Turing Machines

N Geetha
AM & CS
PSG Tech

Computational Models

A **Computational Model** is a mathematical object (Defined on paper) that enables us to reason about computation and to study the properties and limitations of computing.

Will deal with Three principal computational models in increasing order of **Computational Power.**

Computational Models

We will deal with three principal models of computations:

1. Finite Automaton (in short FA).
   recognizes Regular Languages .
2. Push Down Automaton (in short PDA).
   recognizes Context Free Languages .
3. Turing Machines (in short TM).
   recognizes Computable Languages .

Alan Turing - A Short Detour

Dr. Alan Turing is one of the founders of Computer Science (he was an English Mathematician). Important facts:

1. “Invented” Turing machines.
2. “Invented” the **Turing Test.**
3. Broke into the German submarine transmission encoding machine “Enigma”.
4. Was arraigned for being gay and committed suicide soon after.

Different Kinds of Automata

Automata are distinguished by the temporary memory

- **Finite Automata**: no temporary memory

- **Pushdown Automata**: stack

- **Turing Machines**: random access memory

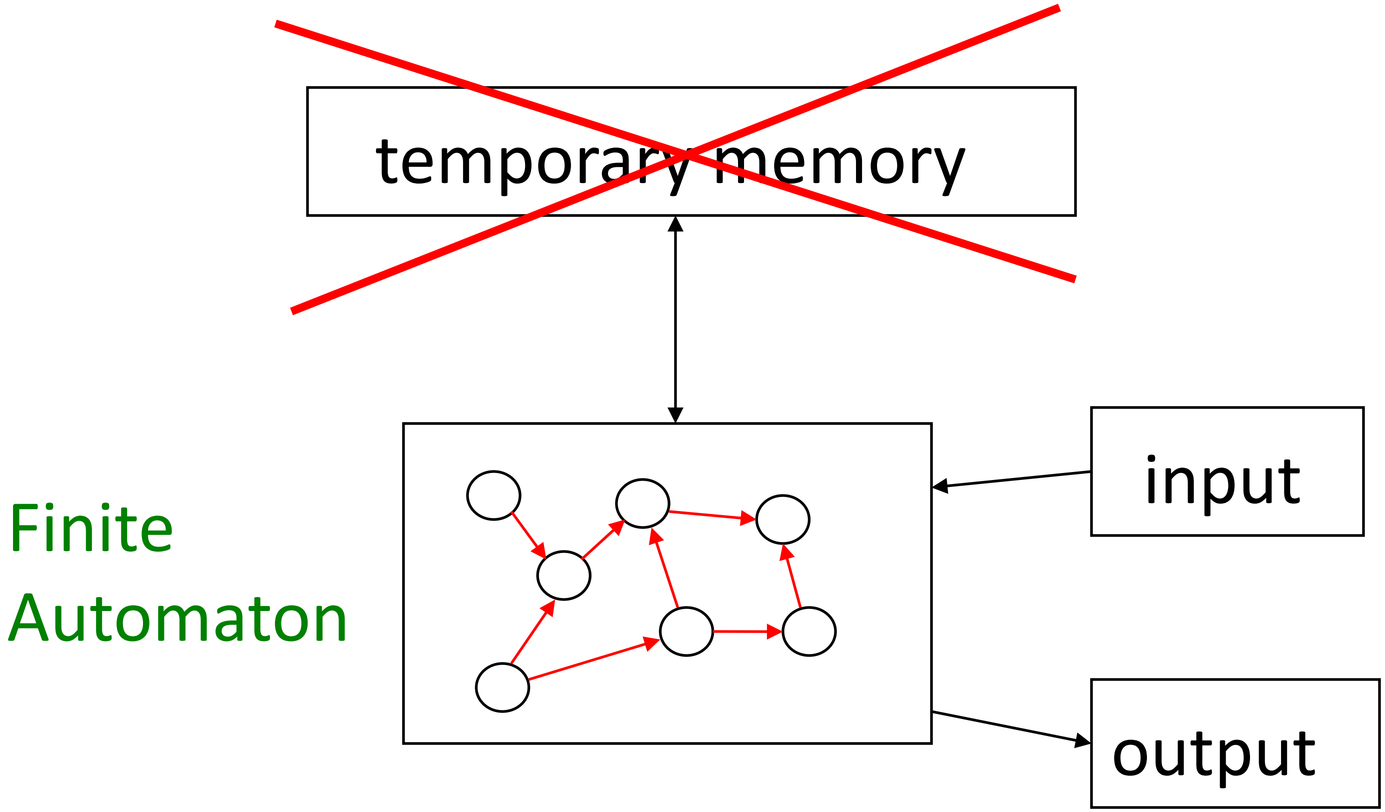
Memory affects computational power:

More flexible memory

results to

The solution of more computational problems

Finite Automaton

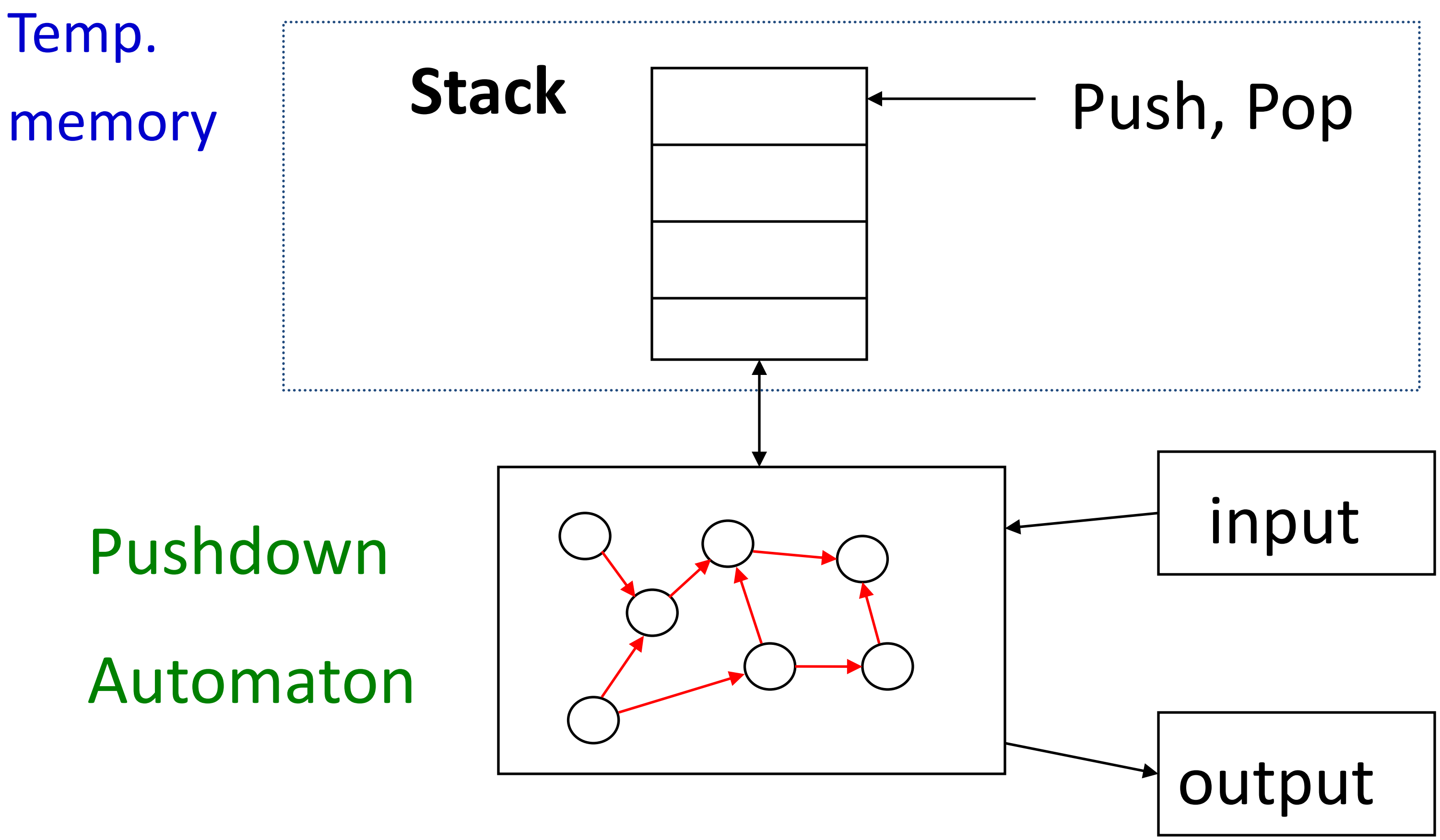


Example: Elevators, Vending Machines,

Lexical Analyzers

(small computing power)

Pushdown Automaton



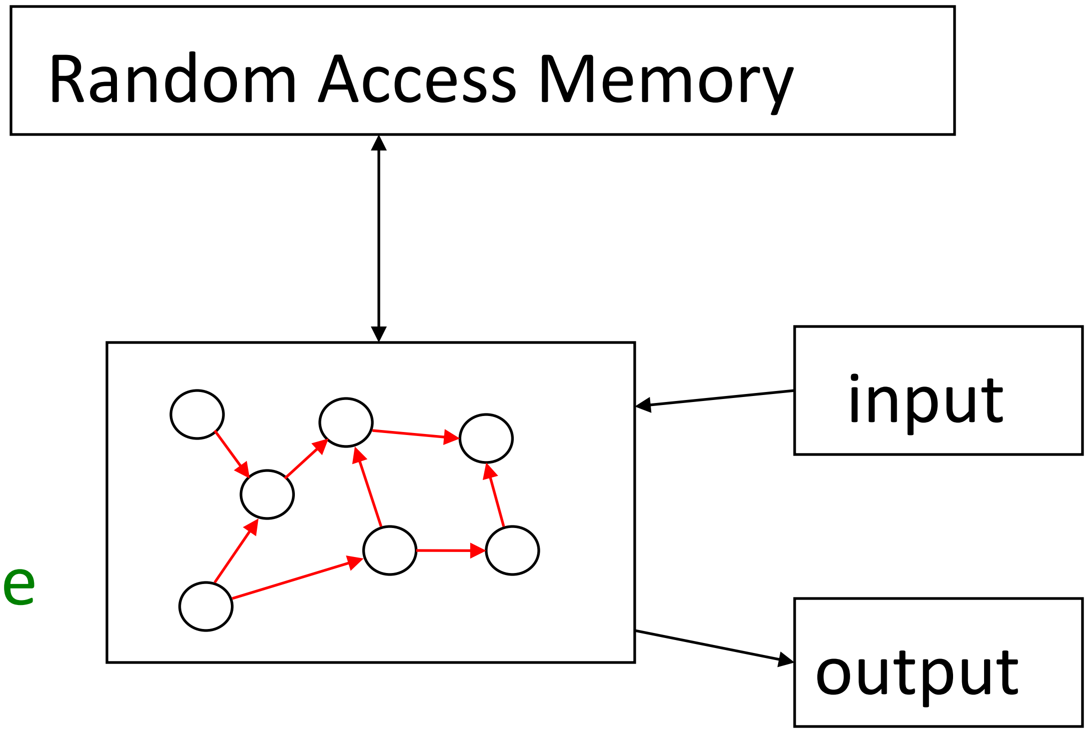
Example: Parsers for Programming Languages

(medium computing power)

Turing Machine

Temp. memory

Turing Machine



Examples: Any Algorithm

(highest known computing power)

Power of Automata

Simple problems		More complex problems		Hardest problems
Finite Automata	<	Pushdown Automata	<	Turing Machine

Less power → More power

Solve more computational problems

Turing Machine is the most powerful known computational model

Question: can Turing Machines solve all computational problems?

Answer: NO
(there are unsolvable problems)

Time Complexity of Computational Problems:

P problems:

(Polynomial time problems)

Solved in polynomial time

NP-complete problems:

(Non-deterministic Polynomial time problems)

Believed to take exponential

time to be solved

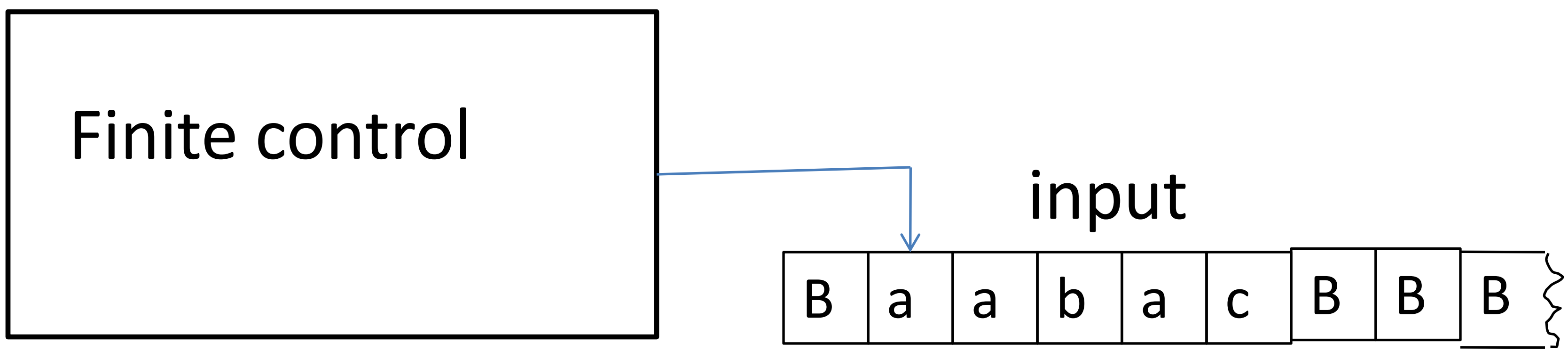
Turing Machines

A **Turing Machine** is a finite state machine augmented with an infinite tape.

The tape head can go in both directions. It can read and write from/to any cell of the semi-infinite tape.

Once the TM reaches an **accept (reject resp.)** state it **accepts (rejects resp.) immediately**.

Schematic of a Turing Machine



The tape head can go in both directions. It can read and write from/to any cell of the semi-infinite tape. The B symbol is bracketed on either side of the input.

TM – A Formal Definition

A ***Turing Machine*** is a 7-tuple $\left(Q, \Sigma, \Gamma, \delta, q_0, B, q_{accept}, \right)$ where:

1. Q is a finite set called the ***states***.
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** , B .
3. Γ is the ***tape alphabet***, $B \in \Gamma$ and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the ***transition function***.
5. $q_0 \in Q$ is the ***start state***.
6. *B is a special tape symbol.*
7. $q_{accept} \in Q$, the ***accept state***, and $q_{reject} \in Q$, the ***reject state***.

The Transition Function - Domain

Let M be a Turing machine defined by $(Q, \Sigma, \Gamma, \delta, q_0, B, q_{accept})$. at any given time M is in some state, $q \in Q$, and its head is on some tape square containing some tape symbol $\gamma \in \Gamma$. The transition function $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$, depends on the machine state q and on the tape symbol γ.

The Transition Function - Range

The range of the transition function are triples of the type (q',γ',d), where q' is M's next state, γ' is the symbol written on the tape cell over which the head was at the beginning of the transition (namely γ is replaced with γ') and $d \in \{L,R\}$ is the direction towards which the tape head has made a step.

Turing machine – A Computation

Computation of M always starts at state q_0, and the input is on the leftmost n cells where n is the input's ***length***. The tape's head is over the tape's cell 0 – the leftmost cell.

Computation of M ends either when it reaches $q_{accept} \in Q$ - this is an ***Accepting Computation***. Or when it reaches $q_{reject} \in Q$ - this is a ***Rejecting Computation***.

Configurations

A configuration of a Turing machine M is a concise description M's state and tape contents. It is written as $C=uqv$. and its meaning is:

1. The state of M is q.
2. The content of M's tape is uv , where u resides on the leftmost part of the tape.
3. The head of M resides over the leftmost (first) symbol of v.
4. The tape cells past the end of v hold blanks.

Configurations

Configuration C_1 of M ***yields*** Configuration C_2, if

M can legally go from C_1 to C_2 in a single step.

For example:

Assume that $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, and $q_i, q_j \in Q$.

We say that uaq_ibv yields uq_jacv, if $\delta(q_i,b)=(q_j,c,L)$,

for a leftward movement of the head.

We say that uaq_ibv yields $uacq_jv$, if $\delta(q_i,b)=(q_j,c,R)$,

for a rightward movement of the head.

Configurations – Special Cases

Configuration $q_i bv$ yields $q_j cv$, if the head is at the beginning of the tape and the transition is **left-moving**, because the head cannot go off the left-hand end of the tape.

Configuration uaq_i is equivalent to $uaq_i \_$, because the empty part of the tape is always filled out with blanks.

Computations

The ***start*** Configuration of M on input w is $q_0 w$, which indicates that M is at its initial state, q_0 , it's head is on the first cell of its tape and the tape's content is the input w.

Any configuration in which of M reaches state q_{accept} , is an ***accepting configuration***.

Any configuration in which M reaches state q_{reject} ,is a ***rejecting configuration***.

Computations

Accepting and rejecting configurations are ***halting configurations***.

A TM M ***accepts*** word w if there exists a computation (a sequence of configurations) of M, $C_1, C_2, ..., C_k$ satisfying:

1. $C_1 = q_0 w$ is the starting state of M on input w.
2. For each $i, 1 \leq i < k$, C_i yields C_{i+1}, and
3. C_k is an accepting configuration.

Computation Outcomes

A Computation of a Turing machine M may result in three different ***outcomes***:

1. M may ***accept*** **–** By halting in q_{accept} .
2. M may ***reject*** **–** By halting in q_{reject} .
3. M may ***loop*** **–** By not halting **for ever**.

Note: When M is running, it is not clear whether it is **looping** . Meaning M **may stop eventually but nobody can tell.**

Turing Recognizers

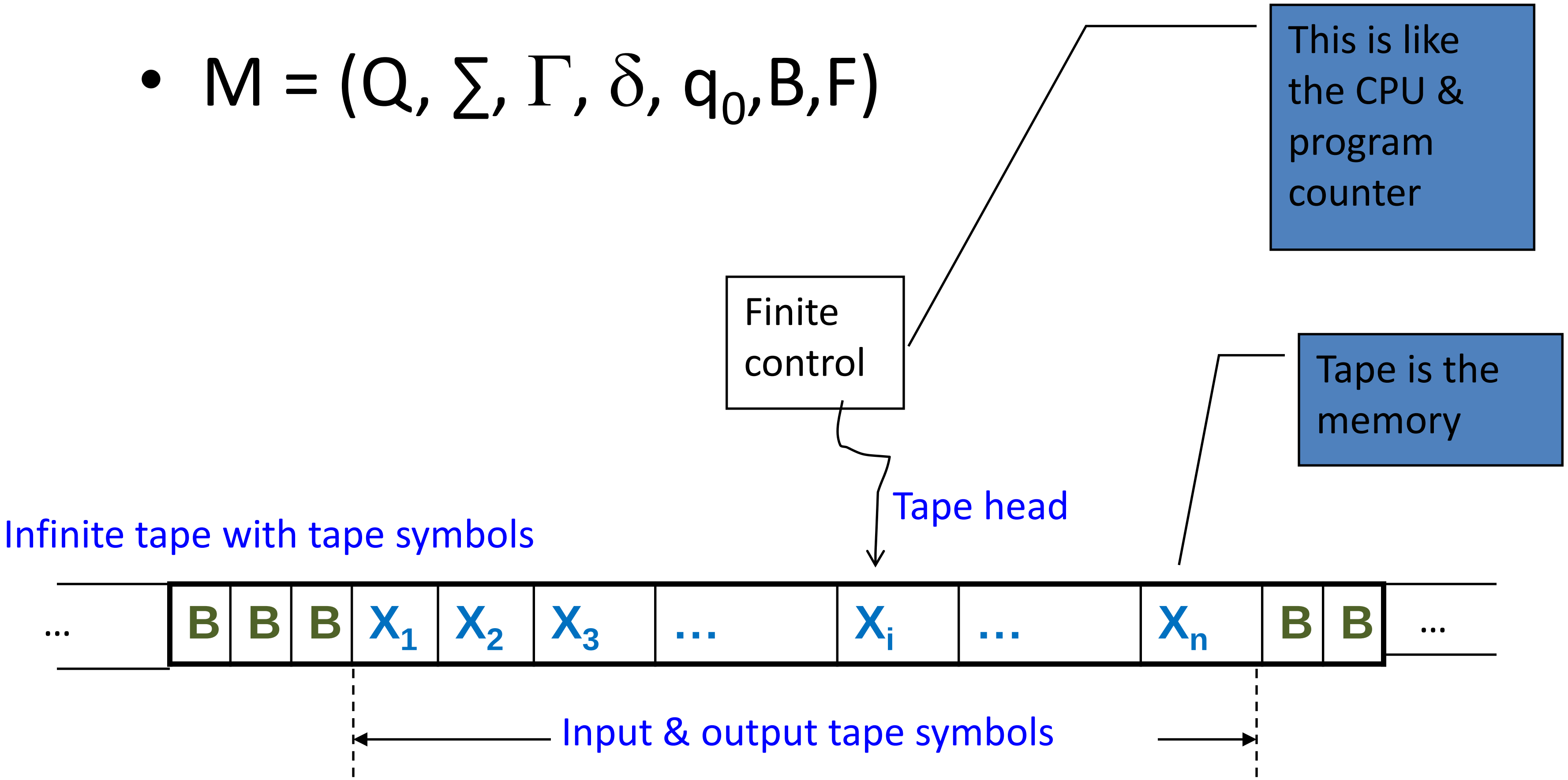
The collection of strings that M accepts is ***the language of*** $\boldsymbol{M}$, denoted T(M).

A string w in $\sum^*$ is accepted by M if $q_0w \vdash^* \alpha q_f\beta$ where α, β belong to Γ^* and q_f is a final state.

A language is ***Turing Recognizable*** if there exists a Turing machine that recognizes it.

A Turing Machine (TM)

- $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

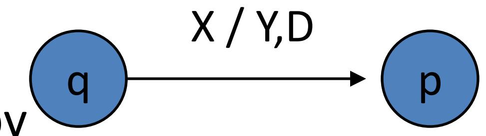


B: blank symbol (special symbol reserved to indicate data boundary)

Transition function

You can also use:
$\rightarrow$ for R
$\leftarrow$ for L

- One move (denoted by |---)
  in a TM does the following:
  - ***$\delta(q,X) = (p,Y,D)$***
    - q is the current state
    - X is the current tape symbol pointed by tape head
    - State changes from q to p
    - <u>After the move:</u>
      - X is replaced with symbol Y
      - If D="L", the tape head moves "left" by one position.
        Alternatively, if D="R" the tape head moves "right" by one position.



ID of a TM

- Instantaneous Description or ID :
  - $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n$

    means:
    - q is the current state
    - Tape head is pointing to X_i
    - $X_1X_2 \ldots X_{i-1}X_iX_{i+1} \ldots X_n$ are the current tape symbols

- $\delta(q,X_i) = (p,Y,R)$ is same as:

  $X_1 \ldots X_{i-1}qX_i \ldots X_n \quad |{-}{-}{-}{-} \quad X_1 \ldots X_{i-1}YpX_{i+1} \ldots X_n$
- $\delta(q,X_i) = (p,Y,L)$ is same as:

  $X_1 \ldots X_{i-1}qX_i \ldots X_n \quad |{-}{-}{-}{-} \quad X_1 \ldots pX_{i-1}YX_{i+1} \ldots X_n$

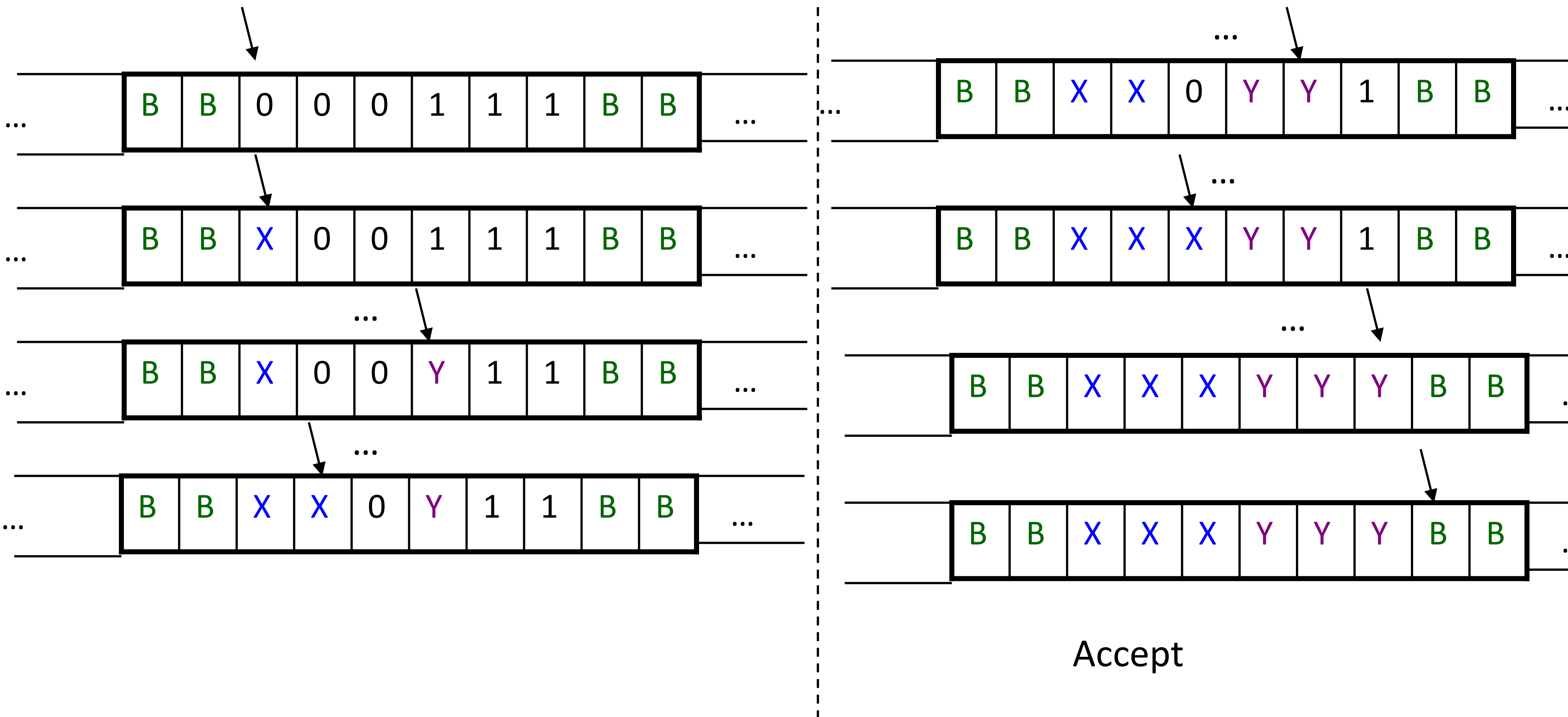
Way to check for Membership

- Is a string *w* accepted by a TM?

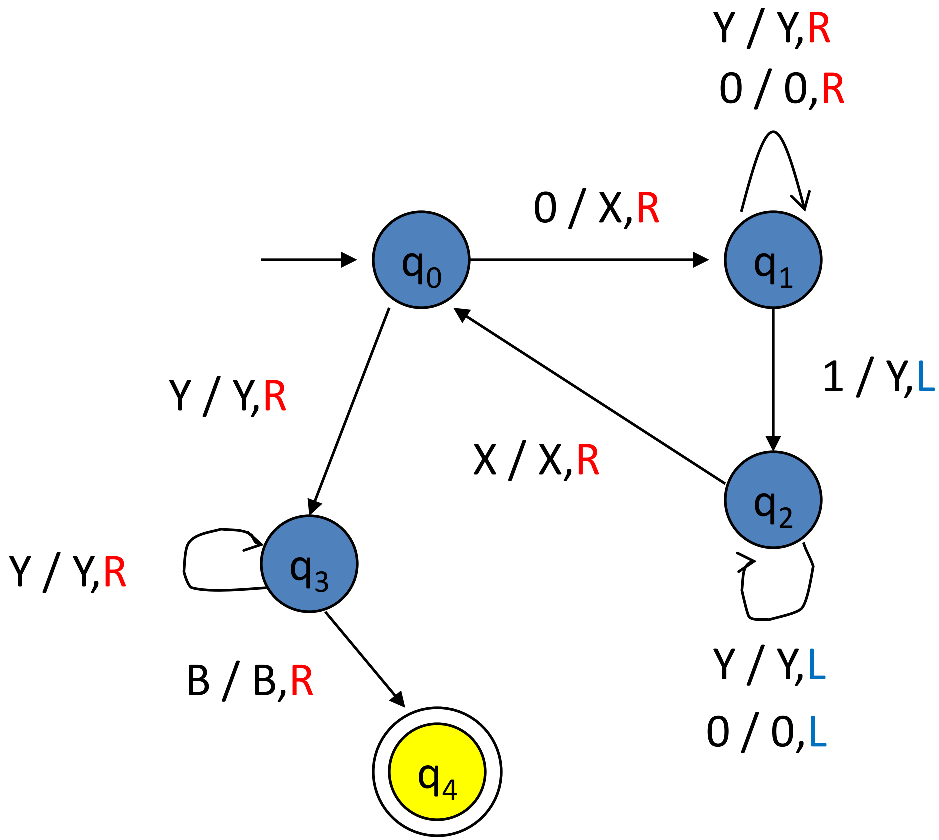
- <u>Initial condition:</u>
  - The (whole) input string *w* is present in TM, preceded and followed by infinite blank symbols
- <u>Final acceptance:</u>
  - Accept *w* if TM enters <u>final state</u> and halts
  - If TM halts and not final state, then reject

Example: $L = \{0^n1^n \mid n \geq 1\}$

- Strategy: w = 000111



1)TM for $\{0^n1^n \mid n \geq 1\}$ Transition Diagram



1. Mark next unread 0 with X and move right
2. Move to the right all the way to the first unread 1, and mark it with Y
3. Move back (to the left) all the way to the last marked X, and then move one position to the right
4. If the next position is 0, then goto step 1.
   Else move all the way to the right to ensure there are no excess 1s. If not move right to the next blank symbol and stop & accept.

*state diagram representation preferred

2)TM for $\{0^n1^n \mid n\geq1\}$ State table

	Next Tape Symbol				
Curr. State	0	1	X	Y	B
$\rightarrow q_0$	(q_1,X,R)	-	-	(q_3,Y,R)	-
q_1	$(q_1,0,R)$	(q_2,Y,L)	-	(q_1,Y,R)	-
q_2	$(q_2,0,L)$	-	(q_0,X,R)	(q_2,Y,L)	-
q_3	-	-	-	(q_3,Y,R)	(q_4,B,R)
$*q_4$	-	--	-	-	-

Table representation of the state diagram

3) 5-tuple configurations for representation

- $(q_0,0,q_1,X,R)$, (q_0,Y,q_3,Y,R),

  $(q_1,0,q_1,X,R)$, $(q_1,1,q_2,Y,L)$,
  (q_1,Y,q_2,Y,L), $(q_2,0,q_2,0,L)$,
  (q_2,X,q_0,X,R), (q_2,Y,q_2,Y,L),
  (q_3,Y,q_3,Y,R), (q_3,B,q_4,B,R)

Derivation for w= $0^2 1^2$

- $Bq_0 0011B \vdash Bxq_1 011B \vdash Bx0q_1 11B \vdash Bx0q_2 Y1B$

  $\vdash Bxq_2 0Y1B \vdash Bq_2 x0Y1B \vdash Bxq_0 0Y1B$

  $\vdash Bxxq_1 Y1B \vdash BxxYq_1 1B \vdash BxxYq_2 YB$

  $\vdash Bxxq_2 YYB \vdash Bxq_2 xYYB \vdash Bxxq_0 YYB$

  $\vdash BxxYq_3 YB \vdash BxxYYq_3 B \vdash BxxYYBq_4$

  Accepted.

Example 2

Consider the language $L = \{w\#w \mid w \in \{0,1\}^*\}$

A simple method to check whether a string w is in L is:

S1. Store the leftmost symbol on the tape and cross it out by writing x.

S2. Go right past #, if # not found, **reject**.

S3. Compare the leftmost non x symbol to the stored symbol. If not equal, **reject**.

S4. Cross out the compared symbol. Return the head to the left-hand end of the tape.

S5. Go to S1.

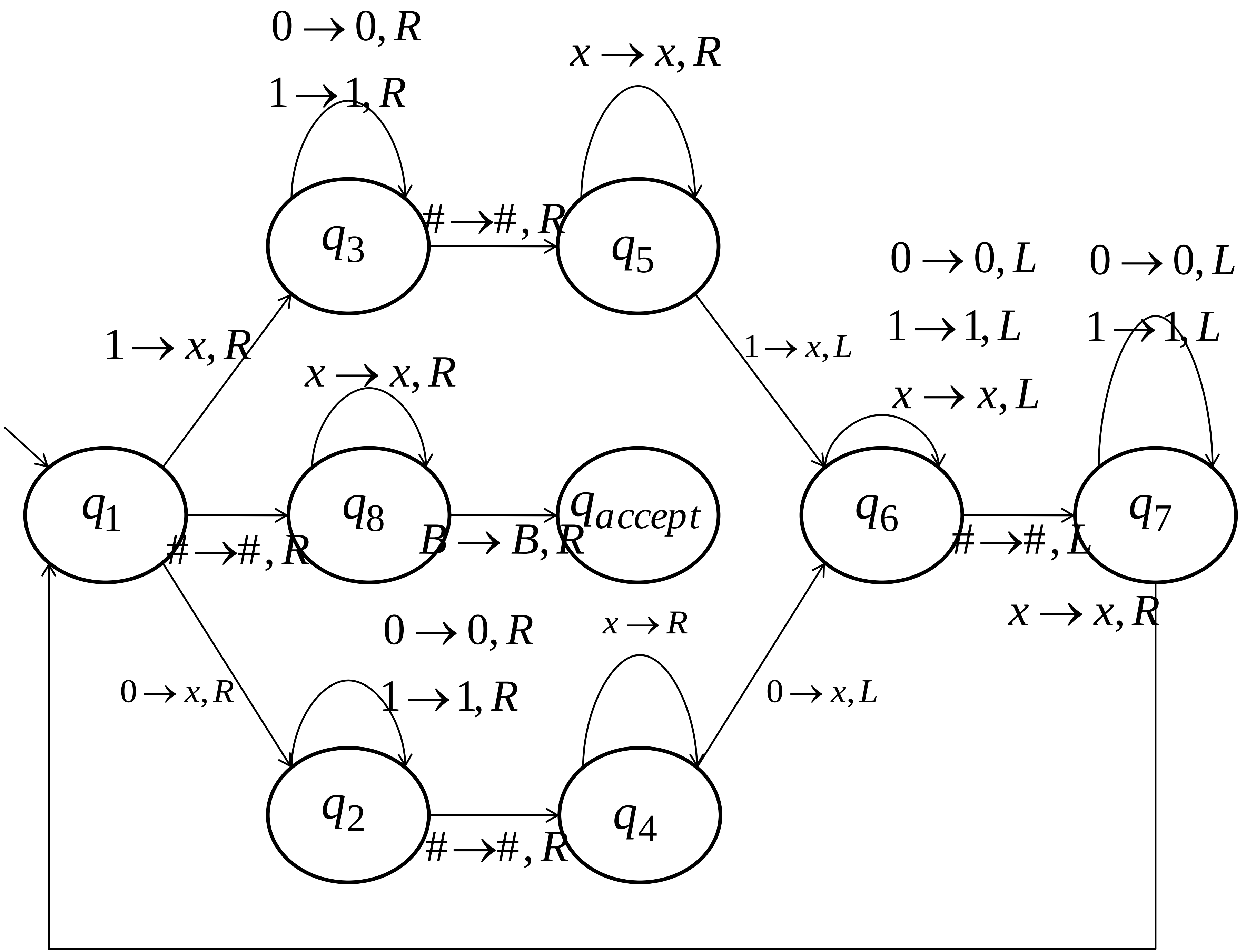
Repeat this procedure until all the string w is scanned. If an unexpected character is found, **reject**. Otherwise, **accept.**

Example 2

In the following slide the transition function of M_2 is presented.

Note:

1. $\Sigma = \{0,1,\#\}\Gamma, = \{0,1,\#, x, B\}$
2. In this description, state q_{reject} and all its incoming transitions are omitted. Wherever there is a missing transition, it goes to q_{reject} .



$0 \to 0, R$
$1 \to 1, R$
$x \to x, R$
$\# \to \#, R$
q_3
q_5
$0 \to 0, L$
$1 \to 1, L$
$0 \to 0, L$
$1 \to 1, L$
$1 \to x, R$
$1 \to x, L$
$x \to x, R$
$x \to x, L$
q_1
q_8
q_{accept}
q_6
q_7
$\# \to \#, R$
$B \to B, R$
$\# \to \#, L$
$x \to x, R$
$0 \to 0, R$
$x \to R$
$0 \to x, R$
$1 \to 1, R$
$0 \to x, L$
q_2
q_4
$\# \to \#, R$

Example 3

Consider the language L = { $a^i b^j c^k$ / i * j=k, i,j,k >=1}

A simple method to check whether a string w is in L is:

S1. Scan the input from left to right to be sure it is $a^+b^+c^+$

S2. Return the tape head to the left end of the input

S3. Cross off an ‘a’. Scan to the right until a ‘b’ occurs. Shuttle between the ‘b’s and ‘c’s crossing off one of each until all b’s are gone. If all ‘c’s have gone off and some ‘b’s remain, then **reject**.

S4. Restore the crossed off ‘b’s and repeat S3 if there is another ‘a’ to cross off. If all ‘a’s are crossed off, determine if all ‘c’s are crossed off. If yes, **accept** otherwise, **reject**.

TMs for calculations

- TMs can also be used for calculating values
  - Like arithmetic computations
  - Eg., addition, subtraction, multiplication, etc.
- TMs can also used for comparison $x > y$
- Such TMs are called as computing machines

Example 4: Addition

- Input is 4 + 3 given as unary ones:
- B11110111B and Output should be B1111111B
- Process:
- S1: scan until 0; change to 1;
- S2 : continue till B
- S3:Move left change 1 to B
- S4: Move till left 1 and position the head.

Example 5: subtraction

"m -- n" = max{m-n,0}

0^m10^n → ...B 0^{m-n} B.. (*if m>n*)
...BB...B.. (*otherwise*)

1. For every 0 on the left (mark X), mark off a 0 on the right (mark Y)
2. Repeat process, until one of the following happens:
   1. // No more 0s remaining on the left of 1
      Answer is 0, so flip all excess 0s on the right of 1 to Bs (and the 1 itself) and halt
   2. //No more 0s remaining on the right of 1
      Answer is m-n, so simply halt after making 1 to B

Give state diagram

Example 6: Multiplication

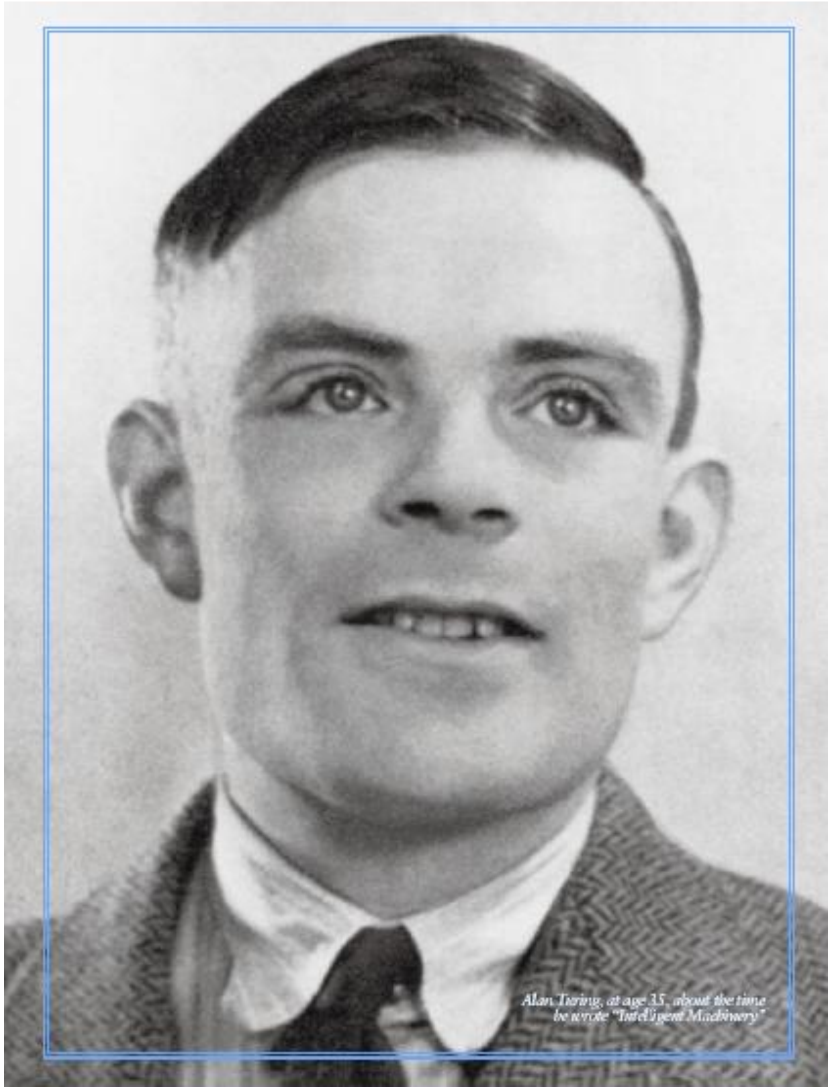
- 0^m10^n1 (input), $0^{mn}1$ (output)

- <u>Pseudocode</u>:
  1. Move tape head back & forth such that for every 0 seen in 0^m, write n 0s to the right of the last delimiting 1
  2. Once written, that zero is changed to B to get marked as finished
  3. After completing on all m 0s, make the remaining n 0s and 1s also as Bs

Give state diagram

Foundations

- The theory of computation and the practical application it made possible — the computer — was developed by an Englishman called Alan Turing.



Alan Turing, at age 35, about the time he wrote "Intelligent Machinery"

Alan Turing

- 1912 (23 June): Birth, Paddington, London
- 1931-34: Undergraduate at King's College, Cambridge University
- 1932-35: Quantum mechanics, probability, logic
- 1936: The Turing machine, computability, universal machine
- 1936-38: Princeton University. Ph.D. Logic, algebra, number theory
- 1938-39: Return to Cambridge. Introduced to German Enigma cipher machine
- 1939-40: The Bombe, machine for Enigma decryption
- 1939-42: Breaking of U-boat Enigma, saving battle of the Atlantic
- 1946: Computer and software design leading the world.
- 1948: Manchester University
- 1949: First serious mathematical use of a computer
- 1950: The Turing Test for machine intelligence
- 1952: Arrested as a homosexual, loss of security clearance
- 1954 (7 June): Death (suicide) by cyanide poisoning, Wilmslow, Cheshire.
- —from Andrew Hodges http://www.turing.org.uk/turing/

The Decision Problem

- In 1928 the German mathematician, David Hilbert (1862-1943), asked whether there could be a mechanical way (i.e. by means of a fully specifiable set of instructions) of determining whether some statement in a formal system like arithmetic was provable or not.
- In 1936 Turing published a paper the aim of which was to show that there was no such method.
- "On computable numbers, with an application to the *Entscheidungs* problem." Proceedings of the London Mathematical Society, 2(42):230-265).

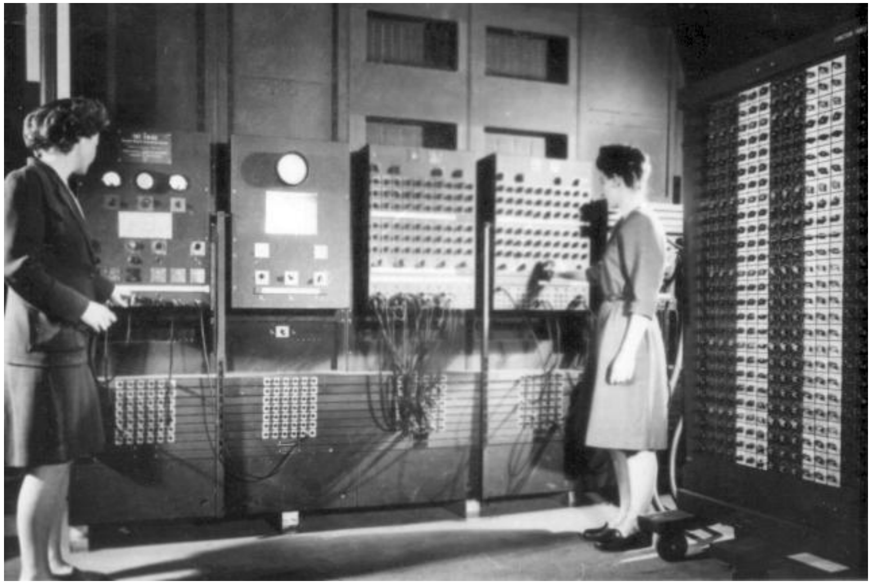
The Turing Machine

- In order to argue for this claim, he needed a clear concept of "mechanical procedure."
- His idea — which came to be called the Turing machine — was this:
- (1) A tape of infinite length
- (2) Finitely many squares of the tape have a single symbol from a finite language.
- (3) Someone (or something) that can read the squares and write in them.
- (4) At any time, the machine is in one of a finite number of internal states.
- (5) The machine has instructions that determine what it does given its internal state and the symbol it encounters on the tape. It can
  - ◆ change its internal state;
  - ◆ change the symbol on the square;
  - ◆ move forward;
  - ◆ move backward;
  - ◆ halt (i.e. stop).

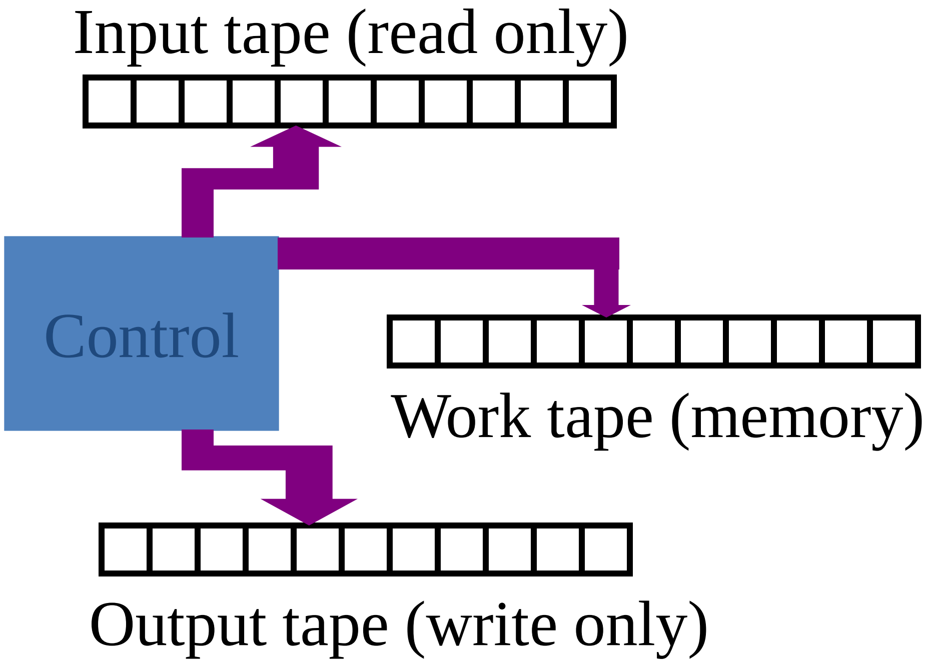
Turing’s Theorem

- In the 1936 paper Turing proved that there are “general-purpose” Turing machines that can compute whatever any other Turing machine.
- This is done by coding the function of the special-purpose machine as instructions of the other machine — that is by “programming” it. This is called Turing’s theorem.
- These are universal Turing machines, and the idea of a coding for a particular function fed into a universal Turing machine is basically our conception of a computer and a stored program.
- The concept of the universal Turing machine is just the concept of the computer as we know it.

First computers: custom computing machines



1950 -- Eniac: the control is hardwired manually for each problem.



1940: **VON NEUMANN:**

DISTINCTION BETWEEN DATA AND INSTRUCTIONS

The Turing Test

- The game runs as follows. You sit at a computer terminal and have an electronic conversation. You don't know who is on the other end; it could be a person or a computer responding as it has been programmed to do.
- If you can't distinguish between a human being and a computer from your interactions, then the computer is intelligent.
- Note that this is meant to be a sufficient condition of intelligence only. There may be other ways to be intelligent.

The Church-Turning Thesis

- Turing, and a logician called Alonzo Church (1903-1995), independently developed the idea (not yet proven but widely accepted) that whatever can be computed by a mechanical procedure can be computed by a Turing machine.
- This is known as the Church-Turing thesis.