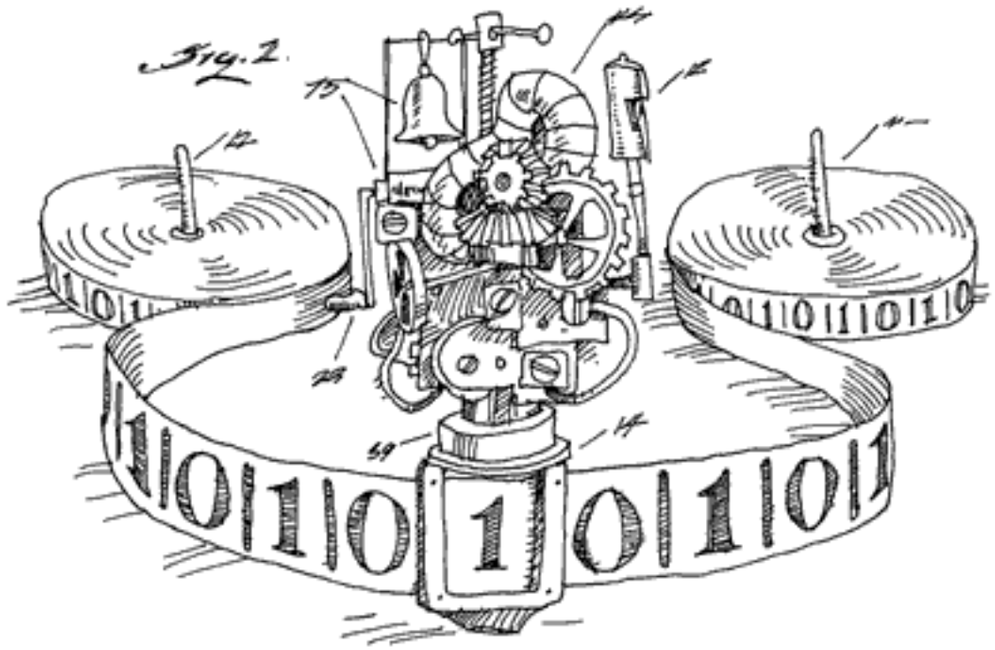
CSC4010: Introduction to Theoretical Computer Science



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.** Any language L that's accepted by the enhanced models was also accepted by the TM (or diverges)

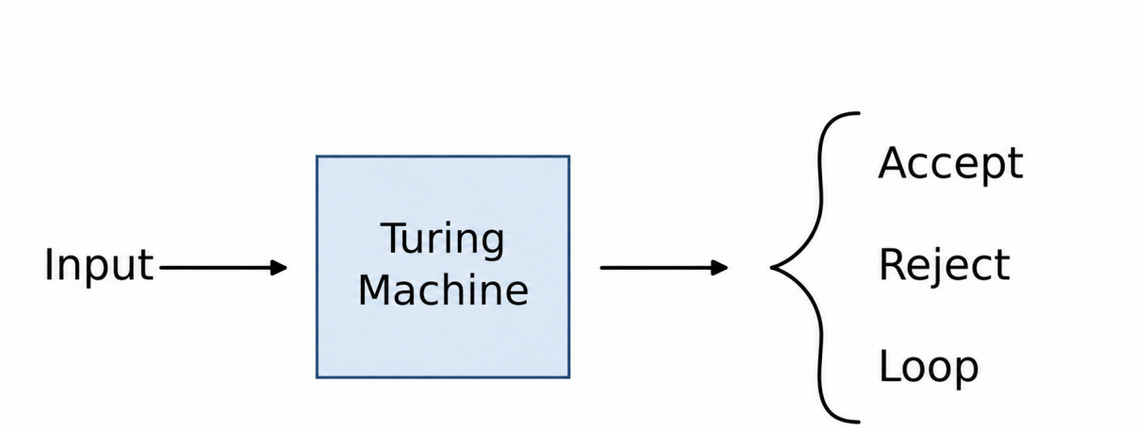
- We can treat **"program as data"** and feed it to a **Universal Turing Machine to simulate any Turing Machine.**

→ given a TM $M = (Q, \Sigma, q_0, q_{acc}, q_{rej}, \sqcup, \delta)$.
$\langle M \rangle$ is the encoding.

* Being able to encode a TM as $\langle M \rangle$.

* given $\langle M \rangle$, another TM can simulate it.

Recall

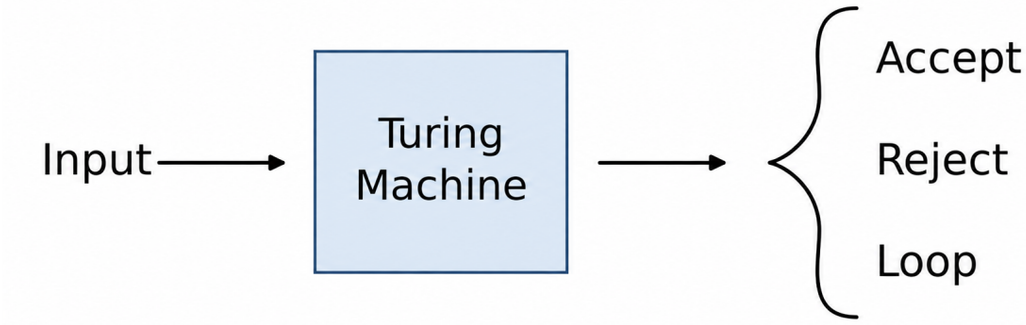


Recall

- 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 $x \in L$.

Every Turing machine M, recognizes $L(M) = \{x \mid M(x)=1\}$

- 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.

$\Rightarrow$ M decides L if 1) M halts on all i/p.
2) $M(x)=1 \Leftrightarrow x \in L$.

Are all problems decidable?

1) Is there a language not solvable in any computational model?

2) What is this language?

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?

"Is there an algorithm that takes in any mathematical statement, and says whether or not it is true"!



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 Machine ≈ λ-calculus model.

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 $\langle M \rangle$.

corresponding to M:
L(M) which it recognizes.
$\langle M \rangle \in L(M)$ or not?

$\langle M \rangle \notin L(M)$ and M halts on $\langle M \rangle$.

- **Definition:** A Turing Machine M is said to be self-rejecting if M rejects $\langle M \rangle$.

$M(\langle M \rangle) = 0$ (halts)
(rejects).

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

M (<M>) = 0 (and it halts).

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

Theorem: L_{self} is undecidable.

Self - Rejecting Turing Machines

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

M_{self} on it $\langle M \rangle$ = { accept if M rejects $\langle M \rangle$; reject if M accepts/loop on $\langle M \rangle$

Theorem: L_{self} is undecidable.

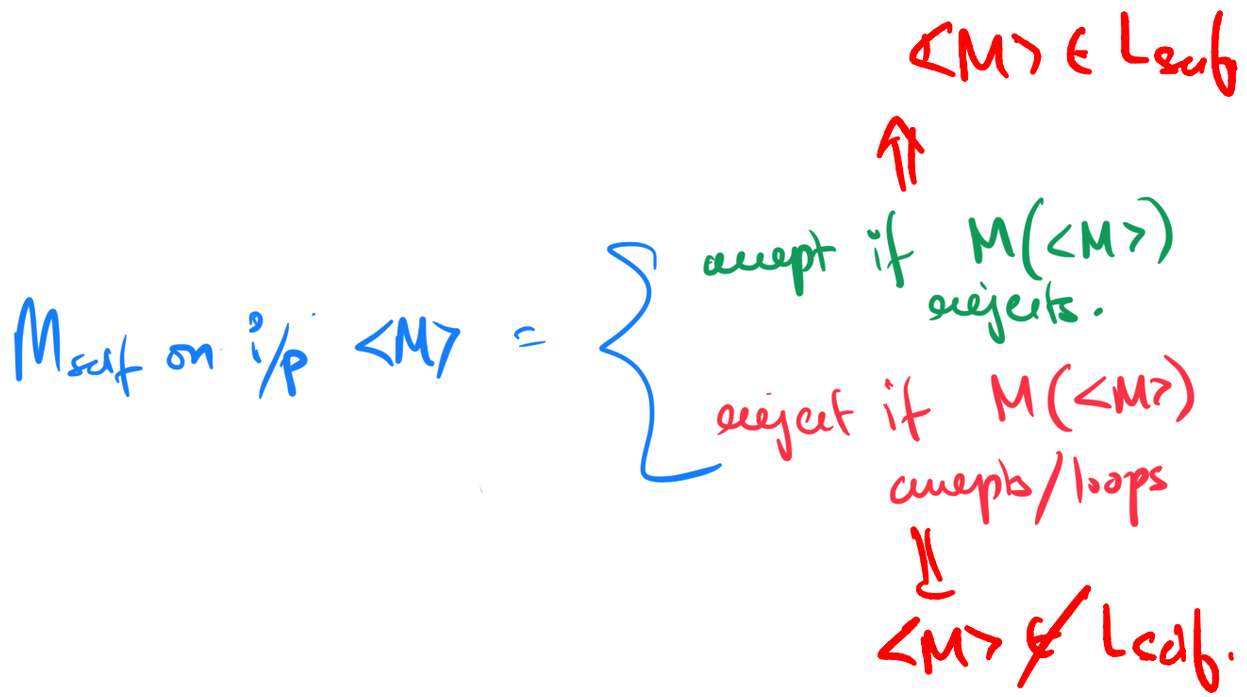
(Contradiction)

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.



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

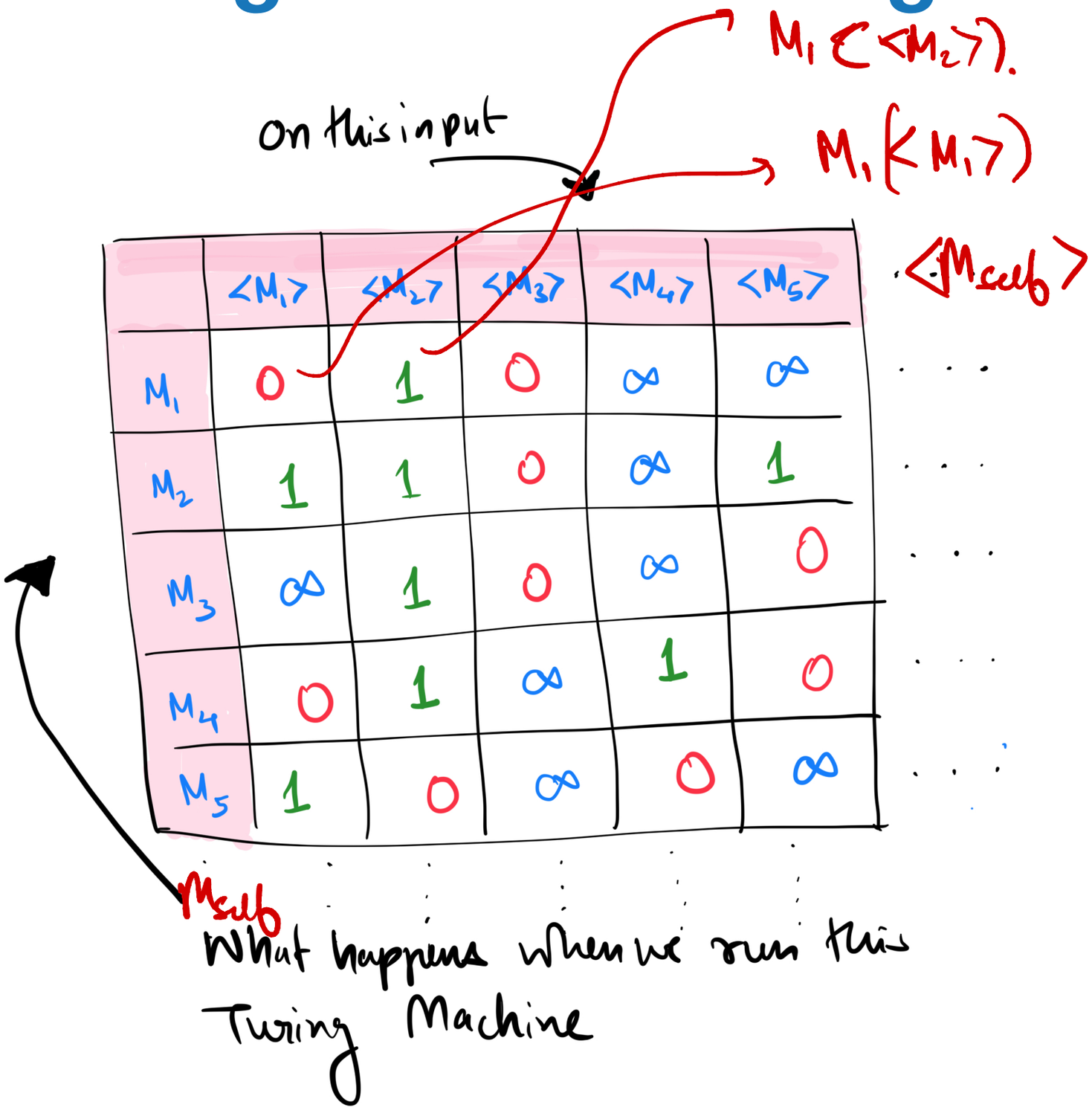
Proof: Suppose L_{self} is a decidable language, decided by Turing Machine M_{self}. ⇒ takes in any tM description and therefore also $\langle M_{self} \rangle$.

- Run M_{self} on $\langle M_{\text{self}} \rangle$.
- If M_{self} rejects $\langle M_{\text{self}} \rangle$, then $\langle M_{\text{self}} \rangle \in L_{\text{self}}$. !! But M_{self} is a decider of L_{self}, so, it should accept $\langle M_{self} \rangle$
- If M_{self} does not reject $\langle M_{\text{self}} \rangle$, then $\langle M_{\text{self}} \rangle \notin L_{\text{self}}$.

↓

and M_{self} is a decider for L_{self}, M_{self} rejected $\langle M_{self} \rangle$.

Visualizing the Proof Using Diagonalization



1 = Accept

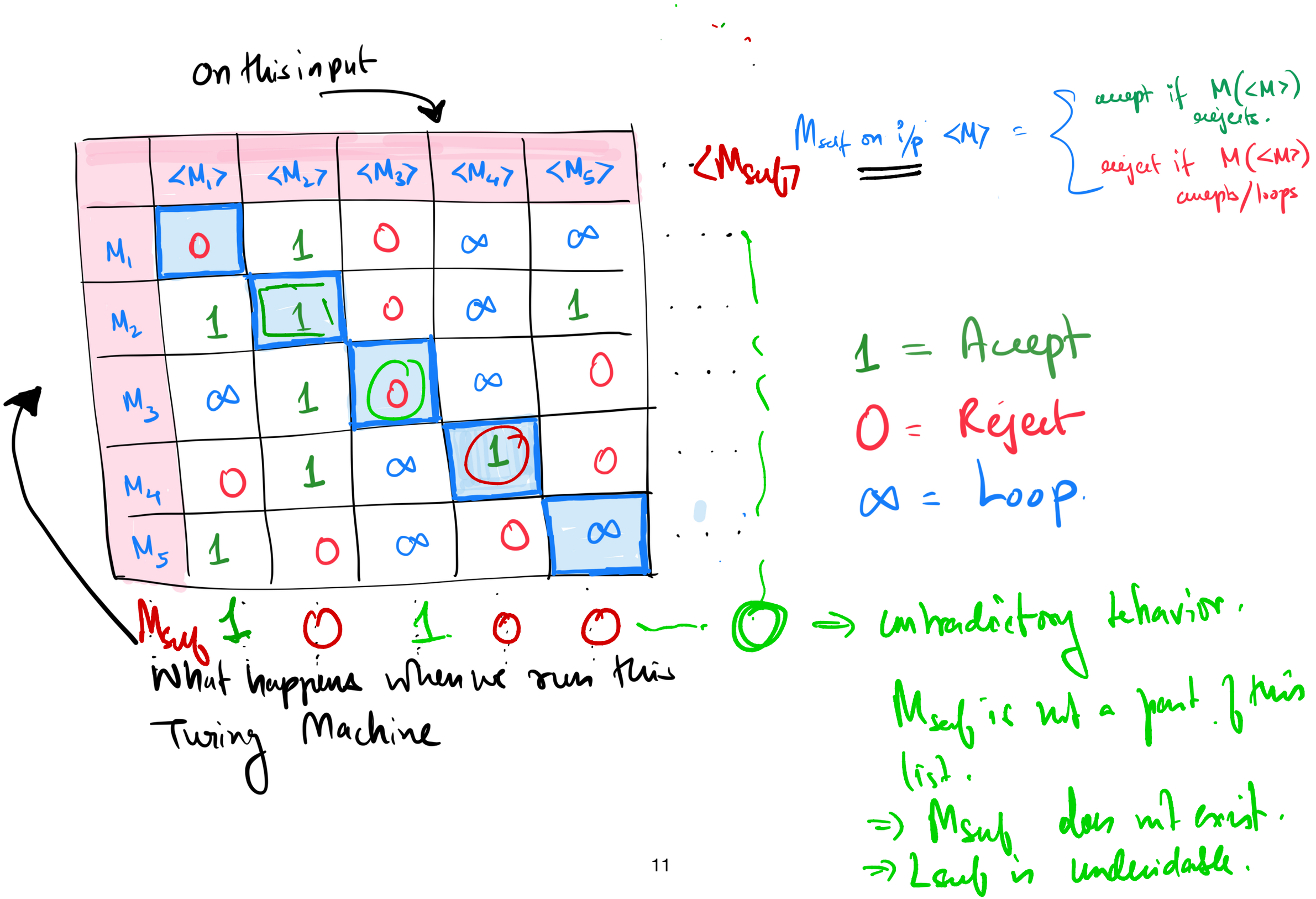
0 = Reject

∞ = Loop.

$Loop$ is decidable

$\Rightarrow M_{self}$ is a decider.

Visualizing the Proof Using Diagonalization



The Accepting Problem: A_{TM}

- **Informal Problem Statement:** Given a Turing Machine M and an input w determine whether M accepts w . ⇒ "kind of like a program checker".

- **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.

⇒ no "perfect" program checker.

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.

H on $\langle M, w \rangle$: { accepts if M accepts w; rejects o.w.

Use H to construct Turing Machine D:

D = "On input $\langle M \rangle$ (M halts.)

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

(is a TM.)

→ H rejects $\langle M, \langle M \rangle \rangle$ ⇒ M does not accept $\langle M \rangle$.

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

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

→ contradiction.

⇒ D can't exist.

⇒ H can't exist ⇒ 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 $\langle D \rangle$.

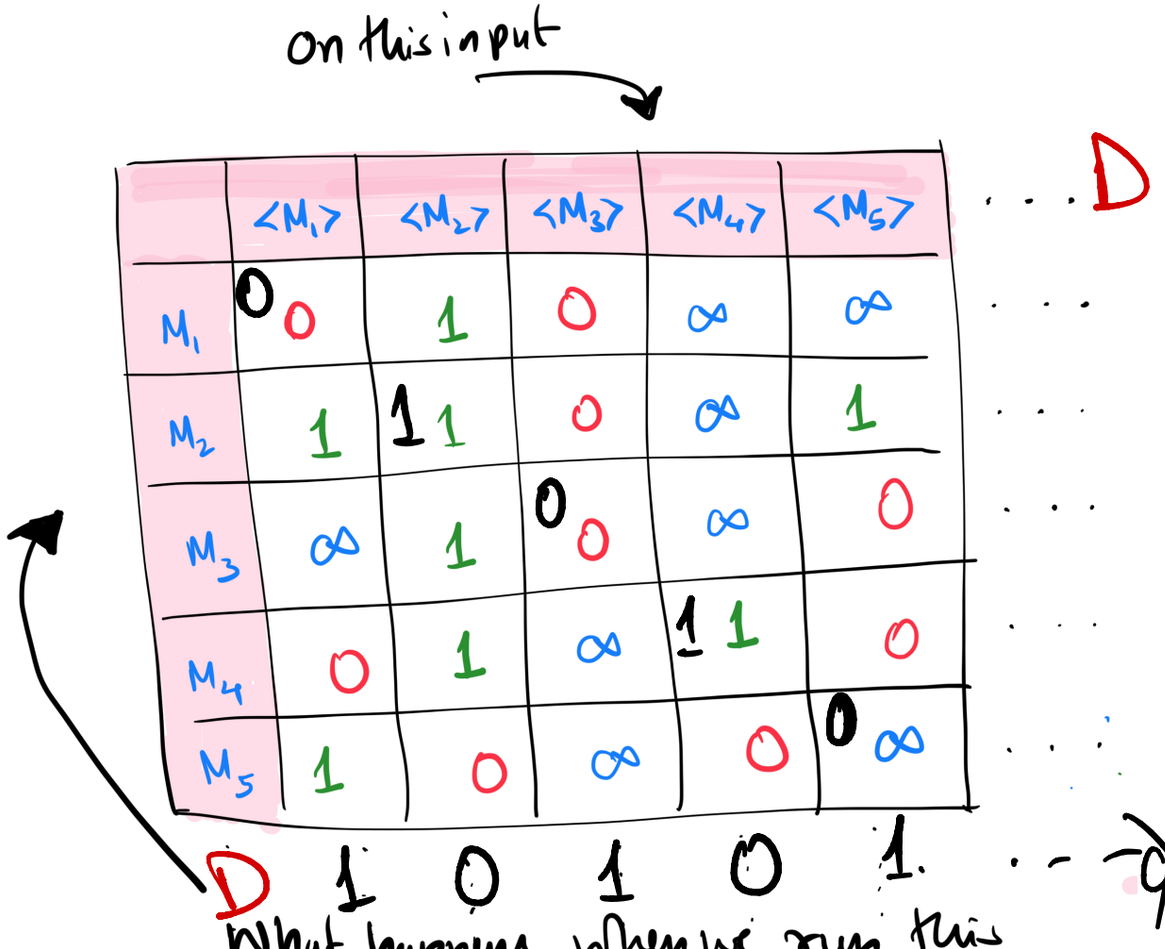
The Accepting Problem: A_{TM}

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."



D cannot exist
⇒ H cannot exist
⇒ A_{TM} is undecidable.

⇐ Entry D, ⟨D⟩ can be 0/1. But run row and the diagonal outputs opposite values.

Are all problems recognizable?

Undecidable language A_{TM} is recognizable

Theorem: A_{TM} is recognizable.

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

1) A recognizer for A_{TM} runs M on w, and accepts if M accepts w

(2) rejects if M rejects w

(3) loops if M loops on w.

Undecidablility and Unrecognizability

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

$\Leftarrow$ L is decidable then, the decider M recognizes L, and if accept/reject state of M were exchanged, then, you also recognize $\bar{L}$.

$\Rightarrow$ T_L and $T_{\bar{L}}$ are the recognizers, decider M, will run T_L and $T_{\bar{L}}$ in parallel on input x:

(1) x is accepted by T_L, then, accept

(2) x is accepted by $T_{\bar{L}}$ then, reject.

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

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

$= \{ \langle M, w \rangle \mid M \text{ does accept } w \}$.

(1) A_{TM} is undecidable.

(2) A_{TM} is recognizable.

(3) If $\bar{A}_{TM}$ is also recognizable $\Rightarrow$ A_{TM} is also decidable.

(not true.)

$\Rightarrow$ $\bar{A}_{TM}$ is not recognizable.

Another Example

The Halting Problem [Turing]

- 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}.

H on i/p <M,w> = { accept if M halts on w ; rej if it loops }

A_{TM} = { <M,w> | M accepts w }

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."

M halts on w

Claim 1: M' decides A_{TM}. ⇒ But A_{TM} is undecidable ⇒ M' cannot exist, H can't exist.

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}.

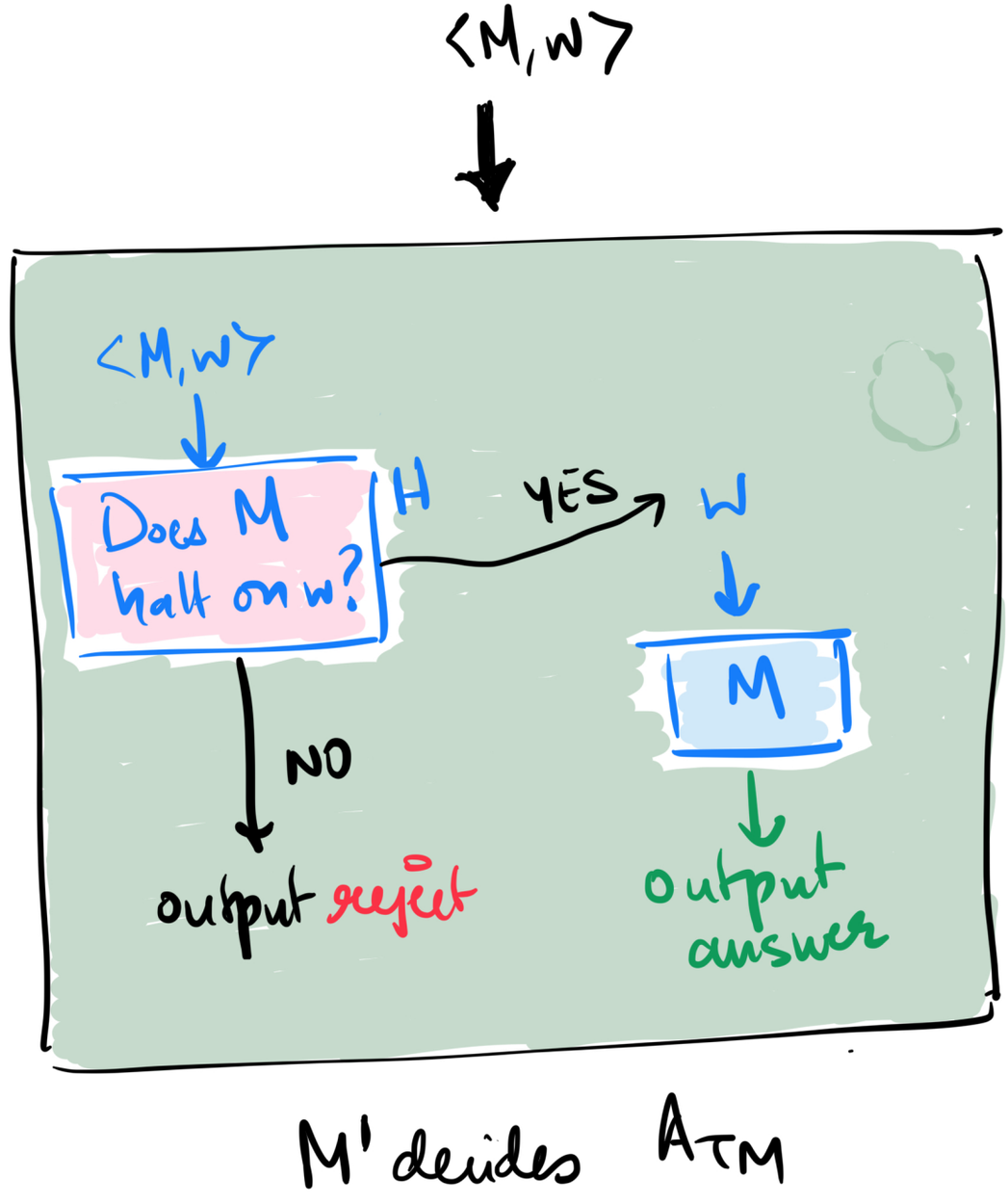
to see M' decides A_{TM}, consider 3 cases:

(1) $\langle M, w \rangle \in A_{TM}$ ⇒ M accepts w. In this case, H accepts $\langle M, w \rangle$ and then, in 3a) M accepts w and M' outputs accept

(2) $\langle M, w \rangle \notin A_{TM}$, and M rejects w. In this case H accepts $\langle M, w \rangle$ and then, in 3b) M rejects w and we reject.

(3) $\langle M, w \rangle \notin A_{TM}$ and M loops on w. In this case, H rejects $\langle M, w \rangle$ and we reject.

The Halting Problem



This is called a Turing Reduction: Using a TM to decide HALT_{TM} we can decide A_{TM}. Reduced the problem of solving A_{TM} to HALT_{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.

"A_{TM} is hard $\Rightarrow$ $HALT_{TM}$ is hard" as A_{TM} can be reduced to $HALT_{TM}$.

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}.