



Formal Definition of a Finite Automaton

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science

Why a formal definition?

- A formal definition is precise.

Note: a formal definition resolves any uncertainties about what is allowed in a finite automaton such as the number of accept states and number of transitions exiting from a state.

- A formal definition provides a notation.

Remember: good notation helps think and express thoughts clearly!

Parts of a finite automaton

- A set of states;
- Rules for going from one state to another depending upon the input symbol;
- An input alphabet that indicates the allowed symbols;
- It has a start state;
- It has a set of accept states.

Note: a formal definition will state that a finite automaton is a list of this five components!

Observation

- In mathematical language a list of five elements is called a 5-tuple, hence a finite automaton can be defined as a 5-tuple.
- We can denote the transition rules by a function called the *transition function*,
 $\delta : States \times Alphabet \rightarrow States$
Example: $\delta(q_0, x) = q_1$

Formal definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set called the *set of states*;
2. Σ is a finite set called the *alphabet*;
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function;
4. $q_0 \in Q$ is the *start state* also called *initial state*;
5. $F \subseteq Q$ is the set of *accept states*, also called the *final states*.

Observations

- Since the set F can be empty set \emptyset a finite automaton may have zero accept states.
- Since transitions are described by a function, the function δ specifies exactly one next state for each possible combination of state and input symbol.

Example finite automaton

The automaton M_1 have been defined by the transition diagram in Figure 1

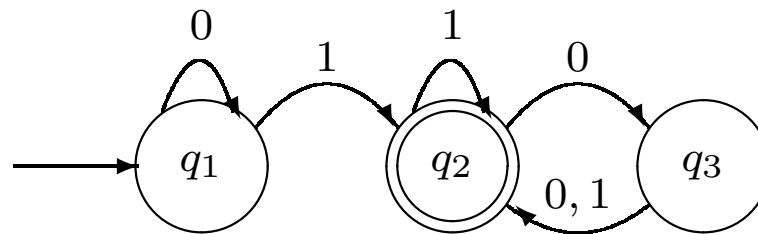


Figure 1: The finite automaton M_1

Formalizing M_1

$M_1 = (Q, \Sigma, \delta, q_1, F)$ where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ is described by the table:

| δ | 0 | 1 |
|----------|-------|-------|
| q_1 | q_1 | q_2 |
| q_2 | q_3 | q_2 |
| q_3 | q_2 | q_2 |

4. q_1 is the start state, and
5. $F = \{q_2\}$.

Language of a machine

Since a finite automaton is used here as the model of a computer we also refer to a finite automaton as a “machine”.

- If A is the set of all strings that a machine M accepts, we say that A is the *language of the machine* M and write $L(M) = A$.

Terminology

The term *accept* has a different meaning when we refer to machines accepting strings and machines accepting languages.

Hence, in order to avoid confusion:

- use the term *accept* when we refer to strings;
- use the term *recognize* when we refer to languages.

Consequences

- A machine may accept several strings, but it always recognizes only one language.
- If a machine accepts no strings, it still recognizes one language, namely the empty language \emptyset .
- Language recognized by machine M_1 is:

$A = \{w \mid w \text{ contains at least one } 1 \text{ and an even number of } 0\text{s follow the last } 1\}$

Conclusion:

$L(M_1) = A$, or equivalently, M_1 recognizes A .

Machine M_2

The state diagram in Figure 2 describes a machine M_2

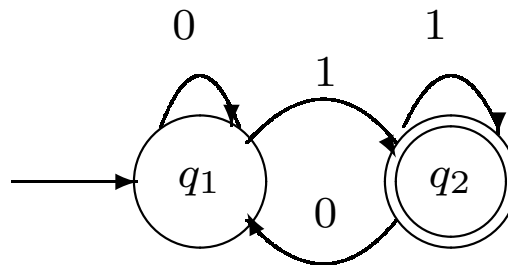


Figure 2: State diagram of the finite automaton M_2

Formally, $M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$ where
 $\delta(q_1, 0) = q_1, \delta(q_1, 1) = q_2, \delta(q_2, 0) = q_1, \delta(q_2, 1) = q_2$

Observations

- State diagram of M_2 and its formal description contain the same information, in different form;
- A good way of understanding any machine is to try it on some sample input string;
- Use this method to discover the language of M_2 , $L(M_2) = \{w \mid w \text{ ends in a } 1\}$.

Another example

Consider the finite automaton M_3 in Figure 3

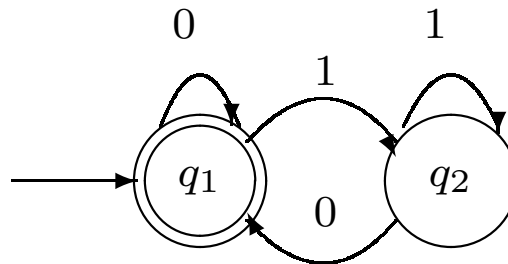


Figure 3: State diagram of the finite automaton M_3

Note: M_3 is similar to M_2 , except for the location of the accept state.

Observations

- As usual, M_3 accepts all strings that leave it in an accept state when it has consumed the input.
- Because M_3 's start state is also accept state, M_3 accepts the empty string ϵ .
Note: as soon as M_3 begins the reading of ϵ it is at the end, so it accepts it!
- In addition to ϵ , M_3 accepts any string that ends in 0.

$$L(M_3) = \{w \mid w \text{ is the empty string } \epsilon \text{ or ends in } 0\}$$

Example 1.4

Consider the five-state machine M_4 , Figure 4

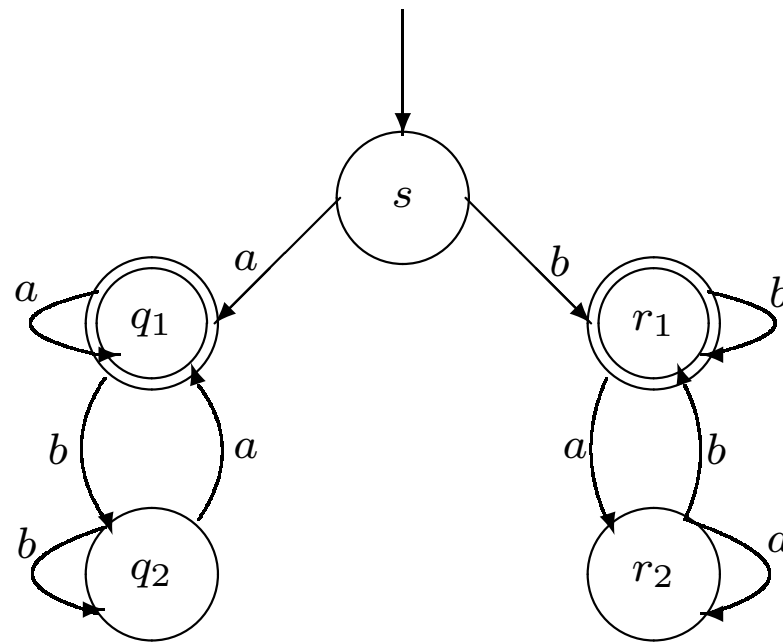


Figure 4: Finite automaton M_4

Observations

- M_4 has two accept states, q_1 and r_1 ;
- M_4 operates over the alphabet $\Sigma = \{a, b\}$;
- Some experimentation with M_4 shows that it accepts strings as a, b, aa, bb, bab and does not accept strings as $ab, ba, bbba$.
- M_4 begins in state s and after it reads the first symbol in the input it either goes to the left to q states or to the right to r states.

More observations

- Once M_4 goes to the left or to the right it can never return to the start state;
- If the first symbol in the string is a then it goes to the left and accept when the strings ends with an a ;
- If the first symbol in the string is b then it goes to the right and accept when the strings ends with a b ;

Conclusion:

$$L(M_4) = \{w|w = axa \vee w = a\} \cup \{w|w = byb \vee w = b\} \text{ for } x, y \in \Sigma^*$$

Example 1.5

The state diagram in Figure 5 shows the machine M_5 which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$ where $\langle RESET \rangle$ is treated as a single symbol.

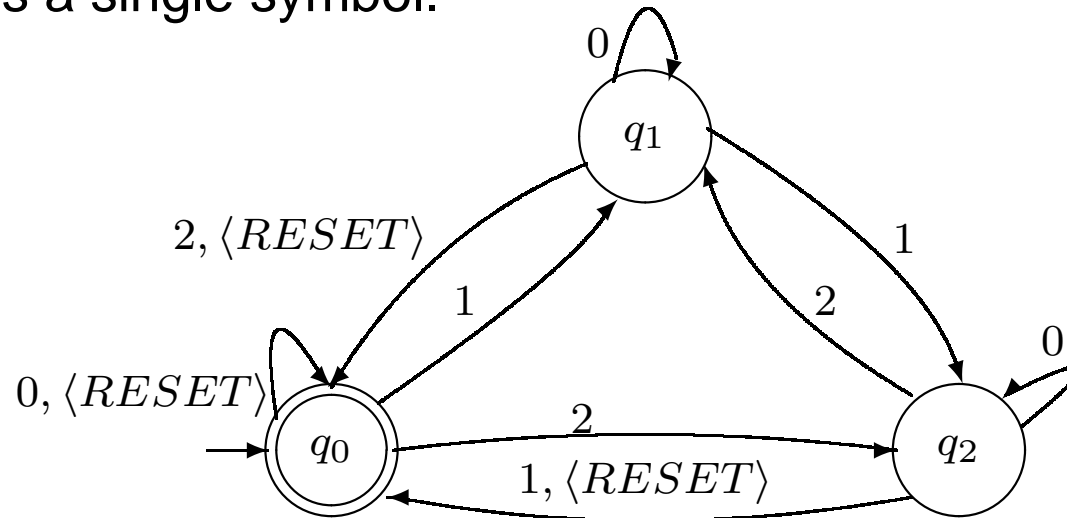


Figure 5: Finite automaton M_5

Running M_5

- M_5 keeps a running count of the sum of the numerical input symbols it reads, modulo 3.
- The three states of the automaton correspond to the three numbers 0,1,2, the sum could ever be.
- Every time the automaton receives the $\langle RESET \rangle$ symbol it resets the count to 0 while moving to state q_0 .
- M_5 accepts if the sum is 0 modulo 3, i.e., if the sum is a multiple of 3.

Limitation of state diagrams

- It is not always possible to describe a finite automaton using a state diagram.
- That may occur when the diagram would be too big to draw or if the description depends on some unspecified parameter.
- In these cases one needs to resort to a formal description to specify the machine.

Example 1.6

Consider a generalization of the Example 1.5 using the same four symbol alphabet Σ :

- For each $i \geq 0$ let A_i be the language of all strings where the sum of the numbers making up the input is a multiple of i , except that the sum is reset to 0 whenever a symbol $\langle RESET \rangle$ appears.
- For each A_i we construct a finite automaton B_i recognizing A_i .

Note: this machine depends on an unspecified parameter, A_i .

The automaton B_i

The machine B_i is described formally as follows:

$B_i = (Q_i, \Sigma, \delta_i, q_0, \{q_0\})$ where:

- $Q_i = \{q_0, q_1, \dots, q_{i-1}\}$
- $\Sigma = \{0, 1, 2, \langle RESET \rangle\}$
- δ_i is designed so that for each j , $0 \leq j \leq i - 1$, if B_i is in the state q_j then the running sum is j modulo i . For each q_j we set:

$$\delta_i(q_j, 0) = q_j$$

$$\delta_i(q_j, 1) = q_k, k = j + 1 \text{ modulo } i$$

$$\delta_i(q_j, 2) = q_k, k = j + 2 \text{ modulo } i$$

$$\delta_i(q_j, \langle RESET \rangle) = q_0$$