

# Turing Machine Extensions, and Enumerators

Prof. (Dr.) K.R. Chowdhary  
*Email: kr.chowdhary@iitj.ac.in*

Formerly, Prof. & HOD, Dept. of CSE MBM Engg. College

Monday 21<sup>st</sup> July, 2025

K.R. Chowdhary

# Theory of Computation

Automata, Formal Languages, Computation and Complexity

Focuses on pedagogy in its writing, that represents a refreshing approach

Ensures comprehensive and enjoyable learning

Undergone a rigorous classroom testing



© 2025

Get 20% off with this code: **SPRAUT**

Available on Springer Nature Link

[link.springer.com/book/  
9789819762347](https://link.springer.com/book/9789819762347)

Please note that promotional coupons are only valid for English-language Springer, Apress, and Palgrave Macmillan books & eBooks and are redeemable on [link.springer.com](https://link.springer.com) only. Titles affected by flood book price laws, forthcoming titles and titles temporarily not available on Springer Nature Link are excluded from promotions, as are reference works, handbooks, encyclopedias, subscriptions, or bulk purchases. The currency in which your order will be invoiced depends on the billing address associated with the payment method used, not necessarily your home currency. Regional VAT/tax may apply. Promotional prices may change due to exchange rates. Promotions are valid for individual customers only. Bookellers, book distributors, and institutions such as libraries and corporations, please visit [springernature.com/contact-us](https://springernature.com/contact-us). Promotions do not work in combination with other discounts or gift cards.

# Ways to Extend Turing Machines

Many variations have been proposed:

- Multiple tape Turing machine
- Multiple track Turing machine
- Two dimensional Turing machine
- Multidimensional Turing machine
- Two-way infinite tape
- Non-determinic Turing machine
- Combinations of the above
- **Theorem:** The operations of a TM allowing some or all the above extensions can be simulated by a standard TM. The extensions do not give us machines more powerful than the TM.
- The extensions are helpful in designing machines to solve particular problems.

# Multiple Tape TM

## Variants of TM:

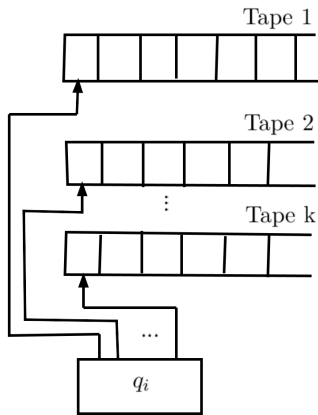
- For example, in two tape Turing machine, each tape has its own read-write head, but the state is common for all tapes and all heads.
- In each step (transitions) TM reads symbols scanned by all heads, depending on head position and current state, each head writes, moves R or L, and control-unit enter into new state.
- Actions of heads are independent of each other.
- Tape position in two tapes:  $[x, y]$ ,  $x$  in first tape, and  $y$  in second, and  $\delta$  is given by:

$$\delta(q_i, [x, y]) = (q_j, [z, w], d, d), \quad d \in \{L, R\}$$

$\delta$	$a, a$	$B, a$	$a, B$	$B, B$
$q_0$	$q_1, a, b, L, R$	$q_2, b, B, L, L$	$q_0, b, B, L, R,$	$\dots$

- A standard TM is multi-tape TM with single tape.
- **Example:** These multi-tape TMs are better suited for specific applications, e.g., copying string from one tape to another tape. However, their time-complexity remains **P** only.

# Multiple Tape TM



A transition in a multi-tape

Turing machine, for  $k \geq 1$   
number of tapes:

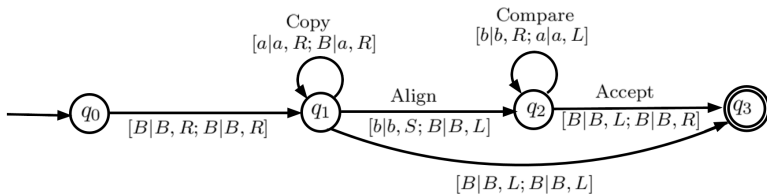
$$\delta : (Q - H) \times \Gamma^k \rightarrow \\ Q \times \Gamma^k \times \{L, R\}^k$$

$$\delta(q_i, a_1, \dots, a_k) = \\ q_j, (b_1, \dots, b_k), (d_1, \dots, d_k)$$

The steps to carry out a  
transition are:

1. change to next state;
2. write one symbols on each  
tape;
3. independently reposition each  
tape heads.

# Two-tape TM to recognize language $L = \{a^n b^n \mid n \geq 0\}$

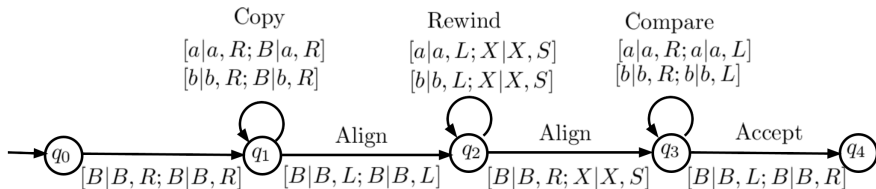


**Working:** with The original string  $w$  is on tape 1

- 1 Copies the string  $a$ 's on tape 2.
- 2 Moves head 1 to begin of  $b$ 's, and head on last  $a$ 's
- 3 Compare number of  $b$ 's tape 1 with number of  $b$ 's on tape 2 by moving heads in opposite directions.
- 4 Time Complexity:  $1 + n + 1 + n + 1 = O(n + 3) = O(n) =$  Polynomial (**P**) (Compare it with standard TM for same problem with complexity  $O(n^2)$ )

# Two-tape TM to recognize language

$$L = \{ww^R \mid w \in \{a, b\}^*\}$$

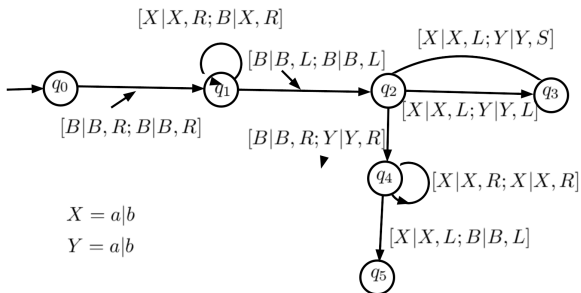


**Working:** with The original string  $w$  is on tape 1

- 1 Copies the string  $w$  on tape 2.
- 2 Moves head 1 to extreme left (rewind)
- 3 Aligns both heads on opposite, i.e, head 1 to extreme left, head 2 to extreme right
- 4 Compare by moving heads in opposite directions. Accept when compare ok.
- 5 Let input is  $|ww^R| = n$ . Time Complexity:  $1 + n + 1 + n + 1 + n + 1 = O(3n + 4) = O(n)$ . This is **Polynomial** Complexity, hence type of complexity is  $P$  (i.e., type  $P$ )

# Multi-Tape TM

**Example:** Construct 2-tape TM to recognize the language  $L = \{ww \mid w \in \{a, b\}^*\}$ .



**steps:**(Note:  $x \in \{a, b\}, y \in \{a, b\}$ )

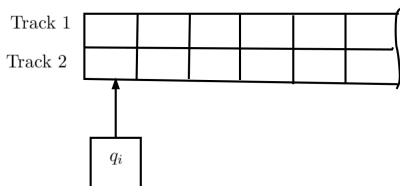
- Initially the string  $ww$  is on tape-1. Copy it to tape-2, at the end both R/W heads are at right most.
  - Move both heads left: Head-1, 2-steps and head-2, 1-step each time.
  - Move both heads right, each one step. Head-1 moves in first  $w$  of  $ww$  and head-2 moves in second  $w$ , comparing these  $w$  in  $q_4$ .
- In  $q_3 \rightarrow q_2$  transition, the move ' $Y|Y, S$ ' keeps head2 *stationary*.

# Multiple Tape TM

## Simulation of a three-tape TM $M$ by standard TM $S$ :

- Let the contents of a three tape TM  $M$  are:  
0**1**010**B** Tape 1: bold character is R/W head position  
**a**a**a****B** Tape 2: bold character is R/W head position  
**b**a**B** Tape 3: bold character is R/W head position
- Tape contents on simulated standard TM  $S$ :  
0**1**010#**a**a**a**#**b**a**B** , # is separator for contents of 3 tapes.
- In practice,  $S$  leaves an extra blank before each symbol to record position of read-write heads
- $S$  reads the symbols under the virtual heads (L to R).
- Then  $S$  makes a second pass to update the tapes according to the way the transition function of  $M$  dictates.
- If, at any point  $S$  moves one of the virtual heads to the right of #, it implies that head moved to unread blank portion of that tape. So  $S$  writes a blank symbol in the right most of that tape. Then continues to simulate.  
⇒ control will need a lot more states.

# Multiple track TM



- The tape is divided into tracks. A tape position in  $n$ -track tape contains  $n$  symbols from tape alphabets.
- Tape position in two-track is represented by  $[x, y]$ , where  $x$  is symbol in track 1 and  $y$  is in track-2. The states,  $\Sigma$ ,  $\Gamma$ ,  $q_0$ ,  $F$  of a two-track machine are same as for standard machine.
- A transition of a two-track machine reads and writes the entire position on both the tracks.
- $\delta$  is:  $\delta(q_i, [x, y]) = [q_j, [z, w], d]$ , where  $d \in \{L, R\}$ . The input for two-track is put at track-1, and all positions on track-2 is initially blank. The acceptance in multi-track is by final state.
- Languages accepted by two-track machines are **Recursively Enumerable** languages.

# Multi-track Turing Machine = Standard TM

## Theorem

A language is accepted by a two-track TM  $M$  if and only if it is accepted by a standard TM  $M'$ .

## Proof.

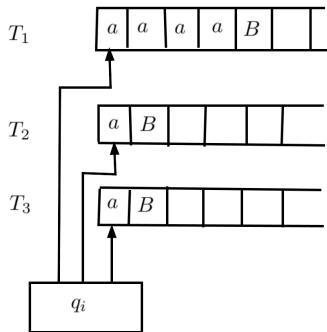
- **Part 1:** If  $L$  is accepted by standard TM  $M'$  then it is accepted by two-track TM  $M$  also (for this, simply ignore 2nd track). Hence track content is tuple  $[a, B]$ .

### Part 2:

- Let  $M = (Q, \Sigma, \Gamma, \delta, q_0, H)$  be two track machine. Find one equivalent standard TM  $M'$
- Create ordered pair  $[x, y]$  on single tape machine  $M'$ .
- $M' = (Q, \Sigma \times \{B\}, \Gamma \times \Gamma, \delta', q_0, F)$  with  $\delta'$  as  $\delta'(q_i, [x, y]) = \delta(q_i, [x, y])$ .



# Construct a 3-tape TM to recognize the set $\{a^k \mid k \text{ is a perfect square}\}$



- 1 Tape 1 holds the input, tape 2 holds progressive  $k^2$  (i.e.,  $k$ ,  $kk$ ,  $kkk$ , ...) and tape 3 the progressive  $k$
- 2 keep  $T_2 = T_3 = \varepsilon$ .

- 3 If Input is compared with  $T_2$ , by scanning both. If both reach to  $B$  together, accept, and terminate. Else, put  $a$  at  $T_2$  and  $T_3$ .
- 4 If  $T_2 \equiv T_1$ , accept and terminate, else if  $|T_2| > |T_1|$ , reject input, else ( $T_2 < T_1$ ), then:
  - append  $T_2$  with  $2 \times T_3$  and append  $a$  to  $T_2$  (this changes content of  $T_2$  from  $a^{k^2}$  to  $a^{k^2+2k+1}$ ).
- 5 append  $a$  to  $T_3$  (increment  $t_3$ ).
- 6 go back to step 3.

# Two-way infinite tape

- There is single tape  $M$  which extends from  $-\infty$  to  $+\infty$ . One R-W head,  $M = (Q, \Sigma, \delta, q_0, F)$   
 $\dots -3 -2 -1 \mathbf{0} \mathbf{1} \mathbf{2} \mathbf{3} \dots$ , is square sequence on TM, with R-W head at  $\mathbf{0}$

This can be simulated by a two-tape TM:

- $M' = (Q' \cup \{q_s, q_t\}) \times \{U, D\}, \Sigma', \Gamma', f')$ , where  $U =$  up tape head,  $D =$  down tape head,  $\Sigma' = \Sigma, \Gamma' = \Gamma \cup \{B\}$ , and  $F' = \{[q_i, U], [q_i, D] \mid q_i \in F\}$ . Initial state of  $M'$  is pair  $[q_s, D]$ . A transition from this writes  $B$  in  $U$  tape at left most position. Transition from  $[q_t, D]$  returns the tape head to its original position to begin simulation of  $M$ .

# Multi-Dimensional Tape:

- Single R-W head, but multiple tapes exists. Let the Dimensions be 2D. For each input symbol and state, this writes a symbols at current head position, moves to a new state, and R-W head moves to left or right or up or down.

## Simulate it on 2-tape TM:

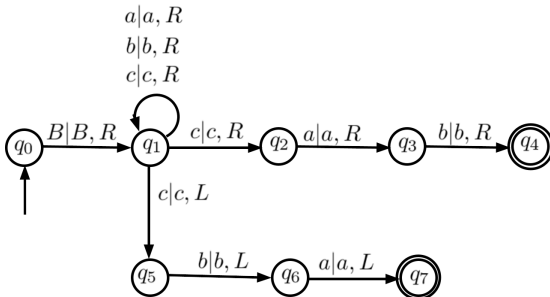
- copy each row of 2-D tape on 2nd tape of 2-tape TM. When 2D TM moves head L or R, move the head on 2nd-tape of two-tape also L or R. When 2D head moves up, 2nd tape of two-tape scans left until it finds \*. As it scans, it writes the symbols on tape-1. Then scans and puts remaining symbols on tape-1. Now it simulates this row (on tape-1).

# Nondeterministic TM (NDTM)

- NDTM has finite number of choices of moves; components are same as standard TM; may have  $> 1$  move with same input and state pair,  $(Q - H \times \Sigma)$ . Nondeterminism is like FA and PDA.
- A NDTM machine accepts by halting if there is at least one computation sequence that halts normally when run with input  $w$ .
- **Example:** Find, if a graph of  $n$  nodes has a connected subgraph of  $k$  nodes. For this, no efficient algorithm exists. A Non-exhaustive based solution is **Guess and check**.
  - 1 **NDTM:** Arbitrarily choose a move when more than one possibility exists for  $\delta(q_i, a)$ .
  - 2 Accept the input if there is at least one computation sequence that leads to accepting state (however, the converse is irrelevant).
- To find a NDTM for  $ww$  input,  $w \in \Sigma^*$ , you need to **guess** the mid point and compare two  $w$ 's.
- A *NDTM* may specify any number of transitions for a given configuration, i.e.  $\delta : (Q - H) \times \Gamma \rightarrow \text{subset of } Q \times \Gamma \times \{L, R\}$

# A NDTM to accept $ab$ preceded or followed with $c$

**Example:**  $w = ucv$ , where  $c$  is preceded by or followed by  $ab$ , and  $u, v \in \{a, b, c\}^*$



**Approach:** Read input  $a, b, c$  and write  $a, b, c$  respectively, and move to  $R$  in each, at start state. Then with input  $c$ , Nondeterministically decide  $c, a, b$  by moving  $R$  in three states transitions or decide  $c, b, a$  by moving  $L$  in three other states transitions (i.e.,  $abc$ )

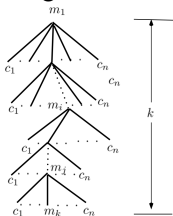
# Simulation of a NDTM on Standard TM

- To accomplish the transformation of a NDTM into a deterministic TM, we show that multiple computations of a single input string can be sequentially generated and examined.
- A NDTM produces multiple computations for a single string. We show that multiple computations  $m_1, \dots, m_i, \dots, m_k$  for a single input string can be sequentially generated and applied. (A computation  $m_i$  is  $\delta(q_i, a) = [q_j, b, d]$ , where  $d \in \{L, R\}$ ).
- These computations can be systematically produced by adding the alternative transitions for each  $Q \times \Sigma$  pair. Each  $m_i$  has  $1 : n$  number of transitions. If  $\delta(q_i, x) = \phi$ , the TM halts.
- Using the ability to sequentially produce the computations, a *NDTM*  $M$  can be simulated by a 3-tape deterministic TM  $M'$ .
- Every nondeterministic TM has an equivalent 3-tape Turing machine, which, in turn, has an equivalent standard TM (1-tape Turing machine).

# Simulation of a NDTM $M$ by 3-tape TM $M'$

## Simulation of NDTM by 3-tape DTM:

- **Approach:** A NDTM may have more than one transition for same input (state  $\times$  input symbol) pair. We may call these configurations as  $c_1 \dots c_n$ .



- If a computation sequence  $m_1, \dots, m_i, \dots, m_k$ , leads to solution, then for a NDTM these are the steps for solution, where

$m_i \in \{c_1, \dots, c_n\}$ . For exhaustive solution, complexity is decided by tree height  $k$ , hence complexity is  $k^n$ .

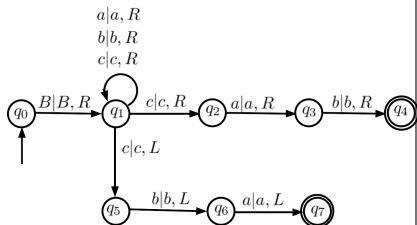
- To perform the simulation of  $M$  on  $M'$ , all the computation sequence, for each (state, input) pair are systematically generated.
- The tree created can be searched in DFS or BFS, until accepting/halting state is reached.
- However, DFS is not preferred because, it may go infinity. Therefore, the generated states are searched in BFS.

# Simulation of a NDTM $M$ by 3-tape TM $m'$

- Tape-1 of  $M'$  stores the input string, tape-2 simulates the tape of  $M$ , and tape-3 holds sequence  $m_1, \dots, m_i, \dots, m_k$  to guide the simulation.
- Computation of  $M'$  consists following:
  - ① A sequence of inputs  $m_1, \dots, m_i, \dots, m_k$ , where each  $i, (i = 1 : n)$  is written on tape-3. ( $m_i \in \{c_1, \dots, c_n\}$ )
  - ② Input string is copied on tape-2.
  - ③ Computation of  $M$  defined by sequence on tape-3 is simulated on tape-2.
  - ④ If simulation halts prior to executing  $k$  transitions, computations of  $M'$  halts and accepts input, **else**
  - ⑤ the Next sequence is generated on tape-3 and computation continues on tape-2.

# Example: Simulation of a NDTM $M$ by 3-tape TM $m'$

Let us consider the NDTM  $M$  given below to be simulated on 3-tape DTM  $M'$ . Let us assume that input string to NDTM is  $w = acab$ .



Since, there are maximum three transitions at any (state, input) pair, we take set  $\{c_1, c_2, c_3\}$  as  $\{1, 2, 3\}$ . In case there is single transition only, we repeat that to make that count equal to 3. The following table shows all pairs of  $(q \times \sigma)$ , where  $\sigma \in \Sigma$ .

$\delta(q_0, B)$	$\begin{matrix} (1)q_1, B, R \\ (2)q_1, B, R \\ (3)q_1, B, R \end{matrix}$	$\delta(q_2, a)$	$\begin{matrix} (1)q_3, a, R \\ (2)q_3, a, R \\ (3)q_3, a, R \end{matrix}$
$\delta(q_1, a)$	$\begin{matrix} (1)q_1, a, R \\ (2)q_1, a, R \\ (3)q_1, a, R \end{matrix}$	$\delta(q_1, c)$	$\begin{matrix} (1)q_1, c, R \\ (2)q_2, c, R \\ (3)q_5, c, R \end{matrix}$
$\delta(q_3, b)$	$\begin{matrix} (1)q_4, b, R \\ (2)q_4, b, R \\ (3)q_4, b, R \end{matrix}$	$\delta(q_5, b)$	$\begin{matrix} (1)q_6, b, R \\ (2)q_6, b, R \\ (3)q_6, b, R \end{matrix}$
...	...	...	...

# Simulation of a NDTM $M$ by 3-tape TM $m'$

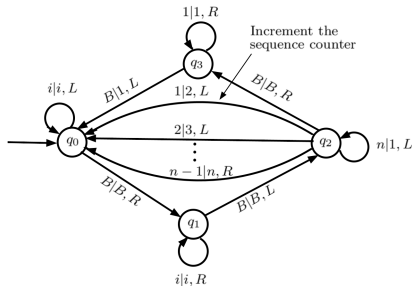
To simulate the NDTM on a 3-tape TM, following  $m_1, \dots, m_i, \dots, m_k$  sequences are possible. All possible sequences are enumerated. The sequence numbers are given as per the table, given in the previous slide. Each sequence indicates a transition from one ID to next ID of the 3-tape DTM.

$q_0BacabB$ 1	$q_0BacabB$ 1	$q_0BacabB$ 2
$\vdash Bq_1acabB$ 1	$\vdash Bq_1acabB$ 1	$\vdash Bq_1acabB$ 2
$\vdash Baq_1cabB$ 1	$\vdash Baq_1cabB$ 2	$\vdash Baq_1cabB$ 3
$\vdash Bacq_1abB$ 1	$\vdash Bacq_2abB$ 1	$\vdash Bq_5acabB$
$\vdash Bacaq_1bB$ 1	$\vdash Bacaq_3bB$ 1	
$\vdash Bacabq_1B$	$\vdash Bacabq_4B$	
sequence=	sequence=	sequence
(1,1,1,1,1)	(1,1,2,1,1)	(2,2,3)

Therefore, what is required to generate the sequences as above: (1,1,1,1,1), then (1,1,2,1,1), then (2,2,3), in lexicographic order. We note, that the sequence (1,1,2,1,1) is accepting sequence; the moment such generated sequence of configurations is simulated on tape-2, the machine is halted.

# Simulation of a NDTM: Generation of lexicographic sequences

Sequence generator is a TM, with no input, but output - sequence, whose generation is interleaved with the computation of the simulation of NDTM on tape-2 of  $M'$ . The figure below shows the sequence generation.



The simulation of NDTM  $M$  on a 3-tape DTM  $M'$  is as follows: (1) Input string  $w$  is put on tape 1, (2)  $w$  is copied on tape 2, (3) next sequence  $(m_1..m_k)$  is generated on tape 3, (4) tape 2 is simulated for the moves (sequence) available on tape 3. (5) If any move leads to halting state, the machine is halted and  $w$  is accepted. (6) process is repeated from step 2. Note that sequence generation is non-terminating. Hence, if  $w \notin L$ , the process will continue indefinitely.

# Turing Machine as language enumerator

- TM can also be designed to enumerate a language. Such machines produce the exhaustive list of string of the language, progressively longer and longer.
- Enumeration has no input, and its computation continues indefinitely if the language is infinite, as well as when it is finite.
- For enumeration, a TM of tape  $k \geq 2$  is used, tape 1 is output tape, which would hold all the generated strings, separated by #, and other tapes are working tapes.
- Output tape 1 has:  $B\#u_1\#u_2\# \dots \#u_i\# \dots$ , where  $u_i \in L$ .
- Tape head 1 always moves R, S, while others may move R, L, S.

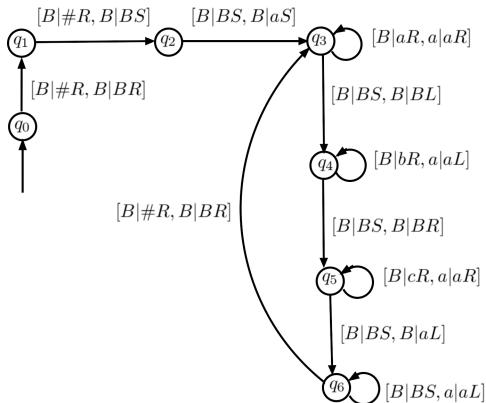
## Example

Enumerate all the strings for  $L = \{a^n b^n c^n \mid n \geq 0\}$

*Solution.* The  $L$  can be generated by two-tape TM  $E$  with following steps: (1) Write ## on tape 1 and 2 for  $\epsilon$ . (2) add  $a$  on tape 2, (3) copy  $a$  equal to size of tape 2 on tape 1, then by same size the  $b$  is written on tape 1, then by same size  $c$  is written tape 1, followed with # (4) goto step 2.

# Turing Machine as language enumerator

The language  $L = \{a^n b^n c^n \mid n \geq 0\}$  is suppose recognized by TM  $M$ . Let the enumerator  $E$  generates all the strings of  $L$ .



# Turing Machine as language enumerator

## Theorem

*If  $L$  is enumerated by a TM then  $L$  is recursively enumerable.*

## Proof.

*Let  $L$  is enumerated by a TM  $E$  of  $k$  tapes. We add a tape new tape  $k+1$  to it. Let this new machine is  $M$ . Consider a string  $w \in L$ , and we write it on  $k+1$ th tape. Every time  $\#$  is written on tape 1 by  $E$ , and its starts generating new string  $u_i$ , simultaneously it is compared with  $w$  on tape  $k+1$ , if found equal,  $M$  halts, otherwise the process of generating and compare is repeated for next string.*

*Since, when  $w \in L$  machine  $M$  halts else continues indefinitely,  $L$  is recursively enumerable. □*

*For enumeration it is necessary that lexicographic order (lo) of strings is generated. We can generate all the strings on alphabet  $\Sigma = \{a_1, \dots, a_n\}$  in lexicographic order using recursion as follows:*

$$lo(\epsilon) = 0, lo(a_i) = i, i = 1, n$$

$$lo(a_i u) = i.n^{\text{lenth}(n)} + lo(u)$$

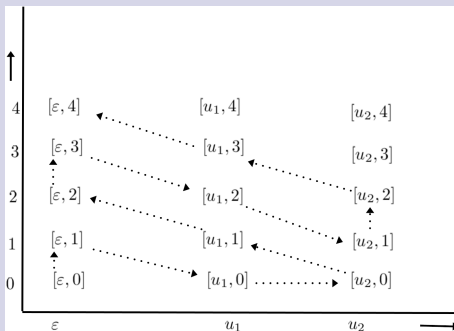
# Turing Machine as language enumerator

## Theorem

For any alphabet  $\Sigma$ , there is TM  $E_{\Sigma^*}$  that enumerates  $\Sigma^*$  in lexicographic order.

## Proof.

Let  $M$  be the TM that accepts  $L$ . The lexicographic ordering produces listing:  $\varepsilon, u_1, u_2, \dots, \in \Sigma^*$ . We Construct a table with columns as strings in  $\Sigma^*$  and rows as natural numbers, in the order as shown.



# Turing Machine as language enumerator . . .

The  $[i, j]$  entry in the table means “run the  $M$  for input  $u_i$  for  $j$  steps. The machine  $E$  is built to enumerate  $L$  such that enumeration of the ordered pairs are interleaved with the computation of  $M$ . The computation of  $E$  is a loop:

- 1 generate an ordered pair  $[i, j]$
- 2 run a simulation of  $M$  with input  $u_i$  for  $j$  transitions or until the simulation halts.
- 3 If  $M$  accepts, write  $u_i$  on the output tape
- 4 continue with step 1.

If  $u_i \in L$ , the computation of  $M$  with input  $u_i$  halts and accepts after  $k$  transitions, for some number  $k$ .

Thus,  $u_i$  will be written on output tape of  $E$  when ordered pair  $[i, j]$  is processed. The 2nd element of  $k$  ensures that simulation is terminated after  $k$  steps. Consequently, no non-terminating computation are allowed, and each string of  $\Sigma^*$  is examined.

This is one more proof of the theorem : "If a language is enumerated by TM then it is RE".



Chowdhary, K.R. (2025). Extensions of Basic Turing Machine. In: Theory of Computation. Springer, Singapore.

[https://doi.org/10.1007/978-981-97-6234-7\\_10](https://doi.org/10.1007/978-981-97-6234-7_10)