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

$\forall\omega \in L, M(\omega)$ **accepts**

$\forall\omega \notin L, M(\omega)$ **doesn't accept** (rejects or runs infinitely)

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

$\forall\omega \in L, M(\omega)$ **doesn't reject** (accepts or loops)

$\forall\omega \notin L, M(\omega)$ **rejects**

Quick Recap

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

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

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

Quick Recap

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

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

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

Some examples of languages that are recursive/decidable:

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

An undecidable language:

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

Quick Recap

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

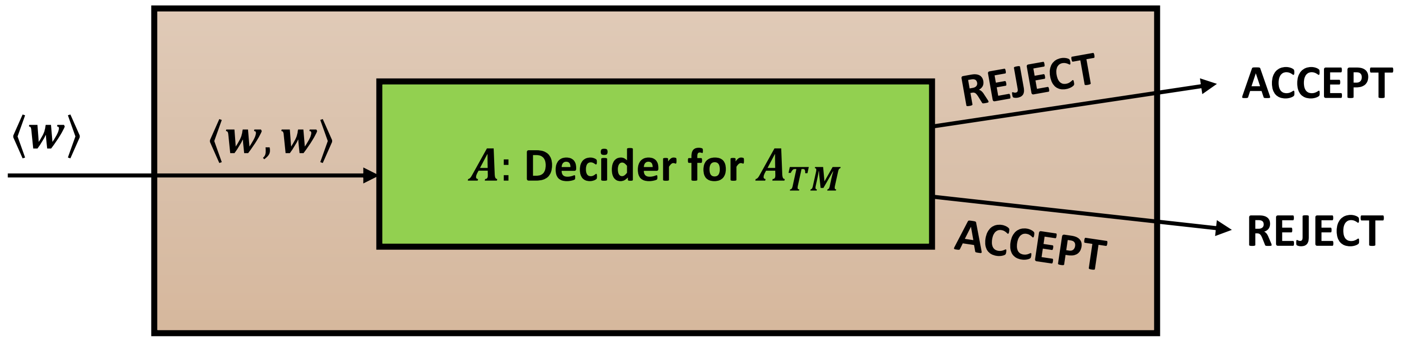
An undecidable language:

- $A_{TM} = \{\langle M, w\rangle | M$ **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.



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

If $A(\langle w, w\rangle)$ accepts, D outputs $REJECT$

If $A(\langle w, w\rangle)$ rejects, D outputs $ACCEPT$

Quick Recap

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

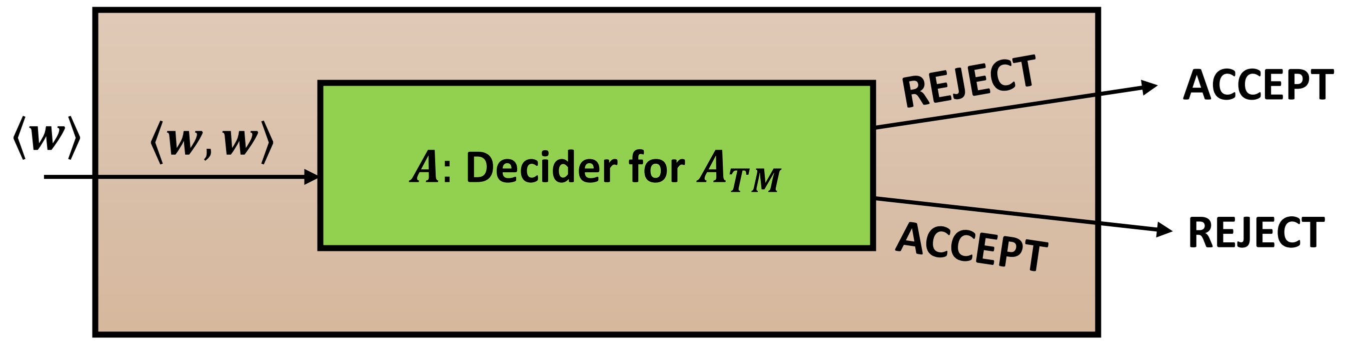
An undecidable language:

- $A_{TM} = \{\langle M, w\rangle | M$ **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.



For $w = \langle M_w\rangle$

$D(\langle M_w\rangle) =$

Run $A(M_w, \langle M_w\rangle)$

$A(M_w, \langle M_w\rangle)$ ***accepts, if*** $M_w(\langle M_w\rangle)$ ***accepts***
(***D outputs*** $REJECT$)

$A(M_w, \langle M_w\rangle)$ ***rejects, if*** $M_w(\langle M_w\rangle)$ ***doesn't accept***
(***D outputs*** $ACCEPT$)

Quick Recap

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

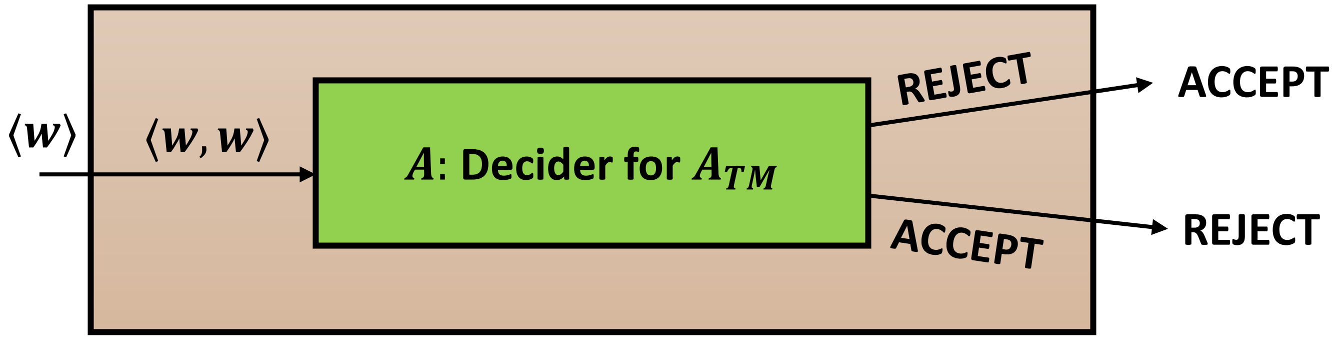
An undecidable language:

- $A_{TM} = \{\langle M, w\rangle | M$ **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.



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

$D(\langle M_w\rangle) =$

Run $A(M_w, \langle M_w\rangle)$

$A(M_w, \langle M_w\rangle)$ accepts, if $M_w(\langle M_w\rangle)$ accepts
(D outputs $REJECT$)

$A(M_w, \langle M_w\rangle)$ rejects, if $M_w(\langle M_w\rangle)$ doesn't accept
(D outputs $ACCEPT$)

Quick Recap

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

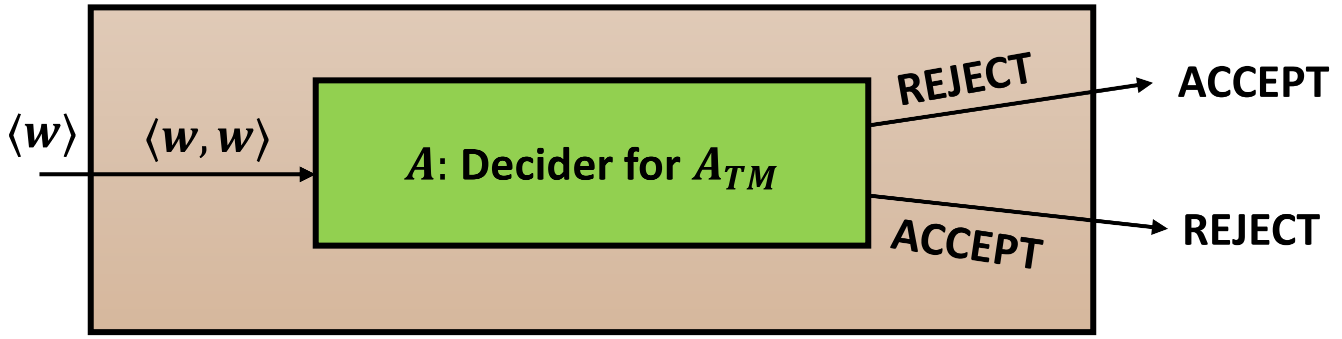
An undecidable language:

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

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

Proof strategy: By contradiction. We assume that a Total TM A exists that decides A_{TM}.

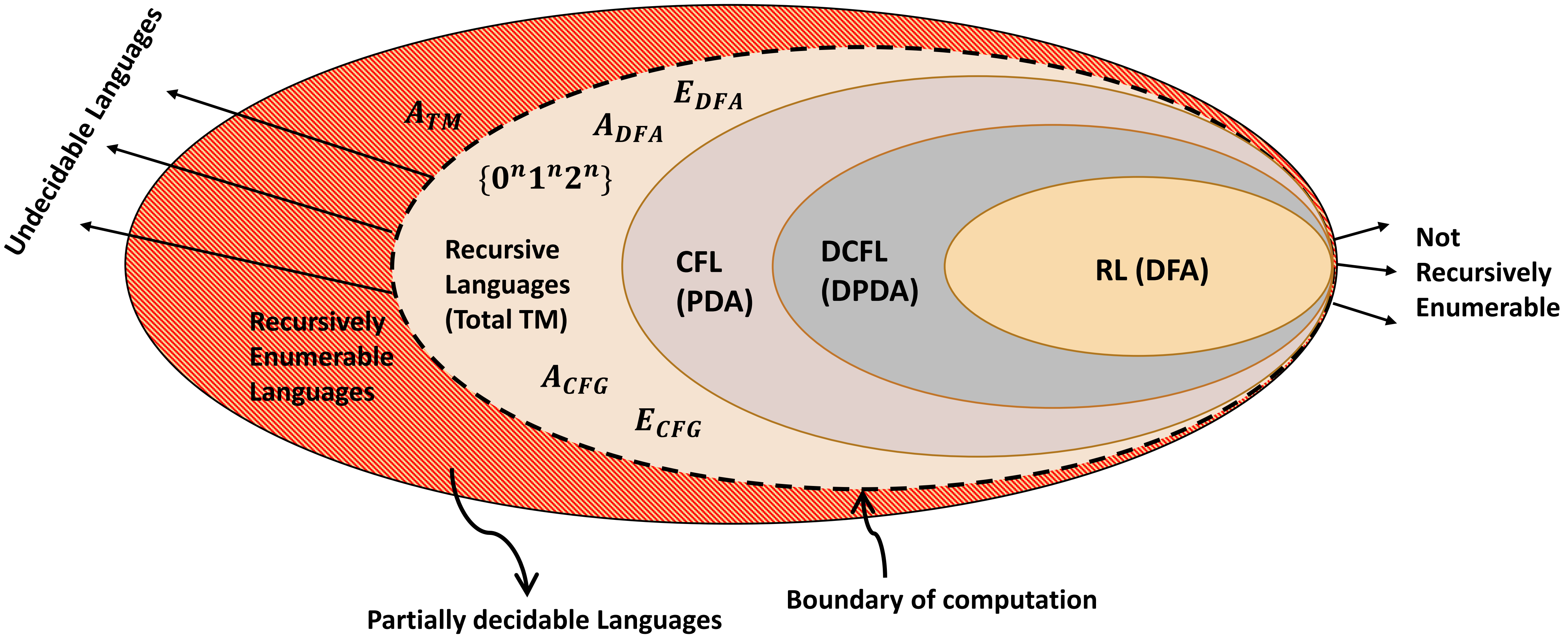
We build a Total TM D that accepts w as input and calls $A(\langle w, w\rangle)$ as a subroutine and outputs the opposite of A.



$D(\langle D\rangle)$ *accepts* $\leftrightarrow$ $D(\langle D\rangle)$ *rejects*
$D(\langle D\rangle)$ *rejects* $\leftrightarrow$ $D(\langle D\rangle)$ *accepts*

For $\boldsymbol{w = \langle D\rangle}$**, there is a contradiction!**

Quick Recap


Undecidable Languages
A_{TM}
E_{DFA}
A_{DFA}
$\{0^n1^n2^n\}$
Recursive Languages (Total TM)
CFL (PDA)
DCFL (DPDA)
RL (DFA)
Not Recursively Enumerable
Recursively Enumerable Languages
A_{CFG}
E_{CFG}
Partially decidable Languages
Boundary of computation

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**.

First, recall that there exists a bijective map (one-one correspondence) between the set of all finite-length 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 finite-length 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

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

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

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

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

	$\boldsymbol{\langle M_0\rangle}$	$\boldsymbol{\langle M_1\rangle}$	$\boldsymbol{\langle M_2\rangle}$	$\boldsymbol{\langle M_3\rangle}$	$\boldsymbol{\langle M_4\rangle}$	$\cdots$
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

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

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

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

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

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

$\boldsymbol{A}$	$\langle M_0\rangle$	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$	$\cdots$
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	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
$\boldsymbol{D}$						$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

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

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

$\}$

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

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

An undecidable problem

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

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

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

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

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

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

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**

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

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**	**Reject**	**Reject**	**Accept**	$\cdots$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

What will be the D^{th} diagonal entry??

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**

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

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**	**Reject**	**Reject**	**Accept**	$\cdots$	**??**	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$

What will be the D^{th} diagonal entry??

Contradiction!

The Halting problem

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

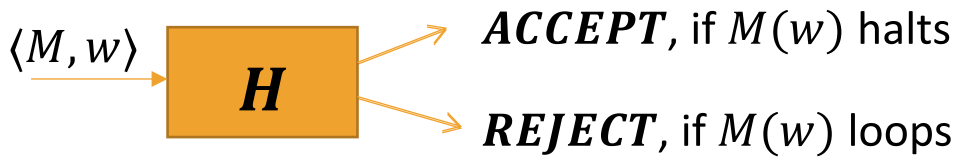
The Halting Problem: Does there exist a Total Turing Machine H that accepts as input a Turing Machine M and an input string w and outputs YES, if $M(w)$ halts (accepts or rejects) and NO, if $M(w)$ does not halt (loops forever), i.e.

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

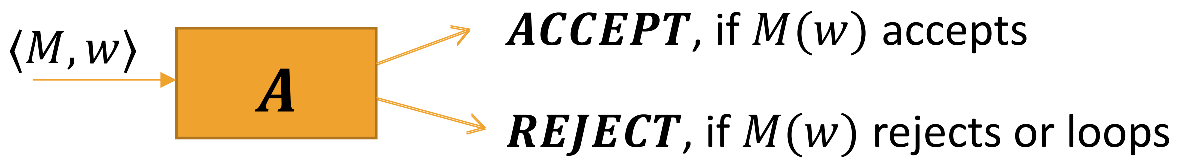
- Turing stated the Halting problem and demonstrated its undecidability in his famous 1936 paper.
- This provided a negative answer to Hilbert's Entscheidungsproblem.
- **Proof strategy:** We will try to show that if we had such a Total TM H, we would be able to build a Total TM for A_{TM}
- But since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

The Halting problem

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



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}

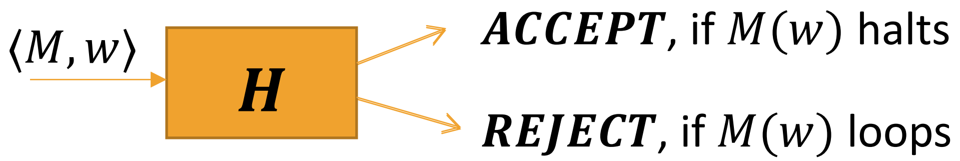


$\langle M, w\rangle$
A
$ACCEPT$, if $M(w)$ accepts
$REJECT$, if $M(w)$ rejects or loops

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

The Halting problem

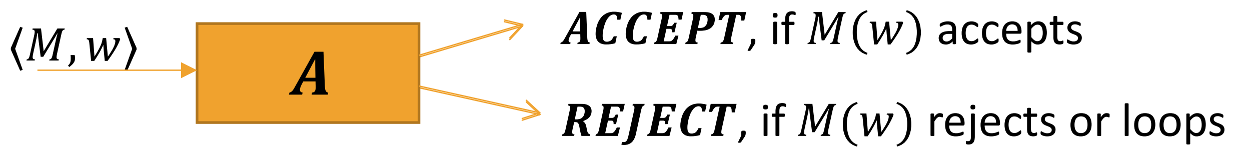
$HALT_{TM} = \{\langle M, w\rangle | M$ halts on input $w\}$. Is $HALT_{TM}$ decidable?



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



$\langle M, w\rangle$
A
$ACCEPT$, if $M(w)$ accepts
$REJECT$, if $M(w)$ rejects or loops

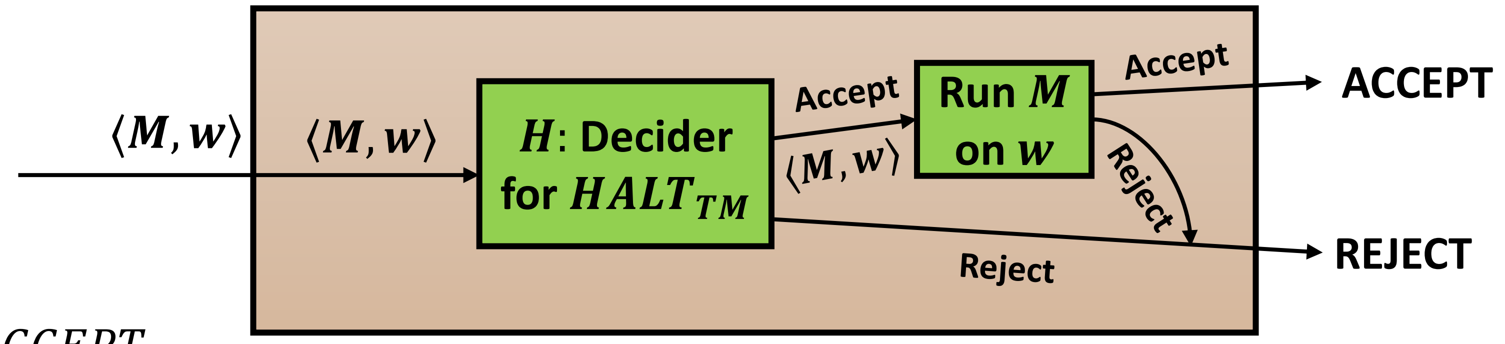
The Halting problem

$HALT_{TM} = \{\langle M, w\rangle | M$ 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

$HALT_{TM} = \{\langle M, w\rangle \mid M$ 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 ⟨M, w⟩
        Run H(⟨M, w⟩)
        If H rejects, output REJECT
        If H accepts,
            Run M(w)
            If M(w) accepts, output ACCEPT
            If M(w) rejects, output REJECT
```

$$\begin{tikzcd}
\boxed{HALT_{TM}} \arrow[r, "\text{Can be used to solve}"] & \boxed{A_{TM}} \arrow[d] \\
\text{$HALT_{TM}$ is undecidable} & \text{Undecidable} \arrow[l]
\end{tikzcd}$$

The Halting problem

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

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

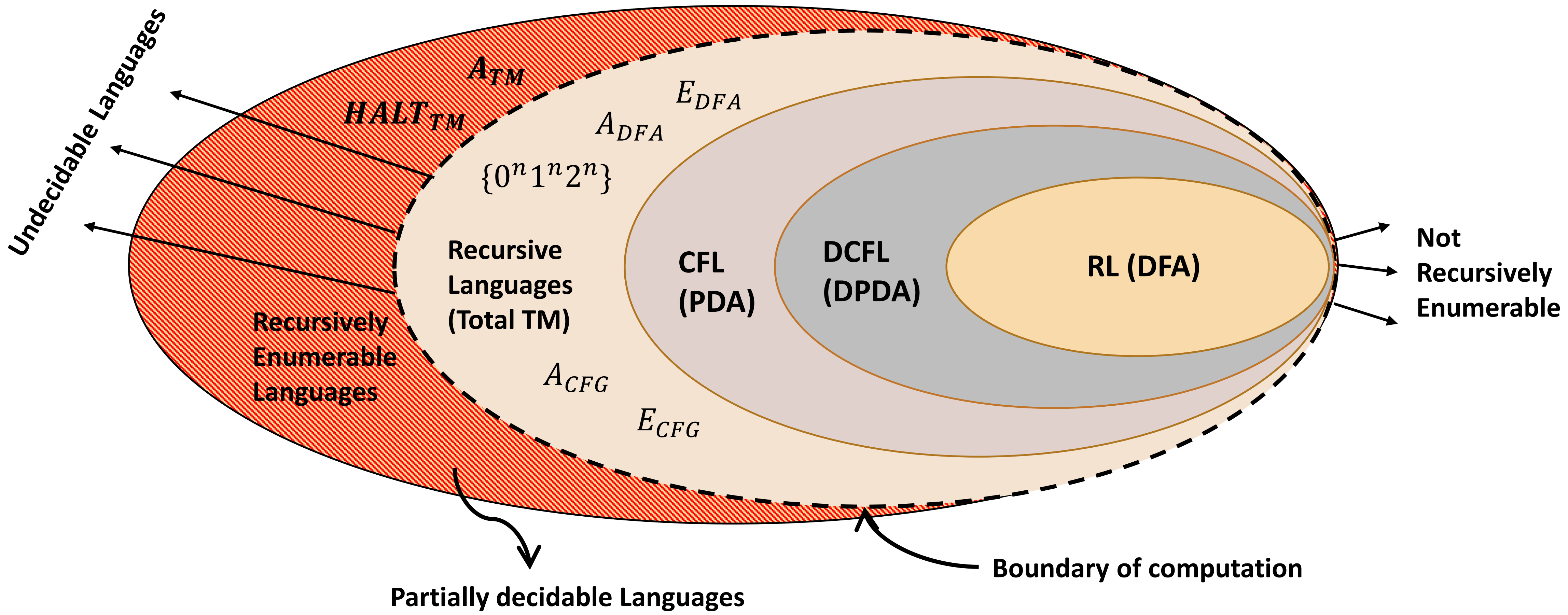
$HALT_{TM} \in RE$ as H halts whenever M accepts or rejects w and so

$Q =$ 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

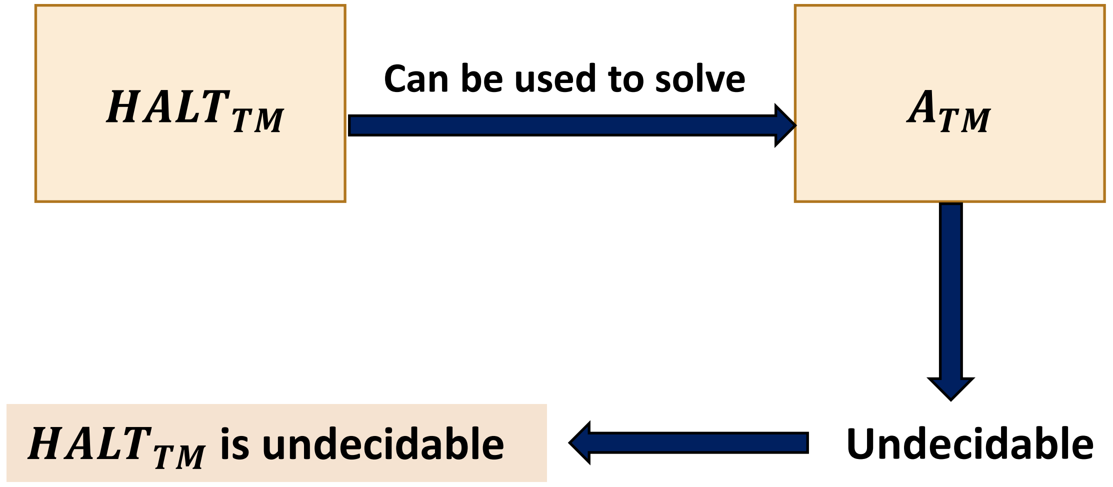

Undecidable Languages
A_{TM}
$HALT_{TM}$
E_{DFA}
A_{DFA}
$\{0^n1^n2^n\}$
Recursive Languages (Total TM)
CFL (PDA)
DCFL (DPDA)
RL (DFA)
Not Recursively Enumerable
Recursively Enumerable Languages
A_{CFG}
E_{CFG}
Partially decidable Languages
Boundary of computation

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.**



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

Reduction

Generally,

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

Suppose, $\boldsymbol{A} \preccurlyeq \boldsymbol{B}$ and

- $\boldsymbol{A}$ **is undecidable. Then** $\boldsymbol{B}$ **is undecidable.**

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

- Intuitively, this means $\boldsymbol{B}$ **is at least as hard as A.**
- So, if $\mathbf{A} \notin \boldsymbol{R} \Rightarrow \boldsymbol{B} \notin \boldsymbol{R}$
- $\boldsymbol{A} \notin \boldsymbol{RE} \Rightarrow \boldsymbol{B} \notin \boldsymbol{RE}$
- $\boldsymbol{A} \notin \boldsymbol{co\ RE} \Rightarrow \boldsymbol{B} \notin \boldsymbol{co\ RE}$

$\boldsymbol{HALT_{TM}} \preccurlyeq \boldsymbol{P} \Rightarrow \boldsymbol{P}$ is undecidable. Also, $\boldsymbol{P} \notin \boldsymbol{R}$

Reduction

Generally,

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

Suppose, $\boldsymbol{A} \preccurlyeq \boldsymbol{B}$ and

- A is undecidable. Then B is undecidable.
- **$\boldsymbol{B}$ is decidable. Then $\boldsymbol{A}$ is decidable.**

- Intuitively, this means **$\boldsymbol{B}$ is at least as hard as A**.
- So, if $\mathbf{A} \notin \boldsymbol{R} \Rightarrow \boldsymbol{B} \notin \boldsymbol{R}$
- $\boldsymbol{A} \notin \boldsymbol{RE} \Rightarrow \boldsymbol{B} \notin \boldsymbol{RE}$
- $\boldsymbol{A} \notin \boldsymbol{co\ RE} \Rightarrow \boldsymbol{B} \notin \boldsymbol{co\ RE}$

- Intuitively, this means **$\boldsymbol{A}$ is not harder than B**.
- So, if $\mathbf{B} \in \boldsymbol{R} \Rightarrow \boldsymbol{A} \in \boldsymbol{R}$
- $\mathbf{B} \in \boldsymbol{RE} \Rightarrow \boldsymbol{A} \in \boldsymbol{RE}$
- $\mathbf{B} \in \boldsymbol{co\ RE} \Rightarrow \boldsymbol{A} \in \boldsymbol{co\ RE}$

Reduction

- A language $\boldsymbol{A}$ **reduces to** another language $\mathbf{B}$ ($\boldsymbol{A} \preccurlyeq \boldsymbol{B}$) iff we can **build a solver for $\boldsymbol{A}$ using a solver for $\boldsymbol{B}$**.
- **If $\boldsymbol{A}$ is undecidable then so is $\boldsymbol{B}$**, i.e. ***B* is at least as hard as *A*.**
- **If B is decidable, then so is A.** Intuitively, this means $\boldsymbol{A}$ **is not harder than B**.
- This is a common technique to show that certain problems are decidable/undecidable.

This is how we can prove several problems are undecidable: by **reducing some known undecidable problem to these problems.**

Examples:

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

$\vdots$

Reduction

This is how we can prove several problems are undecidable: by **reducing some known undecidable problem to these problems.**

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).
- 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 $\boldsymbol{E_{TM}, EQ_{TM}, ALL_{TM}}$ **are undecidable**.

Undecidability

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

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

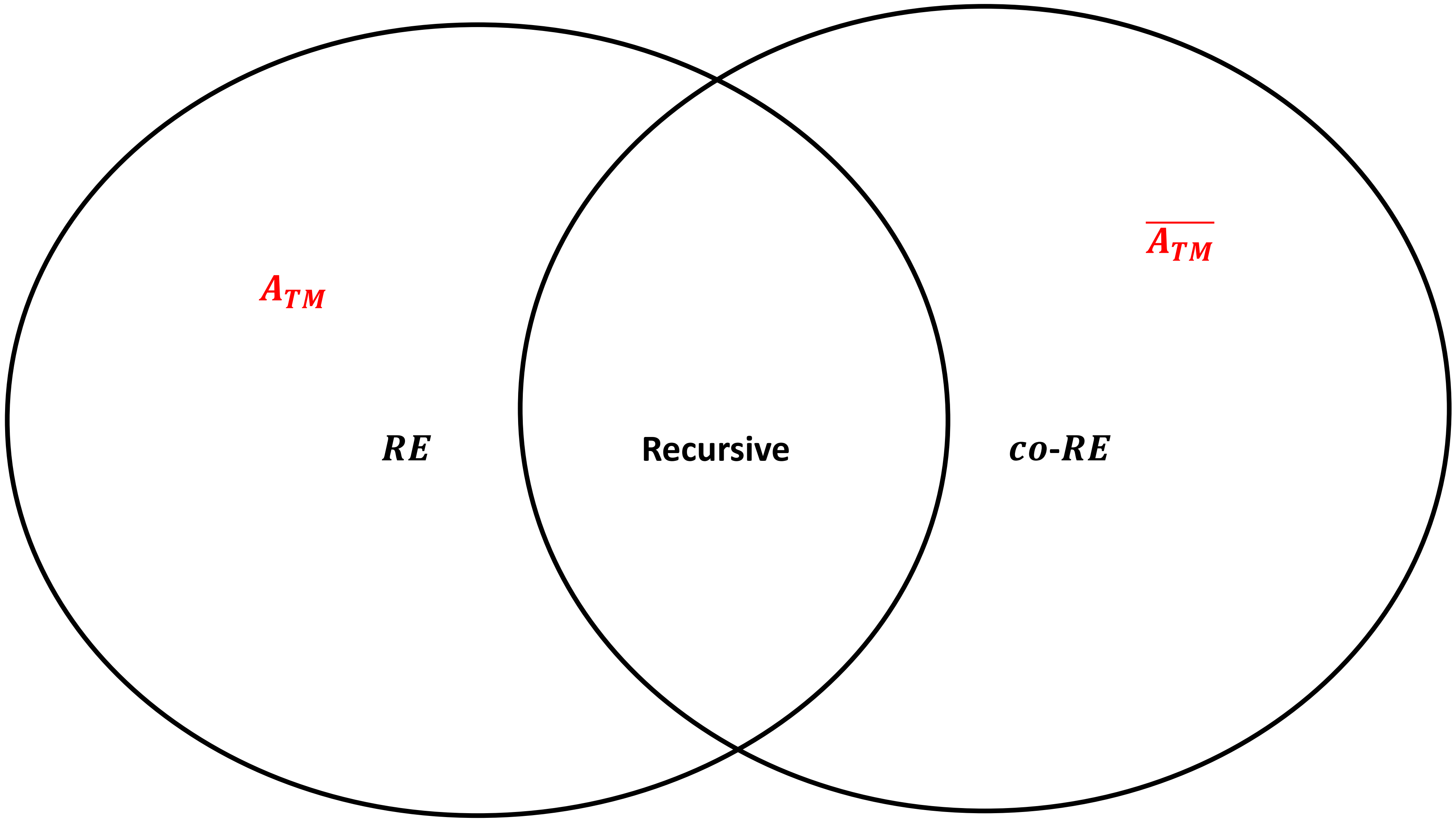
What is $\overline{\boldsymbol{A_{TM}}}$?

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

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

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

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


A_{TM}
$\overline{A_{TM}}$
RE
Recursive
$co\text{-}RE$

Undecidability

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

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

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

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

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

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

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

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

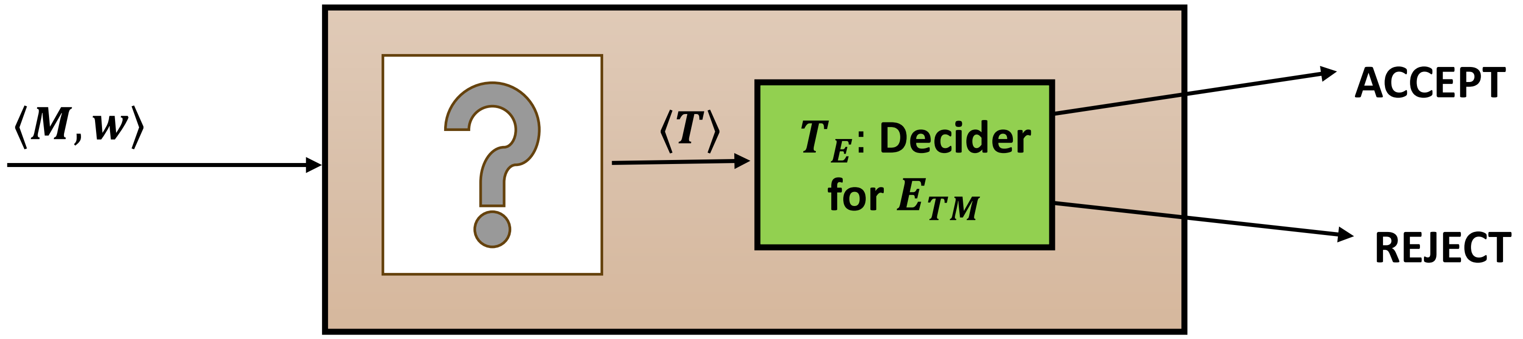
Undecidability

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

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

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

$$\boldsymbol{T_E(\langle T\rangle)} = \begin{cases} \textbf{ACCEPTS, if } \boldsymbol{L(\langle T\rangle) = \Phi} \\ \textbf{REJECTS, if } \boldsymbol{L(\langle T\rangle) \neq \Phi} \end{cases}$$



Undecidability

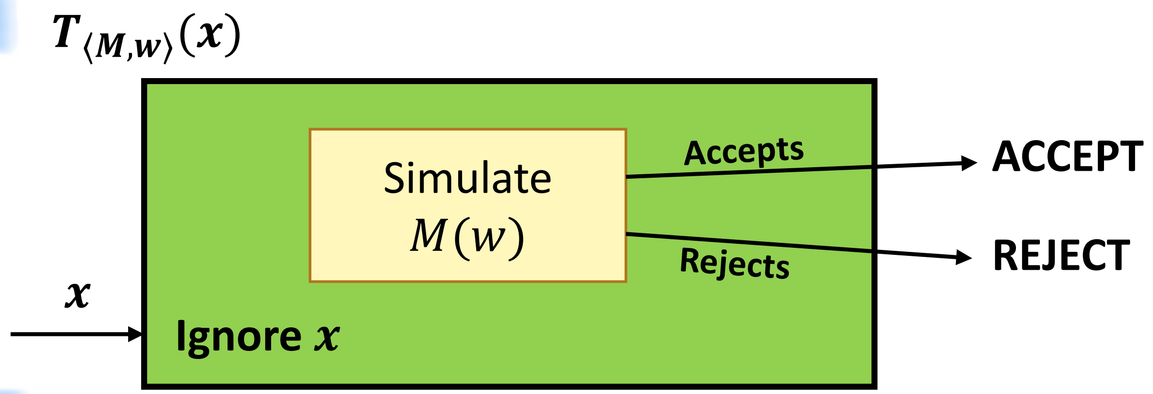
$\boldsymbol{E_{TM}} = \{\langle M \rangle | M$ is a Turing Machine and $L(M) = \Phi\}$

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



- 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

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

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

Key idea:

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

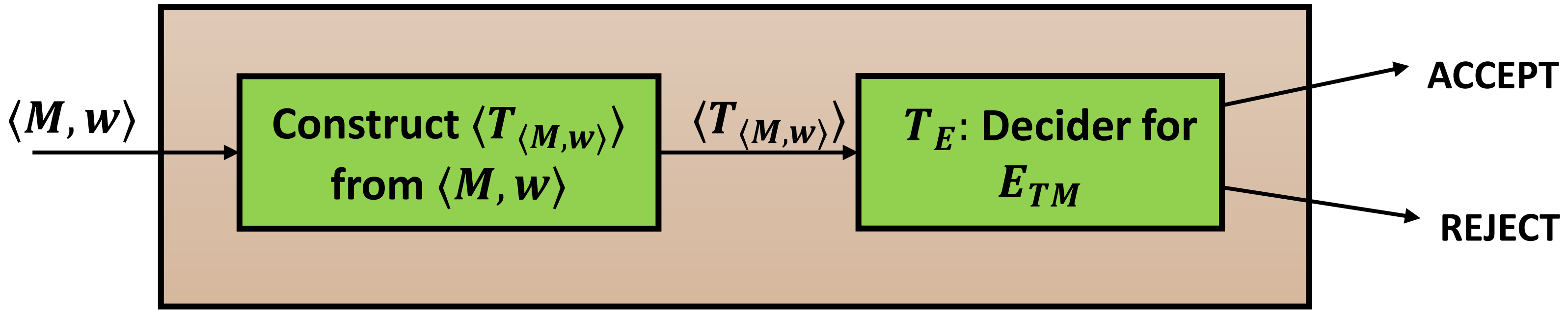
- This means, $L(T) = \Phi$ if M does not accept w and $L(T) = \Sigma^*$ if M accepts w!
- This allows N to call $T_E(\langle T\rangle)$ and
  - ACCEPT if $T_E(\langle T\rangle)$ accepts $\Leftrightarrow L(T) = \Phi$ [M does not accept w]
  - REJECT if $T_E(\langle T\rangle)$ rejects $\Leftrightarrow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

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

Undecidability

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

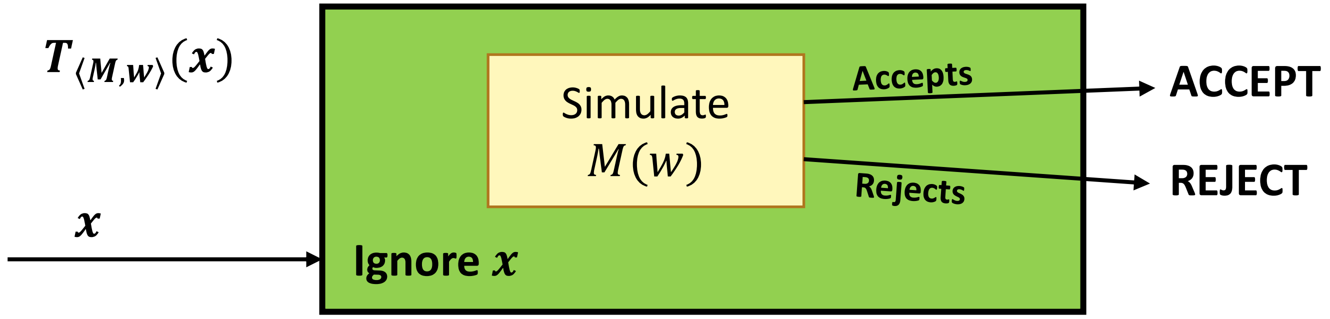
Proof: Let T_E be the Turing Machine that decides E_{TM}. We shall prove that $\overline{\boldsymbol{A_{TM}}} \preccurlyeq \boldsymbol{E_{TM}}$ by constructing a Turing Machine N for $\overline{\boldsymbol{A_{TM}}}$ using $\boldsymbol{T_E}$.



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

$T_{\langle M,w\rangle}(x) =$

- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M rejects w, reject x

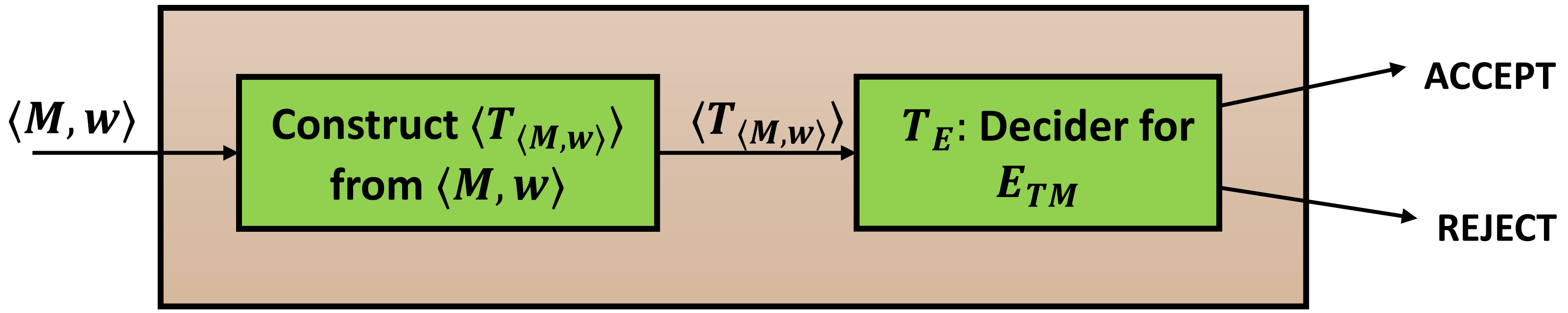


$$L(T) = \begin{cases} \Phi\text{, if } M(w) \text{ doesn't accept} \\ \Sigma^*\text{, if } M(w) \text{ accepts} \end{cases}$$

Undecidability

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

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



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

$\boldsymbol{T_{\langle M,w \rangle}}(x) =$

- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M rejects w, reject x

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

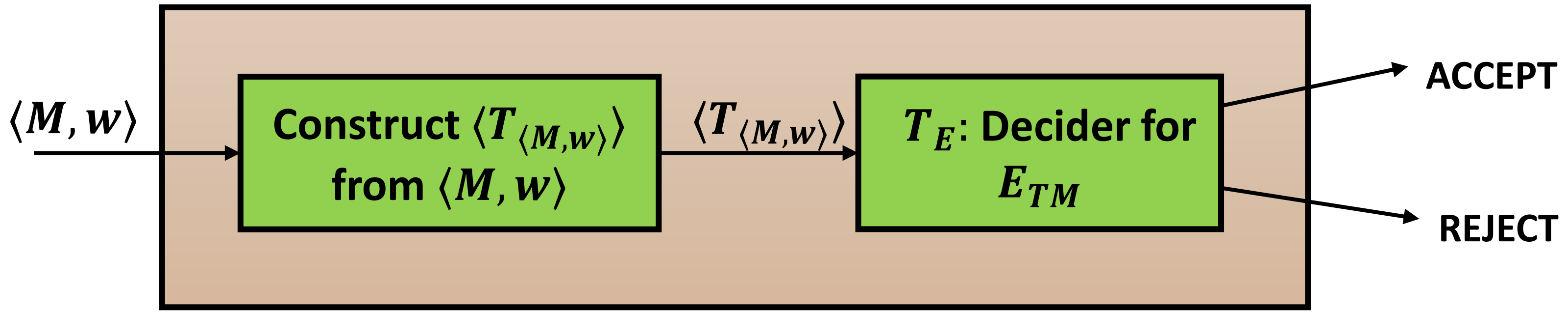
$T_E(\langle \boldsymbol{T} \rangle) =$

- ACCEPT, **if** $\boldsymbol{L(T) = \Phi}$
- REJECT, **if** $\boldsymbol{L(T) \neq \Phi}$

Undecidability

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

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



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

$\boldsymbol{T_{\langle M,w \rangle}}(x) =$

- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M rejects w, reject x

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

$T_E(\langle \boldsymbol{T} \rangle) =$

- ACCEPT, **if** $\boldsymbol{L(T) = \Phi}$ (if $M(w)$ doesn't accept)
- REJECT, **if** $\boldsymbol{L(T) \neq \Phi}$ (if $M(w)$ accepts)

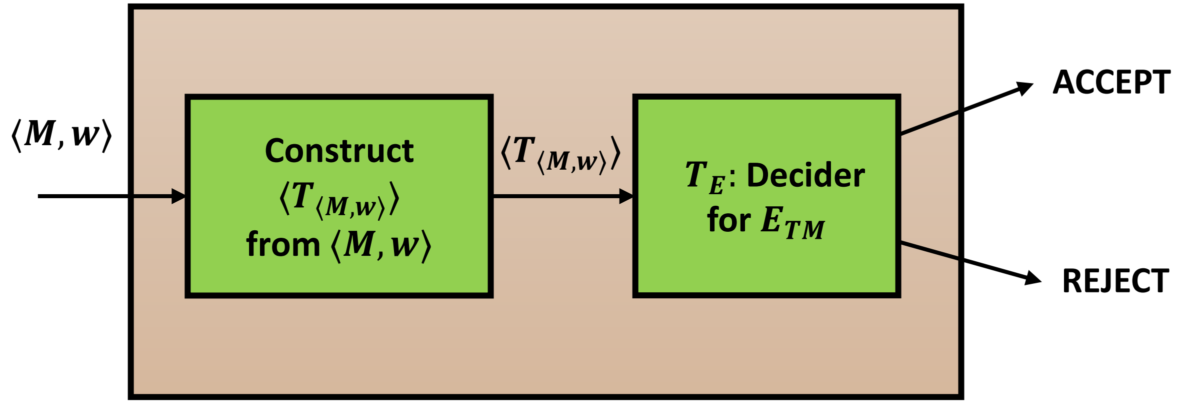
Undecidability

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

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



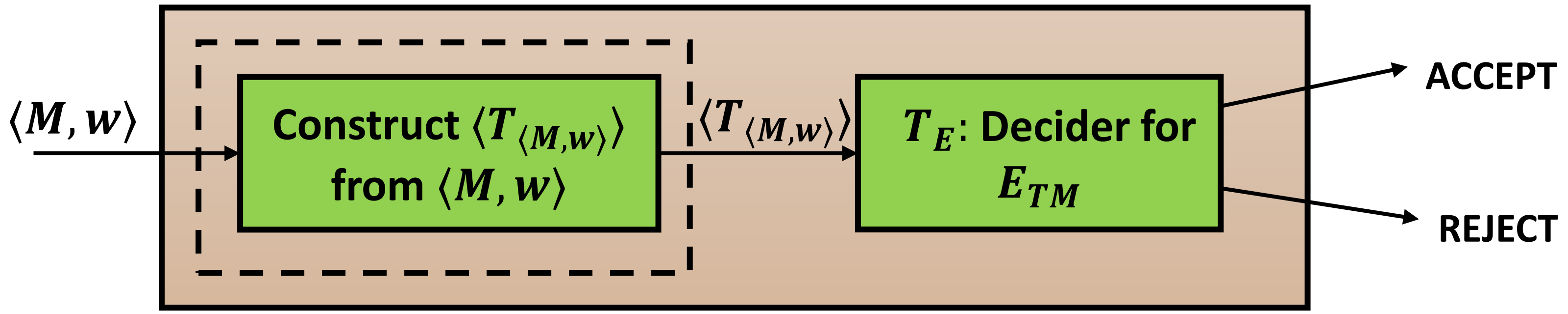
$\langle M, w\rangle$
Construct $\langle T_{\langle M,w\rangle}\rangle$ from $\langle M, w\rangle$
$\langle T_{\langle M,w\rangle}\rangle$
T_E: Decider for E_{TM}
ACCEPT
REJECT

- $\boldsymbol{\overline{A_{TM}}} \preccurlyeq \boldsymbol{E_{TM}}$
- $\boldsymbol{\overline{A_{TM}}}$ **is undecidable**
- $\boldsymbol{E_{TM}}$ **is undecidable!**

Undecidability

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

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



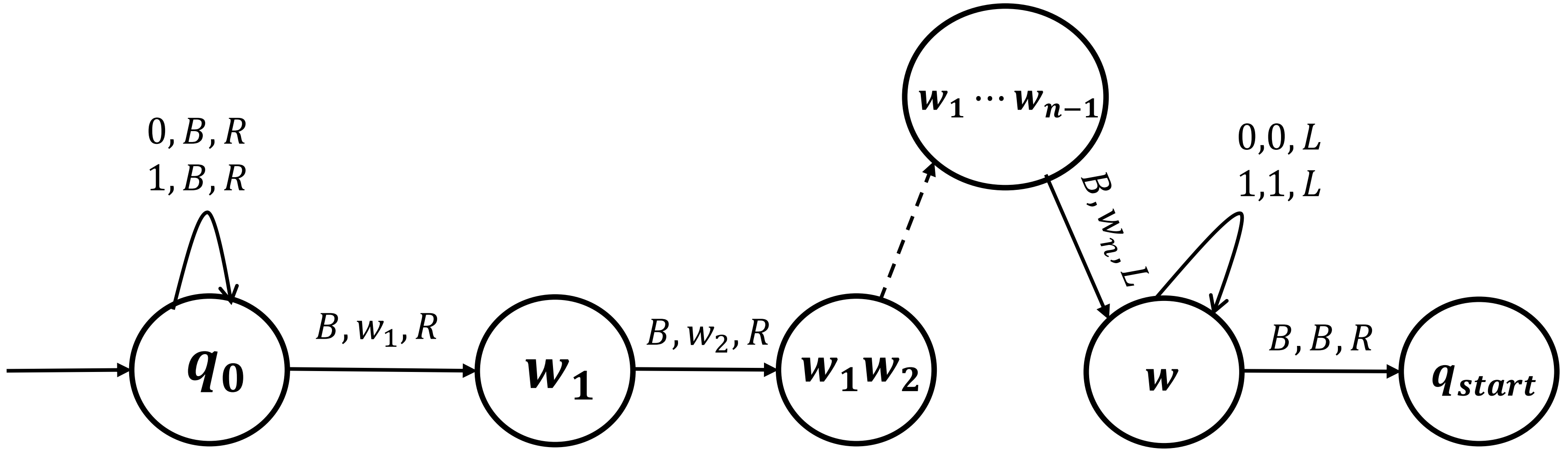
$\langle \boldsymbol{M}, \boldsymbol{w}\rangle$
Construct $\langle \boldsymbol{T_{\langle M,w\rangle}}\rangle$ from $\langle \boldsymbol{M}, \boldsymbol{w}\rangle$
$\langle \boldsymbol{T_{\langle M,w\rangle}}\rangle$
$\boldsymbol{T_E}$: Decider for $\boldsymbol{E_{TM}}$
ACCEPT
REJECT

$\boldsymbol{T_{\langle M,w\rangle}}(x) =$

- Ignore x.
- Run M on w
  - If M accepts w, accept x
  - If M rejects w, reject x

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

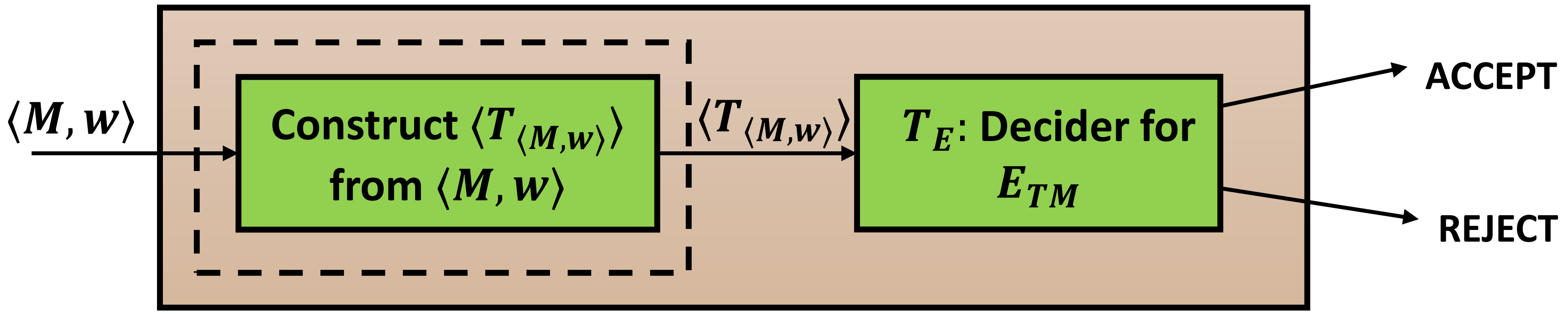
- Remove i/p x from the tape of T
- `Embed' w into the TM of M



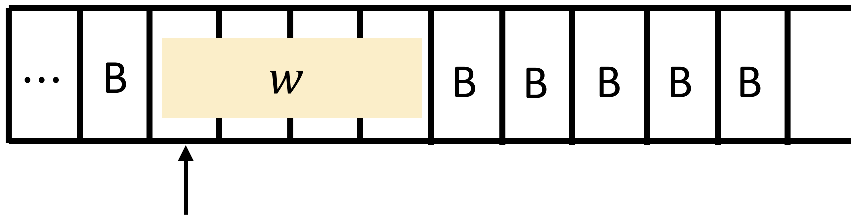
Undecidability

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

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

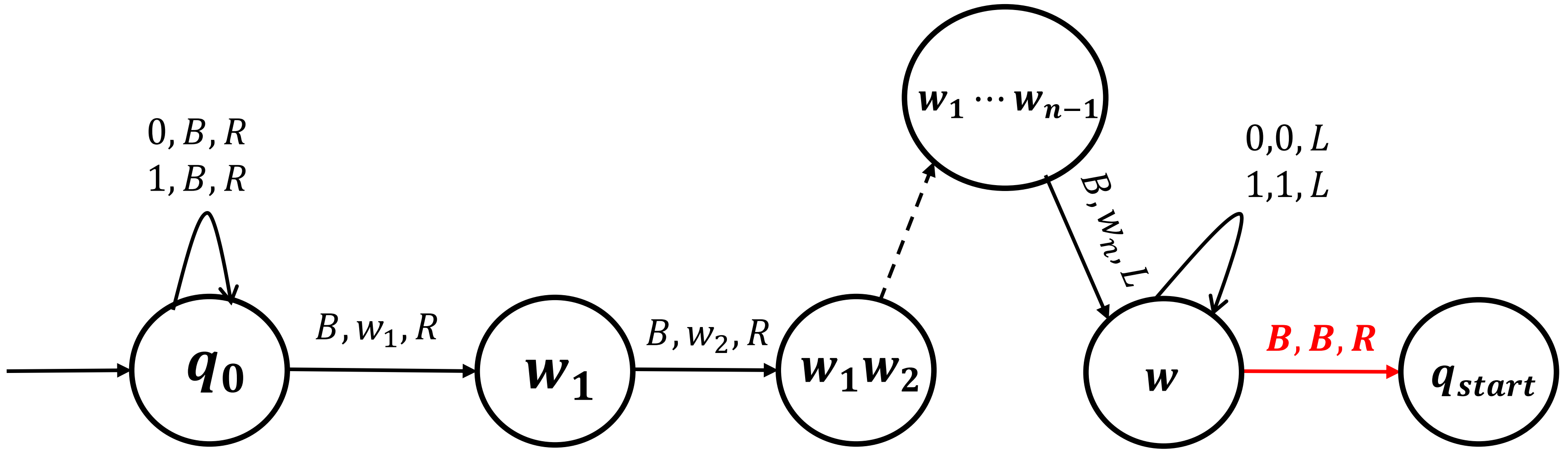


$\langle \boldsymbol{M, w}\rangle$
Construct $\langle \boldsymbol{T_{\langle M,w\rangle}}\rangle$ from $\langle \boldsymbol{M, w}\rangle$
$\langle \boldsymbol{T_{\langle M,w\rangle}}\rangle$
$\boldsymbol{T_E}$: Decider for $\boldsymbol{E_{TM}}$
ACCEPT
REJECT



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

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



Undecidability

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

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

- $\boldsymbol{\overline{A_{TM}}} \preccurlyeq \boldsymbol{E_{TM}}$
- $\boldsymbol{\overline{A_{TM}}}$ **is undecidable**
- $\boldsymbol{E_{TM}}$ **is undecidable!**

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

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

```
C = On input ⟨M⟩
    • For i = 1, 2, 3, ⋯
        • For j = 1, 2, 3, ⋯ i
              Run M on s_j for i steps.
              If M accepts s_j, REJECT.
```

Thank You!