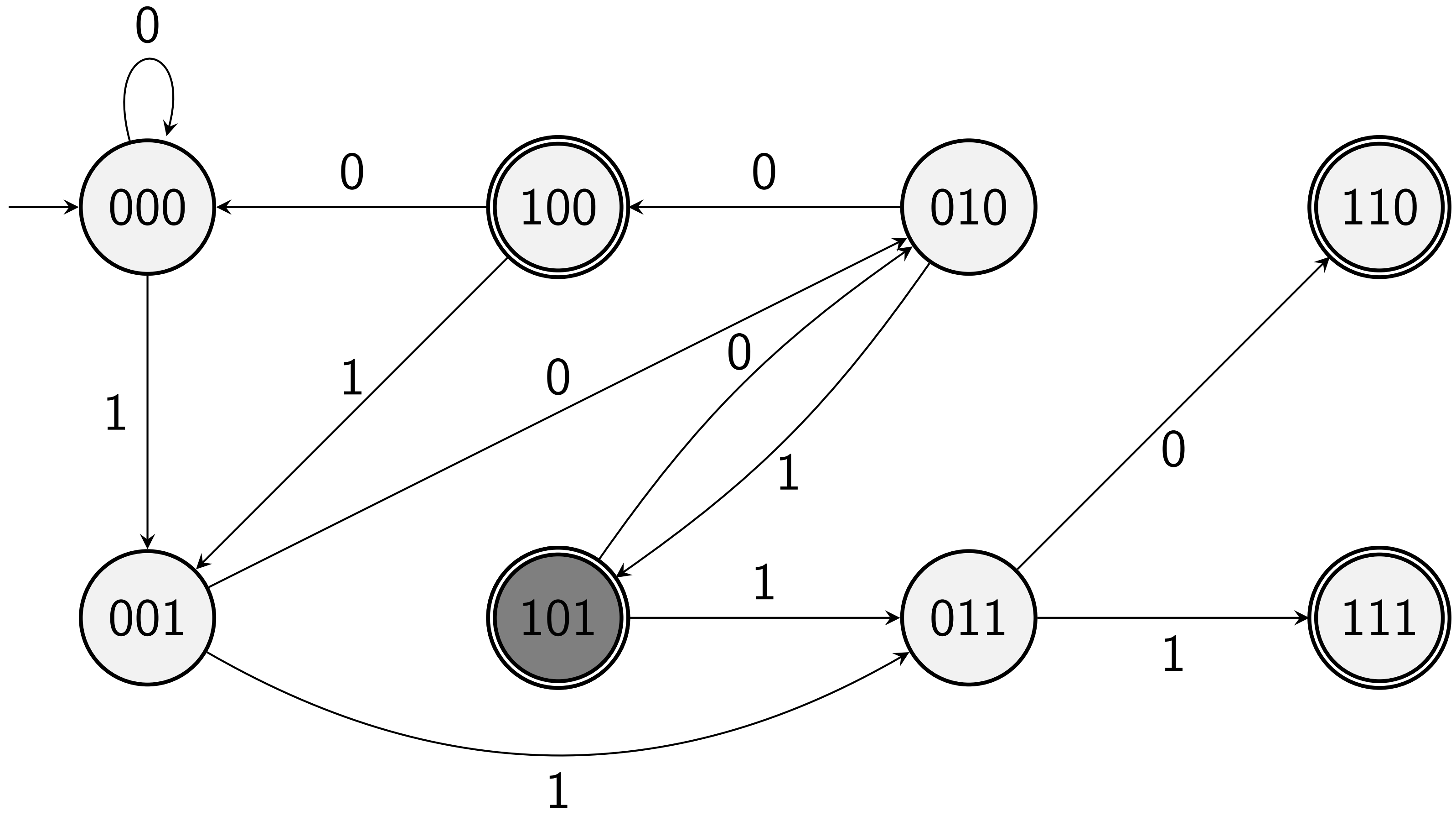
Example



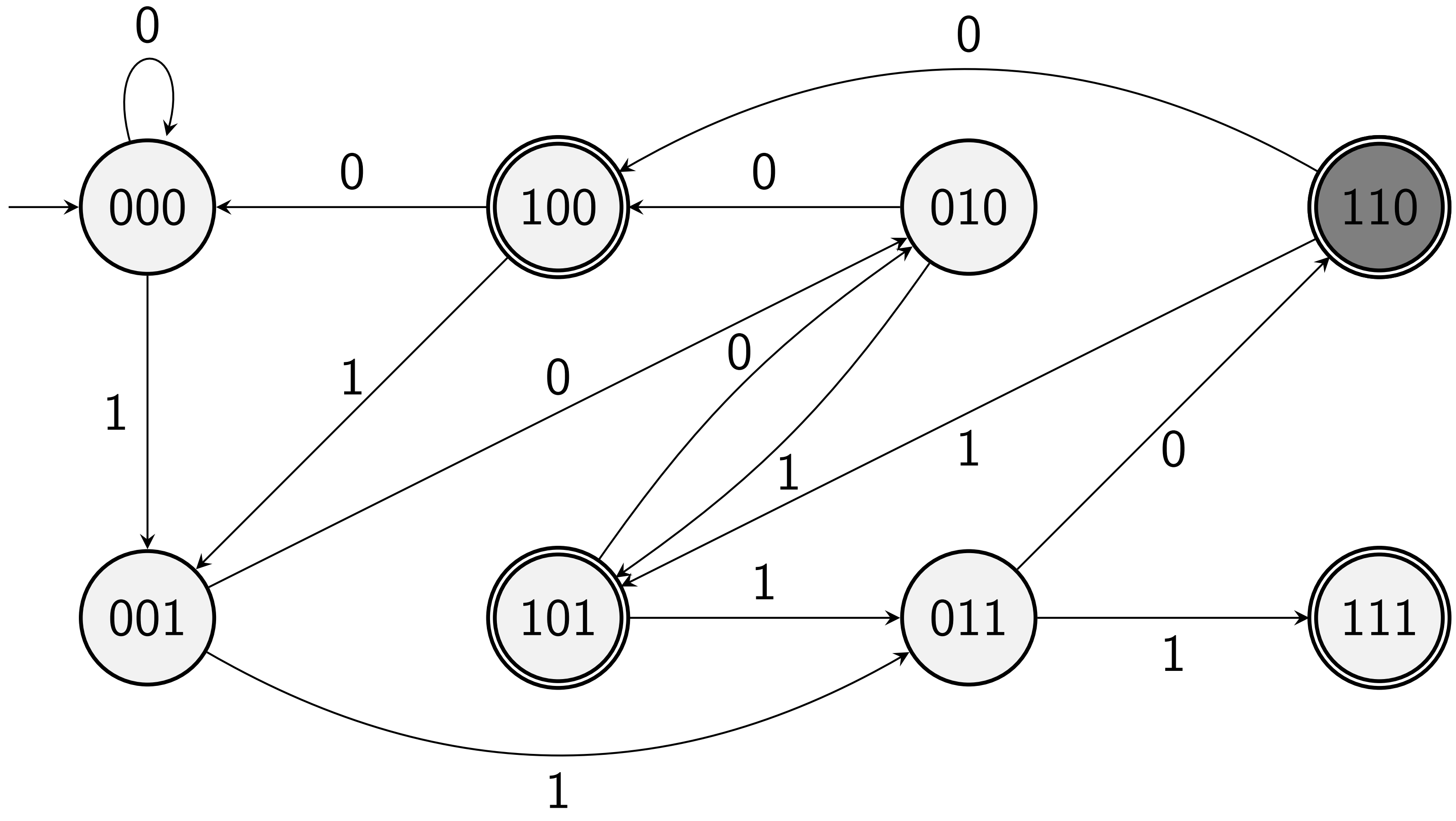
Example



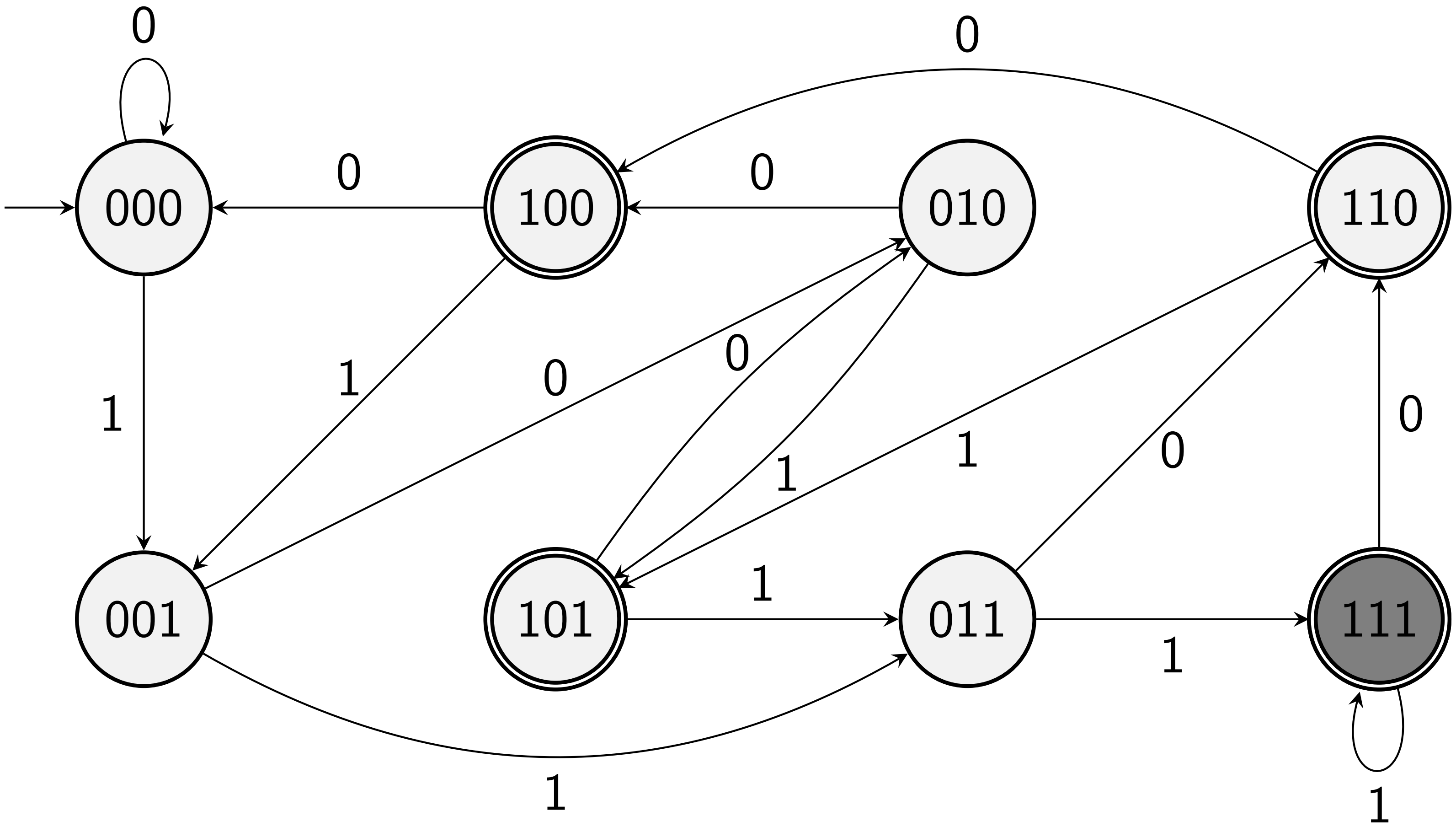
Example



Example



Example



Example

$$L = \{w \text{ over } \{0,1\} : \text{the third last symbol of } w \text{ is } 1\}$$

$$M \text{ with } L(M) = L.$$

Definition

A language L is called **regular**, if there exists a finite automaton M such that

$$L = L(M).$$