Lecture 17: Undecidable Languages

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Outcomes of TM Computation

Possible outcomes of a TM on an input string $\boldsymbol{w}$ are:

- **Accept** (halt and accept)
- **Reject** (halt and reject)
- **Loop infinitely** (considered **non-accept**, but **not a halt reject**)

A Turing machine that **halts on every input** (never loops) is called a **decider**

Turing-recognizable and Turing-decidable

If a language L is **recognized** by some TM, we call L **Turing-recognizable**

- **Halts and accepts** on input strings **in** L
- **Halts and rejects or loops infinitely** on inputs strings **not in** L

We say a language L is **Turing-decidable** or **decidable** if there exists a **decider** that recognizes L (we also say that M decides L)

- **Halts and accepts** on strings **in** L
- **Halts and rejects** on strings **not in** L
- Never loops infinitely

More Decidable Languages / Problems

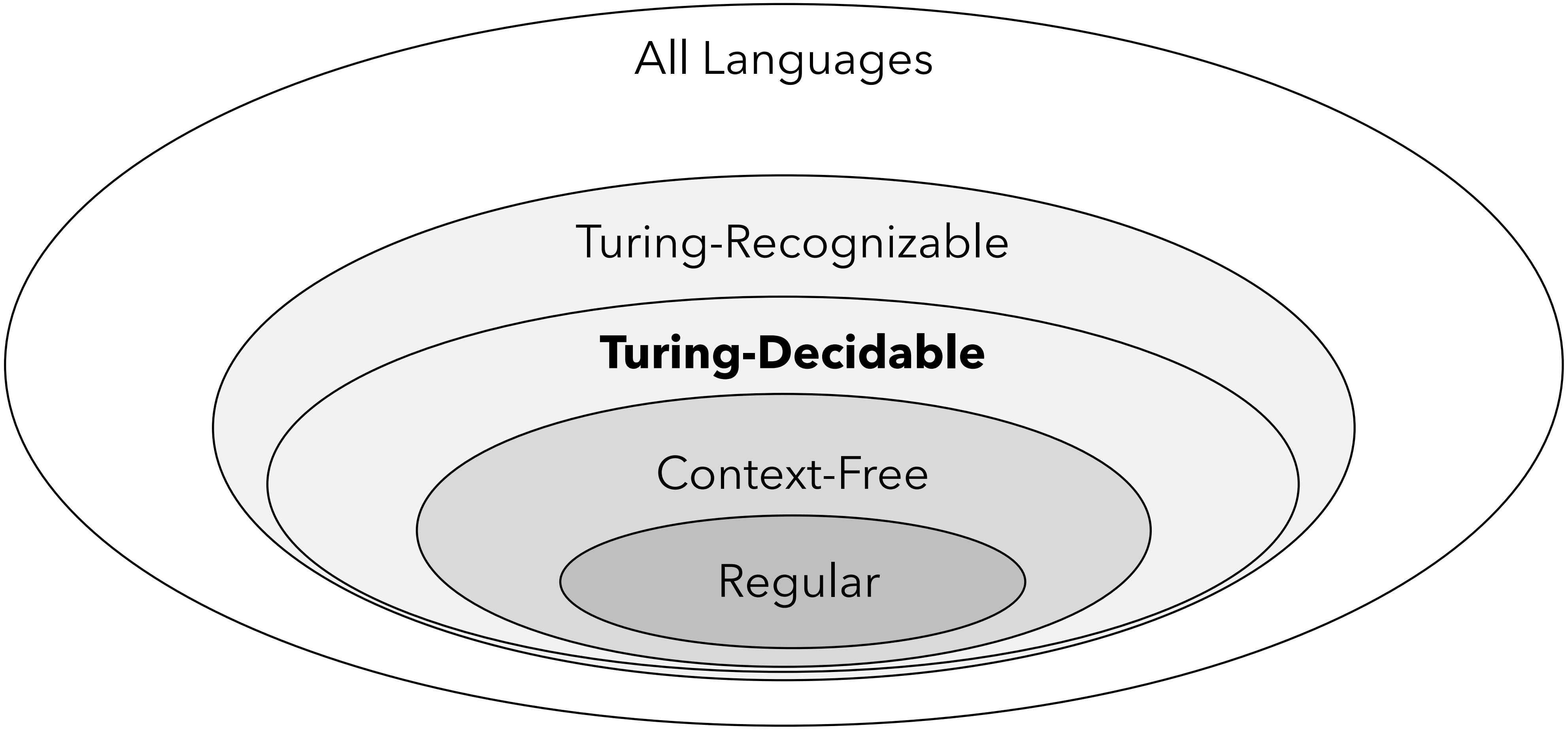
- $A_{DFA} = \{\langle D, w\rangle \mid D \text{ is a } \mathbf{DFA} \text{ that accepts input string } \mathbf{w}\}$
  - Decider takes as input the encoding of a DFA D and a string w and **accepts** if the DFA D accepts the string w, or **rejects** otherwise

- $A_{NFA} = \{\langle N, w\rangle \mid N \text{ is an } \mathbf{NFA} \text{ that accepts input string } \mathbf{w}\}$

- $A_{REX} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$

- $E_{DFA} = \{\langle A\rangle \mid A \text{ is a } \mathbf{DFA} \text{ and } L(A) = \emptyset\}$

- $EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } \mathbf{B} \text{ are } \mathbf{DFAs} \text{ and } L(A) = L(B)\}$



Question: What languages are not Turing-decidable (undecidable), and how do we prove that a language is undecidable?

Barber Paradox

The **barber** **shaves everyone** who **doesn’t shave themselves**.
Who shaves the barber?

- If the **barber** **shaves themself**, then the **barber** **doesn’t shave the** **barber**…
- If the **barber** **doesn’t shave themself**, then the **barber** **shaves the** **barber**…

This is a **paradox**. This **barber** does not exist.

Undecidable Languages

- A language $\boldsymbol{L}$ is **undecidable** if there does not exist a decider which decides the language.

- That is, it is **impossible** creates a TM (decider) for $\boldsymbol{L}$ which
  - Halts and **accepts** on strings in $\boldsymbol{L}$
  - Halts and **rejects** on strings not in $\boldsymbol{L}$
  - Never loops infinitely on any input

- Our first undecidable language:

$$\boldsymbol{A_{TM}} = \{\langle \boldsymbol{M}, \boldsymbol{w} \rangle \mid \boldsymbol{M} \text{ is a TM and } \boldsymbol{M} \text{ accepts } \boldsymbol{w}\}$$

- The strings in $\boldsymbol{A_{TM}}$ are all pairs of **any TM $\boldsymbol{M}$ and string $\boldsymbol{w}$** where $\boldsymbol{M}$ accepts when run on input $\boldsymbol{w}$

Undecidable Languages

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

- To prove that A_{TM} is **undecidable**, we must prove that it is **impossible** to create a decider for A_{TM}

- That is, show it is impossible to create a TM H where:
  - H takes as input another TM M and a string w
  - H halts and **accepts** if M accepts w
  - H halts and **rejects** if M does not accept w (i.e. M rejects / loops on input w)
  - H **never loops forever**

- We will prove this by **contradiction**
  - If we assume a decider for A_{TM} exists, a contradiction occurs

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

A_{TM} is Undecidable

- **Assume for a contradiction that A_{TM} is decidable**

- That means, we assume a TM H exists where on input $\langle M, w\rangle$:
  - H halts and **accepts** if M accepts w
  - H halts and **rejects** if M does not accept (rejects or loops) on w

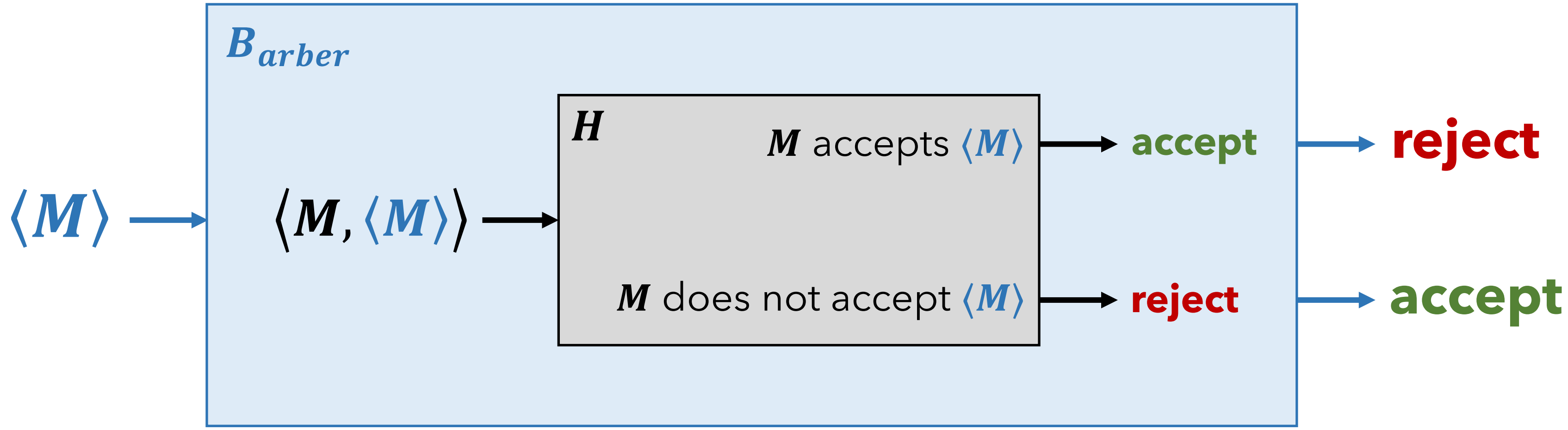
- Using H, we will build a different TM (decider) B_{arber}

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

A_{TM} is Undecidable

- Construct B_{arber} as follows:

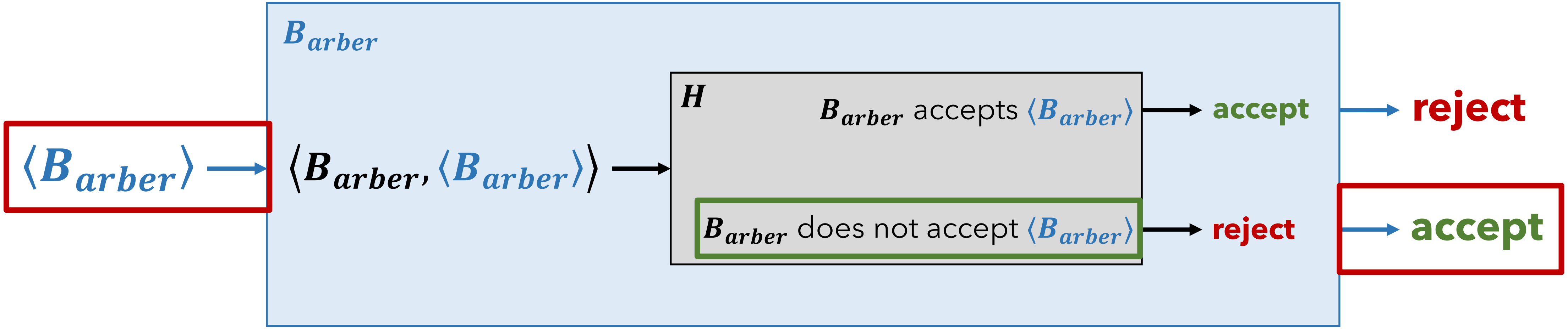
B_{arber} = "On input $\langle M\rangle$, where M is a TM
- Run H on input $\langle M, \langle M\rangle\rangle$
  - See if M accepts the string encoding of itself $\langle M\rangle$
- If H accepts, **reject**
- If H rejects, **accepts**



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

A_{TM} is Undecidable

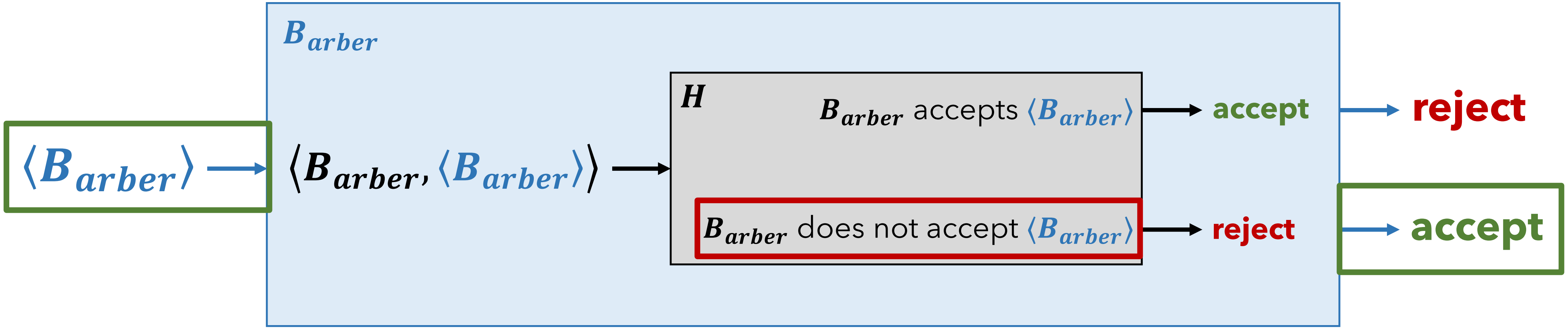
- What if we run B_{arber} on its own string encoding $\langle B_{arber}\rangle$?



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

A_{TM} is Undecidable

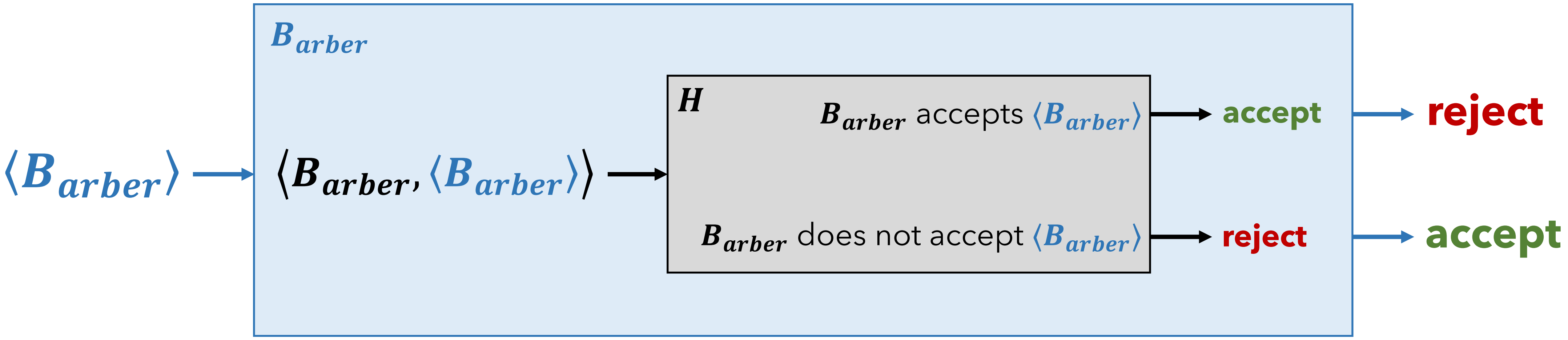
- What if we run B_{arber} on its own string encoding $\langle B_{arber} \rangle$?



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

A_{TM} is Undecidable

- What if we run B_{arber} on its own string encoding $\langle B_{arber}\rangle$?



- If H decides that B_{arber} **accepts** $\langle B_{arber}\rangle$, then B_{arber} **rejects** $\langle B_{arber}\rangle$
- If H decides that B_{arber} **does not accept** $\langle B_{arber}\rangle$, then B_{arber} **accepts** $\langle B_{arber}\rangle$
- This is a **contradiction**

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

A_{TM} is Undecidable Proof Summary

- Assume for a **contradiction** that A_{TM} is decidable
- Then there is a decider H which decides A_{TM}
  - On input $\langle M, w\rangle$, H **accepts** if M accepts w
    H **rejects** if M does not accept w

- Construct a decider B_{arber} which uses H as a subroutine
  - On input $\langle M\rangle$, B_{arber} runs H on input $\langle M, \langle M\rangle\rangle$
    B_{arber} **accepts** if H **rejects**
    B_{arber} **rejects** if H **accepts**

- Running B_{arber} on input $\langle B_{arber}\rangle$ results in a paradox
- Decider B_{arber} cannot exist, which means H cannot exist
- **Therefore, A_{TM} is undecidable**

Reductions

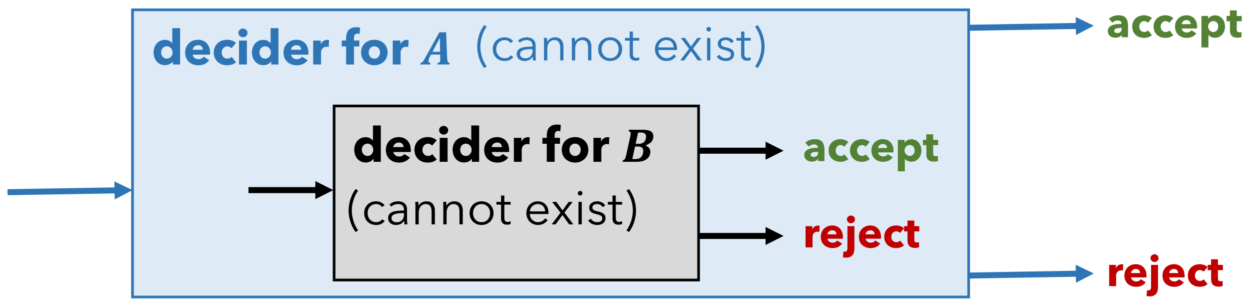
- We have now proven that A_{TM} **is undecidable** using **diagonalization**
- Now that we know that it is **impossible to create a decider for** A_{TM}
- We can use this knowledge to prove that **other languages are undecidable** using **reductions**

Reductions

Suppose **we know** a language $\boldsymbol{A}$ is undecidable and **we want to prove** that a language $\boldsymbol{B}$ is undecidable.

We show a **reduction from $\boldsymbol{A}$ to $\boldsymbol{B}$**:

- Assume for a contradiction that $\boldsymbol{B}$ is decidable (there exists a **decider for $\boldsymbol{B}$**)
- We show that if the **decider for B** exists, we could create a **decider for $\boldsymbol{A}$**
- However, from prior knowledge, we already know that $\boldsymbol{A}$ is undecidable, so a **decider for $\boldsymbol{A}$** cannot exist (**contradiction**)
- So, **decider for B** cannot exist. Therefore, **$\boldsymbol{B}$ is undecidable**.



The Halting Problem

- Does there exist a decider / algorithm, when **given a program with any input**, determines **if the program halts**?

$$\boldsymbol{Halt_{TM}} = \{\langle \boldsymbol{M}, \boldsymbol{w} \rangle \mid \boldsymbol{M} \text{ is a TM and } \boldsymbol{M} \text{ halts on input } \boldsymbol{w}\}$$

- We will show that $\boldsymbol{Halt_{TM}}$ undecidable. That is, no TM $\boldsymbol{R}$ exists where:
  - $\boldsymbol{R}$ takes as input another TM $\boldsymbol{M}$ and a string $\boldsymbol{w}$
  - $\boldsymbol{R}$ halts and **accepts** if $\boldsymbol{M}$ halts (i.e. accepts or rejects) when run on $\boldsymbol{w}$
  - $\boldsymbol{R}$ halts and **rejects** if $\boldsymbol{M}$ loops infinitely when run on $\boldsymbol{w}$
  - $\boldsymbol{R}$ **never loops infinitely**

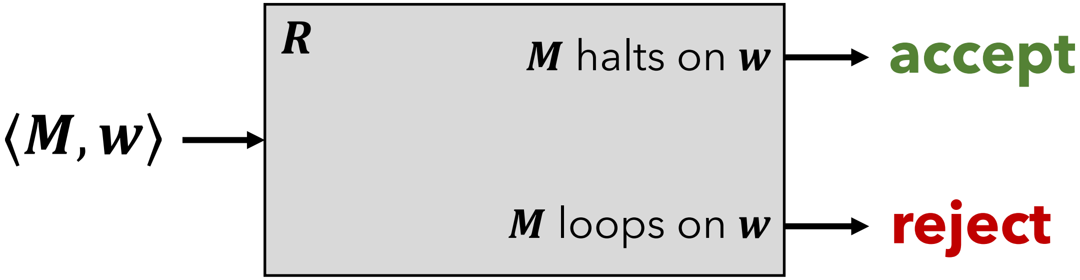
- We will prove that $\boldsymbol{Halt_{TM}}$ is undecidable using a reduction from $\boldsymbol{A_{TM}}$ to $\boldsymbol{Halt_{TM}}$

$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

$Halt_{TM}$ is Undecidable (Reduce A_{TM} to $Halt_{TM}$)

- **Assume for a contradiction that $Halt_{TM}$ is decidable**

- That means, we assume a TM R exists where on input $\langle M, w\rangle$:
  - R halts and **accepts** if M halts on input w
  - R halts and **rejects** if M loops infinitely on input w



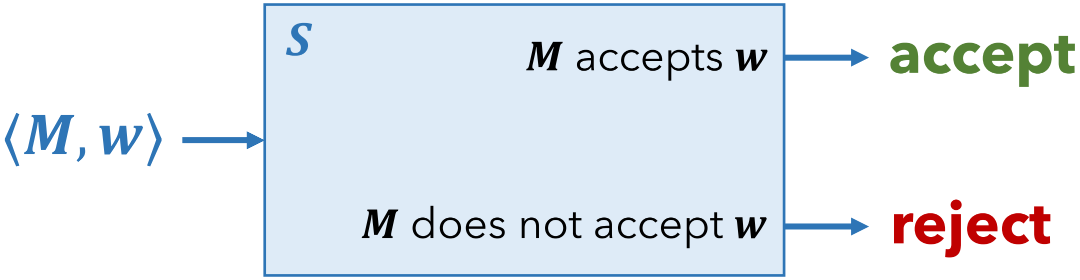
- We will show that using R, we can build a decider S for A_{TM}

$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

$Halt_{TM}$ is Undecidable (Reduce A_{TM} to $Halt_{TM}$)

- Recall, a hypothetical decider S for A_{TM} would do the following:

  On input $\langle M, w\rangle$:
  - **accepts** if M accepts w
  - **rejects** if M does not accept w (rejects or loops)



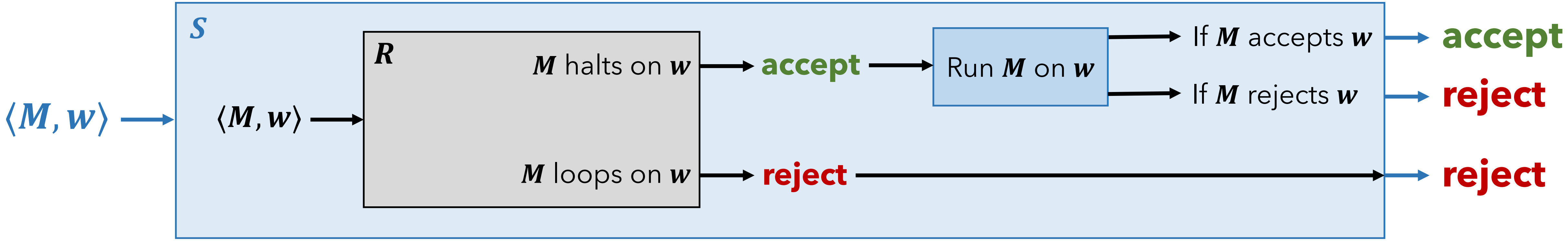
(doesn't actually exist)

- If can somehow build a decider S for A_{TM}, we would have a **contradiction** since A_{TM} is an **undecidable language**

$$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

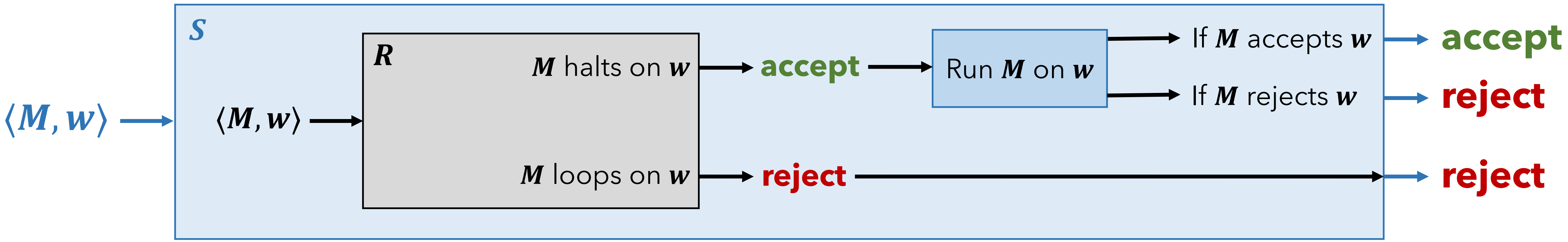
$Halt_{TM}$ is Undecidable (Reduce A_{TM} to $Halt_{TM}$)

- We can create S by using R as follows:
  - Run R on input $\langle M, w\rangle$
  - If R rejects, that means M loops on w, so S outputs **reject**
  - If R accepts, that means M halts (accept or reject) on w, so S runs M on w
    - If M accepts w, S outputs **accept**
    - If M rejects w, S outputs **reject**



$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

$Halt_{TM}$ is Undecidable (Reduce A_{TM} to $Halt_{TM}$)

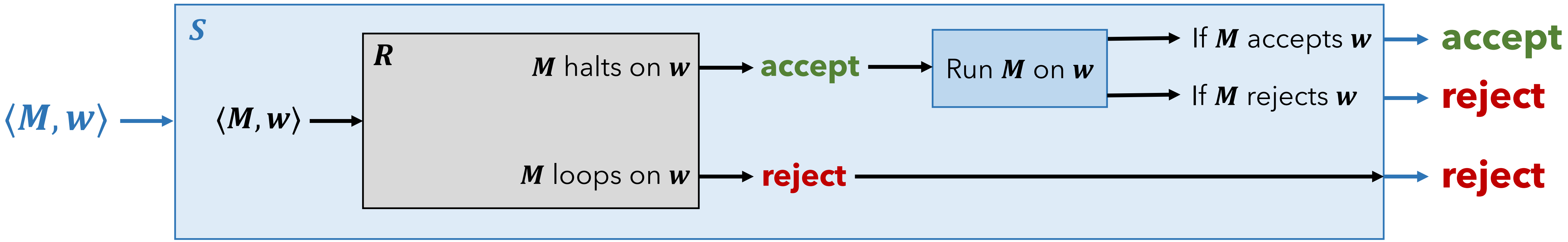


- Verify that S is a decider for A_{TM}
- On input $\langle M, w\rangle$:
  - If M accepts w, S **accepts**
  - If M rejects w, S **rejects**
  - If M loops on w, S **rejects**

- Yes, S is a decider for A_{TM}
- We have a **contradiction** since a **decider for A_{TM} should not exist**

$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

$Halt_{TM}$ is Undecidable (Reduce A_{TM} to $Halt_{TM}$)



- We have shown that if a decider R for $Halt_{TM}$ exists, then we can create a decider for A_{TM}
- This is a **contradiction**, since A_{TM} **is undecidable**
- So, the decider R for $Halt_{TM}$ doesn't exist
- Therefore, $Halt_{TM}$ **is undecidable**