



# Finite Automata: Informal

# Computational models

- The theory of computation should begin with the question: **what is a computer?**

# Computational models

- The theory of computation should begin with the question: **what is a computer?**
- Real computers are however quite **complicated** so it is difficult to set up a manageable mathematical theory for them

# Computational models

- The theory of computation should begin with the question: **what is a computer?**
- Real computers are however quite **complicated** so it is difficult to set up a manageable mathematical theory for them
- Instead we can use an idealized computer called **computational model** in order to begin the theory of computation

# Finite automata

- The simplest computational model is called a **finite state machine** or a **finite automaton**

# Finite automata

- The simplest computational model is called a **finite state machine** or a **finite automaton**
- Before developing the mathematics of finite automata we will examine the usage of a concrete finite automaton: **the controller of an automatic door**

# The automatic door

- Automatic doors are often found at supermarket entrances and exits

# The automatic door

- Automatic doors are often found at supermarket entrances and exits
- An automatic door swing open when sensing that a person is approaching

# The automatic door

- Automatic doors are often found at supermarket entrances and exits
- An automatic door swing open when sensing that a person is approaching
- An automatic door is controlled by a simple automaton seen in Figure 1

# Controller of an automatic door

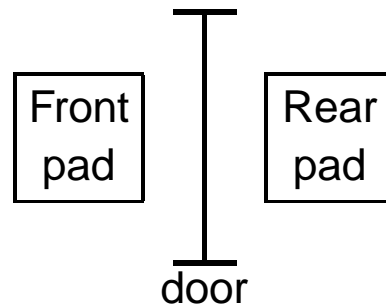


Figure 1: Controller of an automatic door

# Behavior

- An automatic door has a pad in front to detect the presence of a person about to walk through the door

# Behavior

- An automatic door **has a pad in front** to detect the **presence of a person** about to walk through the door
- **Another pad** is **located to the rear** of the doorway so that the controller can **hold the door open** long enough for the person to pass all the way through

# Behavior

- An automatic door **has a pad in front** to detect the **presence of a person** about to walk through the door
- **Another pad** is **located to the rear** of the doorway so that the controller can **hold the door open** long enough for the person to pass all the way through
- **The rear pad** also take care that the **door does not strike** someone standing behind it as it opens

# States of the controller

- The controller is in either of **two states**:

# States of the controller

- The controller is in either of **two states**:  
**open Or closed**

# States of the controller

- The controller is in either of **two states**:  
**open** Or **closed**
- There are **four input conditions**:

# States of the controller

- The controller is in either of **two states**:  
**open** Or **closed**
- There are **four input conditions**:  
**front**, meaning that a person is standing on the front pad

# States of the controller

- The controller is in either of **two states**:  
**open** Or **closed**
- There are **four input conditions**:  
**front**, meaning that a person is standing on the front pad  
**rear**, meaning that a person is standing on the rear pad

# States of the controller

- The controller is in either of **two states**:  
**open** Or **closed**
- There are **four input conditions**:
  - front**, meaning that a person is standing on the front pad
  - rear**, meaning that a person is standing on the rear pad
  - both**, meaning that people are standing on both pads

# States of the controller

- The controller is in either of **two states**:  
**open** Or **closed**
- There are **four input conditions**:
  - front**, meaning that a person is standing on the front pad
  - rear**, meaning that a person is standing on the rear pad
  - both**, meaning that people are standing on both pads
  - neither**, meaning that no one is standing on either pad

# State transition diagram

The state transition diagram of the controller, Figure 2, depicts the movements of the controller depending upon the input it receives:

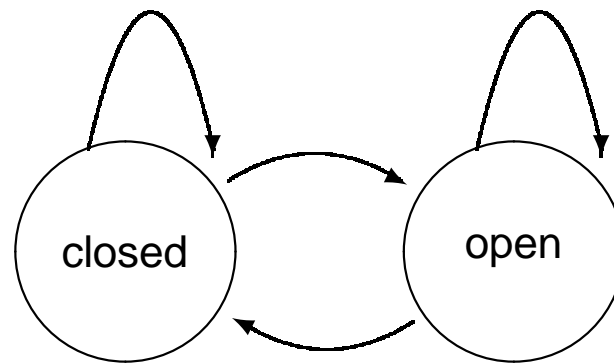


Figure 2: Controller's state transition diagram

# State transition diagram

The state transition diagram of the controller, Figure 2, depicts the movements of the controller depending upon the input it receives:

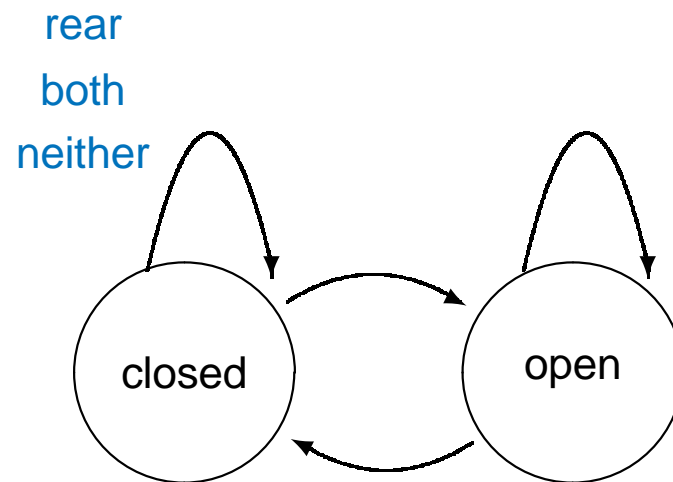


Figure 2: Controller's state transition diagram

# State transition diagram

The state transition diagram of the controller, Figure 2, depicts the movements of the controller depending upon the input it receives:

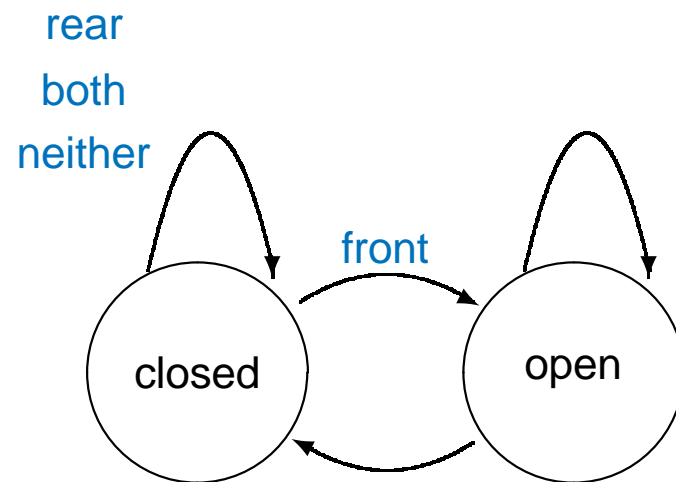


Figure 2: Controller's state transition diagram

# State transition diagram

The state transition diagram of the controller, Figure 2, depicts the movements of the controller depending upon the input it receives:

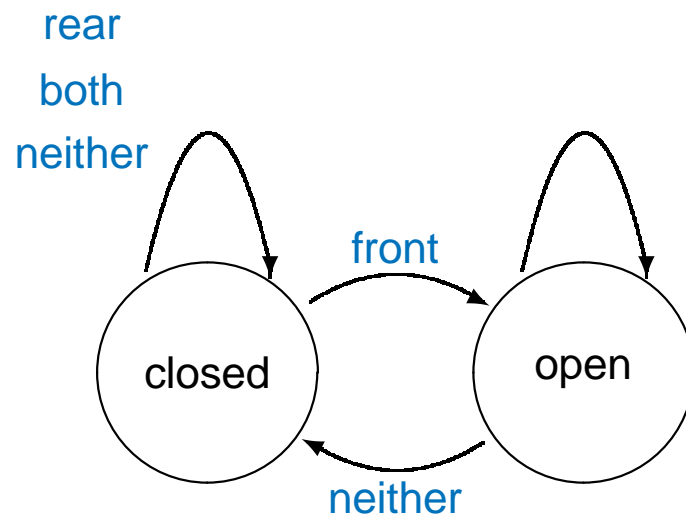


Figure 2: Controller's state transition diagram

# State transition diagram

The state transition diagram of the controller, Figure 2, depicts the movements of the controller depending upon the input it receives:

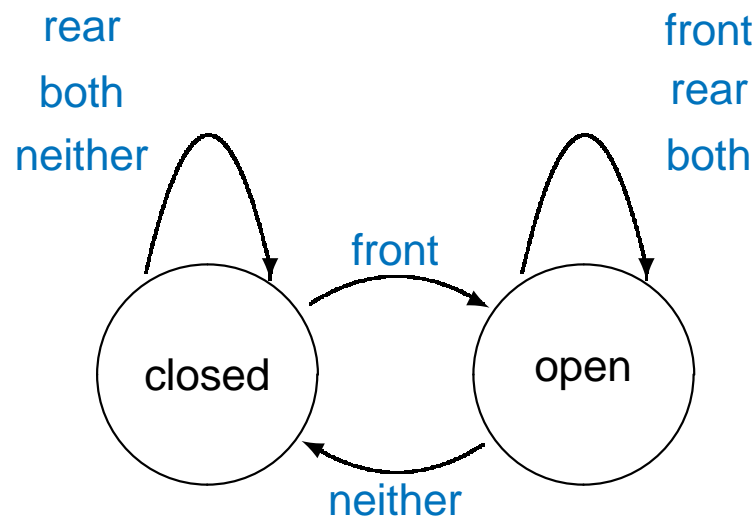


Figure 2: Controller's state transition diagram

# Interpretation

## Door movement:

- When the door is **closed** and there is somebody on **the rear pad** or on **both pads** or there is **no one on the pads**, the door remains **closed**

# Interpretation

## Door movement:

- When the door is **closed** and there is somebody on **the rear pad** or on **both pads** or there is **no one on the pads**, the door remains **closed**
- When the door is **closed** and somebody steps on the **front pad** the door **opens**

# Interpretation

## Door movement:

- When the door is **closed** and there is somebody on **the rear pad** or on **both pads** or there is **no one on the pads**, the door remains **closed**
- When the door is **closed** and somebody steps on the **front pad** the door **opens**
- When the door is **open** and somebody is on the **front pad**, **rear pad**, or on **both pads** the door **stays open**

# Interpretation

## Door movement:

- When the door is **closed** and there is somebody on **the rear pad** or on **both pads** or there is **no one on the pads**, the door remains **closed**
- When the door is **closed** and somebody steps on the **front pad** the door **opens**
- When the door is **open** and somebody is on the **front pad**, **rear pad**, or on **both pads** the door **stays open**
- When the door is **open** and **nobody is on the pads** the door **closes**

# Interpretation

## Controller movement:

- When controller is in the state **closed** and the input is **rear, both,** or **neither** the controller remains in the state **closed**

# Interpretation

## Controller movement:

- When controller is in the state **closed** and the input is **rear, both,** or **neither** the controller remains in the state **closed**
- When controller is in the state **closed** and the input is **front** the controller moves to the state **open**

# Interpretation

## Controller movement:

- When controller is in the state **closed** and the input is **rear, both,** or **neither** the controller remains in the state **closed**
- When controller is in the state **closed** and the input is **front** the controller moves to the state **open**
- When the controller is in the state **open** and the input is one of **front, rear, both,** the controller remains in the state **open**

# Interpretation

## Controller movement:

- When controller is in the state **closed** and the input is **rear, both,** or **neither** the controller remains in the state **closed**
- When controller is in the state **closed** and the input is **front** the controller moves to the state **open**
- When the controller is in the state **open** and the input is one of **front, rear, both,** the controller remains in the state **open**
- When the controller is in the state **open** and the input is **neither** the controller moves to the state **closed**

# Tabular representation

The movement of the controller (and of the door) can also be represented by a table whose lines are labeled by

# Tabular representation

The movement of the controller (and of the door) can also be represented by a table whose lines are labeled by the states

# Tabular representation

The movement of the controller (and of the door) can also be represented by a table whose lines are labeled by the states and whose columns are labeled by

# Tabular representation

The movement of the controller (and of the door) can also be represented by a table whose lines are labeled by the states and whose columns are labeled by the input, as seen in Table 1

# Tabular representation

The movement of the controller (and of the door) can also be represented by a table whose lines are labeled by the states and whose columns are labeled by the input, as seen in Table 1

Table 1: Controller's tabular representation

State/Input	neither	front	rear	both
closed	closed	open	closed	closed
open	closed	open	open	open

Interpretation:  $T(\text{state}, \text{input}) = \text{NewState}$

# Note

- This controller is a computer that has just a single bit of memory that is used to record the controller's state

# Note

- This controller is a computer that has just a single bit of memory that is used to record the controller's state
- Other similar devices need controllers with larger memory.

# Note

- This controller is a computer that has just a single bit of memory that is used to record the controller's state
- Other similar devices need controllers with larger memory.
  - A state of the controller of an elevator may represent the floor the elevator is on and the inputs may be signals received from other floors

# Note

- This controller is a computer that has just a single bit of memory that is used to record the controller's state
- Other similar devices need controllers with larger memory.
  - A state of the controller of an elevator may represent the floor the elevator is on and the inputs may be signals received from other floors
  - Controllers of various household appliances may be more complex.

# Note

- This controller is a computer that has just a single bit of memory that is used to record the controller's state
- Other similar devices need controllers with larger memory.
  - A state of the controller of an elevator may represent the floor the elevator is on and the inputs may be signals received from other floors
  - Controllers of various household appliances may be more complex.
- However the methodology is the same: they are all finite automata

# Other applications

- Finite automata and their probabilistic counterparts, Markov chains, are useful tools for pattern recognition used in speech processing and optical character recognition

# Other applications

- Finite automata and their probabilistic counterparts, Markov chains, are useful tools for pattern recognition used in speech processing and optical character recognition
- Markov chains have been used to model and predict price changes in financial applications

# Generalizing the controller

- All controllers of the applications mentioned above have a common property: they are finite automata

# Generalizing the controller

- All controllers of the applications mentioned above have a common property: they are finite automata
- The mathematical theory of finite automata must be done in abstract, without reference to any particular application

# Generalizing the controller

- All controllers of the applications mentioned above have a common property: they are finite automata
- The mathematical theory of finite automata must be done in abstract, without reference to any particular application
- Hence, the concept of a finite automaton, Figure 3, without reference to application must be developed

# The concept

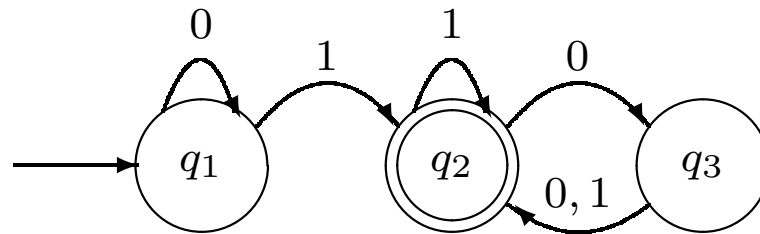


Figure 3: The finite automaton  $M_1$

# The concept

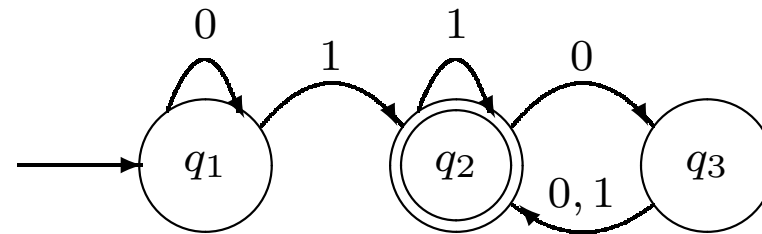


Figure 3: The finite automaton  $M_1$

Finite automaton  $M_1$  has:

# The concept

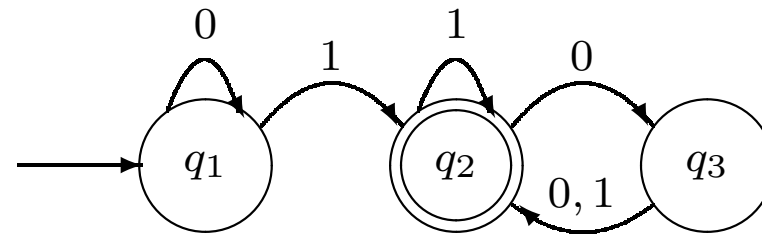


Figure 3: The finite automaton  $M_1$

Finite automaton  $M_1$  has:

- A **finite set of states**  $Q = \{q_1, q_2, q_3\}$

# The concept

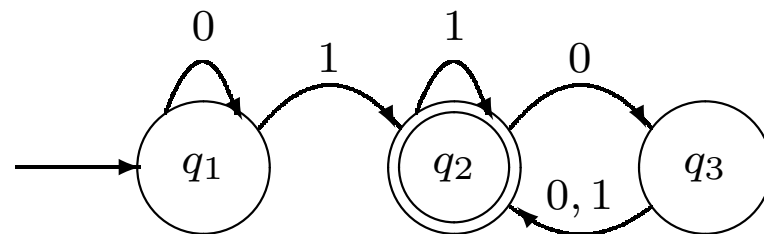


Figure 3: The finite automaton  $M_1$

Finite automaton  $M_1$  has:

- A **finite set of states**  $Q = \{q_1, q_2, q_3\}$
- A **finite input alphabet**  $\Sigma = \{0, 1\}$

# The concept

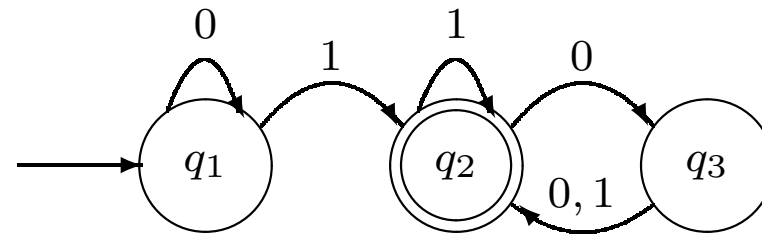


Figure 3: The finite automaton  $M_1$

Finite automaton  $M_1$  has:

- A **finite set of states**  $Q = \{q_1, q_2, q_3\}$
- A **finite input alphabet**  $\Sigma = \{0, 1\}$
- A **transition function**  $\delta : Q \times \Sigma \rightarrow Q$

# The concept

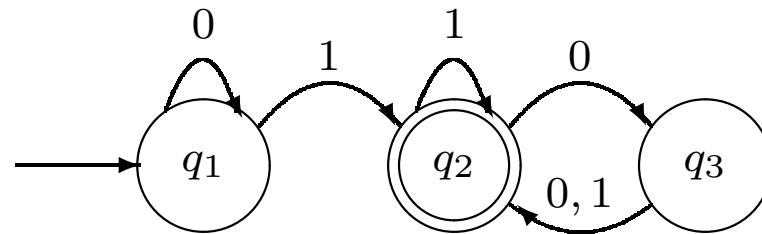


Figure 3: The finite automaton  $M_1$

Finite automaton  $M_1$  has:

- A **finite set of states**  $Q = \{q_1, q_2, q_3\}$
- A **finite input alphabet**  $\Sigma = \{0, 1\}$
- A **transition function**  $\delta : Q \times \Sigma \rightarrow Q$
- A **start state** and an **accept state**

# Note

- Figure 3 is the **state diagram** of the automaton  $M_1$

# Note

- Figure 3 is the **state diagram** of the automaton  $M_1$
- The **start state**,  $q_1$ , is indicated by an arrow pointing to it from nowhere

# Note

- Figure 3 is the **state diagram** of the automaton  $M_1$
- The **start state**,  $q_1$ , is indicated by an arrow pointing to it from nowhere
- The **accept state**,  $q_2$ , is indicated by a double circle

# Note

- Figure 3 is the **state diagram** of the automaton  $M_1$
- The **start state**,  $q_1$ , is indicated by an arrow pointing to it from nowhere
- The **accept state**,  $q_2$ , is indicated by a double circle
- The arrows going from one state to another are called **transitions**

# How does it work?

- When the automaton receives an input string, such as  $1101$ , it processes that string and produces an output which is either **accept** or **reject**

# How does it work?

- When the automaton receives an input string, such as  $1101$ , it processes that string and produces an output which is either **accept** or **reject**
- Processing begins in  $M_1$ 's start state

# How does it work?

- When the automaton receives an input string, such as  $1101$ , it processes that string and produces an output which is either **accept** or **reject**
- Processing begins in  $M_1$ 's start state

# How does it work?

- When the automaton receives an input string, such as  $1101$ , it processes that string and produces an output which is either **accept** or **reject**
- Processing begins in  $M_1$ 's start state

# Processing procedure

- The automaton receives the input symbols one by one from left to right

# Processing procedure

- The automaton receives the input symbols one by one from left to right
- After reading each symbol,  $M_1$  moves from one state to another along the transition that has that symbol as its label

# Processing procedure

- The automaton receives the input symbols one by one from left to right
- After reading each symbol,  $M_1$  moves from one state to another along the transition that has that symbol as its label
- When  $M_1$  reads the last symbol of the input it produces the output: **accept** if  $M_1$  is in an accept state, or **reject** if  $M_1$  is not in an accept state

# Processing procedure

- The automaton receives the input symbols one by one from left to right
- After reading each symbol,  $M_1$  moves from one state to another along the transition that has that symbol as its label
- When  $M_1$  reads the last symbol of the input it produces the output: **accept** if  $M_1$  is in an accept state, or **reject** if  $M_1$  is not in an accept state

# Example processing

Examine the work produced by  $M_1$ , Figure 3, on the input 1101:

1.  $M_1$  starts in state  $q_1$ ;
2.  $M_1$  reads 1 and follows transition from  $q_1$  to  $q_2$ ;
3. In state  $q_2$   $M_1$  reads next symbol 1 and follows transition from  $q_2$  to  $q_2$ ;
4. Then  $M_1$  reads next symbol, 0, and follows transition from  $q_2$  to  $q_3$
5. In state  $q_3$   $M_1$  reads 1 and follows transition to  $q_3$  to  $q_2$
6. In state  $q_2$  the input was consumed,  $q_2$  is an accept state and  $M_1$  outputs **accept**

# Example processing

Examine the work produced by  $M_1$ , Figure 3, on the input 1101:

1.  $M_1$  starts in state  $q_1$ ;
2.  $M_1$  reads 1 and follows transition from  $q_1$  to  $q_2$ ;
3. In state  $q_2$   $M_1$  reads next symbol 1 and follows transition from  $q_2$  to  $q_2$ ;
4. Then  $M_1$  reads next symbol, 0, and follows transition from  $q_2$  to  $q_3$
5. In state  $q_3$   $M_1$  reads 1 and follows transition to  $q_3$  to  $q_2$
6. In state  $q_2$  the input was consumed,  $q_2$  is an accept state and  $M_1$  outputs **accept**

# Example processing

Examine the work produced by  $M_1$ , Figure 3, on the input 1101:

1.  $M_1$  starts in state  $q_1$ ;
2.  $M_1$  reads 1 and follows transition from  $q_1$  to  $q_2$ ;
3. In state  $q_2$   $M_1$  reads next symbol 1 and follows transition from  $q_2$  to  $q_2$ ;
4. Then  $M_1$  reads next symbol, 0, and follows transition from  $q_2$  to  $q_3$
5. In state  $q_3$   $M_1$  reads 1 and follows transition to  $q_3$  to  $q_2$
6. In state  $q_2$  the input was consumed,  $q_2$  is an accept state and  $M_1$  outputs **accept**

# Example processing

Examine the work produced by  $M_1$ , Figure 3, on the input 1101:

1.  $M_1$  starts in state  $q_1$ ;
2.  $M_1$  reads 1 and follows transition from  $q_1$  to  $q_2$ ;
3. In state  $q_2$   $M_1$  reads next symbol 1 and follows transition from  $q_2$  to  $q_2$ ;
4. Then  $M_1$  reads next symbol, 0, and follows transition from  $q_2$  to  $q_3$
5. In state  $q_3$   $M_1$  reads 1 and follows transition to  $q_3$  to  $q_2$
6. In state  $q_2$  the input was consumed,  $q_2$  is an accept state and  $M_1$  outputs **accept**

# Note

- $M_1$  accepts strings that end with  $1$ ; Why?
- $M_1$  accepts strings that end with an even number of  $0$ -s following the last symbol  $1$ ; Why?
- $M_1$  rejects all other strings

Q: can you describe the language consisting from all strings accepted by  $M_1$ ?

# Note

- $M_1$  accepts strings that end with  $1$ ; Why?
- $M_1$  accepts strings that end with an even number of  $0$ -s following the last symbol  $1$ ; Why?
- $M_1$  rejects all other strings

Q: can you describe the language consisting from all strings accepted by  $M_1$ ?

# Note

- $M_1$  accepts strings that end with  $1$ ; Why?
- $M_1$  accepts strings that end with an even number of  $0$ -s following the last symbol  $1$ ; Why?
- $M_1$  rejects all other strings

Q: can you describe the language consisting from all strings accepted by  $M_1$ ?

# Note

- $M_1$  accepts strings that end with  $1$ ; Why?
- $M_1$  accepts strings that end with an even number of  $0$ -s following the last symbol  $1$ ; Why?
- $M_1$  rejects all other strings

Q: can you describe the language consisting from all strings accepted by  $M_1$ ?