The Roots of the Automata Theory

- The present day Automata theory has its roots in the work of the logicians and mathematicians of 1930s.
    - They studied to have an answer to the question:
        - Is an algorithm possible to determine truth or falsity of a mathematical proposition?

- David Hilbert looked for an algorithm to determine if an arbitrary formula in FOPL, applied to integers, was true.

- In 1931, Kurt Godel through his Incompleteness theorem proved that no such algorithm could exist.
    - He constructed a formula in Predicate calculus, applied to integers, whose very definition stated that it could neither be proved nor disproved within this system.

- The quest for an answer to the question *if an algorithm to ~~any~~* arbitrary *problem within a logical or mathematical framework is possible* led to the development of various abstract or mathematical machines in 1930's.
    - Lambda Calculus by Alonzo Church
    - Recursive function by Kurt Godel
    - Formal systems by Stephen Kleene
    - Post's machine by Emil Post
    - Turing Machine by Alan Turing
    - Anything computable with one model is also computable with any other model.

- Turing Machine is accepted as a theoretical model of a computer as per Church's hypothesis.
    - TM is accepted as the most powerful model of computation.

- In 1940s and 1950s, some simpler machines, called Finite Automata (FA), were developed.
    - Automata, originally proposed to model the brain functions, were later found useful for variety of other purposes:
        - Lexical Analysis
        - Network Protocol Verification etc.
- In 1950s, the linguist Noam Chomsky introduced *formal grammars,* shown to be equivalent to automata and TMs.
    - Formal grammars have application in Compilers, and some other important software.

- The Turing Machine is a simple mathematical model of a computer.
- Despite its simplicity, the Turing Machine models the computing capability of a general purpose computer.
- The Turing Machine is studied both for :
    - the class of languages it defines (called the recursively enumerable sets)
    - the class of integer functions it computes (called the partial recursive functions)
- A variety of other models of computation are also there, which are shown to be equivalent to the Turing Machine in computing power.

Clearly one cannot prove that the Turing machine model is equivalent to our intuitive notion of a computer, but there are compelling arguments for this equivalence, which has become known as Church's hypothesis.

The assumption that the intuitive notion of "computable function" can be identified with the class of partial recursive functions is known as Church's hypothesis or the Church Turing Thesis.

✓ Each Turing machine can be thought of as

✓ computing a function from integer to integers

✓ a language recognizer

$f^k_M : I^k \to I$

● A fn. computed by a TM is called a (partial) recursive function.

● If it happens to be defined for all values of its arguments then it is also called a total recursive function.

$f^k_M(i_1, i_2, \ldots, i_k)$

for every TM M & every k

If M halts with 0^j on its tape, then we say $f^k_M(i_1, i_2, \ldots, i_k) = j$

B | 0 | 0 | 0 | B

If M does not halt with a tape consisting of block of 0's with all other cells blank

B | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | B | B

then $f^k_M(i_1, i_2, \ldots, i_k)$ is undefined

Formal Languages & Automata Theory (FLAT)

- It deals with the study of abstract/mathematical machines, sometimes called automata, as well as the computational problems which can be solved on them.

- Automata - A Greek word meaning "self-acting".

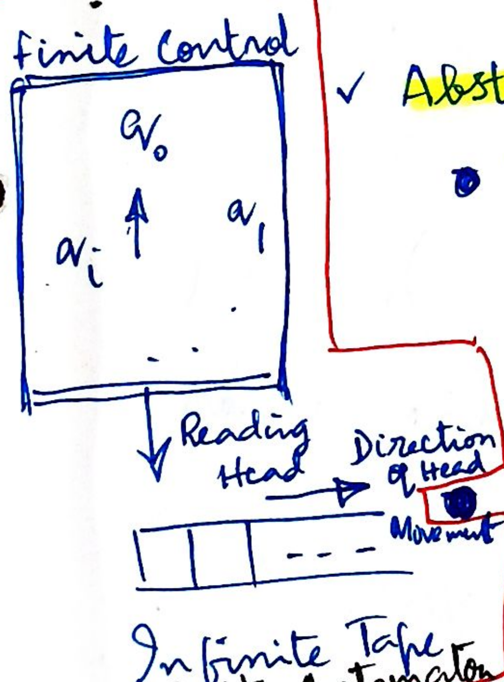

Fig. A Finite Automaton

- Abstract Machines
  - various parts of such m/cs are defined logically just by their functions and not implemented on hardware
  - Such machines may be allowed to run as long as you want (without any chance of getting heated) and may receive input as long as
  - Such m/cs sometimes may have a stack of infinite memory.

- Such machines compute in a stepwise fasion. That is, read an input symbol, perform certain function, read the next input symbol and repeat the same or stop.

- Automata-theory is closely related to formal languag theory.
  - An automaton gives a fini representation of a form language which may be a finite or an infinite s of strings formed with th symbols (taken from a finit set called Alphabet), following spe rules.
    - Examples of Formal Languages

      Σ = alphabet = $\{0, 1\}$

      L_1 = strings with one or more leadi followed by a single 1.

      L_2 = one or more leading 0's follo by an equal number of 1's.

- The Roots of the Automata Theory
  - Work of Logicians in 1930's
  - The Question
  - David Hilbert
  - Godel's Incompleteness Theorem
  - Development of Abstract Machines
  - Turing Machine — A theoretical model of the Computer
  - Finite Automata: Simpler m/cs
  - Chomsky's Grammar

✓ The Roots of the Automata Theory

Work of Logicians in 1930's

✓ The present day Automata theory has its roots in the work of the logicians and mathematicians of 1930's.

The Question

✓ They studied to have an answer to the question :

- Is a step by step solution (algorithm) possible to determine the truth or falsity of any mathematical proposition.
- Loosely speaking, a step by step solution to a problem is called a computation or an algorithm.

✓ David Hilbert

- David Hilbert looked for an algorithm to determine if an arbitrary formula in the First order predicate calculus, applied to integers, was true

✓ Godel's Incompleteness Theorem

- In 1931, Kurt Godel through his Incompleteness theorem proved that no such algorithm could exist.

- He constructed a formula in the predicate calculus, applied to intege[rs] whose very definition stated that could neither be proved nor disp[roved] within this logical system.

✓ The quest for an answer to the question if a step by step solution to any problem within a logical or mathemat[ical] framework is possible led to the deve[lopment] of various abstract or mathematical ma[chines] in 1930's.

✓Development of Abstract M/Cs

✓ Lambda Calculus by Alonzo Church, Recursive Functions by Kurt Godel, Formal Systems by Stephene Kleene, Post's machine by Emil Post, Turing Machines by Alan Turing.

✓ All these models of computation are exactly equivalent. Anythin[g] computable with one model is also computable with any other model.

✓Turing Machines A Theoretical Model Computer

✓ TM is accepted as a theoretical model of a computer as per Church's hypothesis.

✓ That is, TM is accepted as the most powerful model of Computation

A: Simpler M/Cs

✓ In 1940's and 1950s, some simpler machines, called Finite Automata (FA) were developed.

✓ Automata, originally proposed to model the brain functions, were later found useful for variety of other purposes, such as Lexical Analysis, Network Protocol Verification etc..

[Ch]omsky's [Gr]ammars

✓ In 1950's, the linguist Noam Chomsky introduced formal grammars, shown to be equivalent to automata and TMs

✓ Formal grammars have application in compilers, and some other important software.

formal grammars ≡ Automata ≡ TMs

Why to Study Automata Theory

- ✓ Automata are essential for the study of limits of computation.

- ✓ There are two important issues

  1. What can a computer do at all? This study is called "decidability", and the problems that can be solved by computer are called "decidable".

  2. What can a computer do efficiently? This study is called " "intractability", and the problems that can be solved by a computer using no more time than some slowly growing function of the size of the input are called "tractable". Often we take all polynomial functions to be slowly growing" while functions that deemed to grow faster than polynomial are

to grow too fast.