Lecture 18: Undecidable Languages II

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Reductions

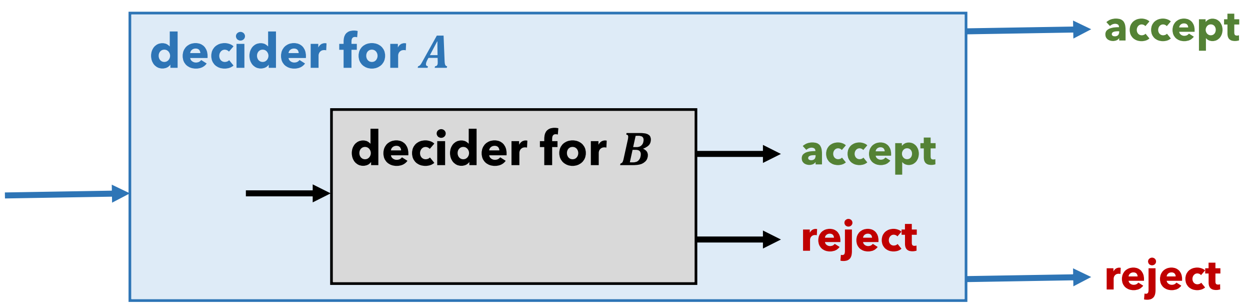
- Reductions convert **a problem A** to **another problem B** such that a **solution to B** can be used to **solve A**
  - If A **reduces to B**, we can use a **solution to B** to **solve A**

- Note: Reducibility says **nothing about how to solve A or B alone**
  - Only **how we could solve A** if we have **a way to solve B**

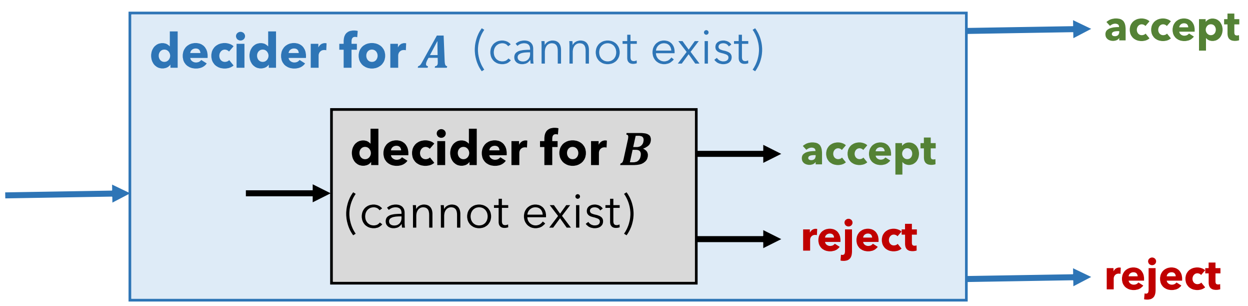
- **If A is reducible to B**, then solving A **cannot be harder** than solving B

Reductions for Undecidability

- If A **is reducible to** B and **we know** B **is decidable**, then A **is decidable**



- If A **is reducible to** B and **we know** A **is undecidable**, then B **is undecidable**



Another Undecidable Language

$$\boldsymbol{E_{TM}} = \{\langle \boldsymbol{M} \rangle \mid \boldsymbol{M} \text{ is a TM and } \boldsymbol{L(M)} = \emptyset\}$$

- $\boldsymbol{E_{TM}}$ contains all (string encodings of) TMs which **do not accept any strings**

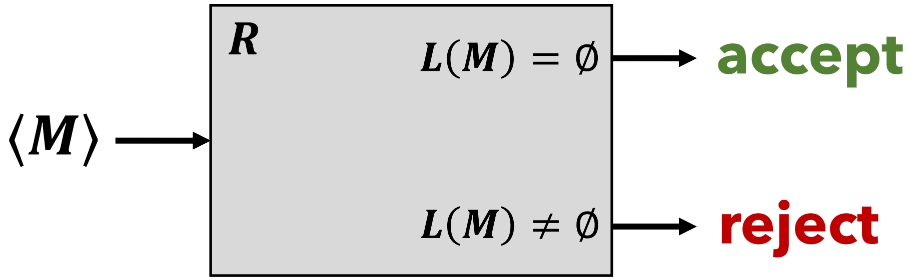
- We will prove that $\boldsymbol{E_{TM}}$ is undecidable, that is does not exist a decider which
  - **Accepts** input $\langle \boldsymbol{M} \rangle$ if $\boldsymbol{L(M)} = \emptyset$
  - **Rejects** input $\langle \boldsymbol{M} \rangle$ if $\boldsymbol{L(M)} \neq \emptyset$

- We will prove that $\boldsymbol{E_{TM}}$ is undecidable by a **reduction from** $\boldsymbol{A_{TM}}$

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

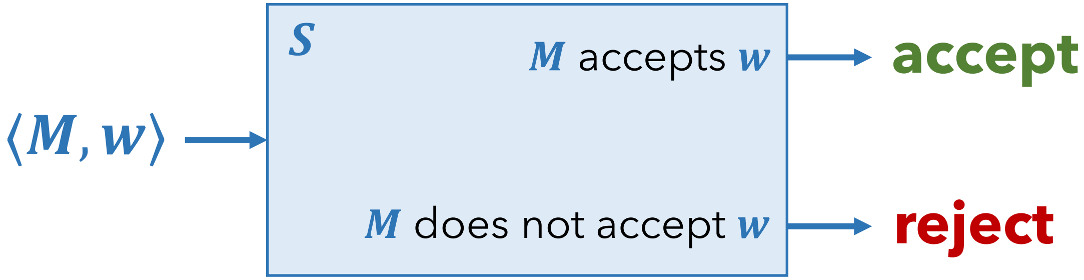
E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})

- **Assume for a contradiction that E_{TM} is decidable**
- That means, we assume a TM R exists which decides E_{TM}



(doesn't actually exist)

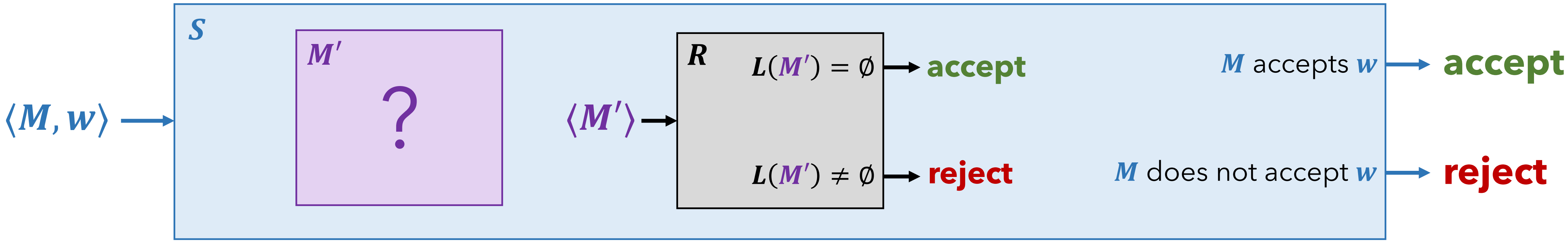
- We will show that using R, we can build a decider S for A_{TM}



(doesn't actually exist)

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})

- This reduction is different than the A_{TM} to $Halt_{TM}$ reduction since S cannot directly input $\langle M, w \rangle$ into R
- What S will do is **create a TM** M' and input $\langle M' \rangle$ into R
- M' is created such that the **result from running** $\langle M' \rangle$ **on** R somehow reveals if
  - M accepts w
  - or M does not accept w



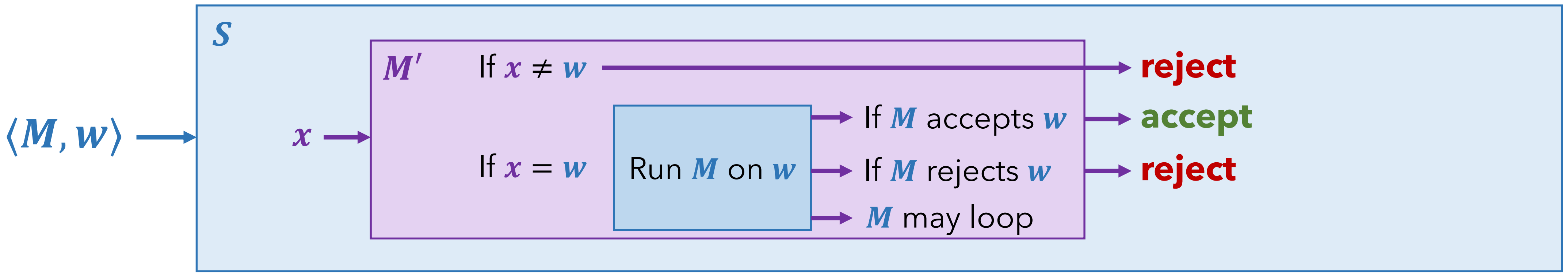
$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})

- We create a decider S for A_{TM} as follows:

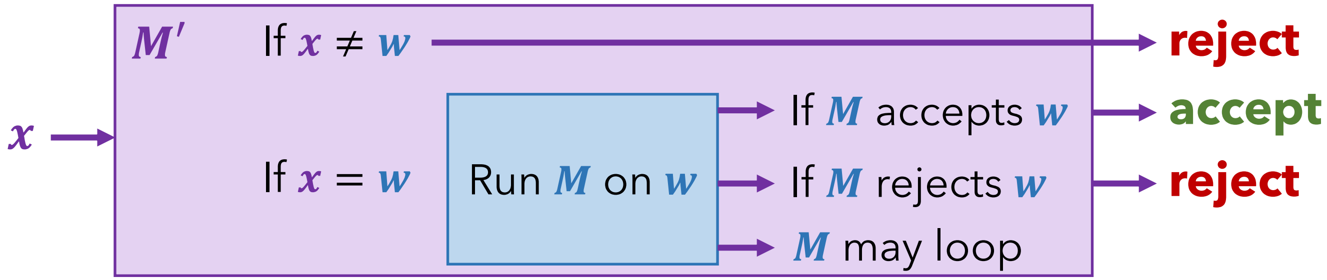
  S = "On input $\langle M, w \rangle$:
  - Create a TM M' as follows:

    M' = "On input x, where x is any string:
    - If $x \neq w$, **reject**
    - If $x = w$, run M on w
      - **accept** if M accepts w, **reject** if M rejects w
      - (M may loop on w)



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

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})



Let's analyze $L(M')$, the strings that are accepted by M':

- If the input to M' is any string in Σ^* that is not w, M' just **rejects**
- If the input to M' is w:
  - If M accepts w, M' **accepts**
  - If M rejects w, M' **rejects**
  - If M loops on w, M' also loops

- **If M accepts w, then $L(M') = \{w\}$**
- **If M does not accept w** (rejects or loops), **then $L(M') = \emptyset$**

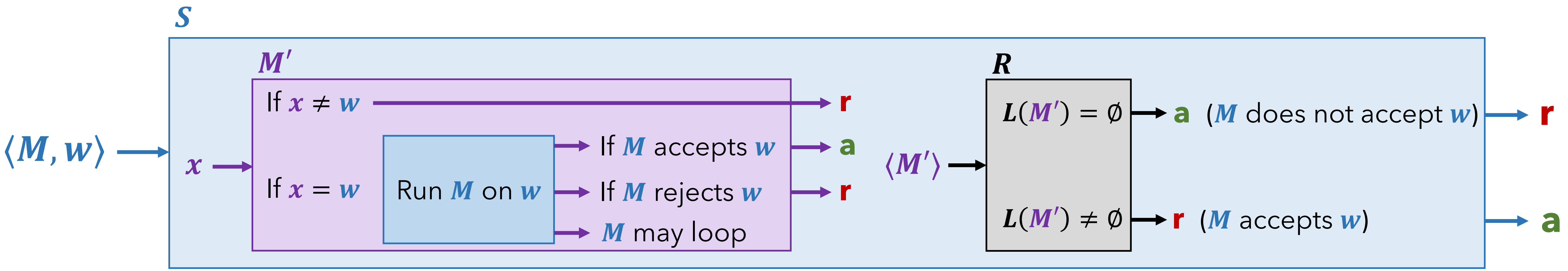
Note: We never actually run M'

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

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})

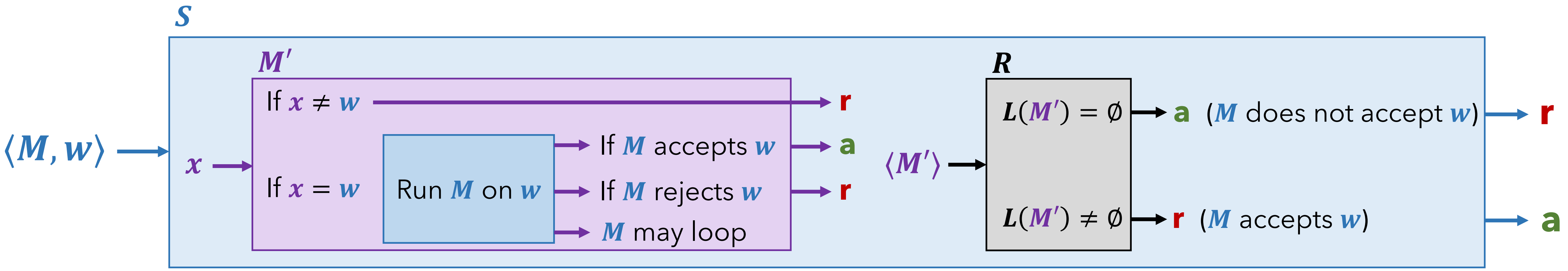
- We create a decider S for A_{TM} as follows:

  S = "On input $\langle M, w \rangle$:
  - Create a TM M' as described
    - $L(M') = \{w\}$ if M accepts w
    - $L(M') = \emptyset$ if M does not accept w
  - Run R on input $\langle M' \rangle$
  - If R **accepts** (decides $L(M') = \emptyset$), then S **rejects**
  - If R **rejects** (decides $L(M') \neq \emptyset$), then S **accepts**



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

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})

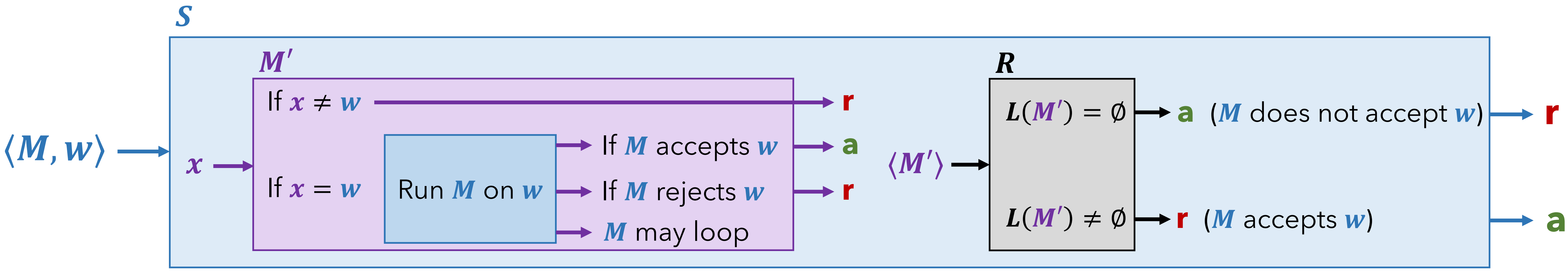


- S is a decider since **it always halts**:
  - Creating M' doesn't loop (we do not run M')
  - Running R on $\langle M' \rangle$ always halts since R is a decider

- S is a decider for A_{TM}:
  - We constructed M' such that
    - $L(M') \neq \emptyset$ if M accepts w and $L(M') = \emptyset$ if M does not accept w
  - So, running R on $\langle M' \rangle$, will halt and tell us if M accepts w or not

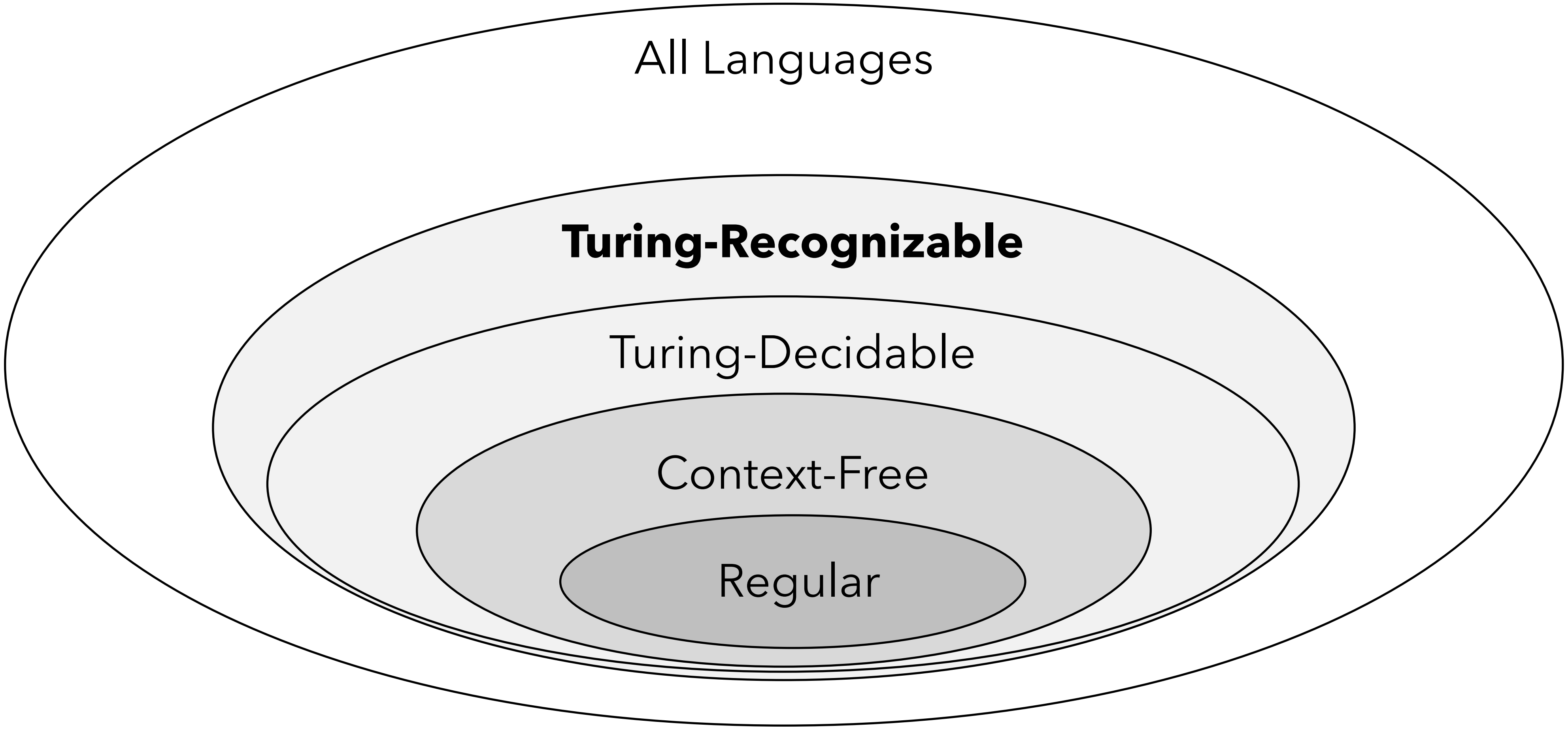
$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$

E_{TM} is Undecidable (Reduce A_{TM} to E_{TM})



- We have shown that if a decider R for E_{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 E_{TM} doesn't exist
- Therefore, E_{TM} **is undecidable**

Non Turing-Recognizable Languages



Question: What languages are **not Turing-recognizable?**

Non Turing-Recognizable Languages

- We now know that the following languages are **undecidable**
  - $\boldsymbol{A_{TM}} = \{\langle \boldsymbol{M}, \boldsymbol{w} \rangle \mid \boldsymbol{M} \text{ is a TM and } \boldsymbol{M} \text{ accepts } \boldsymbol{w}\}$
  - $\boldsymbol{Halt_{TM}} = \{\langle \boldsymbol{M}, \boldsymbol{w} \rangle \mid \boldsymbol{M} \text{ is a TM and } \boldsymbol{M} \text{ halts on input } \boldsymbol{w}\}$

- However $\boldsymbol{A_{TM}}$ and $\boldsymbol{HALT_{TM}}$ are **Turing-recognizable**

- What are examples of languages that are **not Turing-recognizable?**

Co-Turing Recognizable Languages

- **Definition:** A language is **co-Turing recognizable** if it is the **complement** of a Turing-recognizable language

- **Example:** $\overline{A_{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ does not accept } w \}$

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

- The language $\overline{A_{TM}}$ is **co-Turing recognizable** since it is the complement of a Turing-recognizable language (complement of A_{TM})

Decidable Language Theorem

Theorem: A language $\boldsymbol{L}$ is **decidable** if and only if $\boldsymbol{L}$ is both **Turing-recognizable and co-Turing recognizable**

That is, a language $\boldsymbol{L}$ is decidable if and only if

- $\boldsymbol{L}$ is Turing-recognizable and
- $\overline{\boldsymbol{L}}$ is Turing-recognizable

Proof:

$\Rightarrow$ If $\boldsymbol{L}$ **is decidable**, then $\boldsymbol{L}$ and $\overline{\boldsymbol{L}}$ are both **Turing-recognizable**

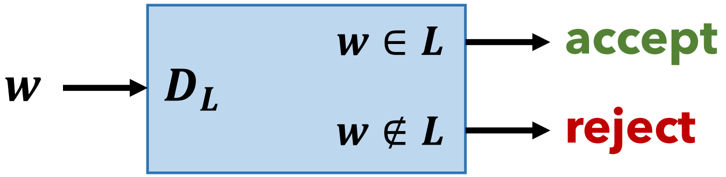
$\Leftarrow$ If $\boldsymbol{L}$ and $\overline{\boldsymbol{L}}$ are both **Turing-recognizable**, then $\boldsymbol{L}$ **is decidable**

Decidable Language Theorem

⇒ If $\boldsymbol{L}$ **is decidable**, then $\boldsymbol{L}$ and $\overline{\boldsymbol{L}}$ are both **Turing-recognizable**

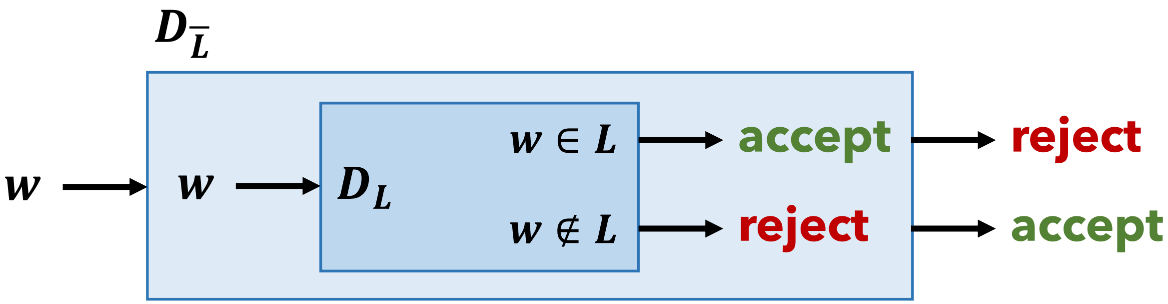
- Clearly if $\boldsymbol{L}$ is decidable then $\boldsymbol{L}$ is Turing-recognizable
- We show that if $\boldsymbol{L}$ is decidable, then $\overline{\boldsymbol{L}}$ is decidable (therefore Turing-recognizable)

- Let $\boldsymbol{D_L}$ be the decider for $\boldsymbol{L}$



- We build a decider $\boldsymbol{D_{\overline{L}}}$ for $\overline{\boldsymbol{L}}$ as follows:

  $\boldsymbol{D_{\overline{L}}}$ = "On input $\boldsymbol{w}$
  - Run $\boldsymbol{D_L}$ on input $\boldsymbol{w}$
  - If $\boldsymbol{D_L}$ **accepts**, then **reject**
  - If $\boldsymbol{D_L}$ **rejects**, then **accept**"



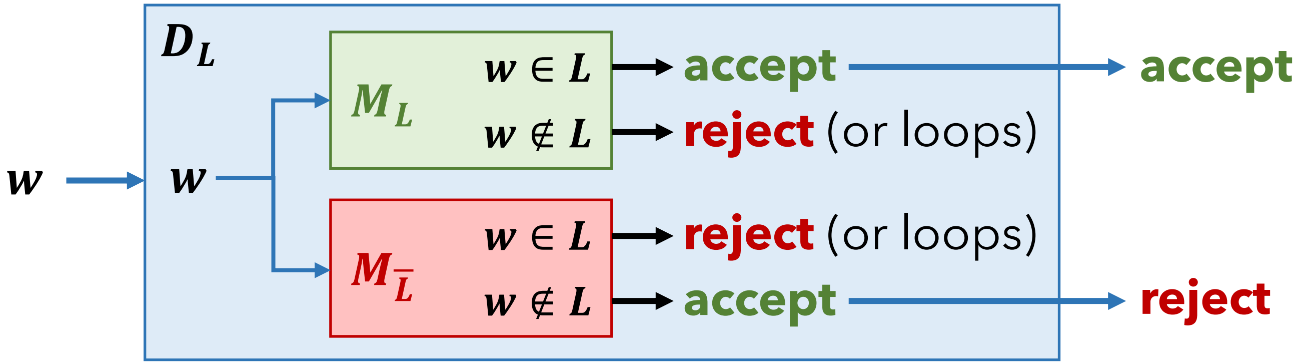
Decidable Language Theorem

$\Leftarrow$ If L and $\overline{L}$ are both **Turing-recognizable**, then L **is decidable**

- Let M_L and $M_{\overline{L}}$ be TMs which **recognize** L and $\overline{L}$ respectively (may loop)
- We show that L **is decidable** by building a decider D_L that decides L

D_L = "On input w

- Run M_L and $M_{\overline{L}}$ on w simultaneously
- If M_L accepts, then **accept**
- If $M_{\overline{L}}$ accepts, then **reject**



D_L is a decider for L.
For each $w \in \Sigma^*$:
- M_L halts and accepts if $w \in L$
- $M_{\overline{L}}$ halts and accepts if $w \notin L$

$\overline{A_{TM}}$ is Non Turing Recognizable Language

Theorem: A language L is **decidable** if and only if L is **Turing-recognizable** and $\overline{L}$ is **Turing-recognizable**

We can use the theorem to prove that $\overline{A_{TM}}$ is **not Turing-recognizable**

$$\overline{A_{TM}} = \{\, \langle M, w \rangle \mid M \text{ is a TM and } M \text{ does not accept } w \,\}$$

Proof:

- We know A_{TM} **is Turing-recognizable** and **not decidable**
- If $\overline{A_{TM}}$ is **Turing-recognizable**, then by the theorem A_{TM} would be decidable
- Therefore, $\overline{A_{TM}}$ is **not Turing-recognizable**