

# Rice's Theorem

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

Formerly at department of Computer Science and Engineering  
MBM Engineering College, Jodhpur

Thursday 13<sup>th</sup> November, 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.

Formally, reduction from  $P$  to  $Q$  is a TM that takes as instance of  $P$  on its tape and halts with an instance of  $Q$  written on its tape.

## Theorem

*If there is a reduction from  $P$  to  $Q$ , then (1) if  $P$  is undecidable then so  $Q$ , (2) if  $P$  is non-R.E, the so is  $Q$ .*

## Proof.

*(1) Suppose  $P$  is undecidable. If it is possible to decide  $Q$ , then we can combine the reduction from  $P$  to  $Q$  with the algorithm that decides  $Q$  to construct an algorithm that decides  $P$ . If given an instance  $p_1$  of  $P$ , apply algorithm to convert it to an instance  $r(p_1)$  ( $r$  is reduction TM) of  $Q$ , then apply the algorithm that decides  $Q$  to decide  $r(p_1)$ . If that algorithm says "yes", then  $P$  is decidable. The sequential execution of reducer TM and algorithm (i.e., TM)  $Q$ , solves  $P$ . (2) can be proved in same lines as (1). □*

# TM that accepts empty languages

- Let  $L_e$  (empty language) and  $L_{ne}$  (not empty language) are languages based on  $\{0,1\}^*$  strings. If  $L(M_i) = \emptyset$ ,  $M_i$  does not accept any input. However,  $w \in L_e$ . We define  $L_e$  as language consisting of strings of all TMs whose language is empty
- If  $L(M_i)$  is not empty language, then  $w$  is in  $L_{ne}$ .  $L_{ne}$  is language of all the codes of TM that accept at least one string.

$$L_e = \{R\langle M \rangle \mid L(M) = \emptyset\}$$

$$L_{ne} = \{R\langle M \rangle \mid L(M) \neq \emptyset\}$$

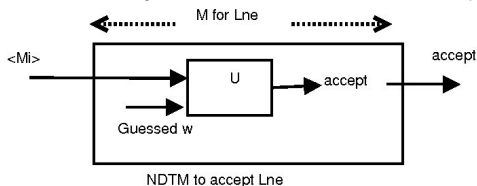
- we will show that  $L_{ne}$  is RE but not Recursive.  $L_e$  is non-RE.

## Theorem

$L_{ne}$  is RE.

## Proof.

- We have only to show that a TM  $M$  accepts  $L_{ne}$ .



- The operation of  $M$  is:  $M$  takes input  $\langle M_i \rangle$ ; guesses input  $w$  which  $M_i$  might accept;  $M$  tests whether  $M_i$  accepts  $w$ . For this  $M_i$  simulates UTM  $U$ . If  $M_i$  accepts  $w$ , then  $M$  accepts its own input  $\langle M_i \rangle$ . Hence, if  $M_i$  accepts even one string,  $M$  accepts  $M_i$ .  $\therefore$ ,  $L(M) = L_{ne}$ , and  $L_{ne}$  is RE since it is accepted by TM.  $\square$

$\square$

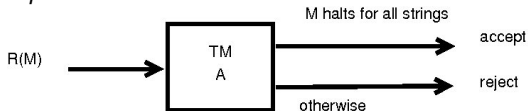
# An arbitrary TM halts for all inputs is undecidable.

## Theorem

*An arbitrary TM halts for all inputs is undecidable*

## Proof.

- *Let TM A halts for input  $v \in \{0,1\}^*$ . Input is accepted if  $v = R(M)$  for some TM M, that halts for all the inputs strings. Input is rejected if either v is not the representation of a TM or it is representation for some TM that does not halt for some input string.*



*continued ...*

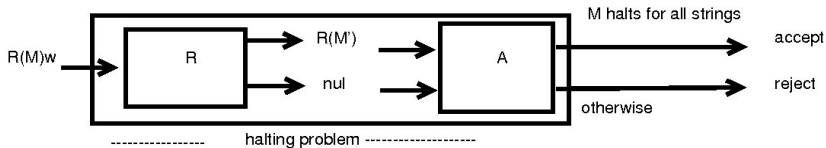


# An arbitrary TM halts for all inputs is undecidable ...

- **Reduction:** is used to create a solution to the halting problem from machine A.
- First action of reduction R is find out if i/p has format of R(M), followed by w. If no, R erases i/p and leaves tape blank.
- When i/p is in correct format, (R(M)w), R constructs the R(M') of M', that when run with i/p y:-
  - erases y from tape, writes w on tape, and runs M on w.

∴, R(M') = erase original input of tape + write w on tape + R(M).  
(R(M') is designed to ignore the i/p w).
- M' halts iff computation of M with i/p w halts.

# An arbitrary TM halts for all inputs is undecidable ...



- If  $i/p$  is not of the form  $R(M)w$ ,  $R$  produces null  $o/p$ , which is rejected by  $A$ . Otherwise  $R$  generates  $R(M')$ .  $\therefore$ ,  $i/p$  is accepted iff it is of the form  $R(M)w$ , where  $M$  halts on input  $w$ . Since halting problem is undecidable, there is no TM  $A$  that solves the halt for all strings  $w$ .  $\square$

# Non-trivial properties

- Rice's theorem after Henry Rice, states that if  $S$  is a non-trivial property of Turing-recognizable languages, then the problem, Given a  $TM M$ , "does  $L(M)$  have the property  $S$ ?" is undecidable.
- An example of a property of Turing-recognizable languages: the property of being a regular language, is undecidable.
- A property  $S$  of Turing-recognizable languages is non-trivial if some recognizable languages have the property and others do not.
- **Definition:** Let  $S$  be a property of Turing-recognizable languages. We say that  $S$  is **non-trivial** if there exist Turing-recognizable languages  $L_1$  and  $L_2$  such that  $L_1 \in S$  but  $L_2 \notin S$ . If a property of Turing-recognizable languages is not non-trivial, we call it trivial.
- **Example 1:** The class of languages

$$S_{REGULAR} = \{L \mid L \text{ is regular}\}$$

is a non-trivial property of Turing-recognizable languages, since the language  $L_1 = \{a^n \mid n \geq 0\}$  is a member of  $S_{REGULAR}$  (and therefore also recognizable, as  $S_{REGULAR}$  is a property), whereas the language  $L_2 = \{a^n b^n \mid n \geq 0\}$  is also recognizable, but not a member of  $S_{REGULAR}$ , as it is a non-regular language.

# Language problem

- **Example 2:** There are only two ways of violation the conditions for being non-trivial. The properties

$$S_{all} = \{L | L \text{ is Turing-recognizable}\} \quad (1)$$

$$S_{none} = \emptyset \quad (2)$$

are both trivial, and they are the only such properties.  $S_{all}$  is trivial, since there does not exist a Turing-recognizable  $L_2$  such that  $L_2 \notin S_{all}$ .  $S_{none}$  is trivial, since there does not exist a Turing-recognizable  $L_1$  such that  $L_1 \notin S_{none}$ .

- We know that there is no algorithm that decides whether a TM halts for all the inputs  $w$ .  $\therefore L_{\Sigma^*}$  **is undecidable.**
- Language of property  $P$  is,  $L_P = \{R(M) | L(M) \text{ satisfies the property } P\}$ . In fact  $L_{\varepsilon}, L_{\emptyset}, L_{reg}, L_{\Sigma^*}$  are language properties.
- A property  $P$  of  $RE$  set is trivial if: there are no  $RE$  sets  $L(M)$  (i.e., empty) satisfying  $P$ , or every  $RE$  set  $L(M)$  (i.e., all  $R(TM)$ ) satisfies  $P$ . Otherwise it is nontrivial.
- **Rice's theorem:** Any property that is satisfied by some (not all, and not empty) sets is undecidable. I.e., any nontrivial of a language accepted Turing machine is undecidable.

# Examples of non-trivial properties

- Let  $L = \{ \text{all ASCII character sequences } s_1, s_2, \dots \mid s_i \text{ is a C program } c_i \text{ with the property that each } c_i \text{ when run on input } x \in \{0, 1\}^* \text{ accepts iff } x \text{ is Regexp} \}$
- Rice's theorem says that languages such as  $L$  above are not decidable, i.e., it is impossible to classify all C programs (or equivalently TMs) into those whose languages are *regular* and those whose languages are *not regular*.
- Mathematically, given the property  $P$ , let

$$L = \{ \langle M \rangle \mid M \text{ is TM and } P(\text{Lang}(M)) \}$$

**Example 3:** The following are examples of non-trivial properties of Turing-recognizable languages:

- The class  $\{ \emptyset \}$  whose only member is the empty language (not to be confused with the empty property  $S_{\text{none}}$ )
- The class  $\{ \Sigma^* \}$  whose only member is the language of all strings over alphabet  $\Sigma$ .
- The class of languages that have the empty string  $\epsilon$  as an element.
- The class of context-free languages
- The class of Turing-decidable languages

# Examples of non-trivial properties

- **Definition:** Let  $S$  be a property. Then the language  $S_{TM}$  is defined as

$$S_{TM} = \{ \langle M \rangle \mid L(M) \in S \}.$$

The result that we shall now state and prove, states that  $S_{TM}$  is undecidable whenever  $S$  is non-trivial.

- **Example** Consider the property  $S_{REGULAR}$  defined above. Then,

$$S_{TM} = \{ \langle M \rangle \mid L(M) \text{ is a regular language} \}$$

- We show that if  $S$  is a non-trivial property of Turing-recognizable languages, then we can reduce  $A_{TM}$  to  $S_{TM}$ .
- In other words, we show that, given  $\langle M, w \rangle$  we can construct the description of a machine a  $\langle M' \rangle$  such that  $M$  accepts  $w$  if and only if  $L(M') \in S$ .

## Theorem

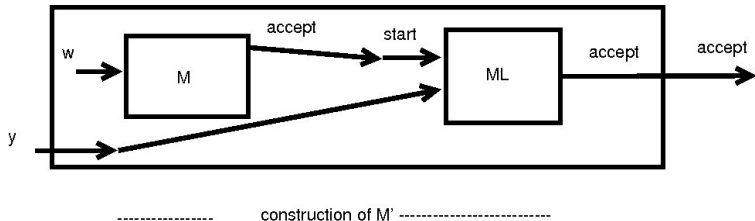
*If  $P$  is nontrivial property of of r.e. language then  $L_P$  is non recursive(Undecidable)*

## Proof.

- *Let  $P$  is nontrivial property that is not satisfied by empty language. We will show that  $L_P = \{R(M) \mid L(M) \text{ satisfies } P\}$  is not recursive.*
  - *Since,  $L_P$  is nontrivial,  $\exists L, L \in L_P$ .  $L$  cannot be  $\emptyset$ . Let TM  $M_L$  accepts  $L$ .*
  - *we design  $R$  (reducer) to transform  $R(M)w$  to  $R(M')$ . The  $M'$  runs on  $y$  as:*
    - *write  $w$  to right of  $y$ , producing  $ByBwB$ ;*
    - *run  $M$  on  $w$ ; (acts as gate keeper), (1st tm)*
    - *if  $M$  halts when run with  $w$ , then run  $M_L$  with i/p  $y$ . (2nd tm).*
- Contd. next slide . . .*



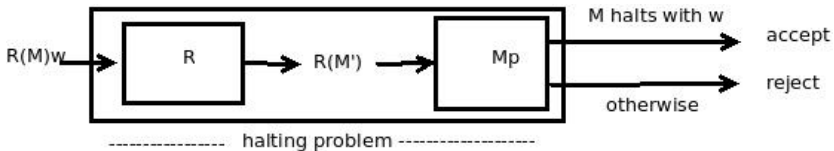
## Rice's Theorem proof continued ...



- In this case result of computation of  $M'$  with i/p  $y$  is exactly that of computation by  $M_L$  with  $y$ .  $\therefore$ ,  $L(M') = L(M_L) = L$ , and  $L(M')$  satisfies  $P$ .
- If  $M$  does not halt with i/p  $w$ , then  $M'$  never halts regardless of  $y$ .  $\therefore$ , no string is accepted by  $M'$ , and  $L(M') = \emptyset$ . (does not satisfy  $P$ )
- $\therefore$ ,  $M'$  accepts  $\emptyset$ , when  $M$  does not halt with i/p  $w$ , and  $M'$  accepts  $L$  when  $M$  halts with  $w$ . Since  $L$  satisfies  $P$  and  $\emptyset$  does not,  $L(M')$  satisfies  $P$  iff  $M$  halts when run with i/p  $w$ .

# Rice's Theorem proof continued ...

- Assume that  $L_P$  is recursive. Then there is a machine  $M_P$  that decides membership in  $L_P$ . The machines  $R$  and  $M_P$  combine to produce a solution to halting problem.  $\therefore$ , the property  $P$  is not decidable.



□

# Problem of determining whether a language accepted by TM is CFL, is undecidable

- By Rice's theorem, it is sufficient to show that property "is CF", is nontrivial property of *RE* language. It is accomplished by one *RE* language that is CF and another not. The  $\emptyset$  and  $\{a^i b^i c^i \mid i \geq 0\}$  are both *RE* and former as CF and latter as not.
- Applications of Rice's theorem: We now have any number of undecidable questions about TM's:
  - Is  $L(M)$  a regular language?
  - Is  $L(M)$  a CFL?
  - Does  $L(M)$  include any palindromes?
  - Is  $L(M)$  empty?
  - Does  $L(M)$  contain more than 1000 strings? Etc., etc.

# Some Conclusions from Rice's Theorem

- Undecidability: whether a TM accepts a particular string;
- whether a TM recognizes a particular language
- whether a languages recognized by TM can be recognized by a simpler machine, like FA.

*Rice's theorem does not say any thing about those properties of functions or programs, which are not properties functions or languages.* E.g., whether the machine will run for 100 steps; whether machine has five or more states; no general purpose computer can solve general problem of determining whether or not a program is virus free - is undecidable.



Chowdhary, K.R. (2025). Decidability, Undecidability, and Unsolvability. In: Theory of Computation. Springer, Singapore.  
[https://doi.org/10.1007/978-981-97-6234-7\\_12](https://doi.org/10.1007/978-981-97-6234-7_12)