Week 1: Lecture Notes

Finite State Systems

- State : Summarizes the information concerning past inputs that is needed to determine the behaviour of the systems on subsequent inputs.

Example:

- Elevator (Control Mechanism)
  - current floor
  - up / down
  - collection not yet satisfied requests for service

  } states

- Switching circuit, such as the control unit of a computer
  - finite number of gates each of which is either on/off
  - n gates $\Rightarrow$ 2^n assignments of 0 or 1 to various gates.

- Programs such as text editors and the lexical analyzers used in most compilers are often designed as finite state system.

- a lexical analyzer scans the symbol of a computer program to locate strings of characters corresponding to identifiers, numerical constants, reserved words and so on.

- The lexical analyzer needs to remember only a finite amount of information, such as how long a prefix of a reserved word it has seen since startup.

- Computer itself can be viewed as a finite state system.

- Human brain → finite state system.
  - \# of brain cells or neurons is limited. (2^{35} at most)

Finite Automata / Automaton (FA)

- The most basic model of a computer
- Computer without memory / the amount of memory is fixed, regardless of the size of the input.
- involves
  - → states
    - → accept states
    - → reject states
  - → transition among states in response to inputs

- an abstract computing device that recognizes a collection of strings.

  $w = a_1 a_2 \cdots a_n$ → strings

  $$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \longrightarrow \cdots \xrightarrow{a_n} q_n$$

  if accept state (i.e. $q_n \in F$) w is recognized by the FA otherwise FA rejects w

> A finite automata modeling recognition of the string $w = a_1 a_2 \cdots a_n$ may be a part of lexical analyzer.

Scanned by CamScanner

Applications of FA

- useful model for many important kinds of hardware and software.
  - design of lexical analyzer of a typical compiler.

- software for designing and checking the behaviour of digital circuits / protocols.

- software for scanning large bodies of text (e.g. web page) to find occurrences of words, phrases or other patterns (pattern recognition)

- protocols (with finite number of states)
  → communication protocol
  → protocol for secure exchange of information

- lexical analyzer of a compiler:
  the compiler component that breaks the input text into logical units, e.g. identifiers, keywords and punctuations.

- Automata are essential to study the limits of computation:
  → Decidable problems (solvable by computer) (What can a computer do at all?)
  → Tractable problems (solvable by computer efficiently)

(What can a computer do efficiently?) → time complexity is a slowly growing function in the size of input

studies ↓

Intractable / tractable problems

Turing Machines: automata that models the power of real computers.

- allows us to study decidability → the question of what can or cannot be done by a computer
- also allows us to distinguish tractable (solvable in polynomial time) from intractable (not solvable in polynomial time) problems.

Content-free grammars and Push down automata

- useful tool for describing structure of programming languages and design of parser → another key partion of a compiler which deals with recursively nested featurs of the typical programming language (e.g. arithmetic, conditional etc.)

Regular Expressions

- useful for describing same patterns that can be represented by finite automata and design of lexical analyzer (compiler component that groups character into tokens)

Example: Unix-style regular expression

```
[A-Z][a-z]* [ ][A-Z][A-Z]
```

Kolkata WB

```
[A-Z][a-z]*([ ][A-Z][a-z]*)*[ ][A-Z][A-Z]
```

Kolkata West Bengal IN

Deterministic finite Automata (DFA)

A 5-tuple $(Q, \Sigma, \delta, q_0, F)$

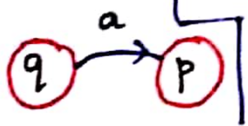
- $\Sigma = \{0,1\}$ binary
- $\Sigma = \{a, b, \dots, z\}$ → lower case letters
- Σ. set of all ASCII characters.

Q : a finite set of states

Σ : a finite input alphabet

δ : $Q \times \Sigma \to Q$, the transition function:

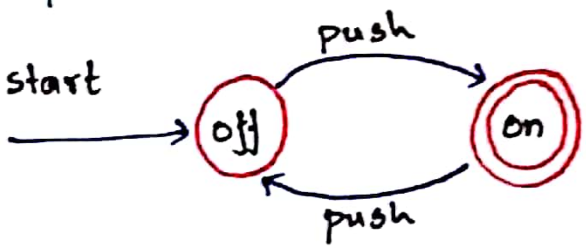
$p = \delta(q, a)$



$q_0 \in Q$: the initial state

$F \subseteq Q$: the set of final/accepting states

Example: (A finite automata modeling an on/off switch)



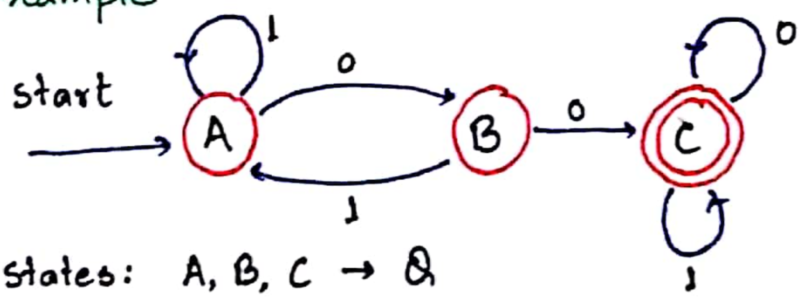
(Transition diagram)

- states: on, off → Q
- input : push → Σ
- accepting state: on → F
- initial state : off → q_0

δ	push
→ off	on
* on	off

Transition table

Example



States: A, B, C → Q

input: 0, 1 → Σ

accepting state: C → F

initial state : A → q_0

Transition table

δ	0	1
→ A	B	A
B	C	A
* C	C	C

- Alphabet (Σ)
  - a finite non-empty set of symbols

  e.g. $\Sigma = \{0, 1\}$ → binary alphabet

- Strings / words (w)
  - a finite sequence of symbols

  e.g. 01101 is a string from the binary alphabet $\Sigma = \{0, 1\}$

- Empty string (ε)
  - zero occurrences of symbols

- Length of a string w
  - $|w|$: # of positions of symbols in w

  e.g. $|\varepsilon| = 0$, $|01101| = 5$

- Convention:
  - lower case letters at the begining of the alphabet (or digits) → symbols e.g. a, b, c, ..
  - lower case letters near the end of the alphabet → strings e.g. w, x, y, z

- Concatenation of strings

  $x = a_1 a_2 \ldots a_i$, $y = b_1 b_2 \ldots b_j$

  $xy = a_1 a_2 \ldots a_i b_1 b_2 \ldots b_j$ → string of length $i+j$

For any string w, we have

$$w\varepsilon = \varepsilon w = w$$

Power of an alphabet

- Σ - an alphabet
- Σ^k - set of strings of length k, each of whose symbols is in Σ.

Example:

- $\Sigma = \{a, b, c\}$ - alphabet
- $\Sigma^0 = \varepsilon$ - ε is the only string of length 0
- $\Sigma^1 = \{a, b, c\}$ - strings of length 1.
- $\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
- $\Sigma^3 =$ strings of length 3

- Σ^* - set of all strings over an alphabet Σ

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$
$$= \{\varepsilon\} \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$

$$\Sigma^+ - \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

So,

$$\Sigma^* = \{\varepsilon\} \cup \Sigma^+$$

Languages

- $L \subseteq \Sigma^*$ → a language over Σ

Example:

1. Languages of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$ is

   $\{\varepsilon, 01, 0011, 000111\}$

   → $\{0^n 1^n \mid n \geq 0\}$

2. The set of strings of 0's and 1's with equal number of each $\{\varepsilon, 01, 10, 0011, 0101, 1001, \dots\}$

3. The set of binary numbers whose value is a prime
   - $\{w \mid w$ is a binary integer that is prime$\}$
   - $\{10, 11, 101, 111, \dots\}$

4. Σ^* - a language over Σ

5. ϕ - the empty language (a language over any alphabet)

6. $\{\varepsilon\}$ - a language over any alphabet.

   $\phi \neq \{\varepsilon\}$

   no string — are string of length 0

Language: may contain an infinite number of strings, but strings are drawn from one fixed, finite alphabet.

Problem - Decisional Problems

- Membership in a language
- Σ – an alphabet
- L - language over Σ

Problem: $L \rightarrow$ Given a string w in Σ^*, decide whether or not $w \in L$

Example:

Primality Testing - Given an integer decide whether it is prime or not

Reformulation: Express the problem by the language L_p consisting of all binary strings whose value as a binary number is prime.

$\Rightarrow$ Given a string w of 0's and 1's

output $\rightarrow$ YES if $w \in L_p$

$\rightarrow$ NO if $w \notin L_p$

How a DFA processes strings?

Consider a DFA $A = (Q, \Sigma, \delta, q_0, F)$ and a string $w = a_1 a_2 \dots a_n$

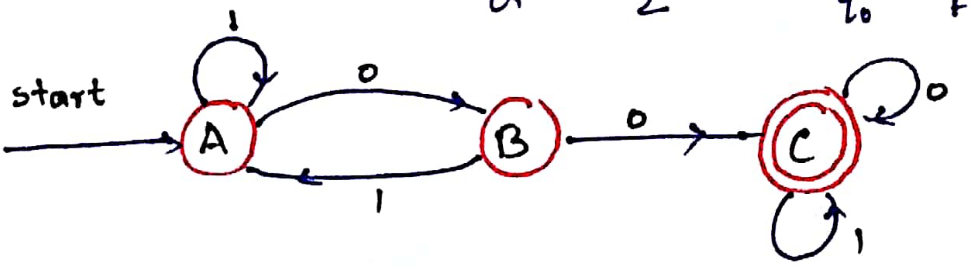
$$q_1 = \delta(q_0, a_1)$$
$$q_2 = \delta(q_1, a_2)$$
$$\vdots$$
$$q_i = \delta(q_{i-1}, a_i)$$
$$\vdots$$
$$q_n = \delta(q_{n-1}, a_n)$$

DFA A accepts the string $w = a_1 a_2 \dots a_n$ if $q_n \in F$

if not, A rejects w.

Example:

Consider DFA : $(\underbrace{\{A, B, C\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, \delta, \underbrace{\{A\}}_{q_0}, \underbrace{\{C\}}_{F})$



Consider string $w = 101001$

current state	symbol read	New state
A	1	A
A	0	B
B	1	A
A	0	B
B	0	C
C	1	C

final state which is the accept state

The above DFA accepts string w

Also, the above DFA

- doesnot accept 11101
- accepts 0001
- accepts all strings of 0's and 1's with two consecutive zeros somewhere.

• Language accepted by the given DFA
  ↳ regular language.

Extending the transition function to strings.

- DFA $\rightarrow (Q, \Sigma, \delta, q_0, F)$
- $\delta : Q \times \Sigma \rightarrow Q$ (transition function)
- $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ (extended transition function)

we define it by induction on the length of input string.

Base : $\hat{\delta}(q, \varepsilon) = q$, we are in state q and read no input so we are still in state q

induction:

let $w \in \Sigma^*$, $w = xa$, $x \in \Sigma^*$, $a \in \Sigma$

(x: string consisting of all but the last symbol of w

a: last symbol of w)

Then $\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

Example:

$w = 1101$

$= xa$; $x = 110$, $a = 1$

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$$

i.e. to compute $\hat{\delta}(q, w)$, first compute $\hat{\delta}(q, x)$

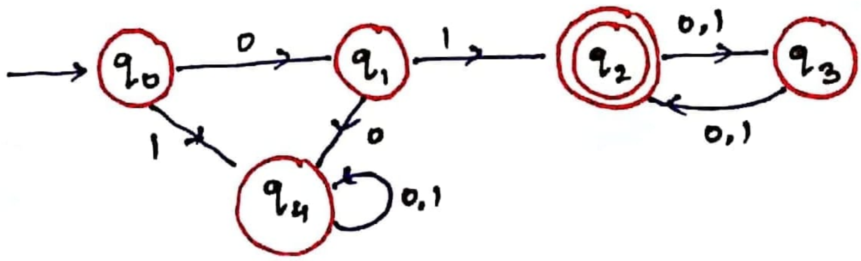
Example

Consider the DFA: $(Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0,1\}$
$\delta, q_0, F = \{q_2\})$

the transition table:

δ	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_4	q_2
$* q_2$	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

So the transition diagram will be:



If $w = 011101$, then is w accepted by the above DFA, i.e. is $\hat{\delta}(q_0, w) \in F$?

$\Rightarrow$ chech each prefix x of $w = 011101$ starting at 1 and going in increasing size.

$\hat{\delta}(q_0, \varepsilon) = q_0$

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \varepsilon), 0)$
$= \delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$
$= \delta(q_1, 1)$
$= q_2$

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$
$= \delta(q_2, 1) = q_3$

$\hat{\delta}(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1)$
$= \delta(q_3, 1)$
$= q_2$

$\hat{\delta}(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0)$
$= \delta(q_2, 0)$
$= q_3$

$\hat{\delta}(q_0, 011101) = \delta(\hat{\delta}(q_0, 01110), 1)$
$= \delta(q_3, 1)$
$= q_2 \in F$

$\Rightarrow$ w is accepted by the given DFA

$w = 011101$
$\hat{\delta}(q_0, w)$
$= \delta(\hat{\delta}(q_0, 01110), 1)$
$\downarrow$
$\delta(\hat{\delta}(q_0, 0111), 0)$
$\vdots$

→ accepts all strings of 0's and 1's of even length and begins with 01

The languages of DFA

- $A = (Q, \Sigma, \delta, q_0, F)$, a DFA
- $L(A) = \{ w \mid \hat{\delta}(q_0, w) \in F \}$

  is the language of the DFA A

  i.e. "the set of strings w that takes the start state q_0 to one of the accepting state."

Regular Language / Regular Set

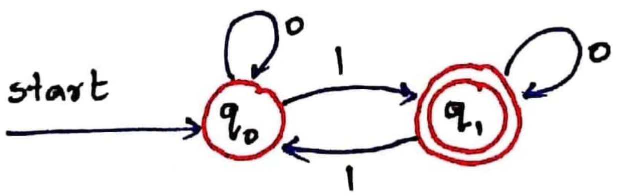
- L, a language over Σ is a regular language

if $L = L(A)$ for some DFA A.

Example:

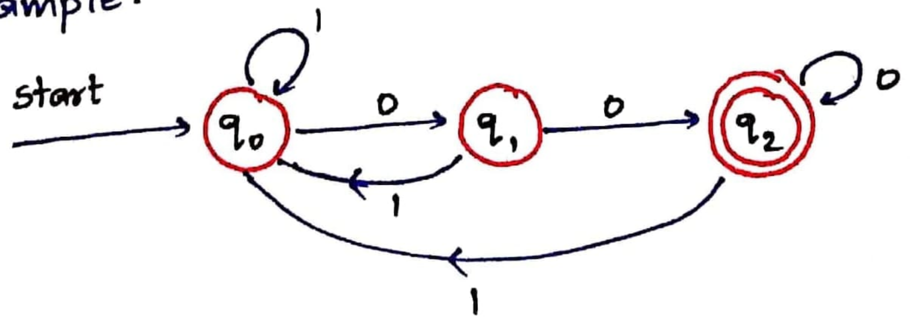
$L = \{ w \mid w$ is a binary string with odd numbers of 1's$\}$

is a regular language as the following DFA accepts L



$1 \in L$
$111 \in L$
$01 \in L$
$1011 \notin L$

Example:



This DFA accepts all binary strings ending in 00