CS605
5 February 2025

1.1

FINITE AUTOMATA

Finite automata are good models for computers with an extremely limited amount of memory. What can a computer do with such a small memory? Many useful things! In fact, we interact with such computers all the time, as they lie at the heart of various electromechanical devices.

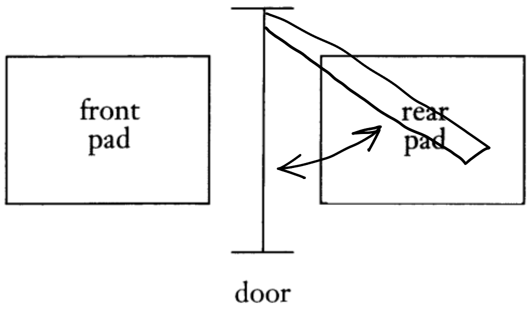
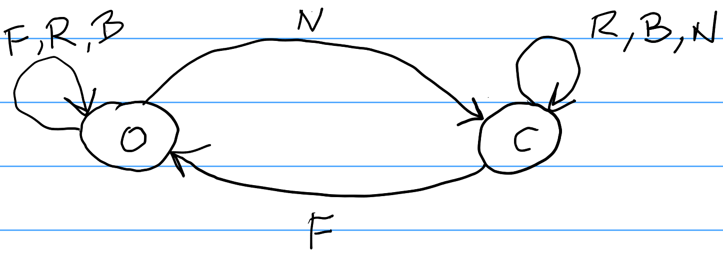


FIGURE 1.1
Top view of an automatic door



DEFINITION 1.5

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

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[2]

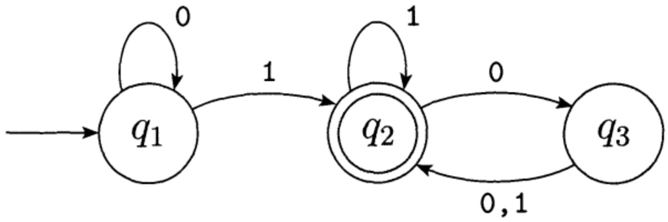


FIGURE 1.6
The finite automaton M_1

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

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

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

$\delta =$

$Q \backslash \Sigma$	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

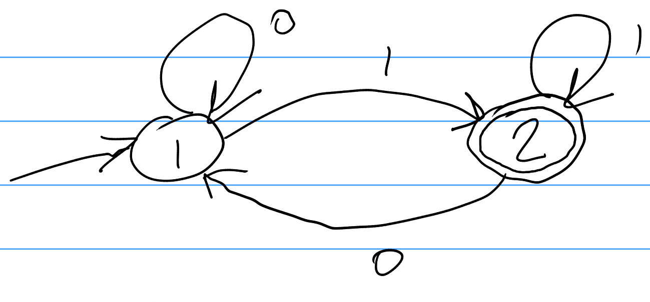
$q_0 = q_1$

$F = \{q_2\}$

$M = (\{q_1, q_2, q_3\}, \{0,1\}, \{((q_1, 0), q_1), ((q_1, 1), q_2),$

$\ldots\ldots\ldots \}, q_1, \{q_2\})$

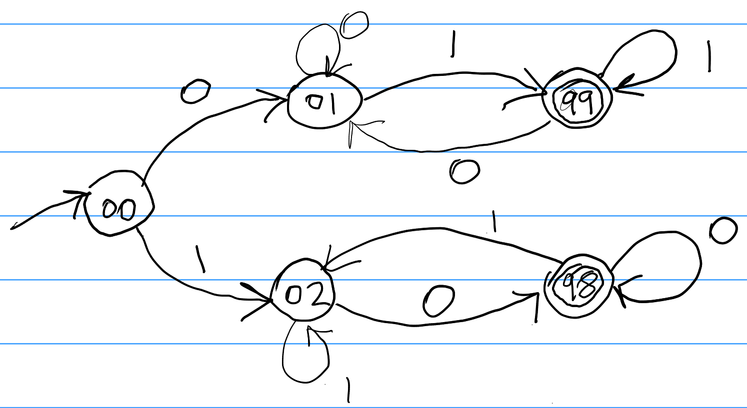
If A is the set of all strings that machine M accepts, we say that A is the ***language of machine M*** and write $L(M) = A$. We say that ***M recognizes A*** or that ***M accepts A***. Because the term *accept* has different meanings when we refer to machines accepting strings and machines accepting languages, we prefer the term *recognize* for languages in order to avoid confusion.



$M = (\{1,2\}, \{0,1\}, \{((1,0),1), ((1,1),2), ((2,1),2), ((2,0),1)\}, 1, \{2\})$

$L_{BD} = \{w : w \in \{0,1\}^*$, w begins and ends with a different symbol, i.e. either begins with 0 and ends with 1, or begins with 1 and ends with 0.$\}$

E.g. $e \notin L$ $\quad 0 \notin L$ $\quad 010 \notin L$ $\quad 01 \in L$ $\quad 011 \in L$



FORMAL DEFINITION OF COMPUTATION

So far we have described finite automata informally, using state diagrams, and with a formal definition, as a 5-tuple. The informal description is easier to grasp at first, but the formal definition is useful for making the notion precise, resolving any ambiguities that may have occurred in the informal description. Next we do the same for a finite automaton's computation. We already have an informal idea of the way it computes, and we now formalize it mathematically.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,

2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and

3. $r_n \in F$.

Condition 1 says that the machine starts in the start state. Condition 2 says that the machine goes from state to state according to the transition function. Condition 3 says that the machine accepts its input if it ends up in an accept state. We say that M ***recognizes language*** A if $A = \{w \mid M \text{ accepts } w\}$.

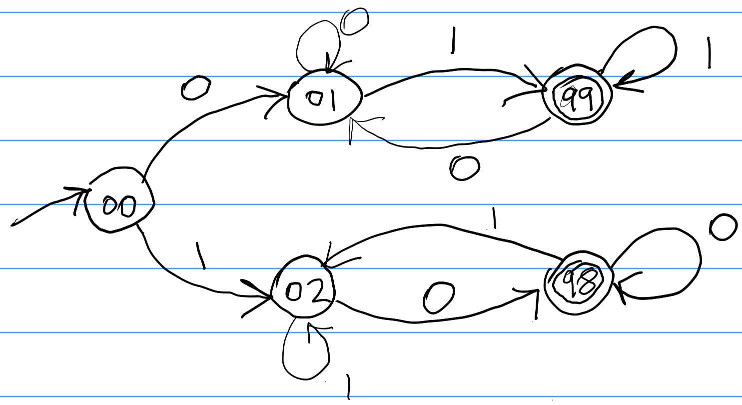
DEFINITION 1.16

A language is called a ***regular language*** if some finite automaton recognizes it.

Prove that L_{BD} (from above) is regular or prove it is not regular.

Proof We will prove L_{BD} is regular by constructing a FA M to recognise it.

M =



M is a FA that recognises L_{BD} therefore this proves that L_{BD} is regular.

DEFINITION 1.23

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows.

- **Union**: $A \cup B = \{x | \, x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy | \, x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \ldots x_k | \, k \geq 0 \text{ and each } x_i \in A\}$.

$L_1 = \{w : w \in \{0,1\}^*, \ w \text{ begins and ends with } 0\}$

$L_2 = \{w : w \in \{0,1\}^*, \ w \text{ begins and ends with } 1\}$

$L_1 \cup L_2$ is regular.

$L_1 = \{w : w \in \{0,1\}^* \ w \text{ begins with } 1\}$

$L_2 = \{e\}$

$L_1 \cup L_2$ is regular

Tomorrow:

Prove that the complement of a regular language is regular

All finite languages are regular

Languages to program using MU Machines on Wednesday afternoon 15:00-17:00

W1 = $\{w : w \in \{0,1\}^*$, w is empty or ends with 0$\}$

W2 = $\{w : w \in \{0,1\}^*$, w has an odd number of 1s$\}$

W3 = $\{w : w \in \{0,1\}^*$, w contains the substring 001$\}$

w4 = $\{w : w \in \{a,b\}^*$, w starts and ends with a or starts and ends with b$\}$

W5 = $\{w : w \in \{a,b\}^*$, w does not contain the substring ab$\}$

W6 = $\{w : w \in \{a,b\}^*$, $|w|$ is even$\}$

W7 = $\{w : w \in \{a,b\}^*, |w| > 0\}$

W8 = $\{w : w \in \{a,b\}^*$, w has at least two bs$\}$

W9 = $\{w : w \in \{a,b\}^*$, each a is followed by at least one b$\}$

W10 = $\{w : w \in \{0,1\}^*$, w begins with 1 and ends with 0$\}$

W11 = $\{w : w \in \{a,b\}^*$, w does not contain the substring baba$\}$