

# Lec 18. Reduction and undecidable languages I

Eunjung Kim

# HOW TO SHOW THAT A LANGUAGE IS UNDECIDABLE

- 1 Diagonal argument: used to show that  $A_{TM}$  is undecidable.

# HOW TO SHOW THAT A LANGUAGE IS UNDECIDABLE

- 1 Diagonal argument: used to show that  $A_{TM}$  is undecidable.
- 2 Reduction, a.k.a. Turing-reduction:

## $A$ TURING-REDUCIBLE TO $B$

We say that  $A$  is **Turing-reducible** to  $B$  if there is a TM deciding  $A$  which uses a (hypothetical) TM deciding  $B$ .

Observe: Undecidability of  $A$  implies the undecidability of  $B$ . (" $B$  is as hard as  $A$ ")

# HOW TO SHOW THAT A LANGUAGE IS UNDECIDABLE

- 1 Diagonal argument: used to show that  $A_{TM}$  is undecidable.
- 2 Reduction, a.k.a. Turing-reduction:

## A TURING-REDUCIBLE TO B

We say that  $A$  is **Turing-reducible** to  $B$  if there is a TM deciding  $A$  which uses a (hypothetical) TM deciding  $B$ .

Observe: Undecidability of  $A$  implies the undecidability of  $B$ . (" $B$  is as hard as  $A$ ")

- 3 We shall later see a third technique called **mapping-reduction** (a.k.a many-one reduction).

# SOME TERMINOLOGY IN CLASS / TEXT

We say a TM  $D$

- **runs** a TM  $M$  on  $w$  if  $M$  is a subroutine of  $D$ ; it is hardwired in  $D$ .
- **simulates** a TM  $M$  on  $w$  if  $\langle M \rangle$  is written on the tape of  $D$  and  $D$  has a universal TM as hardwired subroutine. In this case  $D$  is **running** the universal TM on  $\langle M \rangle$  and  $\langle w \rangle$ .

# HALTING PROBLEM IS UNDECIDABLE

Halting problem:  $\text{HALT}_{TM} = \{\langle M, w \rangle \mid M \text{ is TM and } M \text{ halts on } w\}$ .

FROM UNDECIDABILITY OF  $A_{TM}$ , WE DERIVE

$\text{HALT}_{TM}$  is undecidable.

# HALTING PROBLEM IS UNDECIDABLE

Halting problem:  $\text{HALT}_{TM} = \{\langle M, w \rangle \mid M \text{ is TM and } M \text{ halts on } w\}$ .

FROM UNDECIDABILITY OF  $A_{TM}$ , WE DERIVE

$\text{HALT}_{TM}$  is undecidable.

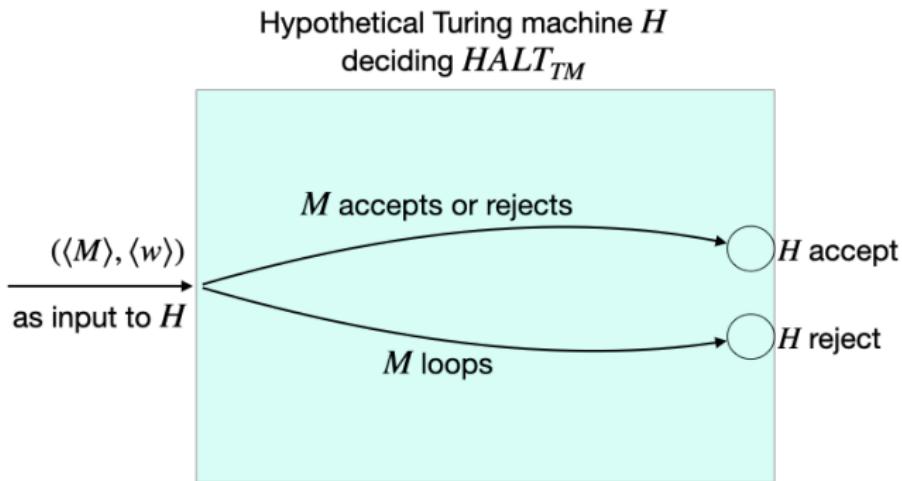
Proof: we crucially use that  $A_{TM} = \{\langle M, w \rangle \mid M \text{ is TM and } M \text{ accepts } w\}$  is undecidable.

# HALTING PROBLEM IS UNDECIDABLE

Suppose the contrary; let  $H$  be a decider TM for  $\text{HALT}_{TM}$ .

# HALTING PROBLEM IS UNDECIDABLE

Suppose the contrary; let  $H$  be a decider TM for  $\text{HALT}_{TM}$ .



# HALTING PROBLEM IS UNDECIDABLE

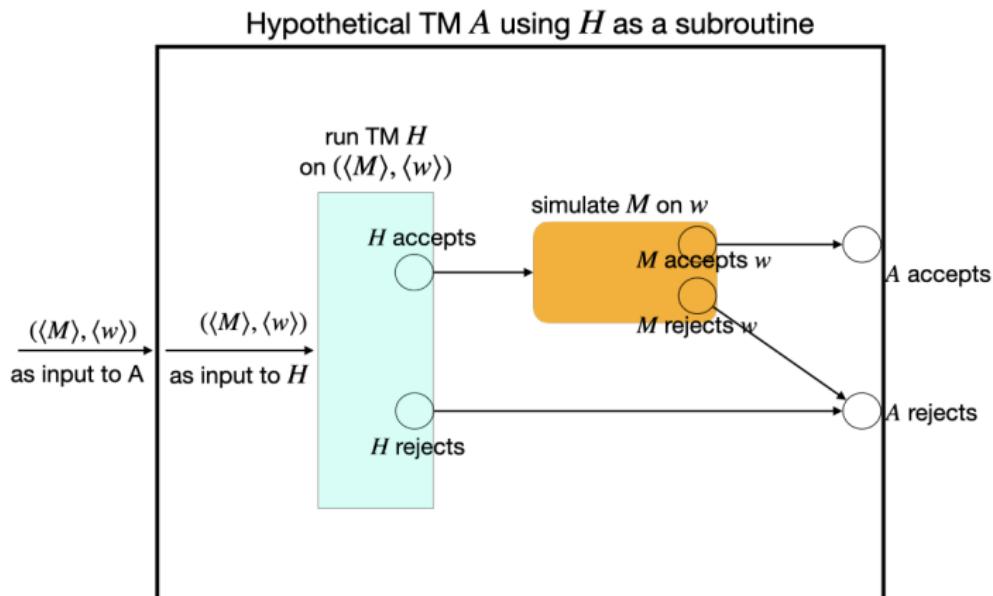
Suppose the contrary; let  $H$  be a decider TM for  $\text{HALT}_{TM}$ .

Using the hypothetical decider  $H$  for  $\text{HALT}_{TM}$  as a subroutine, we can build a decider  $A$  for  $A_{TM}$  as follows (Turing-reduction):

# HALTING PROBLEM IS UNDECIDABLE

Suppose the contrary; let  $H$  be a decider TM for  $\text{HALT}_{TM}$ .

Using the hypothetical decider  $H$  for  $\text{HALT}_{TM}$  as a subroutine, we can build a decider  $A$  for  $A_{TM}$  as follows (Turing-reduction):



# HALTING PROBLEM IS UNDECIDABLE

Using the hypothetical decider  $H$  for  $\text{HALT}_{TM}$ , we can build a decider  $A$  for  $A_{TM}$  as follows:

## DECIDER $A$ FOR $A_{TM}$ USING $H$ AS A SUBROUTINE

Decider  $A$  upon input  $\langle M, w \rangle$

- 1 Run  $H$  on the string  $\langle M, w \rangle$ .
- 2 if  $H$  accepts  $\langle M, w \rangle$  (i.e.  $M$  halts on  $w$ ), then  $A$  simulates  $M$  on  $w$  and outputs

$$A(\langle M, w \rangle) = \begin{cases} \text{YES} & \text{if } M(w) = \text{YES} \\ \text{No} & \text{if } M(w) = \text{No} \end{cases}$$

- 3 if  $H$  rejects  $\langle M, w \rangle$  (i.e.  $M$  loops on  $w$ ), then  $A$  rejects  $\langle M, w \rangle$ .

# HALTING PROBLEM IS UNDECIDABLE

$A$  is a decider.

- 1 case 1. when  $M$  accepts  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs YES.

# HALTING PROBLEM IS UNDECIDABLE

$A$  is a decider.

- 1 case 1. when  $M$  accepts  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs YES.
- 2 case 2. when  $M$  rejects  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs No.

# HALTING PROBLEM IS UNDECIDABLE

$A$  is a decider.

- 1 case 1. when  $M$  accepts  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs YES.
- 2 case 2. when  $M$  rejects  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs No.
- 3 case 3. when  $M$  loops on  $w$ : then  $H$  rejects  $(M, w)$ . By design of  $A$ ,  $A$  outputs No.

# HALTING PROBLEM IS UNDECIDABLE

$A$  is a decider.

- 1 case 1. when  $M$  accepts  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs YES.
- 2 case 2. when  $M$  rejects  $w$ : then  $H$  accepts  $(M, w)$ , meaning that  $M$  halts on  $w$ . Therefore, when  $A$  simulates  $M$  on  $w$ ,  $A$  also halts. By design of  $A$ ,  $A$  outputs No.
- 3 case 3. when  $M$  loops on  $w$ : then  $H$  rejects  $(M, w)$ . By design of  $A$ ,  $A$  outputs No.

$$L(A) = \{(M, w) \mid M \text{ accepts } w\} = A_{TM}.$$

This contradicts that  $A_{TM}$  is undecidable.

# EMPTINESS PROBLEM IS UNDECIDABLE

Emptiness problem:  $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$ .

## UNDECIDABILITY OF $E_{TM}$

$E_{TM}$  is undecidable.

# EMPTINESS PROBLEM IS UNDECIDABLE

Emptiness problem:  $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$ .

## UNDECIDABILITY OF $E_{TM}$

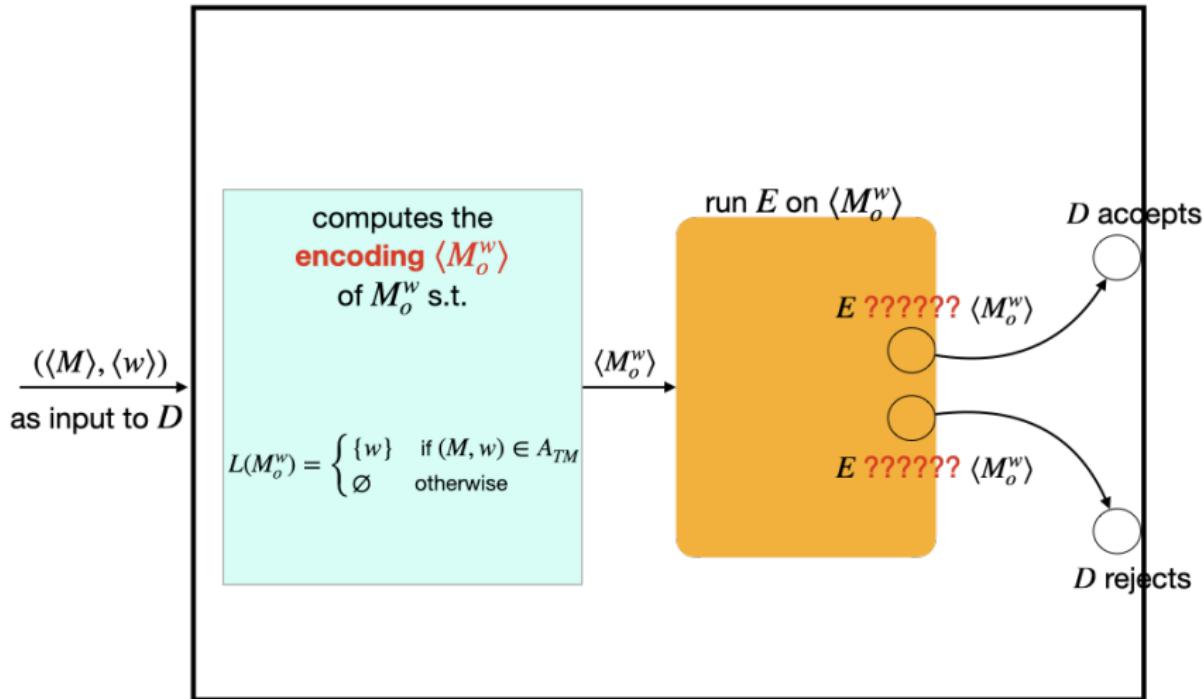
$E_{TM}$  is undecidable.

Proof outline: Turing-reduction from  $A_{TM}$  to  $E_{TM}$ .

- Prove that if  $E_{TM}$  has a decider  $E$ , then this can be used (as a subroutine) to build a decider  $D$  for  $A_{TM} = \{(M, w) : M \text{ is TM and } M \text{ accepts } w\}$ .
- This contradicts the undecidability of  $A_{TM}$ .

# EMPTINESS PROBLEM IS UNDECIDABLE

Hypothetical TM  $D$  using  $E$  as a subroutine



# EMPTINESS PROBLEM IS UNDECIDABLE

Emptiness problem:  $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$ .

## UNDECIDABILITY OF $E_{TM}$

$E_{TM}$  is undecidable.

- $D$  upon an input  $\langle M, w \rangle$  does the following:
  - 1 Compute an **encoding** of a TM  $M_o^w$  such that

$$L(M_o^w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

- 2 Run  $E$  on  $\langle M_o^w \rangle$ .
- 3  $D$  outputs

$$\begin{cases} ?????????? & \text{if } E(\langle M_o^w \rangle) = \text{YES} \\ ?????????? & \text{if } E(\langle M_o^w \rangle) = \text{NO} \end{cases}$$

- How to design (compute) such TM  $M_o^w$ ?

# EMPTINESS PROBLEM IS UNDECIDABLE

We want  $D$  to compute (i.e. write the encoding of) another TM  $M_o^w$  such that

$$L(M_o^w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

How does a TM  $M_o^w$  work internally? And how can a TM output its encoding?

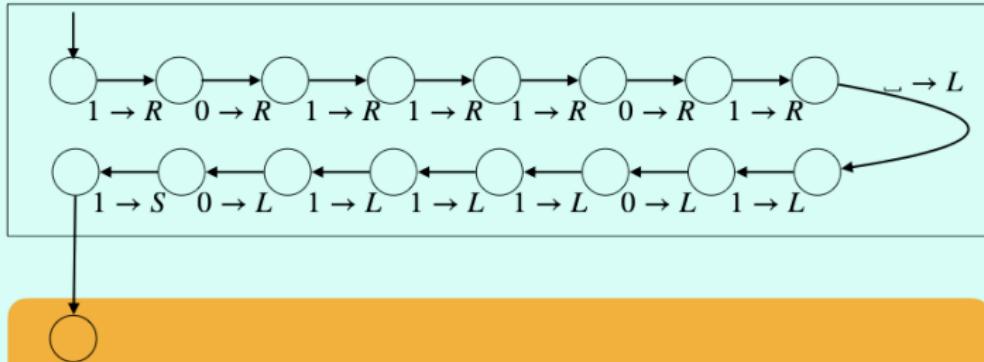
TM  $M_o^w$  does the following on input string  $x$ :

$$M_o^w(x) = \begin{cases} \text{YES} & \text{if } x = w \text{ and } M \text{ accepts } w \\ \text{No or loops on } x & \text{otherwise} \end{cases}$$

$M_o^w$  may loop on  $w$  (this happens when  $M$  loops on  $w$ ). But it's fine for the purpose of obtaining a desired decider  $D$  as  $D$  does not simulate  $M_o^w$  **but only computes its encoding  $\langle M_o^w \rangle$** .

# EMPTINESS PROBLEM IS UNDECIDABLE

TM  $M_o^w$  with a hardwired string  $w = 1011101$



Replicate the TM  $M$  here

# NON-EMPTINESS IS UNDECIDABLE

$SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$ .

## UNDECIDABILITY OF $SOME_{TM}$ FROM THAT OF $E_{TM}$

$SOME_{TM}$  is undecidable.

- Key idea: (Turing-)reduce from  $E_{TM}$  to  $SOME_{TM}$  by designing a decider  $D$  of  $E_{TM}$  using the hypothetical TM  $S$  deciding  $SOME_{TM}$ .
- $D$  upon an input  $\langle M \rangle$  does the following:
  - 1 Check if  $\langle M \rangle$  is a legit encoding of a TM.
  - 2 Run  $S$  on  $\langle M \rangle$ .
  - 3  $D$  outputs

$$D(\langle M \rangle) = \begin{cases} ??????? & \text{if } S(\langle M \rangle) = \text{YES} \\ ??????? & \text{if } S(\langle M \rangle) = \text{NO} \end{cases}$$

# $REG_{TM}$ IS UNDECIDABLE

$REG_{TM} = \{M : M \text{ is TM and } L(M) \text{ is a regular language}\}.$

## UNDECIDABILITY OF $REG_{TM}$

$REG_{TM}$  is undecidable.

- Key idea: reduction from  $A_{TM}$  to  $REG_{TM}$ . Suppose that  $R$  is a decider for  $REG_{TM}$ .
- $D$ , upon an input  $\langle M, w \rangle$ , does the following:

- 1 Compute & write an encoding  $\langle M_o^{M,w} \rangle$  of TM  $M_o^{M,w}$  such that

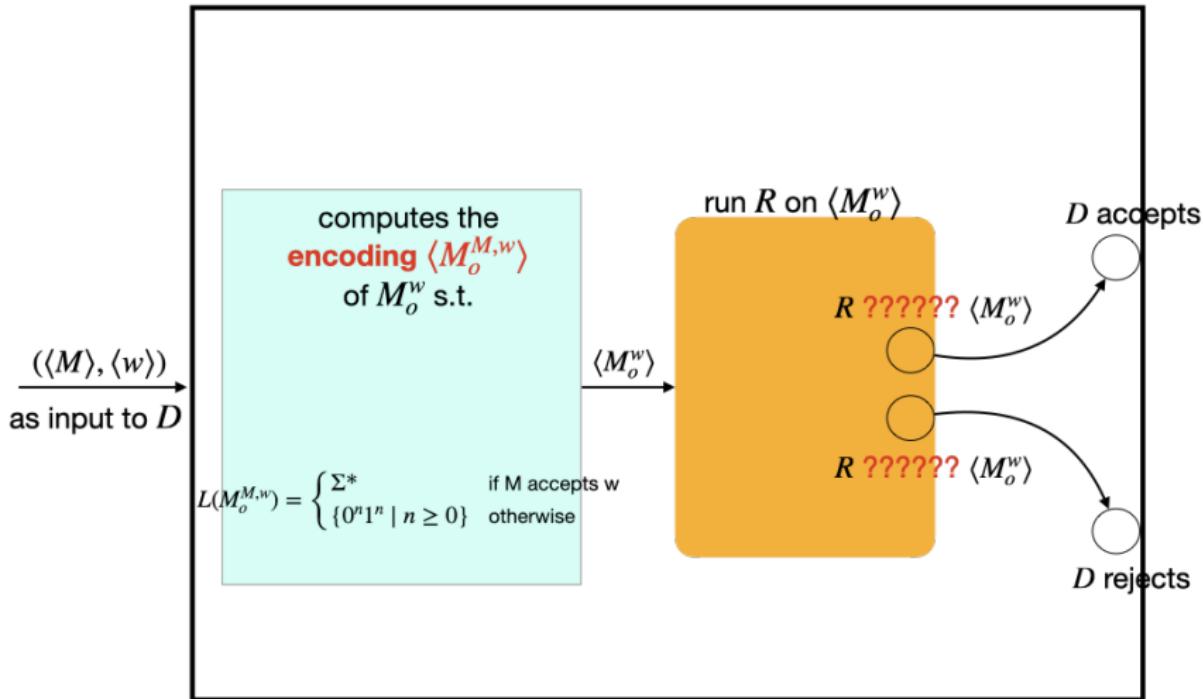
$$L(M_o^{M,w}) = \begin{cases} \{0^n 1^n \mid n \geq 0\} & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$$

- 2 Run  $R$  on  $\langle M_o^{M,w} \rangle$ .
- 3  $D$  outputs

$$\begin{cases} \text{YES} & \text{if } R \text{ outputs YES} \\ \text{No} & \text{if } R \text{ outputs No} \end{cases}$$

# EMPTINESS PROBLEM IS UNDECIDABLE

Hypothetical TM  $D$  using  $R$  as a subroutine



# $REG_{TM}$ IS UNDECIDABLE

We want  $D$  to compute the encoding of  $M_o^{M,w}$  such that

$$L(M_o^{M,w}) = \begin{cases} \{0^n1^n \mid n \geq 0\} & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$$

How does a TM  $M_o^{M,w}$  work internally?

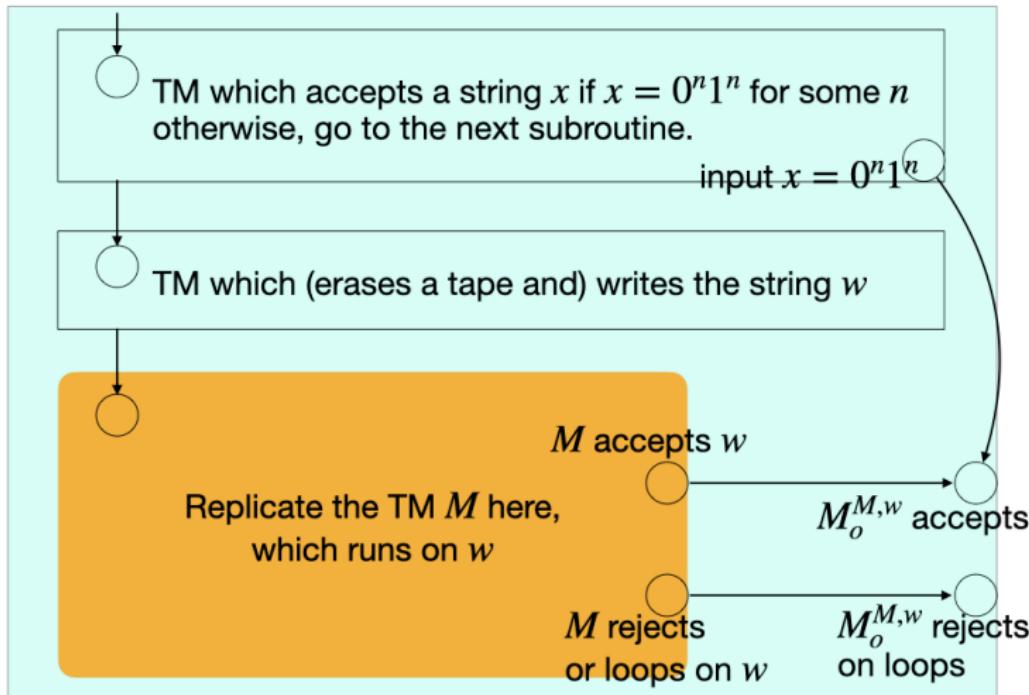
TM  $M_o^{M,w}$  does the following on input string  $x$ :

$$M_o^{M,w}(x) = \begin{cases} \text{YES} & \text{if } x \text{ is of the form } 0^n1^n \text{ for some } n \geq 0 \\ & \quad \text{or } M \text{ accepts } w \\ \text{No or loop} & \text{otherwise} \end{cases}$$

$M_o^{M,w}$  may loop on  $w$ . But it's fine to obtain a decider  $D$  as  $D$  does not simulate  $M_o^{M,w}$  but **only computes its encoding**.

# EMPTINESS PROBLEM IS UNDECIDABLE

TM  $M_o^{M,w}$  with a hardwired string  $w = 1011101$   
and using  $M$  as a subroutine



# LINEAR BOUNDED AUTOMATON

## LINEAR BOUNDED AUTOMATON

A linear bounded automaton is a Turing machine with the following restriction: its header is not allowed to move off the portion of the (single) tape containing the input. When the TM instructs the header to move to the right of the right-end of the input, then it stays where it is.

Key observation: For an input of length  $n$ , a linear bounded automaton on  $w$  can go through at most

$$|Q| \cdot n \cdot |\Gamma|^n$$

distinct configurations. This means

## HALTING PROBLEM FOR LBA

A linear bounded automaton  $M$  halts on an input  $w$  of length  $n$  if it halts in the first  $|Q| \cdot n \cdot |\Gamma|^n$  steps. Does the converse hold?