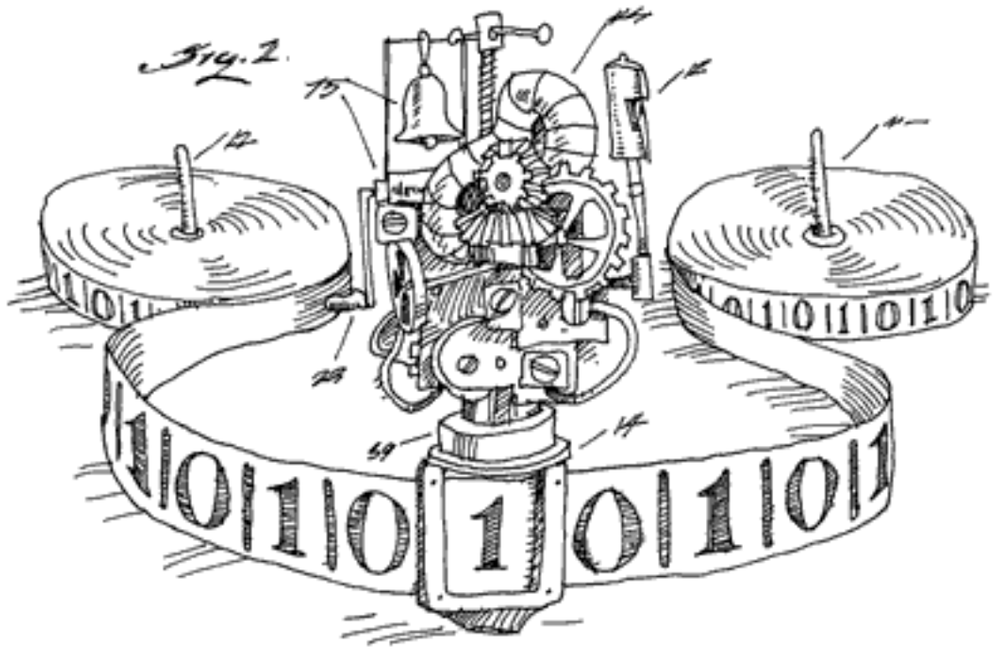
CSC4010: Introduction to Theoretical Computer Science



Lecture 7

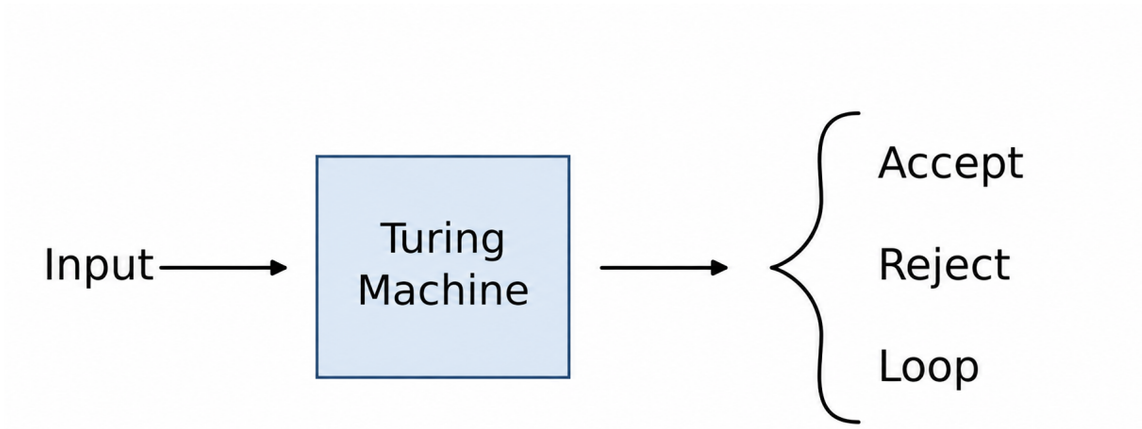
Last Time

- **Church-Turing Thesis:** The Turing Machine model is the **“correct”** way of modeling arbitrary computation. The informal concept of an **“algorithm”** is successfully modeled by the **rigorous definition of a Turing Machine.**

- We added other “features” to the vanilla model. **None of these changes have any affect on the model.**

- We can treat **“program as data”** and feed it to a **Universal Turing Machine to simulate any Turing Machine.**

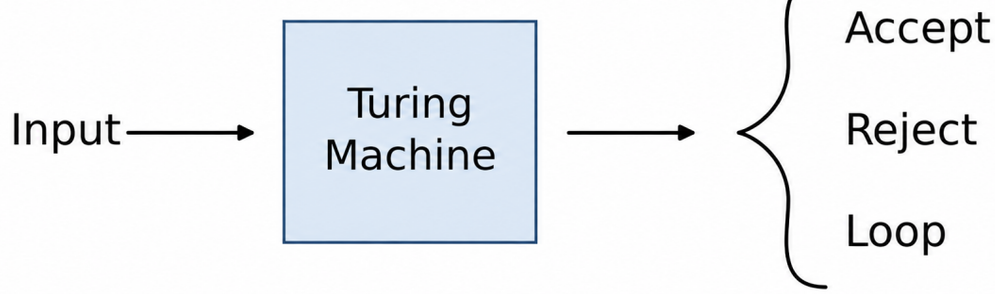
Recall



Recall

- The Turing Machine M recognizes the language $L \subseteq \{0,1\}^*$ if for every $x \in \{0,1\}^*$, M accepts x if and only if $x \in L$.

- Note that by this definition, the machine M can reject or loop forever on inputs $x \notin L$.



- A language L is recognizable if there exists a Turing Machine M that recognizes it. Otherwise, it is unrecognizable.

- A language L is decidable if and only if there exists a Turing Machine M that decides it. Otherwise, it is undecidable.

Are all problems decidable?

1928: The *Entscheidungsproblem*

Wilhelm Ackermann
1896-1962

The "Decision Problem":

Is there an algorithm which takes as input a formula (in first-order logic) and decide whether it is logically valid?

David Hilbert
1862-1943

1936: Solution to the *Entscheidungsproblem*

Alonzo Church

"An unsolvable problem of elementary number theory."

Model of Computation: λ-calculus (CSC3120)

Alan Turing

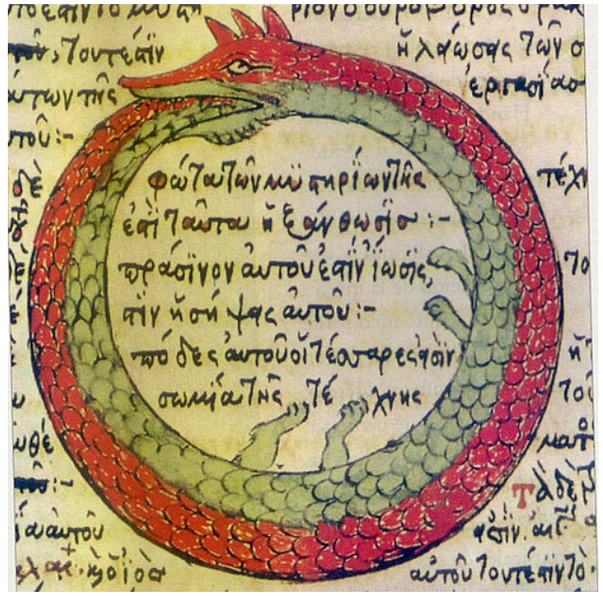
"On computable numbers, with an application to the Entscheidungsproblem."

Model of Computation: Turing Machines

Turing Machines Analyzing Turing Machines…

- **Recall:** A Turing Machine M has an encoding $\langle M \rangle \in \{0,1\}^*$.
- Then $\langle M \rangle$ can be the input for another Turing Machine (such as the Universal Turing Machine!)
- Compilers, syntax highlighters, linters, etc.

Self - Rejecting Turing Machines



- Let M be a Turing Machine.
- What if we run M on $\langle M \rangle$? Strange, but legal.
- Three possibilities:
  - M accepts $\langle M \rangle$.
  - M rejects $\langle M \rangle$.
  - M loops on $\langle M \rangle$.
- **Definition:** A Turing Machine M is said to be self-rejecting if M rejects $\langle M \rangle$.

Self - Rejecting Turing Machines

Self - Rejecting Turing Machines

- Let $L_{\text{self}} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$.

Self - Rejecting Turing Machines

- Let $L_{\text{self}} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$.

Theorem: L_{self} is undecidable.

Self - Rejecting Turing Machines

- Let $L_{\text{self}} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$.

Theorem: L_{self} is undecidable.

Proof: Suppose L_{self} is a decidable language, decided by Turing Machine M_{self}:

Self - Rejecting Turing Machines

- Let $L_{\text{self}} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$

Theorem: L_{self} is undecidable.

M_{self} on i/p $\langle M \rangle$ = { accept if M($\langle M \rangle$) rejects.
reject if M($\langle M \rangle$) accepts/loops

Proof: Suppose L_{self} is a decidable language, decided by Turing Machine M_{self}:

Self - Rejecting Turing Machines

- Let $L_{\text{self}} = \{\langle M \rangle : M \text{ rejects } \langle M \rangle\}$

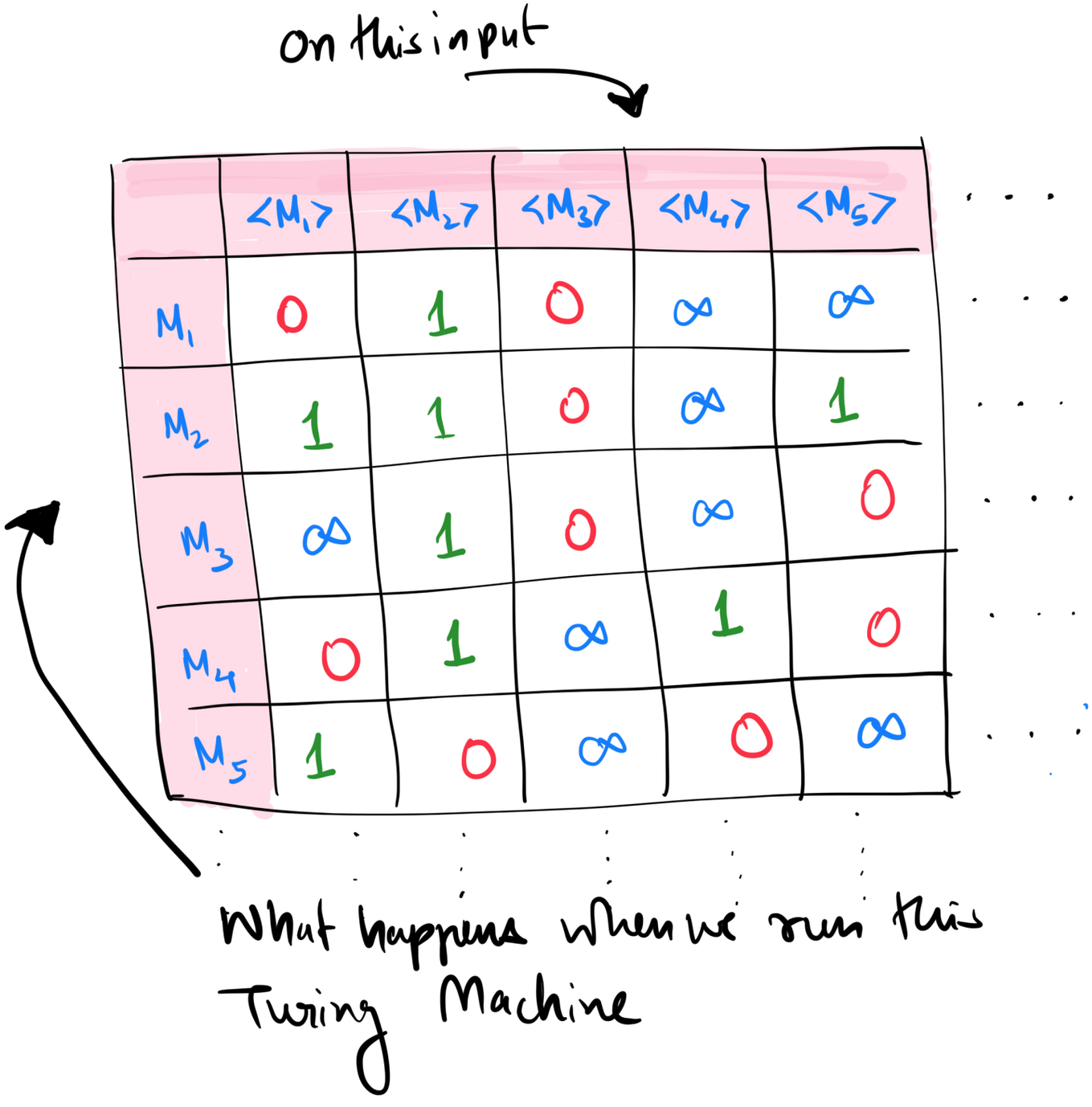
Theorem: L_{self} is undecidable.

M_{self} on i/p $\langle M \rangle$ = { accept if M($\langle M \rangle$) rejects. ; reject if M($\langle M \rangle$) accepts/loops

Proof: Suppose L_{self} is a decidable language, decided by Turing Machine M_{self}:

- Run M_{self} on $\langle M_{\text{self}} \rangle$.
- If M_{self} rejects $\langle M_{\text{self}} \rangle$, then $\langle M_{\text{self}} \rangle \in L_{\text{self}}$.
- If M_{self} does not reject $\langle M_{\text{self}} \rangle$, then $\langle M_{\text{self}} \rangle \notin L_{\text{self}}$.

Visualizing the Proof Using Diagonalization

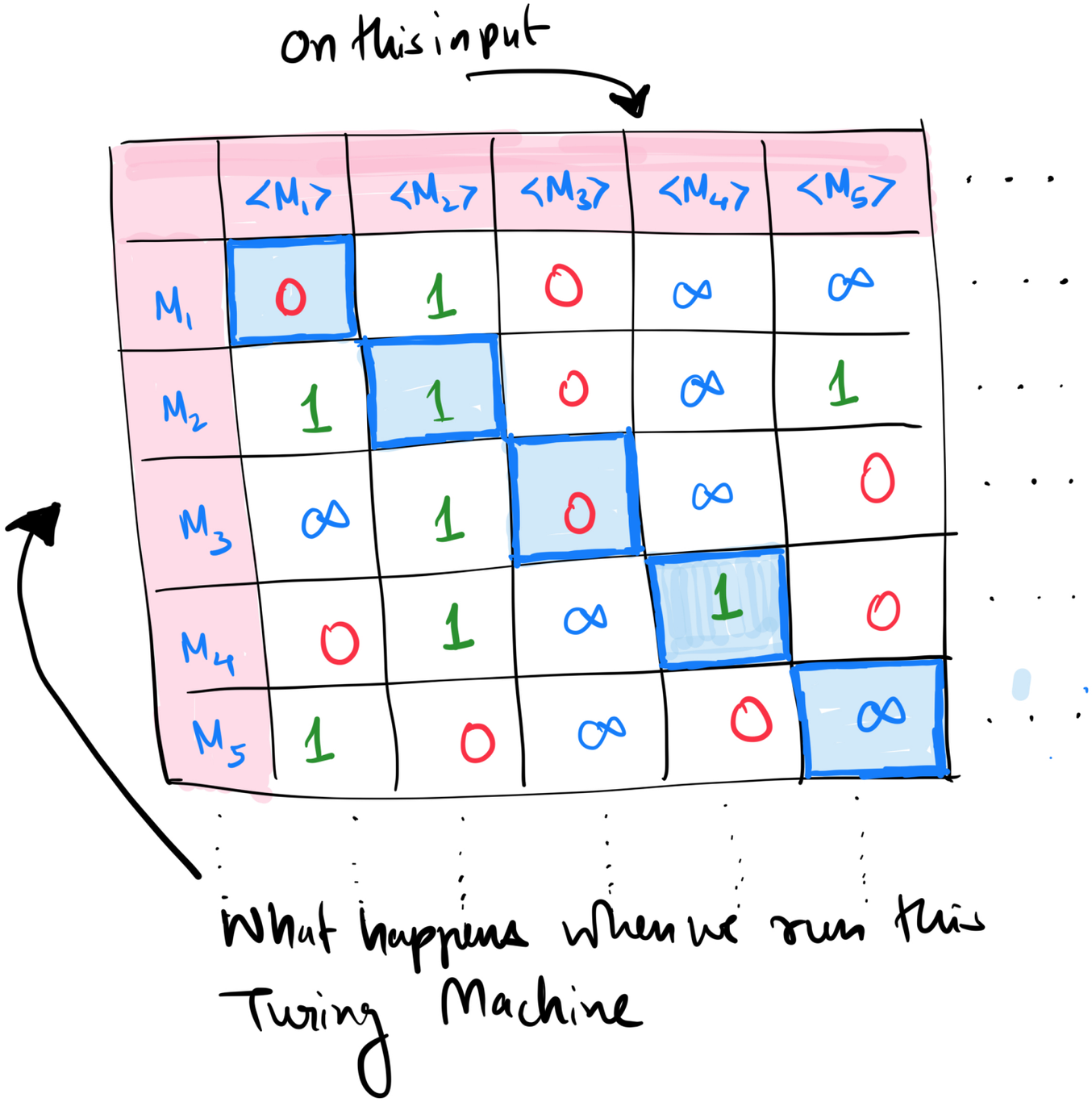


1 = Accept

0 = Reject

∞ = Loop.

Visualizing the Proof Using Diagonalization



M_{self} on i/p $\langle M \rangle$ = { accept if $M(\langle M \rangle)$ rejects. / reject if $M(\langle M \rangle)$ accepts/loops

1 = Accept

0 = Reject

∞ = Loop.

The Accepting Problem: A_{TM}

- **Informal Problem Statement:** Given a Turing Machine M and an input w determine whether M accepts w .

- **The same problem, formulated as a language:**

$$A_{TM} = \{\langle M, w\rangle : M \text{ accepts } w\} .$$

- A_{TM} and L_{self} is about predicting Turing Machines' behavior.

Theorem: A_{TM} is undecidable.

The Accepting Problem: A_{TM}

Theorem: A_{TM} is undecidable.

- **Proof:** Proof by contradiction. Assume there exists H deciding A_{TM}. Use H on $\langle M, w \rangle$ and accept if M accepts w, else reject.

Use H to construct Turing Machine D:

$D =$ ``On input $\langle M \rangle$

1. Simulate H on input $\langle M, \langle M \rangle \rangle$.
2. Accept if H rejects. Reject if H accepts."

D accepts $\langle M \rangle \iff M$ doesn't accept $\langle M \rangle$.

D accepts $\langle D \rangle \iff D$ doesn't accept $\langle D \rangle$.

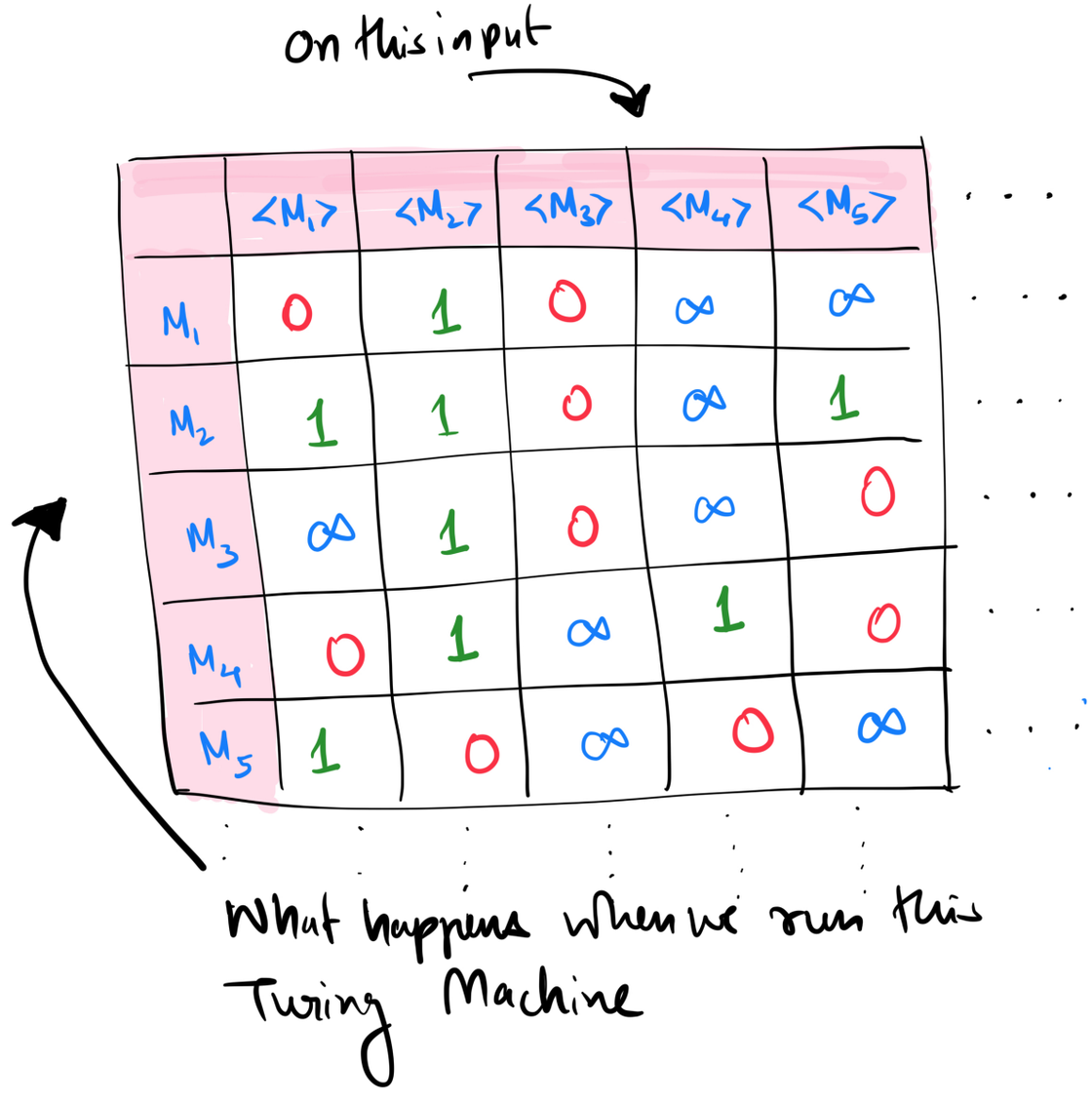
The Accepting Problem: A_{TM}

Theorem: A_{TM} is undecidable.

Use H to construct Turing Machine D:

$D =$ ``On input $\langle M \rangle$

1. Simulate H on input $\langle M, \langle M \rangle \rangle$.
2. Accept if H rejects.
   Reject if H accepts."



Are all problems recognizable?

Undecidable language A_{TM} is recognizable

Theorem: A_{TM} is recognizable.

Undecidablility and Unrecognizability

Theorem: L and $\bar{L}$ are recognizable if and only if L is decidable.

$\bar{A}_{TM}$ is unrecognizable

Theorem: $\bar{A}_{TM}$ is unrecognizable.

Another Example

The Halting Problem

- Let $\text{HALT}_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$.

Theorem: HALT_{TM} is undecidable.

- **Proof:** Proof by contradiction. Assume there exists H deciding HALT_{TM}.

The Halting Problem

- Let $\text{HALT}_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$.

Theorem: HALT_{TM} is undecidable.

- **Proof:** Proof by contradiction. Assume there exists H deciding HALT_{TM}.

Idea: Use H to construct a Turing Machine M' deciding A_{TM}.

The Halting Problem

- Let $\text{HALT}_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$.

Theorem: HALT_{TM} is undecidable.

- **Proof:** Proof by contradiction. Assume there exists H deciding HALT_{TM}.

Idea: Use H to construct a Turing Machine M' deciding A_{TM}.

$M' =$ ``On input $\langle M, w\rangle$

1. Simulate H on input $\langle M, w\rangle$.
2. Reject if H rejects.
3. If H accepts, run M on w until it halts.
   - **(a)** If M accepts, then accept.
   - **(b)** If M rejects, then reject."

The Halting Problem

- Let $\text{HALT}_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$.

Theorem: HALT_{TM} is undecidable.

- **Proof:** Proof by contradiction. Assume there exists H deciding HALT_{TM}.

Idea: Use H to construct a Turing Machine M' deciding A_{TM}.

$M' =$ ``On input $\langle M, w\rangle$

1. Simulate H on input $\langle M, w\rangle$.
2. Reject if H rejects.
3. If H accepts, run M on w until it halts.
   - **(a)** If M accepts, then accept.
   - **(b)** If M rejects, then reject."

Claim: If H exists, then M' decides A_{TM}.

The Halting Problem

The Halting Problem

M' = "On input $\langle M, w \rangle$

1. Simulate H on input $\langle M, w \rangle$.
2. Reject if H rejects.
3. If H accepts, run M on w until it halts.
   - **(a)** If M accepts, then accept.
   - **(b)** If M rejects, then reject."

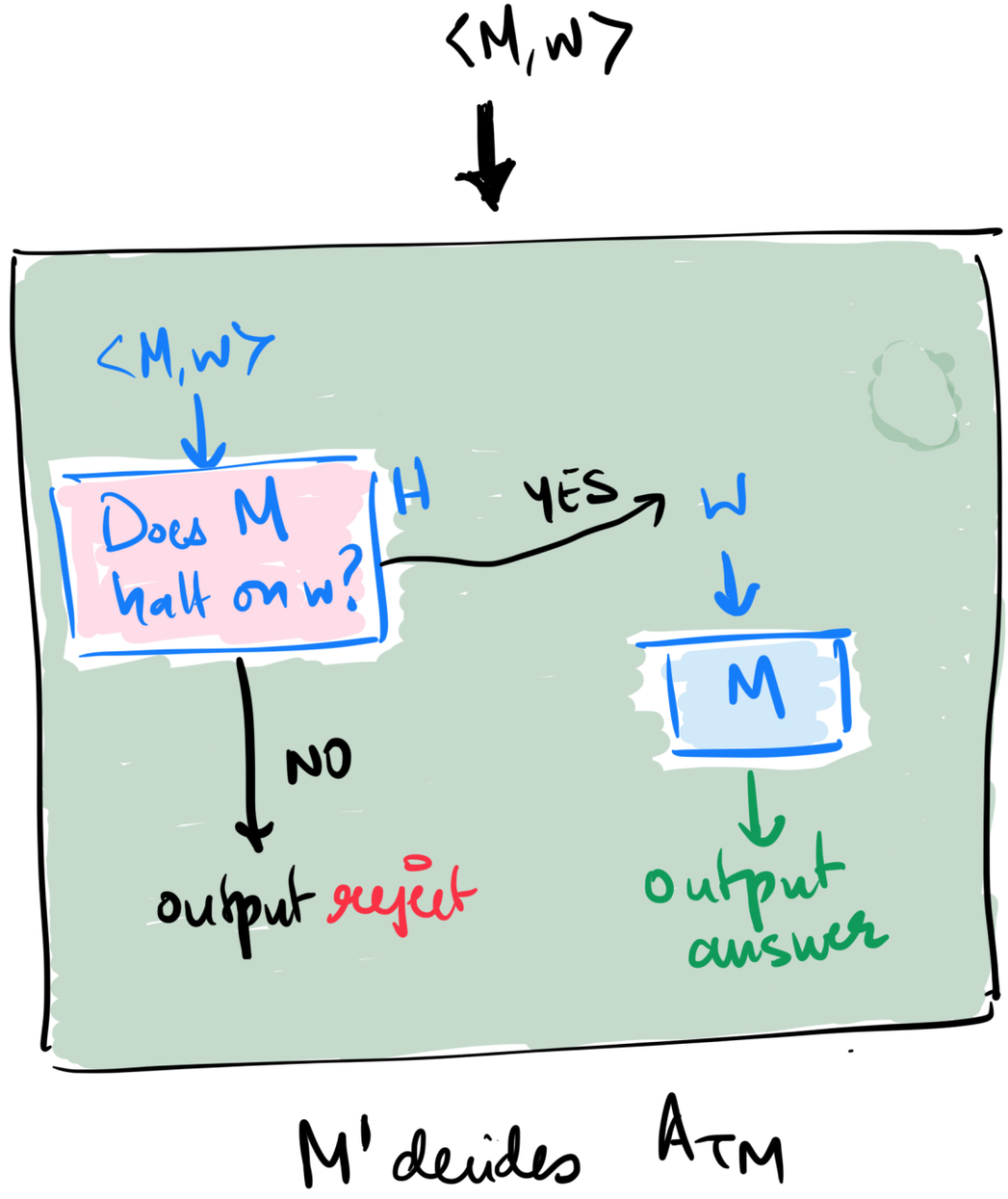
The Halting Problem

M' = "On input $\langle M, w \rangle$

1. Simulate H on input $\langle M, w \rangle$.
2. Reject if H rejects.
3. If H accepts, run M on w until it halts.
   (a) If M accepts, then accept.
   (b) If M rejects, then reject."

Claim: Suppose H decides HALT_{TM} . If H exists, then M' decides A_{TM}.

The Halting Problem



This is called a Turing Reduction: Using a TM to decide HALT_{TM} we can decide A_{TM}.

Reducibility

- We already prove that A_{TM} is undecidable (by diagonalization).
- Our goal was to show HALT_{TM} is undecidable.
- We showed if HALT_{TM} is decidable, then so is A_{TM} .
  - Design an algorithm for A_{TM} using the hypothetical algorithm for HALT_{TM}.
- "Reduction from A_{TM} to HALT_{TM}."

Reducibility - Concept

- If we have two languages (or problems) A and B, then A being reducible to B implies we can use B to solve A .

- **Example:** Measuring the area of a rectangle is reducible to measuring the lengths of its sides.

- If A is reducible to B, then solving B gives a solution to A .
  - Then, B is easy implies A is easy.
  - Then, A is hard implies B is hard.

This is the form we will use.

Summary

- We showed that L_{self}, and A_{TM} are undecidable via diagonalization.
- A language L is decidable if and only if L and $\bar{L}$ are both recognizable. We used this to conclude that $\bar{A}_{TM}$ is undecidable.
- We showed HALT_{TM} is undecidable by reducing A_{TM} to HALT_{TM}.