



Finite Automata: Informal

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science

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 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**.
- 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;
- 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

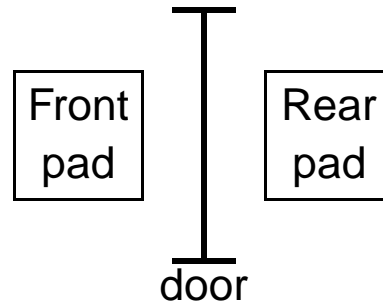


Figure 1: Controller of an automatic door

Behavior

- An automatic door has a FrontPad that detects the presence of a person about to walk through the door;
- An automatic door has a RearPad that holds the door open long enough for the person to pass all the way through;
- An automatic door has a controller that handles FrontPad and RearPad.

Note: the RearPad must ensure 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:
open **Or** closed
- There are four input conditions:
front, meaning that a person is standing on the FrontPad;
rear, meaning that a person is standing on the RearPad;
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:

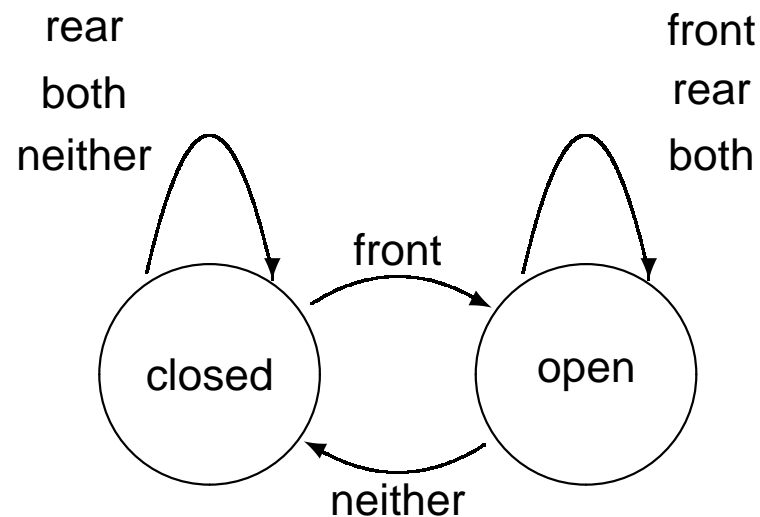


Figure 2: Controller's state transition diagram

Interpretation

Door movement:

1. When the door is closed and there is somebody on the RearPad or on both pads or there is no one on the pads, the door remains closed;
2. When the door is closed and somebody steps on the FrontPad the door opens;
3. When the door is open and somebody is on the FrontPad, RearPad, or on both pads the door stays open;
4. When the door is open and nobody is on the pads the door closes.

Interpretation

Controller movement:

1. When controller is in the state `closed` and the input is `rear`, `both`, or `neither` the controller remains in the state `closed`
2. When controller is in the state `closed` and the input is `front` the controller moves to the state `open`
3. When the controller is in the state `open` and the input is one of `front`, `rear`, `both`, the controller remains in the state `open`
4. 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 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: $T(state, input) = NewState$

Observations

- Controller is a computer that has a single bit of memory used to record its states;
- Other similar devices need larger memory;
Example: the controller of an elevator may have states to 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 chain*, are useful tools for pattern recognition;
These are 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
- 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

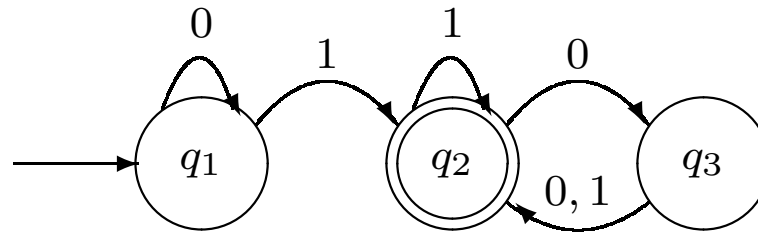


Figure 3: The finite automaton M_1

Note:

- M_1 has a set of states, $Q = \{q_1, q_2, q_3\}$
- M_1 has an input alphabet, $\Sigma = \{0, 1\}$
- M_1 has a transition function $\delta : Q \times \Sigma \rightarrow Q$
- M_1 has a start state and an accept state

Observations

1. Figure 3 is the **state diagram** of the automaton M_1 ;
2. The *start state*, q_1 , is indicated by an arrow pointing to it from nowhere;
3. The *accept state*, q_2 , is indicated by a double circle;
4. 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;
- The output is either *accept* or *reject*;
- Processing begins in M_1 's start state.

Processing procedure

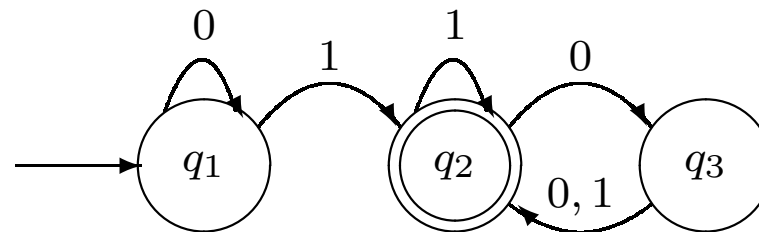
1. The automaton receives the input symbols one by one from left to right;
2. After reading each symbol, M_1 moves from one state to another along the transition that has that symbol as its label;
3. When M_1 reads the last symbol of the input it produces the output:
accept if M_1 is in an accept state, 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*.

Observation



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

The language of M_1

Can you describe the language consisting from all strings accepted by M_1 ?