CS 302.1 - Automata Theory

Lecture 12

Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)

Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

Quick Recap

Recursive Language/Turing Decidable/Decidable: A language L is called Recursive or Turing decidable or Decidable if there exists a Total Turing Machine M and

$$\left.\begin{array}{l} \forall \omega \in L, M(\omega) \text{ accepts} \\ \forall \omega \notin L, M(\omega) \text{ rejects} \end{array}\right\} \text{Halts on all inputs}$$

The Church Turing thesis: An algorithm can be written for a problem if and only if it is decidable, i.e. there exists a Total Turing machine that solves the problem. **Total TM ⇔ Algorithms!**

Recursively Enumerable Language/Turing Recognizable (RE): A language L is called Recursively Enumerable (RE) or Turing Recognizable if

$$\forall \omega \in L, M(\omega) \textbf{ accepts}$$
$$\forall \omega \notin L, M(\omega) \textbf{ doesn't accept} \qquad \text{(rejects or runs infinitely)}$$

Co-Recursively Enumerable Language/co-Turing Recognizable (Co-RE/$\overline{RE}$/nRE): A language L is **Co-Recursively Enumerable (co-RE/$\overline{RE}$) or Co-Turing Recognizable** if

$$\forall \omega \in L, M(\omega) \textbf{ doesn't reject} \qquad \text{(accepts or loops)}$$
$$\forall \omega \notin L, M(\omega) \textbf{ rejects}$$

Quick Recap

There exists a one-one mapping (bijective relationship) between the set of binary strings and TMs.

Universal Turing Machine: A Universal Turing Machine, denoted as U_{TM} accepts as input (i) the encoding of a Turing Machine M and (ii) an input string w and **simulates M running on w**, i.e.

$$U_{TM}(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \textbf{REJECTS, if } M(w) \textbf{ rejects} \\ \textbf{LOOPS INFINITELY, if } M(w) \textbf{ loops infinitely} \end{cases}$$

Quick Recap

There exists a one-one mapping (bijective relationship) between the set of binary strings and TMs.

Universal Turing Machine: A Universal Turing Machine, denoted as U_{TM} accepts as input (i) the encoding of a Turing Machine M and (ii) an input string w and **simulates M running on w**, i.e.

$$U_{TM}(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \textbf{REJECTS, if } M(w) \textbf{ rejects} \\ \textbf{LOOPS INFINITELY, if } M(w) \textbf{ loops infinitely} \end{cases}$$

Some examples of languages that are recursive/decidable:

- $A_{DFA} = \{\langle DFA \rangle, w | w \in L(DFA)\}$
- $E_{DFA} = \{\langle DFA \rangle | L(DFA) = \Phi\}$
- $A_{CFG} = \{\langle CFG, w \rangle | w \in L(CFG)\}$
- $E_{CFG} = \{\langle CFG \rangle | L(CFG) = \Phi\}$

An undecidable language:

- $A_{TM} = \{\langle M, w \rangle | M \textbf{ accepts input } w\}$

Quick Recap

There exists a one-one mapping (bijective relationship) between the set of binary strings and TMs.
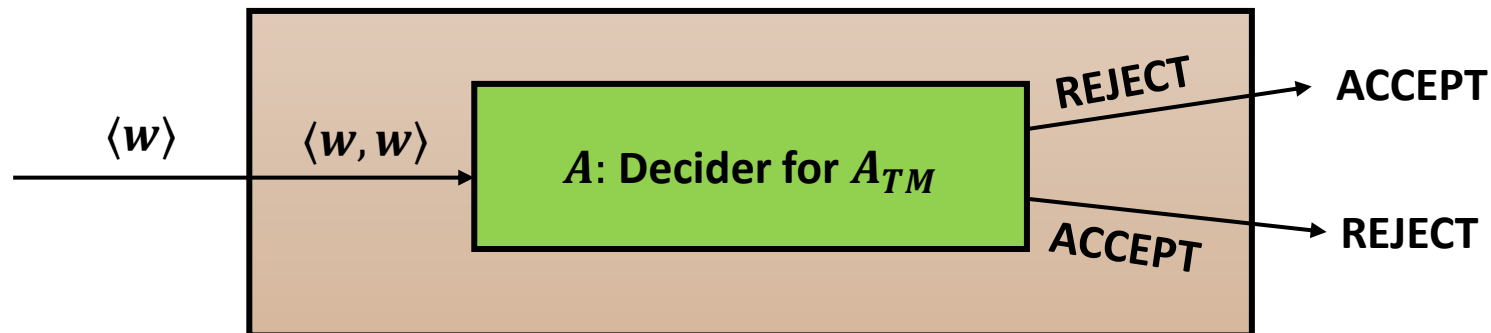
An undecidable language:

- $A_{TM} = \{\langle M, w\rangle | M \text{ accepts input } w\}$

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

Proof strategy: By contradiction. We assume that a Total TM A exists that decides A_{TM}.
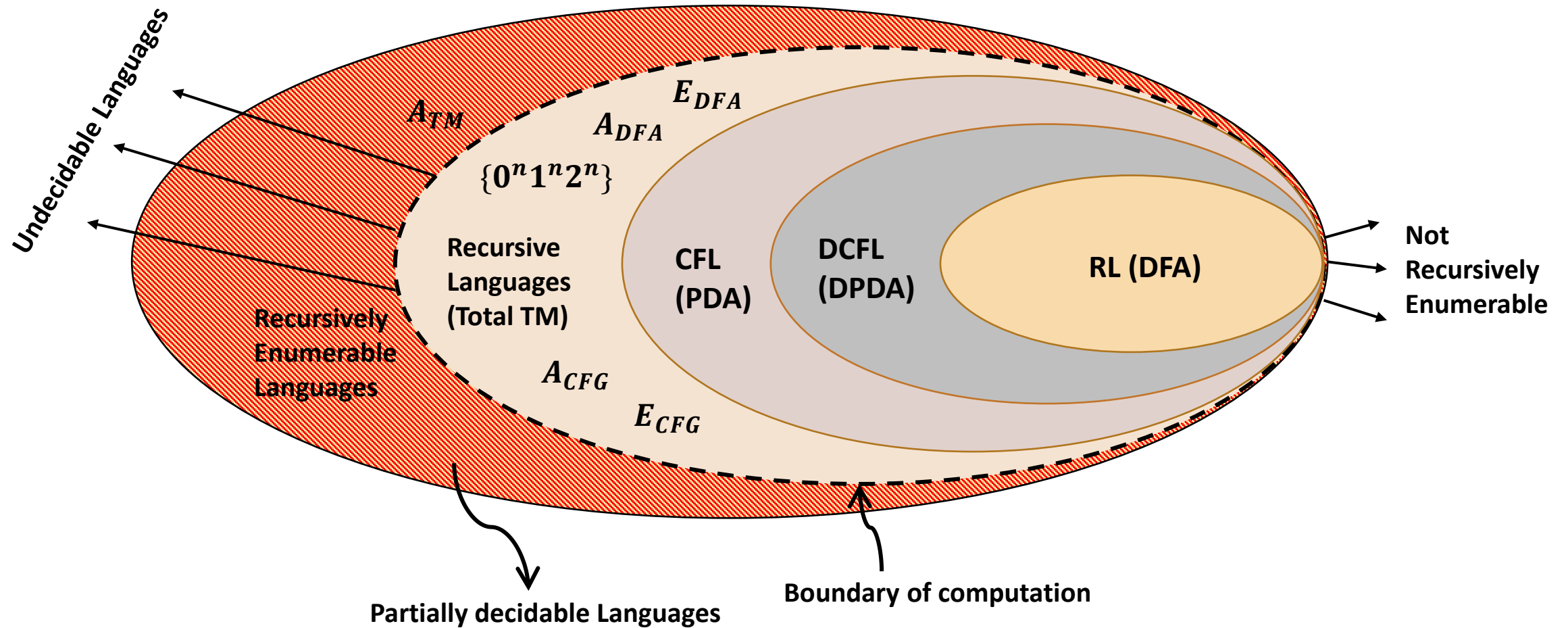
We build a Total TM D that accepts w as input and calls $A(\langle w, w\rangle)$ as a subroutine and outputs the opposite of A.

The TM D cannot decide the input $w = \langle D\rangle$. So A is undecidable.

For $w = \langle D\rangle$, there is a contradiction!

Quick Recap

Undecidable Languages

A_{TM}

E_{DFA}

A_{DFA}

$\{0^n 1^n 2^n\}$

Recursive Languages (Total TM)

CFL (PDA)

DCFL (DPDA)

RL (DFA)

Recursively Enumerable Languages

A_{CFG}

E_{CFG}

Not Recursively Enumerable

Partially decidable Languages

Boundary of computation

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M \text{ accepts input } w\}$. A_{TM} **is undecidable**

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\[2em] \textbf{REJECTS, if } M(w) \textbf{ rejects or loops infinitely} \end{cases}$$

The proof uses a technique called **Diagonalization**.

First, recall that there exists a bijective map (one-one correspondence) between the set of all binary strings and Turing Machines.

We can list all the Turing Machines and write down the result of running any M_i on input $\langle M_j \rangle$.

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M$ accepts input $w\}$. A_{TM} is undecidable

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ rejects or loops infinitely} \end{cases}$$

The proof uses a technique called **Diagonalization**.

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	Accept	Accept	Loops	Reject	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_2	Reject	Loops	Accept	Loops	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

First, recall that there exists a bijective map (one-one correspondence) between the set of all binary strings and Turing Machines.

We can list all the Turing Machines and write down the result of running any M_i on input $\langle M_j \rangle$.

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M \text{ accepts input } w\}$. A_{TM} is undecidable

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ rejects or loops infinitely} \end{cases}$$

The proof uses a technique called **Diagonalization**.

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	Accept	Accept	Loops	Reject	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_2	Reject	Loops	Accept	Loops	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

How would this Table look for A?

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M \text{ accepts input } w\}. \ A_{TM}$ **is undecidable**

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ rejects or loops infinitely} \end{cases}$$

- A_{TM} **is undecidable**
- $A_{TM} \in RE$ **but not recursive**
- A_{TM} **is partially decidable**

The proof uses a technique called **Diagonalization**.

How would this Table look for A?

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	Accept	Accept	Loops	Reject	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_2	Reject	Loops	Accept	Loops	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

A	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	Accept	Accept	**Reject**	Reject	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_2	Reject	**Reject**	Accept	**Reject**	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M \text{ accepts input } w\}$. A_{TM} is undecidable

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

The proof uses a technique called **Diagonalization**.

A	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_0	Accept	Accept	**Reject**	Reject	Accept	$\cdots$	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$	Accept	$\cdots$
M_2	Reject	**Reject**	Accept	**Reject**	Accept	$\cdots$	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
D						$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

$D(w) = \{$ **Run** $A(\langle w, w \rangle)$

If $A(\langle w, w \rangle)$ **accepts, then** $REJECT$
If $A(\langle w, w \rangle)$ **rejects, then** $ACCEPT$

$\}$

$$D(\langle M_w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M_w(\langle M_w \rangle) \textbf{ doesn't accept} \\ \\ \textbf{REJECTS, if } M_w(\langle M_w \rangle) \textbf{ accepts} \end{cases}$$

- Somewhere we will also have the TM D.
- Note that D by definition **computes the opposite of the diagonal entries** of the table.

An undecidable problem

$A_{TM} = \{\langle M, w\rangle | M \text{ accepts input } w\}$. A_{TM} is undecidable

The proof uses a technique called **Diagonalization**.

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

Note that D by definition **computes the opposite of the diagonal entries** of the table.

A	$\langle M_0\rangle$	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$	$\cdots$
M_0	**Accept**	Accept	Reject	Reject	Accept	$\cdots$	Accept	$\cdots$
M_1	Accept	Reject	Reject	Accept	Reject	$\cdots$	Accept	$\cdots$
M_2	Reject	Reject	Accept	Reject	Accept	$\cdots$	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
D	**Reject**					$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

An undecidable problem

$A_{TM} = \{\langle M, w \rangle | M \text{ accepts input } w\}. A_{TM}$ **is undecidable**

The proof uses a technique called **Diagonalization**.

- A_{TM} **is undecidable**
- $A_{TM} \in RE$ **but not recursive**
- A_{TM} **is partially decidable**

Note that D by definition **computes the opposite of the diagonal entries** of the table.

A	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_0	**Accept**	Accept	Reject	Reject	Accept	$\cdots$	Accept	$\cdots$
M_1	Accept	**Reject**	Reject	Accept	Reject	$\cdots$	Accept	$\cdots$
M_2	Reject	Reject	Accept	Reject	Accept	$\cdots$	Accept	$\cdots$
M_3	Accept	Reject	Reject	Accept	Reject	$\cdots$	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	Reject	$\cdots$	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
D	**Reject**	**Accept**				$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

An undecidable problem

$A_{TM} = \{\langle M, w\rangle | M \text{ accepts input } w\}. A_{TM}$ **is undecidable**

The proof uses a technique called **Diagonalization**.

A	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_0	**Accept**	Accept	Reject	Reject	Accept	$\cdots$	Accept	$\cdots$
M_1	Accept	**Reject**	Reject	Accept	Reject	$\cdots$	Accept	$\cdots$
M_2	Reject	Reject	**Accept**	Reject	Accept	$\cdots$	Accept	$\cdots$
M_3	Accept	Reject	Reject	**Accept**	Reject	$\cdots$	Reject	$\cdots$
M_4	Accept	Accept	Accept	Accept	**Reject**	$\cdots$	Reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
D	**Reject**	**Accept**	**Reject**	**Reject**	**Accept**	$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

- A_{TM} **is undecidable**
- $A_{TM} \in RE$ **but not recursive**
- A_{TM} **is partially decidable**

Note that D by definition **computes the opposite of the diagonal entries** of the table.

What will be the Dth diagonal entry??

An undecidable problem

$A_{TM} = \{\langle M, w \rangle \mid M$ **accepts input** $w\}$. A_{TM} **is undecidable**

The proof uses a technique called **Diagonalization**.

- A_{TM} is undecidable
- $A_{TM} \in RE$ but not recursive
- A_{TM} is partially decidable

Note that D by definition **computes the opposite of the diagonal entries** of the table.

A	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$	...
M_0	**Accept**	Accept	Reject	Reject	Accept	...	Accept	...
M_1	Accept	**Reject**	Reject	Accept	Reject	...	Accept	...
M_2	Reject	Reject	**Accept**	Reject	Accept	...	Accept	...
M_3	Accept	Reject	Reject	**Accept**	Reject	...	Reject	...
M_4	Accept	Accept	Accept	Accept	**Reject**	...	Reject	...
⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	...
D	**Reject**	**Accept**	**Reject**	**Reject**	**Accept**	...	**??**	...
⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	...

What will be the Dth diagonal entry??

Contradiction!

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$**. Is** $HALT_{TM}$ **decidable?**

The Halting Problem: Does there exist a Total Turing Machine H that accepts as input a Turing Machine M and an input string w and outputs YES, if $M(w)$ halts (accepts or rejects) and NO, if $M(w)$ does not halt (loops forever), i.e.

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPTS, if } M(w) \text{ HALTS, i.e. accepts or rejects} \\ \\ \text{REJECTS, if } M(w) \text{ does not HALT, i.e. loops infinitely} \end{cases}$$

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M \text{ halts on input } w\}$. Is $HALT_{TM}$ decidable?

The Halting Problem: Does there exist a Total Turing Machine H that accepts as input a Turing Machine M and an input string w and outputs YES, if $M(w)$ halts (accepts or rejects) and NO, if $M(w)$ does not halt (loops forever), i.e.

$$H(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ HALTS, i.e. accepts or rejects} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ does not HALT, i.e. loops infinitely} \end{cases}$$

- Turing stated the Halting problem and demonstrated its undecidability in his famous 1936 paper.
- This provided a negative answer to Hilbert's Eintscheidungsproblem.
- **Proof strategy:** We will try to show that if we had such a Total TM H, we would be able to build a Total TM for A_{TM}
- But since we proved that A_{TM} is undecidable, this would mean that **H is undecidable**.

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M \text{ halts on input } w\}$. Is $HALT_{TM}$ decidable?

$\langle M, w \rangle \longrightarrow \boxed{H} \longrightarrow ACCEPT$, if $M(w)$ halts

$\longrightarrow REJECT$, if $M(w)$ loops

Proof idea: We first assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM}

$\langle M, w \rangle \longrightarrow \boxed{A} \longrightarrow ACCEPT$, if $M(w)$ accepts

$\longrightarrow REJECT$, if $M(w)$ rejects or loops

But A cannot be Total and so a total TM that decides H cannot exist

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$. **Is** $HALT_{TM}$ **decidable?**

$\langle M, w \rangle \rightarrow \boxed{H} \rightarrow$ **ACCEPT**, if $M(w)$ halts

REJECT, if $M(w)$ loops

Proof idea: We first assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM}

Outlining the steps for building A using H:

- A calls $H(\langle M, w \rangle)$
- If H rejects, then we know that $M(w)$ loops forever and so A would output $REJECT$.

$\langle M, w \rangle \rightarrow \boxed{A} \rightarrow$ **ACCEPT**, if $M(w)$ accepts
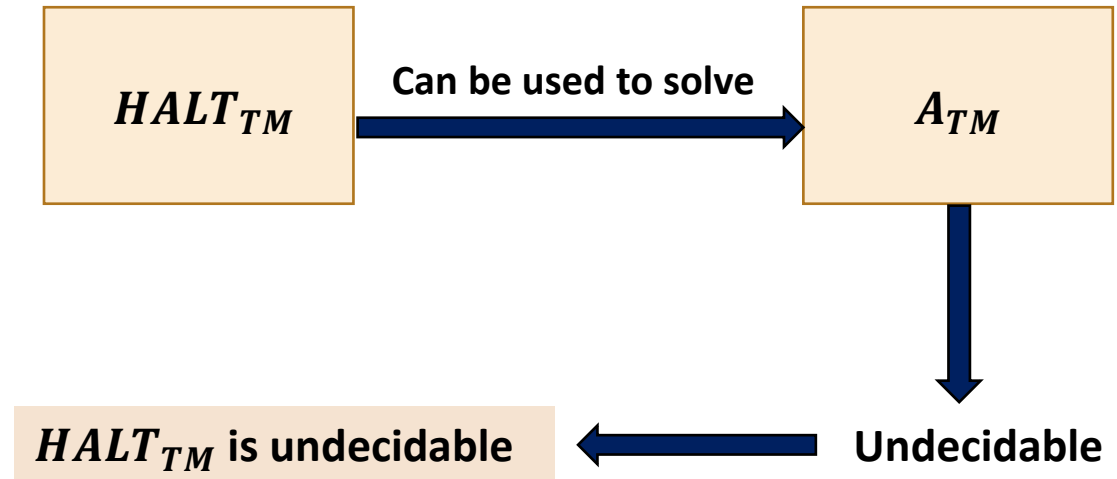
REJECT, if $M(w)$ rejects or loops

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$. **Is** $HALT_{TM}$ **decidable?**

$\langle M, w \rangle \rightarrow \boxed{H} \rightarrow ACCEPT$, if $M(w)$ halts

$\rightarrow REJECT$, if $M(w)$ loops

Proof idea: We first assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM}

Outlining the steps for building A using H:

- A calls $H(\langle M, w \rangle)$
- If H rejects, then we know that $M(w)$ loops forever and so A would output $REJECT$.
- If H accepts,
  - $M(w)$ surely halts (either accepts or rejects).

$\langle M, w \rangle \rightarrow \boxed{A} \rightarrow ACCEPT$, if $M(w)$ accepts

$\rightarrow REJECT$, if $M(w)$ rejects or loops

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$**. Is** $HALT_{TM}$ **decidable?**

$\langle M, w \rangle \rightarrow \boxed{H}$

$ACCEPT$, if $M(w)$ halts

$REJECT$, if $M(w)$ loops

Proof idea: We first assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM}

Outlining the steps for building A using H:

- A calls $H(\langle M, w \rangle)$
- If H rejects, then we know that $M(w)$ loops forever and so A would output $REJECT$.
- If H accepts,
  - $M(w)$ surely halts (either accepts or rejects).
  - Simply run $M(w)$ and
    - $ACCEPT$ if $M(w)$ accepts
    - $REJECT$ if $M(w)$ rejects

$\langle M, w \rangle \rightarrow \boxed{A}$

$ACCEPT$, if $M(w)$ accepts

$REJECT$, if $M(w)$ rejects or loops

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$. **Is** $HALT_{TM}$ **decidable?**

$$H(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ HALTS, i.e. accepts or rejects} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ does not HALT, i.e. loops infinitely} \end{cases}$$

Proof: Assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM} as follows:

$A =$ On input $\langle M, w \rangle$
       Run $H(\langle M, w \rangle)$
      If H rejects, output $REJECT$
      If H accepts,
            Run $M(w)$
         If $M(w)$ accepts, output $ACCEPT$
         If $M(w)$ rejects, output $REJECT$

https://www.udiprod.com/halting-problem/

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ **halts on input** $w\}$. **Is** $HALT_{TM}$ **decidable?**

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPTS, if } M(w) \text{ HALTS, i.e. accepts or rejects} \\ \\ \text{REJECTS, if } M(w) \text{ does not HALT, i.e. loops infinitely} \end{cases}$$

Proof: Assume that there exists such a Total Turing Machine H. Then, we use H as a subroutine to construct a Total Turing Machine A for A_{TM} as follows:

$A =$ On input $\langle M, w \rangle$
       Run $H(\langle M, w \rangle)$
      If H rejects, output $REJECT$
      If H accepts,
            Run $M(w)$
          If $M(w)$ accepts, output $ACCEPT$
          If $M(w)$ rejects, output $REJECT$

$HALT_{TM}$ **Can be used to solve** → A_{TM}

↓ **Undecidable**

$HALT_{TM}$ **is undecidable** ←

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ halts on input $w\}$. Is $HALT_{TM}$ decidable?

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPTS, if } M(w) \text{ HALTS, i.e. accepts or rejects} \\ \\ \text{REJECTS, if } M(w) \text{ does not HALT, i.e. loops infinitely} \end{cases}$$

$HALT_{TM} \in RE$ as H halts whenever M accepts or rejects w and so

$Q = $ On input $\langle M, w \rangle$:
- Simulate M on input w
- If M accepts w, $ACCEPT$; if M rejects w, $ACCEPT$

Q recognizes $HALT_{TM}$

The Halting problem

$HALT_{TM} = \{\langle M, w \rangle | M$ halts on input $w\}$. Is $HALT_{TM}$ decidable?

$$H(\langle M, w \rangle) = \begin{cases} \text{ACCEPTS, if } M(w) \text{ HALTS, i.e. accepts or rejects} \\ \\ \text{REJECTS, if } M(w) \text{ does not HALT, i.e. loops infinitely} \end{cases}$$

$HALT_{TM} \in RE$ as H halts whenever M accepts or rejects w and so

$Q = $ On input $\langle M, w \rangle$:
- Simulate M on input w
- If M accepts w, $ACCEPT$; if M rejects w, $ACCEPT$

Q recognizes $HALT_{TM}$

- $HALT_{TM}$ is undecidable
- $HALT_{TM} \in RE$ but not recursive
- $HALT_{TM}$ is partially decidable
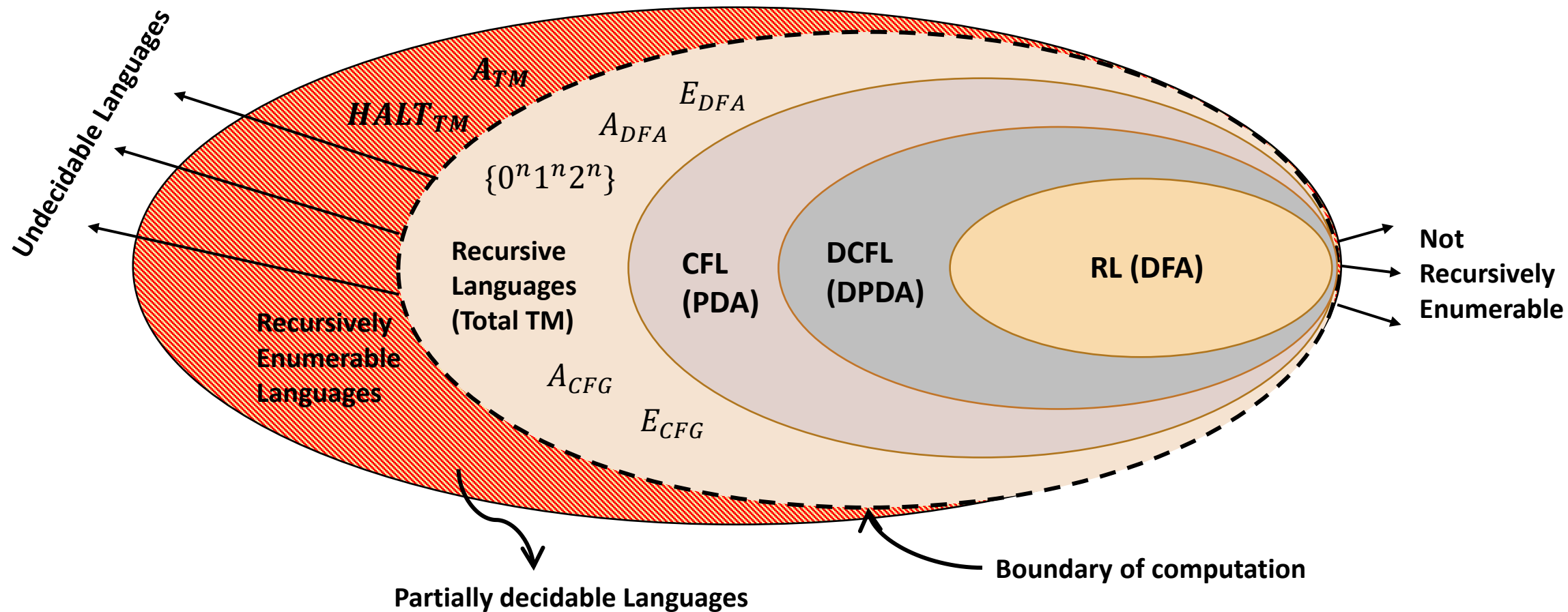
Reduction

Recall the proof of the undecidability of the Halting Problem

What did we do there?

- We used a (supposed) decider for the Halting Problem to build a decider for A_{TM}.
- This established that $HALT_{TM}$ can be used to solve A_{TM}.
- As A_{TM} is undecidable, this established that $HALT_{TM}$ is undecidable too.
- The key underlying concept is an idea called **Reduction.**
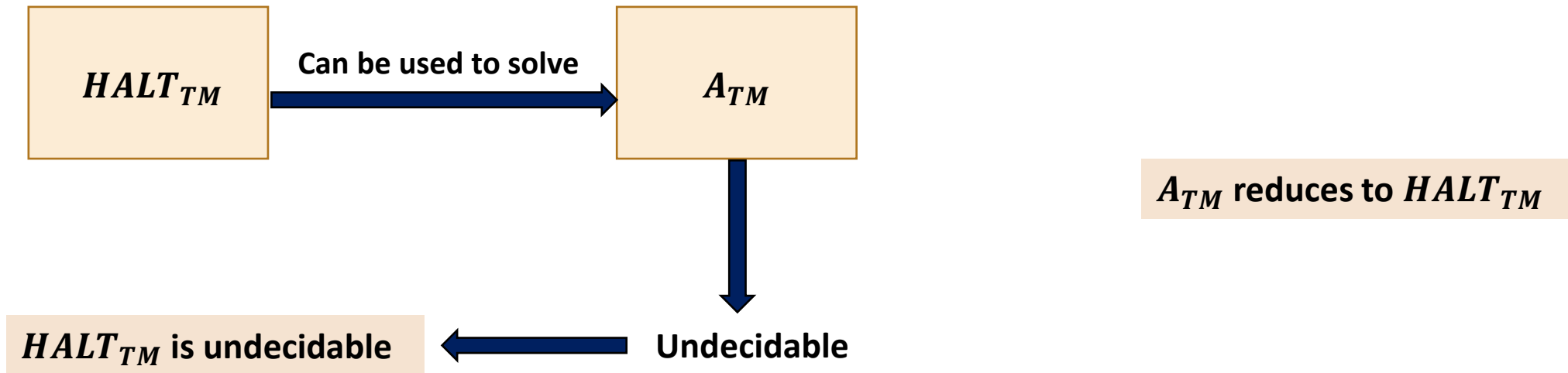
$$HALT_{TM} \xrightarrow{\textbf{Can be used to solve}} A_{TM}$$

A_{TM} **reduces to** $HALT_{TM}$

$HALT_{TM}$ **is undecidable** $\leftarrow$ **Undecidable**

Reduction

Generally,

- A language A **reduces to** another language **B** ($A \leqslant B$) iff we can build a **solver for** A using a **solver for** B

- In terms of computability, suppose using B we can compute A. Then, if A **is undecidable then so is** B.

- So, in the last proof we showed: $A_{TM} \leqslant HALT_{TM}$ to prove that $HALT_{TM}$ is undecidable.

Reduction

Generally,

- A language A **reduces to** another language B ($A \leqslant B$) iff we can build a solver for A using **solver for B**.
.
- In terms of computability, suppose using B we can compute A. Then, if A **is undecidable then so is B**.

- So, in the last proof we showed: $A_{TM} \leqslant HALT_{TM}$ to prove that $HALT_{TM}$ is **undecidable**.

- This is a common technique to show that certain problems are decidable/undecidable.

Suppose, $A \leqslant B$ and

- A is undecidable. Then B is undecidable.

For example, we can prove that a problem P is undecidable by reducing the Halting problem to P.

$$A_{TM} \leqslant HALT_{TM}$$

Reduction

Generally,

- A language A **reduces to** another language B ($A \leqslant B$) iff we can build a solver for A using **solver for B**.
- In terms of computability, suppose using B we can compute A. Then, if A **is undecidable then so is B**.
- So, in the last proof we showed: $A_{TM} \leqslant HALT_{TM}$ to prove that $HALT_{TM}$ is **undecidable**.
- This is a common technique to show that certain problems are decidable/undecidable.

Suppose, $A \leqslant B$ and

- A is undecidable. Then B is undecidable.
- **B is decidable. Then A is decidable.**

For example, we can prove that a problem P is undecidable by reducing the Halting problem to P.

$$A_{TM} \leqslant HALT_{TM}$$

If there existed a decider for $HALT_{TM}$, then we would have a decider for A_{TM}.

Reduction

- A language A **reduces to** another language **B** ($A \leqslant B$) iff we can build a solver for A using **solver for B**.
.
- In terms of computability, suppose using B we can compute A. Then, if A **is undecidable then so is B**.

- So, in the last proof we showed: $A_{TM} \leqslant HALT_{TM}$ to prove that $HALT_{TM}$ is undecidable.

- If **B is decidable, then so is A.**

- This is a common technique to show that certain problems are decidable/undecidable.

We can prove that several problems are undecidable by the **reduction to some known undecidable problem.**

Examples:

- $E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle | M_1$ and M_2 are Turing Machines having $L(M_1) = L(M_2)\}$
- $ALL_{TM} = \{\langle M \rangle | M$ halts on all inputs$\}$
  $\vdots$

Reduction

We can prove that several problems are undecidable by the **reduction of these problems** to some known undecidable problem.

Examples:

- $E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle | M_1$ and M_2 are Turing Machines having $L(M_1) = L(M_2)\}$
- $ALL_{TM} = \{\langle M \rangle | M$ halts on all inputs$\}$

$$\vdots$$

Generic strategy for proof:

- Consider that a Total TM for the given problem exists (say T_M).
- Show that can build a total TM that can decide some undecidable problem (e.g. $HALT_{TM}$, A_{TM}) using T_M as a subroutine.

This reduction would prove that $\boldsymbol{E_{TM}}, \boldsymbol{EQ_{TM}}, \boldsymbol{ALL_{TM}}$ **are undecidable**.

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

What is $\overline{A_{TM}}$?

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

What is $\overline{A_{TM}}$?

$A_{TM} = \{\langle M, w \rangle | M$ **accepts input** $w\}$

$\overline{A_{TM}} = \{\langle M, w \rangle | M$ **DOES NOT accept input** $w\}$

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

What is $\overline{A_{TM}}$?

$A_{TM} = \{\langle M, w \rangle | M$ accepts input $w\}$

$\overline{A_{TM}} = \{\langle M, w \rangle | M$ DOES NOT accept input $w\}$

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ rejects/loops} \end{cases}$$

$$N(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ rejects/loops} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ accepts} \end{cases}$$

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using T_E.

What is $\overline{A_{TM}}$?

$\overline{A_{TM}} \in$ co-RE$- R$.

$A_{TM} = \{\langle M, w \rangle | M$ accepts input $w\}$

$\overline{A_{TM}} = \{\langle M, w \rangle | M$ DOES NOT accept input $w\}$

$$A(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ accepts} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ rejects/loops} \end{cases}$$

$$N(\langle M, w \rangle) = \begin{cases} \textbf{ACCEPTS, if } M(w) \textbf{ rejects/loops} \\ \\ \textbf{REJECTS, if } M(w) \textbf{ accepts} \end{cases}$$

Undecidability

$E_{TM} = \{\langle M\rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

Key idea:

- $N(\langle M, w\rangle)$ first builds the encoding of a TM $T_{\langle M,w\rangle}$
- $T_{\langle M,w\rangle}$ does not accept any string if and only if M does not accept w.
- That is, running $T_{\langle M,w\rangle}(x)$ on **ANY** input x, runs M on w.

What does this achieve??

Undecidability

$E_{TM} = \{\langle M \rangle | M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using T_E.

Key idea:

- $N(\langle M, w \rangle)$ first builds the encoding of a TM $T_{\langle M,w \rangle}$
- $T_{\langle M,w \rangle}$ does not accept any string if and only if M does not accept w.
- That is, running $T_{\langle M,w \rangle}(x)$ on **ANY** input x, runs M on w.

What does this achieve??

- This means, $L(T) = \Phi$ if M does not accept w and $L(T) \neq \Phi$ if M accepts w!
- This allows N to call $T_E(\langle T \rangle)$

Undecidability

$E_{TM} = \{\langle M \rangle | M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

Key idea:

- $N(\langle M, w \rangle)$ first builds the encoding of a TM $T_{\langle M, w \rangle}$
- $T_{\langle M, w \rangle}$ does not accept any string if and only if M does not accept w.
- That is, running $T_{\langle M, w \rangle}(x)$ on **ANY** input x, runs M on w.

- This means, $L(T) = \Phi$ if M does not accept w and $L(T) = \{w\}$ if M accepts w!
- This allows N to call $T_E(\langle T \rangle)$ and

  - ACCEPT if $T_E(\langle T \rangle)$ accepts $\Leftrightarrow L(T) = \Phi$ [M does not accept w]
  - REJECT if $T_E(\langle T \rangle)$ rejects $\Leftrightarrow L(T) \neq \Phi$ [M accepts w]

Undecidability

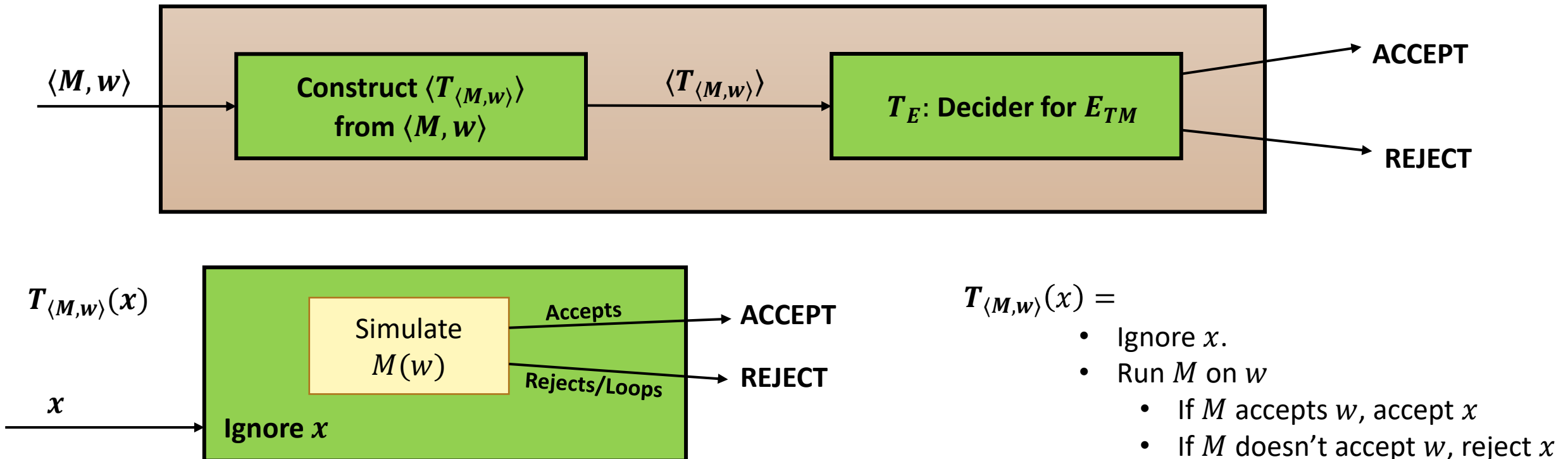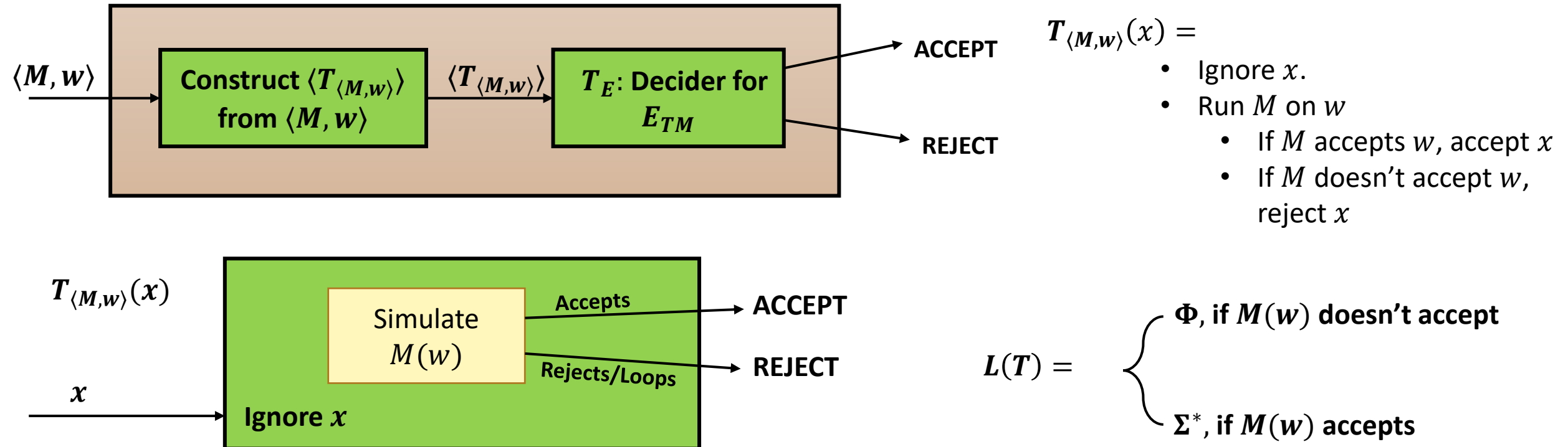$E_{TM} = \{\langle M \rangle | M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

Key idea:

- $N(\langle M, w \rangle)$ first builds the encoding of a TM $T_{\langle M,w \rangle}$
- $T_{\langle M,w \rangle}$ does not accept any string if and only if M does not accept w.
- That is, running $T_{\langle M,w \rangle}(x)$ on **ANY** input x, runs M on w.

- This means, $L(T) = \Phi$ if M does not accept w and $L(T) = \Sigma^*$ if M accepts w!
- This allows N to call $T_E(\langle T \rangle)$ and

    - ACCEPT if $T_E(\langle T \rangle)$ accepts $\Leftrightarrow L(T) = \Phi$ [M does not accept w]
    - REJECT if $T_E(\langle T \rangle)$ rejects $\Leftrightarrow L(T) \neq \Phi$ [M accepts w]

- **Deciding $\overline{A_{TM}}$ is tied to whether $L(T) = \Phi$!**

Undecidability
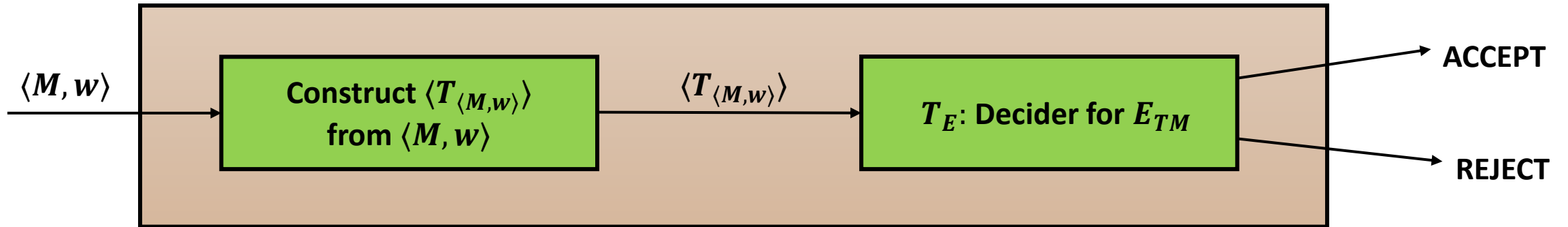
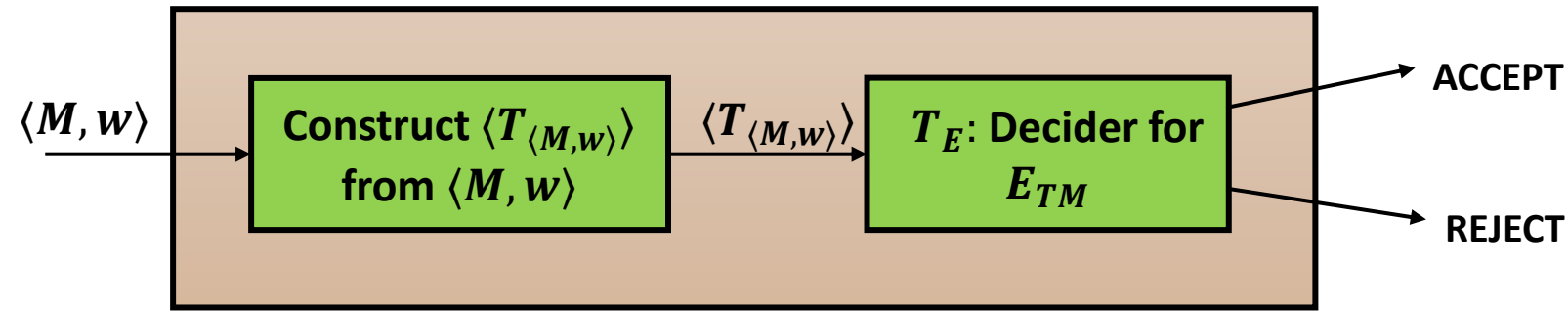$E_{TM} = \{\langle M \rangle | M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using T_E.

$T_{\langle M,w \rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M doesn't accept w, reject x

Undecidability

$E_{TM} = \{\langle M \rangle \mid M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using T_E.

$T_{\langle M,w \rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M doesn't accept w, reject x

$L(T) = \begin{cases} \Phi, \text{ if } M(w) \text{ doesn't accept} \\ \\ \Sigma^*, \text{ if } M(w) \text{ accepts} \end{cases}$

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N for $\overline{A_{TM}}$ using T_E.

$\langle M, w \rangle \longrightarrow$ **Construct $\langle T_{\langle M,w \rangle} \rangle$ from $\langle M, w \rangle$** $\xrightarrow{\langle T_{\langle M,w \rangle} \rangle}$ **T_E: Decider for E_{TM}** $\longrightarrow$ **ACCEPT** / **REJECT**

$T_{\langle M,w \rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M doesn't accept w, reject x

$L(T) = \begin{cases} \Phi, \text{ if } M(w) \text{ doesn't accept} \\ \\ \Sigma^*, \text{ if } M(w) \text{ accepts} \end{cases}$
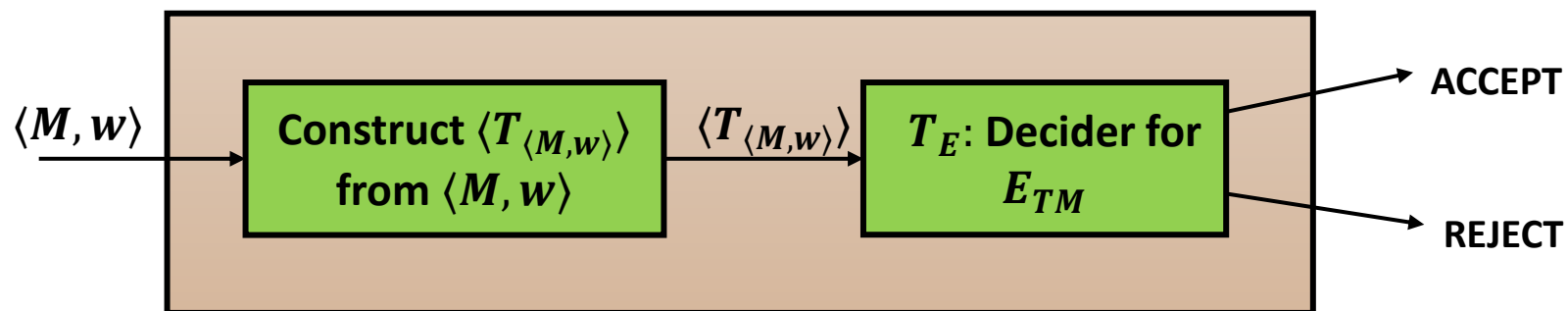
$T_E(\langle M \rangle) =$
- ACCEPT, if $L(M) = \Phi$
- REJECT, if $L(M) \neq \Phi$

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.
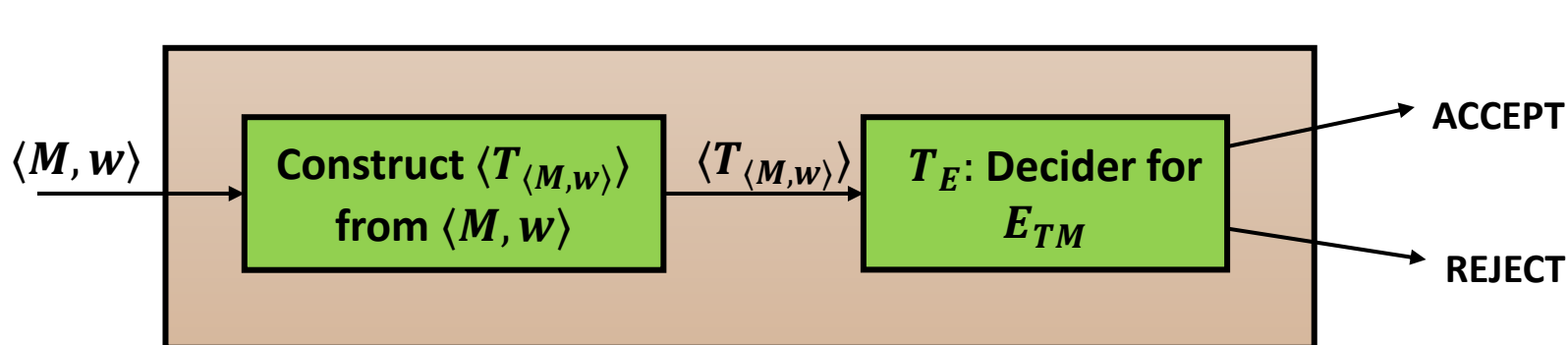
$T_{\langle M,w \rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M doesn't accept w, reject x

$$L(T) = \begin{cases} \Phi, \text{ if } M(w) \text{ doesn't accept} \\ \\ \Sigma^*, \text{ if } M(w) \text{ accepts} \end{cases}$$

$T_E(\langle T \rangle) =$
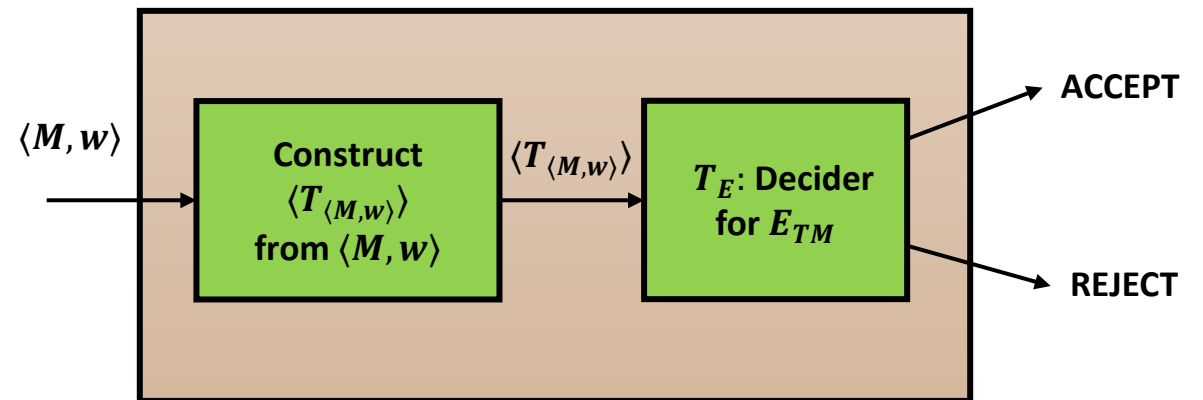- ACCEPT, **if** $L(T) = \Phi$
- REJECT, **if** $L(T) \neq \Phi$

Undecidability

$E_{TM} = \{\langle M\rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

$T_{\langle M,w\rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M doesn't accept w, reject x

$$L(T) = \begin{cases} \Phi, \textbf{ if } M(w) \textbf{ doesn't accept} \\ \\ \Sigma^*, \textbf{ if } M(w) \textbf{ accepts} \end{cases}$$

$T_E(\langle T\rangle) =$
- ACCEPT, **if** $L(T) = \Phi$ (if $M(w)$ doesn't accept)
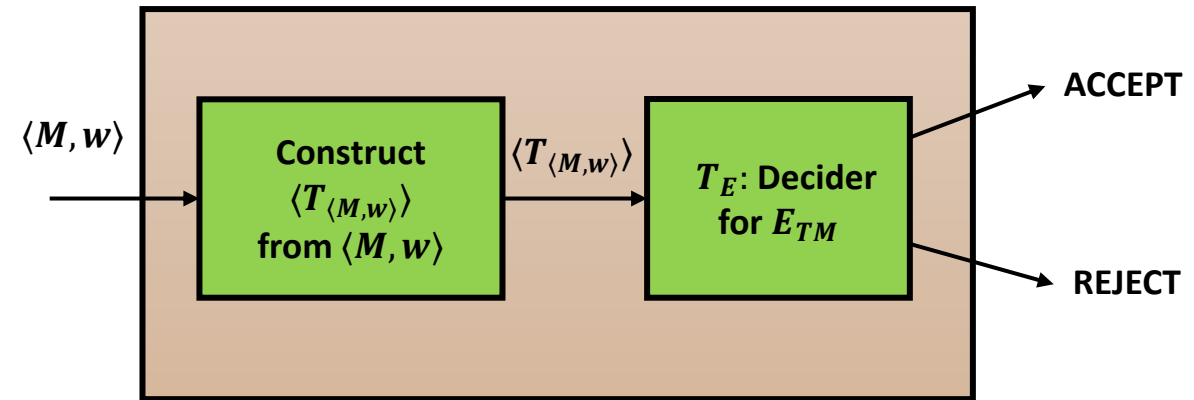- REJECT, **if** $L(T) \neq \Phi$ (if $M(w)$ accepts)

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

$N(\langle M, w \rangle) =$

- Construct $\langle T_{\langle M,w \rangle} \rangle$, the encoding of $T_{\langle M,w \rangle}$ such that for any input x it works as follows:
  - Ignore x.
  - Run M on w
    - If M accepts w, accept x
    - If M rejects w, reject x

- Send $\langle T_{\langle M,w \rangle} \rangle$ to T_E and Output
  - ACCEPT if T_E accepts
  - REJECT if T_E rejects

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using $\boldsymbol{T_E}$.

$N(\langle M, w \rangle) =$

- Construct $\langle T_{\langle M,w \rangle} \rangle$, the encoding of $T_{\langle M,w \rangle}$ such that for any input x it works as follows:
    - Ignore x.
    - Run M on w
        - If M accepts w, accept x
        - If M rejects w, reject x

- Send $\langle T_{\langle M,w \rangle} \rangle$ to T_E and Output
    - ACCEPT if T_E accepts
    - REJECT if T_E rejects

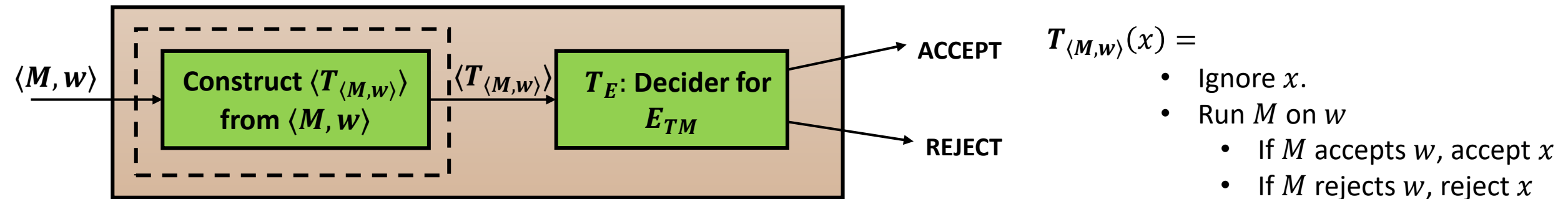

- $\overline{A_{TM}} \leqslant E_{TM}$
- $\overline{A_{TM}}$ is undecidable
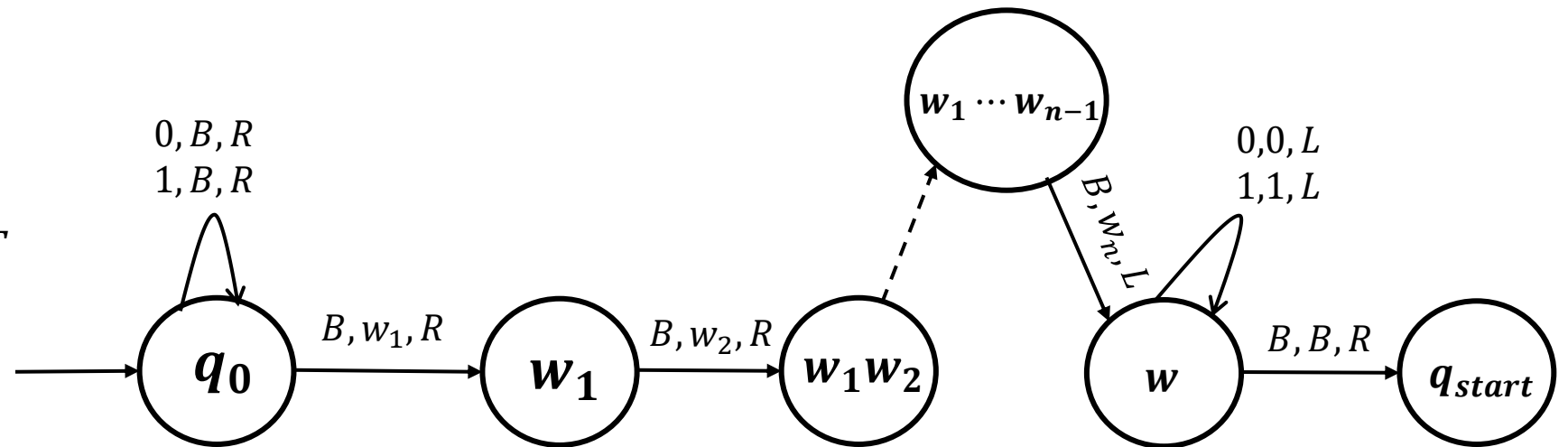- E_{TM} is undecidable!

Undecidability

$E_{TM} = \{\langle M\rangle | M \text{ is a Turing Machine and } L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

$T_{\langle M,w\rangle}(x) =$
- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M rejects w, reject x

$T_{\langle M,w\rangle}$ from $\langle M,w\rangle$:

- Remove i/p x from the tape of T
- `Embed' w into the TM of M

Undecidability

$E_{TM} = \{\langle M \rangle \mid M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \preccurlyeq E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

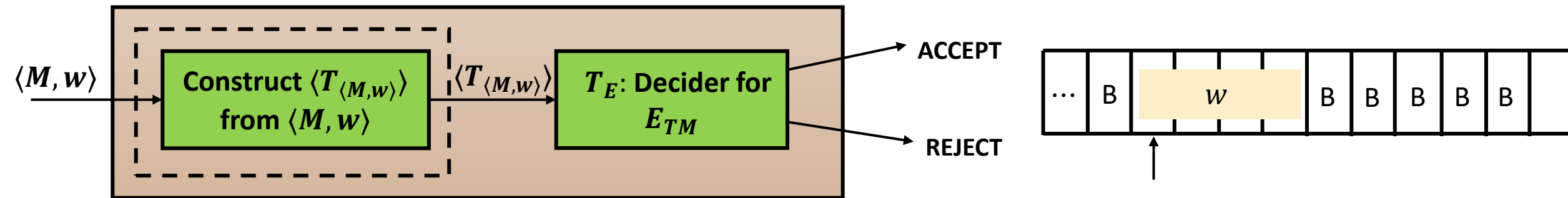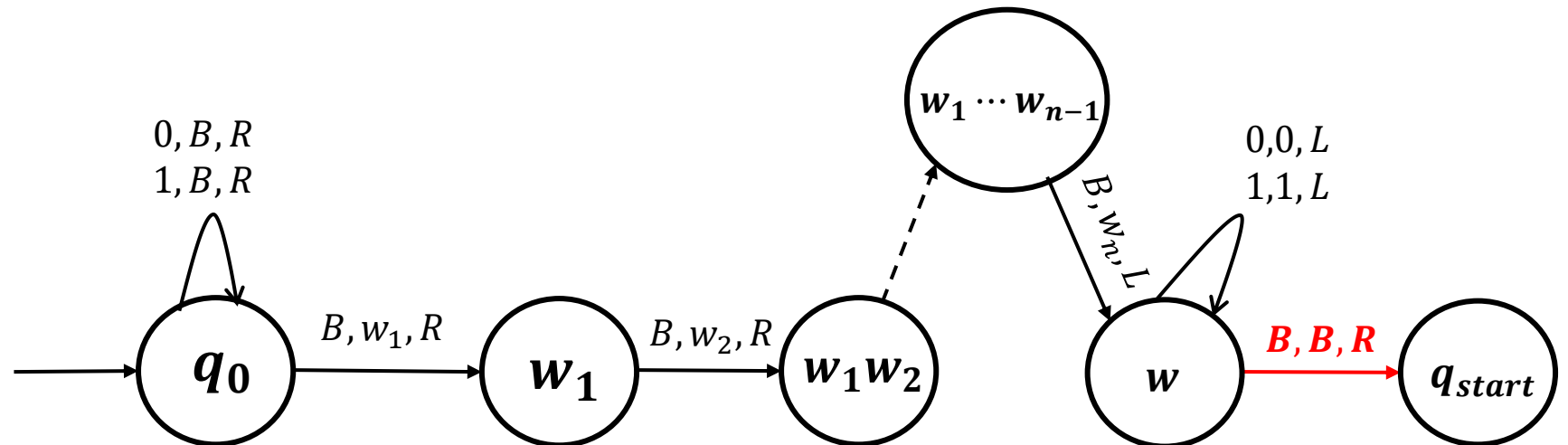$T_{\langle M,w \rangle}$ from $\langle M, w \rangle$:

- Remove i/p x from the tape of T
- `Embed' w into the TM of M

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

$N(\langle M, w \rangle) =$

- Construct $\langle T_{\langle M,w \rangle} \rangle$, the encoding of $T_{\langle M,w \rangle}$ such that for any input x it works as follows:
  - Ignore x.
  - Run M on w
    - If M accepts w, accept x
    - If M rejects w, reject x

- Send $\langle T_{\langle M,w \rangle} \rangle$ to T_E and Output
  - ACCEPT if T_E accepts
  - REJECT if T_E rejects

- $\overline{A_{TM}} \leqslant E_{TM}$
- $\overline{A_{TM}}$ is undecidable
- E_{TM} is undecidable!

Claim: $E_{TM} \in$ co-RE$-R$

WHY?

Undecidability

$E_{TM} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{A_{TM}} \leqslant E_{TM}$ by constructing a Turing Machine N that decides $\overline{A_{TM}}$ using T_E.

$N(\langle M, w \rangle) =$

- Construct $\langle T_{\langle M,w \rangle} \rangle$, the encoding of $T_{\langle M,w \rangle}$ such that for any input x it works as follows:
  - Ignore x.
  - Run M on w
    - If M accepts w, accept x
    - If M rejects w, reject x

- Send $\langle T_{\langle M,w \rangle} \rangle$ to T_E and Output

  ACCEPT if T_E accepts

  REJECT if T_E rejects

- $\overline{A_{TM}} \leqslant E_{TM}$
- $\overline{A_{TM}}$ is undecidable
- E_{TM} is undecidable!

$E_{TM} \in$ **co-RE**$-R$

Proof idea: We can build a co-recognizer for E_{TM}.

$C =$ On input $\langle M \rangle$
- For $i = 1, 2, 3, \cdots$
  - For $j = 1, 2, 3, \cdots i$

    Run M on s_j for i steps.

    If M accepts s_j, REJECT.

Thank You!