Mapping Reductions

Announcements

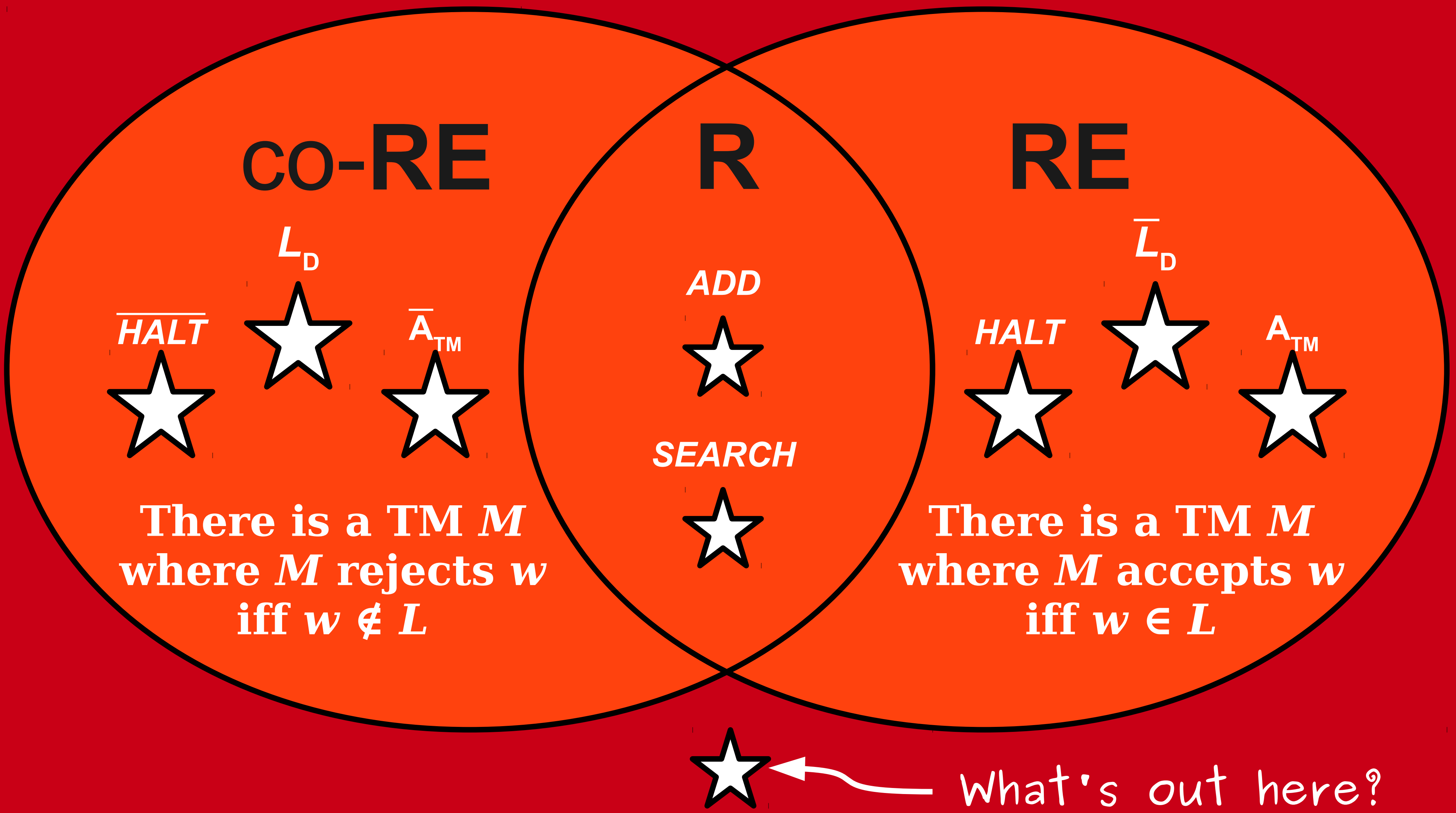
- Casual CS Dinner for Women Studying Computer Science: **Thursday, March 7** at **6PM** in **Gates 219**!
- RSVP through the email link sent out earlier today.

Announcements

- All Problem Set 6's are graded, will be returned at end of lecture.
- Problem Set 7 due right now, or due at Thursday at 12:50PM with a late day.
  - **Please submit no later than 12:50PM**; we're hoping to get solutions posted then. This is a hard deadline.
- Problem Set 8 out, due next Monday, March 11 at 12:50PM.
  - Explore the limits of computation!

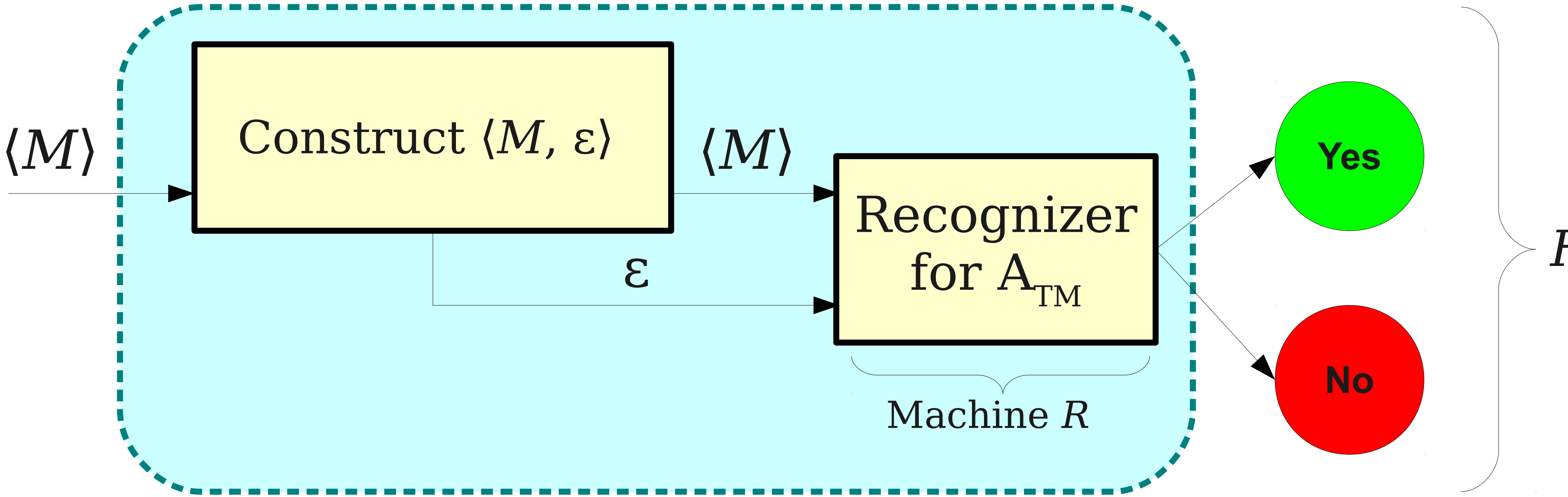
Recap from Last Time

The Limits of Computability



A Repeating Pattern

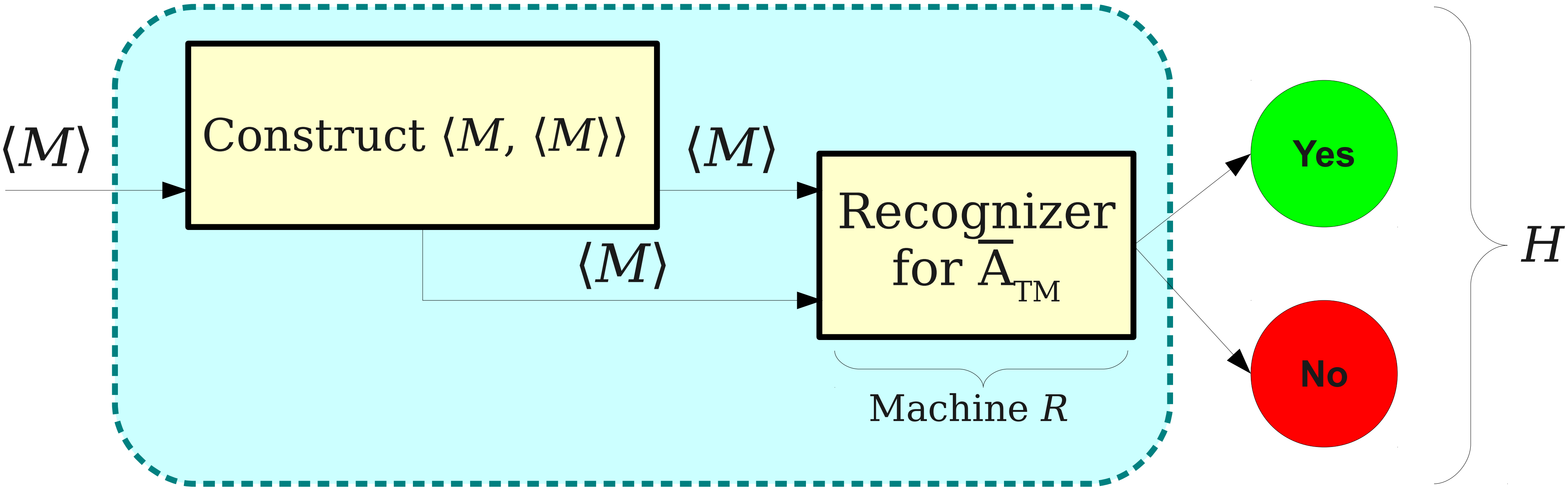
$L = \{ \langle M \rangle \mid M$ is a TM that accepts ε }



H = "On input $\langle M \rangle$:

- Construct the string $\langle M, \varepsilon \rangle$.
- Run R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts $\langle M, \varepsilon \rangle$.
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects $\langle M, \varepsilon \rangle$."

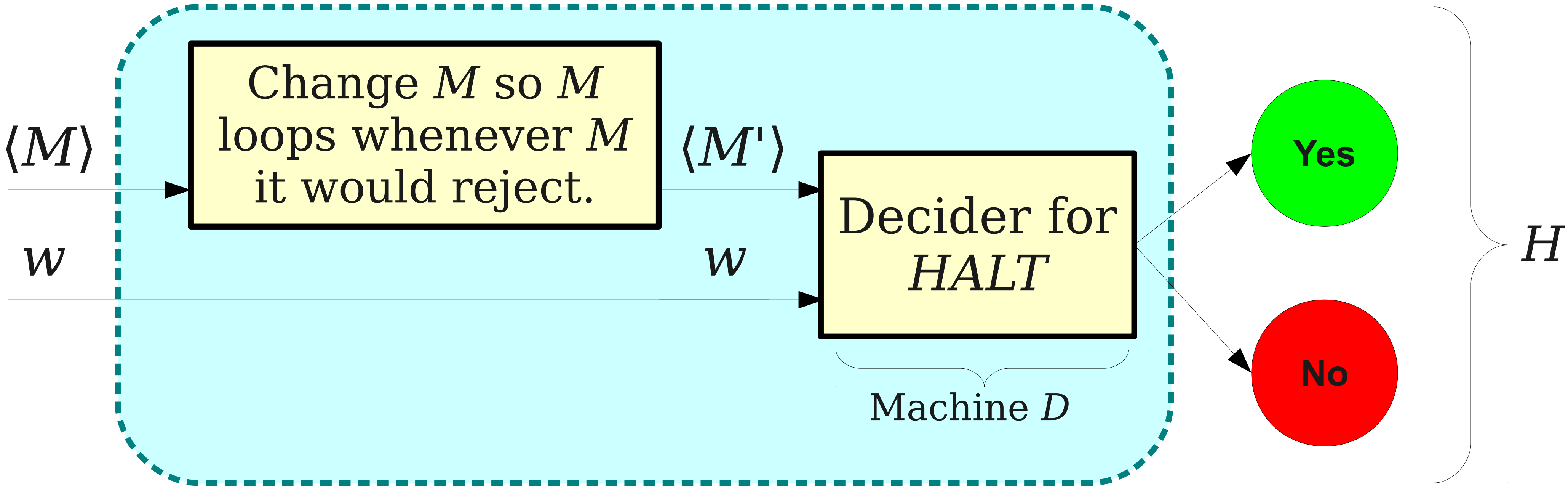
From $\overline{\text{A}}_{\text{TM}}$ to L_{D}



H = "On input $\langle M \rangle$:

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts $\langle M, \langle M \rangle \rangle$.
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects $\langle M, \langle M \rangle \rangle$."

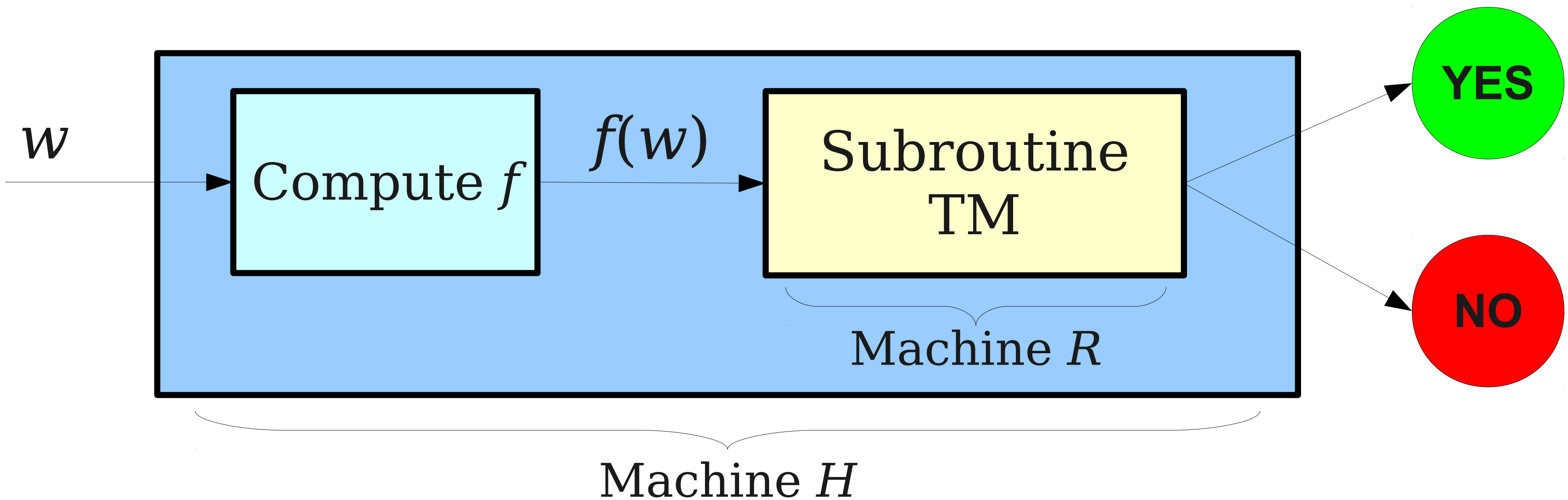
From *HALT* to A_{TM}



H = "On input ⟨*M*, *w*⟩:

- Build *M* into *M*' so *M*' loops when *M* rejects.
- Run *D* on ⟨*M*', *w*⟩.
- If *D* accepts ⟨*M*', *w*⟩, then *H* accepts ⟨*M*, *w*⟩.
- If *D* rejects ⟨*M*', *w*⟩, then *H* rejects ⟨*M*, *w*⟩."

The General Pattern

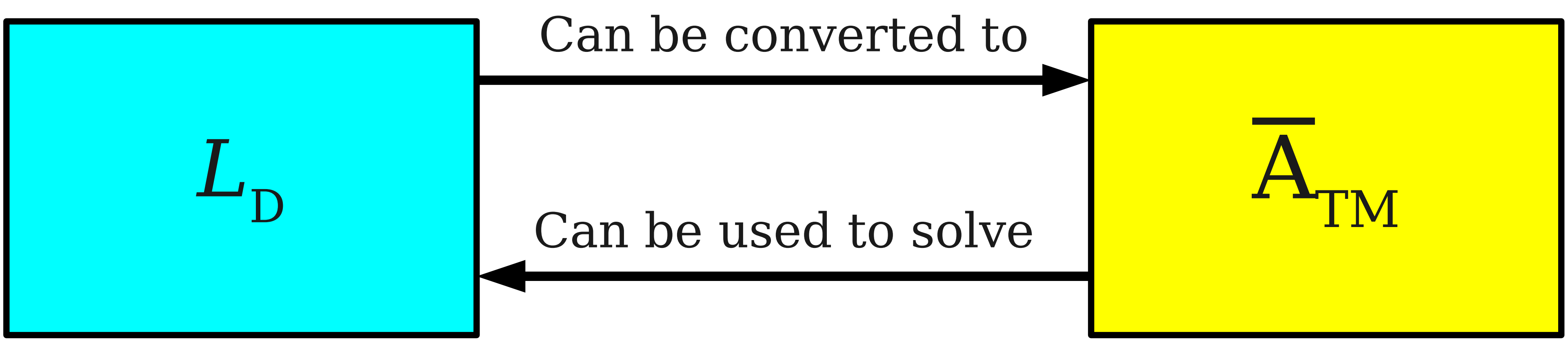


H = "On input w:

- Transform the input w into $f(w)$.
- Run machine R on $f(w)$.
- If R accepts $f(w)$, then H accepts w.
- If R rejects $f(w)$, then H rejects w."

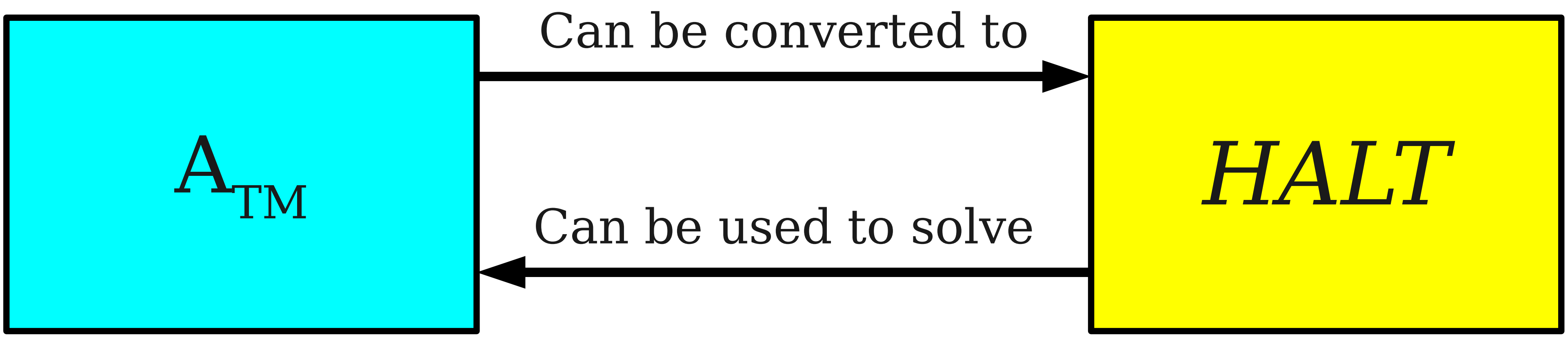
Reductions

- Intuitively, problem *A* **reduces** to problem *B* iff a solver for *B* can be used to solve problem *A*.



Reductions

- Intuitively, problem *A* **reduces** to problem *B* iff a solver for *B* can be used to solve problem *A*.



Reductions

- Intuitively, problem A **reduces** to problem B iff a solver for B can be used to solve problem A.
- Reductions can be used to show certain problems are "solvable:"

If *A* reduces to *B* and *B* is "solvable," then *A* is "solvable."

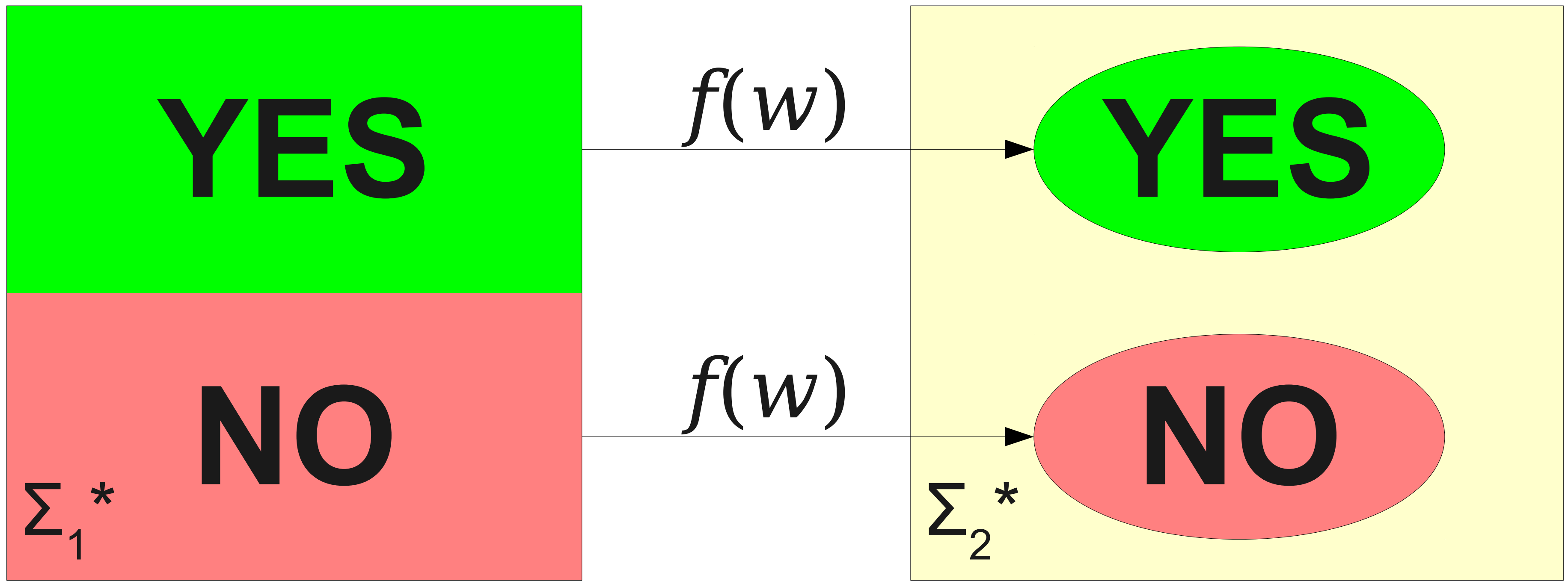
- Reductions can be used to show certain problems are "*unsolvable*:"

If *A* reduces to *B* and *A* is "unsolvable," then *B* is "unsolvable."

Defining Reductions

- A **reduction** from A to B is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$



Defining Reductions

- A **reduction** from A to B is a function $f : \Sigma_1^* \to \Sigma_2^*$ such that

  For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$

- Every $w \in A$ maps to some $f(w) \in B$.
- Every $w \notin A$ maps to some $f(w) \notin B$.
- f does not have to be injective or surjective.

Computable Functions

- Not all mathematical functions can be computed by Turing machines.
- A function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is called a **computable function** if there is some TM M with the following behavior:

“On input w:

Compute $f(w)$ and write it on the tape.

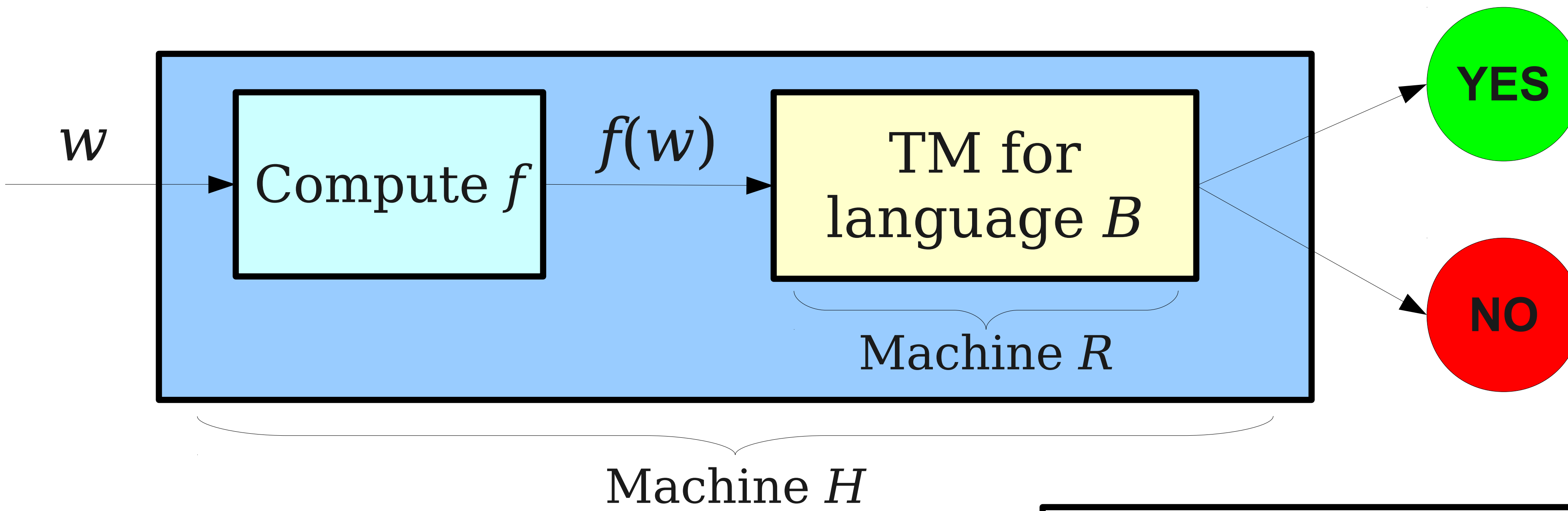
Move the tape head to the start of $f(w)$.

Halt.”

Mapping Reductions

- A function $f : \Sigma_1^* \to \Sigma_2^*$ is called a **mapping reduction** from A to B iff
  - For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$.
  - f is a computable function.
- Intuitively, a mapping reduction from A to B says that a computer can transform any instance of A into an instance of B such that the answer to B is the answer to A.

$w \in A$ iff $f(w) \in B$



H = "On input w:

- Transform the input w into $f(w)$.
- Run machine R on $f(w)$.
- If R accepts $f(w)$, then H accepts w.
- If R rejects $f(w)$, then H rejects w."

H accepts w

iff

R accepts $f(w)$

iff

$f(w) \in B$

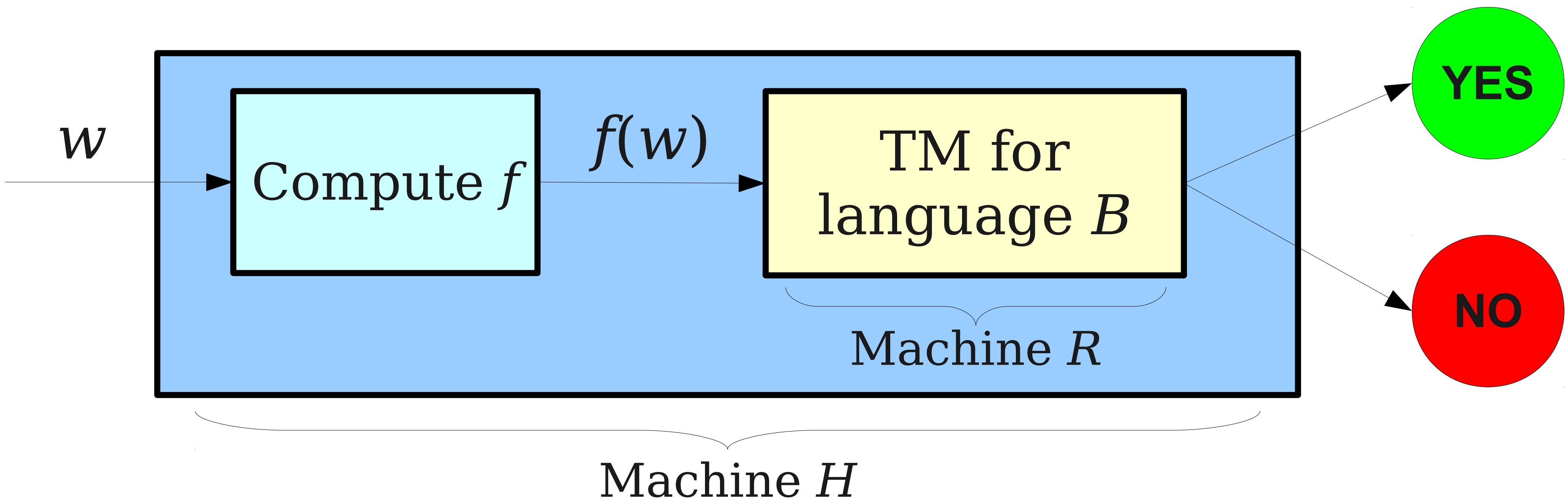
iff

$w \in A$

Mapping Reducibility

- If there is a mapping reduction from language *A* to language *B*, we say that language *A* is **mapping reducible** to language *B*.
- Notation: $\boldsymbol{A \leq_M B}$ iff language *A* is mapping reducible to language *B*.
- Note that we reduce ***languages***, not ***machines***.
- Interesting exercise: Show $\leq_M$ is reflexive and transitive, but not antisymmetric.

$A \leq_M B$



H = "On input w:

- Compute $f(w)$.
- Run machine R on $f(w)$.
- If R accepts $f(w)$, then H accepts w.
- If R rejects $f(w)$, then H rejects w."

If R is a decider for B, then H is a decider for A.

If R is a recognizer for B, then H is a recognizer for A.

If R is a co-recognizer for B, then H is a co-recognizer for A.

Why Mapping Reducibility Matters

- **Theorem**: If $B \in \mathbf{R}$ and $A \leq_M B$, then $A \in \mathbf{R}$.
- **Theorem**: If $B \in \mathbf{RE}$ and $A \leq_M B$, then $A \in \mathbf{RE}$.
- **Theorem**: If $B \in \text{co-}\mathbf{RE}$ and $A \leq_M B$, then $A \in \text{co-}\mathbf{RE}$.
- *Intuitively:* $A \leq_M B$ means "A is not harder than B."

Why Mapping Reducibility Matters

- **Theorem**: If $A \notin \mathbf{R}$ and $A \leq_{\mathrm{M}} B$, then $B \notin \mathbf{R}$.
- **Theorem**: If $A \notin \mathbf{RE}$ and $A \leq_{\mathrm{M}} B$, then $B \notin \mathbf{RE}$.
- **Theorem**: If $A \notin \text{co-}\mathbf{RE}$ and $A \leq_{\mathrm{M}} B$, then $B \notin \text{co-}\mathbf{RE}$.
- *Intuitively:* $A \leq_{\mathrm{M}} B$ means "B is at at least as hard as A."

Why Mapping Reducibility Matters

If this one is "easy" (R, RE, co-RE)…

$$A \leq_{\mathrm{M}} B$$

… then this one is "easy" (R, RE, co-RE) too.

Why Mapping Reducibility Matters

If this one is "hard" (not **R**, not **RE**, or not co-**RE**)…

$$A \leq_M B$$

… then this one is "hard" (not **R**, not **RE**, or not co-**RE**) too.

Using Mapping Reductions

Revisiting our Proofs

- Consider the language

$$\boldsymbol{L = \{ \langle M \rangle \mid M \textbf{ is a TM and } M \textbf{ accepts } \varepsilon \}}$$

- We have already proven that this language is **RE** by building a TM for it.
- Let's repeat this proof using mapping reductions.
- Specifically, we will prove

$$\boldsymbol{L \leq_M A_{TM}}$$

$L = \{ \langle M \rangle \mid M$ is a TM and M accepts $\varepsilon \}$

- To prove $L \leq_M A_{TM}$, we will need to find a computable function f such that

$$\langle M \rangle \in L \quad \textbf{iff} \quad f(\langle M \rangle) \in A_{TM}$$

- Since A_{TM} is a language of TM/string pairs, let's assume $f(\langle M \rangle) = \langle N, w \rangle$ for some TM N and string w (which we'll pick later):

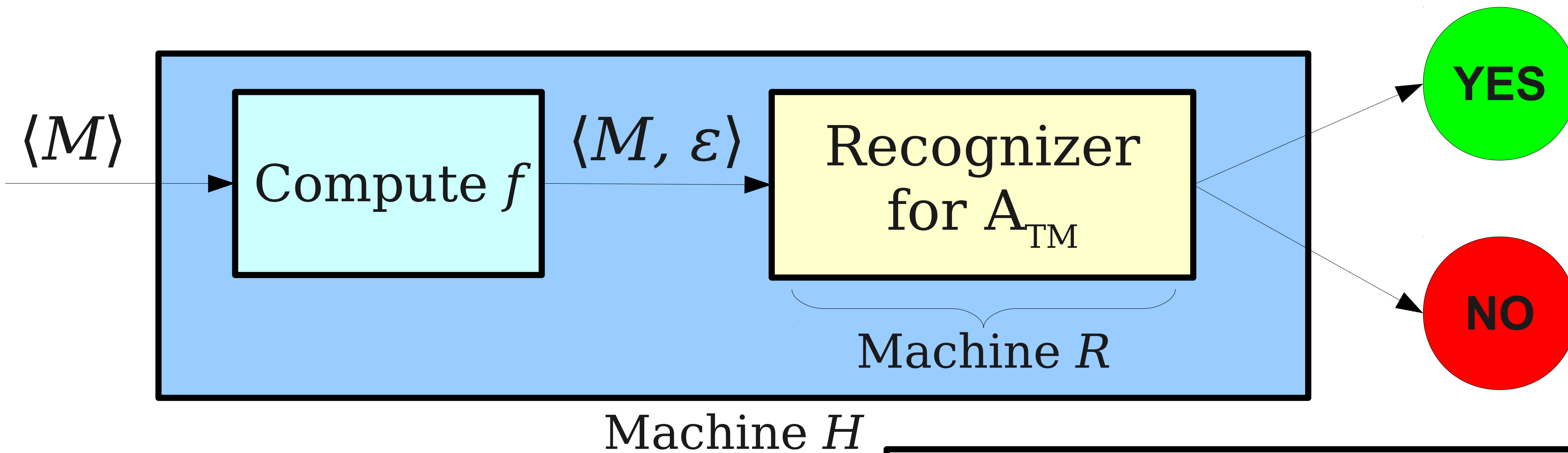
$$\langle M \rangle \in L \quad \textbf{iff} \quad \langle N, w \rangle \in A_{TM}$$

- Substituting definitions:

$$M \textbf{ accepts } \varepsilon \quad \textbf{iff} \quad N \textbf{ accepts } w$$

- Choose $N = M$, $w = \varepsilon$. So $f(\langle M \rangle) = \langle M, \varepsilon \rangle$.

One Interpretation of the Reduction



H = "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w.
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \varepsilon \rangle$

iff

M accepts ε

iff

$\langle M \rangle \in L$

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

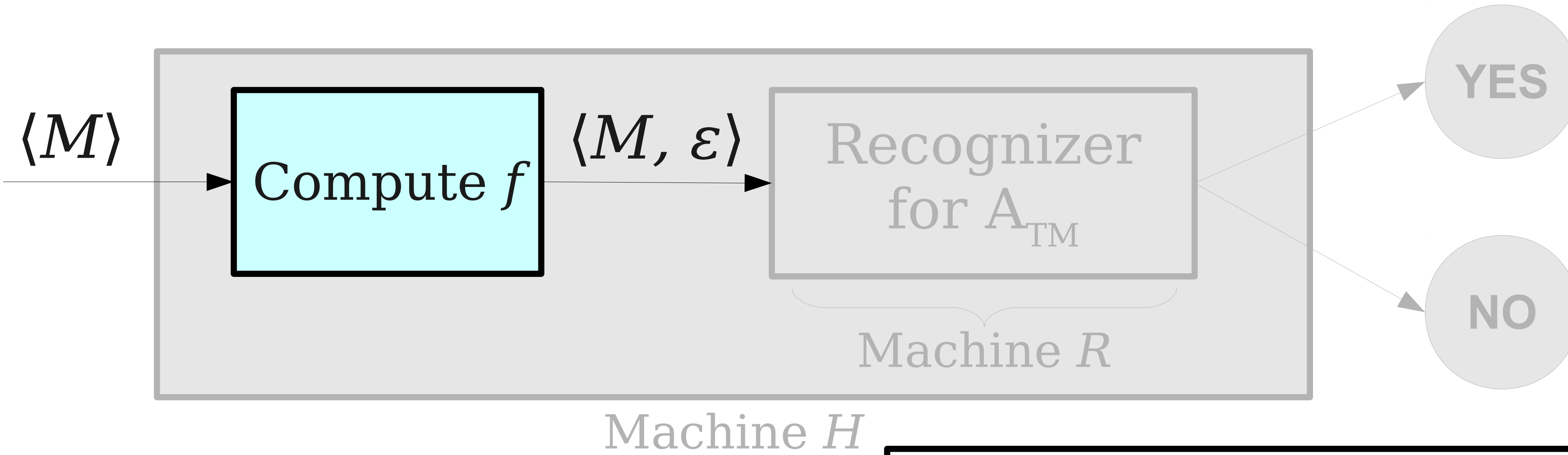
Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{TM}$. Since $A_{TM} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

Consider the function $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. We state without proof that this function is computable and claim that f is a mapping reduction from L to A_{TM}. To see this, note that $f(\langle M \rangle) = \langle M, \varepsilon \rangle \in A_{TM}$ iff M accepts ε iff $\langle M \rangle \in L$, so $\langle M \rangle \in L$ iff $f(\langle M \rangle) \in A_{TM}$.

Since f is a mapping reduction from L to A_{TM}, we have $L \leq_M A_{TM}$, and thus $L \in \mathbf{RE}$. ■

What Did We Prove?



H = "On input ⟨M⟩:

- Run machine R on ⟨M, ε⟩.
- If R accepts ⟨M, ε⟩, then H accepts w.
- If R rejects ⟨M, ε⟩, then H rejects w."

H accepts ⟨M⟩

iff

R accepts ⟨M, ε⟩

iff

M accepts ε

iff

⟨M⟩ ∈ L

Interpreting Mapping Reductions

- If $A \leq_M B$, there is a known construction to turn a TM for B into a TM for A.
- When doing proofs with mapping reductions, you do *not* need to show the overall construction.
- You just need to prove that
  - f is a computable function, and
  - $w \in A$ iff $f(w) \in B$.

Another Mapping Reduction

L_D and $\overline{A}_{TM}$

- Earlier, we proved $\overline{A}_{TM} \notin$ **RE** by proving that

  If $\overline{A}_{TM} \in$ **RE**, then $L_D \in$ **RE**.

- The proof constructed this TM, assuming R was a recognizer for $\overline{A}_{TM}$.

H = "On input $\langle M \rangle$:
- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts $\langle M \rangle$.
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects $\langle M \rangle$."

- Let's do another proof using mapping reductions.

$L_D \leq_M \overline{A}_{TM}$

- To prove that $\overline{A}_{TM} \notin \mathbf{RE}$, we will prove

$$\boldsymbol{L_D \leq_M \overline{A}_{TM}}$$

- By our earlier theorem, since $L_D \notin \mathbf{RE}$, we have that $\overline{A}_{TM} \notin \mathbf{RE}$.
- *Intuitively*: $\overline{A}_{TM}$ is "at least as hard" as L_D, and since $L_D \notin \mathbf{RE}$, this means $\overline{A}_{TM} \notin \mathbf{RE}$.

$L_D \leq_M \overline{A}_{TM}$

- Goal: Find a computable function f such that

$$\langle M \rangle \in L_D \quad \text{iff} \quad f(\langle M \rangle) \in \overline{A}_{TM}$$

- Simplifying this using the definition of L_D

$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad f(\langle M \rangle) \in \overline{A}_{TM}$$

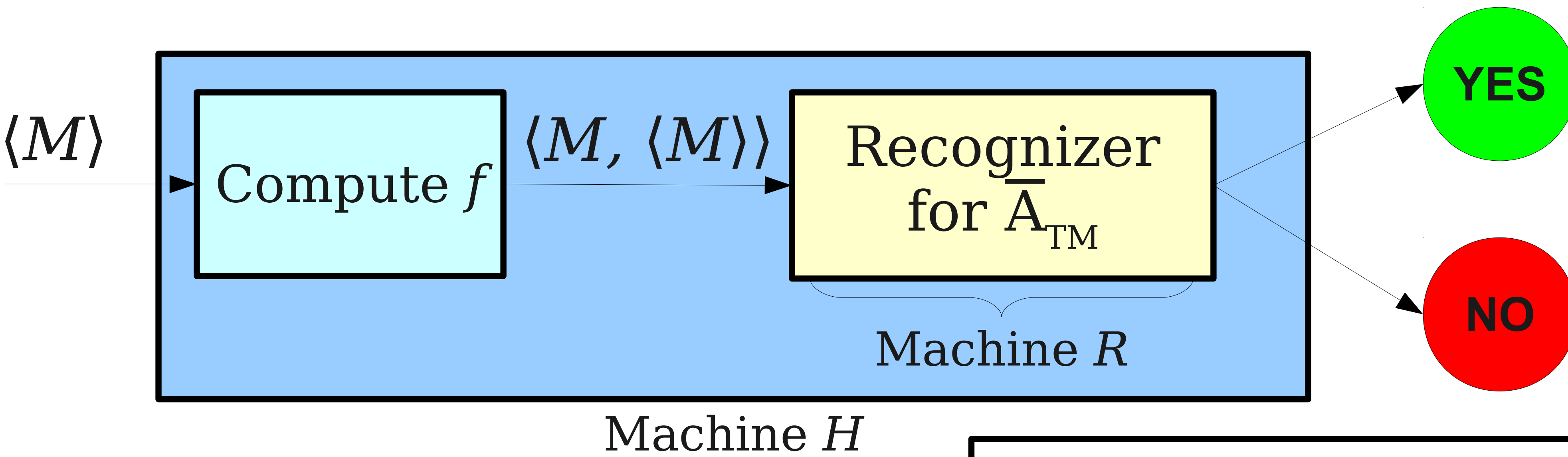
- Let's assume that $f(\langle M \rangle)$ has the form $\langle N, w \rangle$ for some TM N and string w. This means that

$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad \langle N, w \rangle \in \overline{A}_{TM}$$

$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad N \text{ does not accept } w$$

- If we can choose w and N such that the above is true, we will have our reduction from L_D to $\overline{A}_{TM}$.
- Choose $N = M$ and $w = \langle M \rangle$.

One Interpretation of the Reduction



H = "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w.
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \langle M \rangle \rangle$

iff

M does not accept $\langle M \rangle$

iff

$\langle M \rangle \in L_D$

Theorem: $\overline{\mathrm{A}}_{\mathrm{TM}} \notin \mathbf{RE}$.

Proof: We exhibit a mapping reduction f from L_{D} to $\overline{\mathrm{A}}_{\mathrm{TM}}$.

Consider the function f defined as follows:

$$f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$$

We claim that f can be computed by a TM and omit the details from this proof. We will prove that $\langle M \rangle \in L_{\mathrm{D}}$ iff $f(\langle M \rangle) \in \overline{\mathrm{A}}_{\mathrm{TM}}$. Note that $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$, so $f(\langle M \rangle) \in \overline{\mathrm{A}}_{\mathrm{TM}}$ iff $\langle M, \langle M \rangle \rangle \in \overline{\mathrm{A}}_{\mathrm{TM}}$. By definition of $\overline{\mathrm{A}}_{\mathrm{TM}}$, $\langle M, \langle M \rangle \rangle \in \overline{\mathrm{A}}_{\mathrm{TM}}$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Finally, note that $\langle M \rangle \notin \mathscr{L}(M)$ iff $\langle M \rangle \in L_{\mathrm{D}}$. Thus $f(\langle M \rangle) \in \overline{\mathrm{A}}_{\mathrm{TM}}$ iff $\langle M \rangle \in L_{\mathrm{D}}$, so f is a mapping reduction from L_{D} to $\overline{\mathrm{A}}_{\mathrm{TM}}$.

Since f is a mapping reduction from L_{D} to $\overline{\mathrm{A}}_{\mathrm{TM}}$, we have $L_{\mathrm{D}} \leq_{\mathrm{M}} \overline{\mathrm{A}}_{\mathrm{TM}}$. Since $L_{\mathrm{D}} \notin \mathbf{RE}$ and $L_{\mathrm{D}} \leq_{\mathrm{M}} \overline{\mathrm{A}}_{\mathrm{TM}}$, this means $\overline{\mathrm{A}}_{\mathrm{TM}} \notin \mathbf{RE}$, as required. ■

Another Example of Mapping Reductions

A More Elaborate Reduction

- Since $\overline{A}_{TM} \notin$ **RE**, there is no algorithm for determining whether a TM will not accept a given string.
- Could we check instead whether a TM *never* accepts a string?
- Consider the language

$$\boldsymbol{L_e = \{ \langle M \rangle \mid M \textbf{ is a TM and } M \textbf{ never accepts } \}}$$

- How "hard" is L_e? Is it **R**, **RE**, co-**RE**, or none of these?

Building an Intuition

- Before we even try to prove how “hard” this language is, we should build an intuition for its difficulty.
- L_e is *probably* not in **RE**, since if we were convinced a TM never accepted, it would be hard to find positive evidence of this.
- L_e is *probably* in co-**RE**, since if we were convinced that a TM *did* accept some string, we could exhaustively search over all strings and try to find the string it accepts.
- Best guess: $L_e \in$ co-**RE** – **R**.

$\overline{\mathrm{A}}_{\mathrm{TM}} \leq_{\mathrm{M}} L_{\mathrm{e}}$

- We will prove that $L_{\mathrm{e}} \notin \mathbf{RE}$ by showing that $\overline{\mathrm{A}}_{\mathrm{TM}} \leq_{\mathrm{M}} L_{\mathrm{e}}$. (This also proves $L_{\mathrm{e}} \notin \mathbf{R}$).
- We want to find a function f such that

$$\langle M, w\rangle \in \overline{\mathrm{A}}_{\mathrm{TM}} \quad \textbf{iff} \quad f(\langle M, w\rangle) \in L_{\mathrm{e}}$$

- Since L_{e} is a language of TM descriptions, let's assume $f(\langle M, w\rangle) = \langle N\rangle$ for some TM N. Then

$$\langle M, w\rangle \in \overline{\mathrm{A}}_{\mathrm{TM}} \quad \textbf{iff} \quad \langle N\rangle \in L_{\mathrm{e}}$$

- Expanding out definitions, we get

***M* doesn't accept *w* iff *N* doesn't accept any strings**

- How do we pick the machine N?

The Reduction

- Find a TM N such that N does not accept any strings iff M does not accept w.
- **Key idea:** Build N such that running N on any input runs M on w.
- Here is one choice of N:

  N = "On input x:

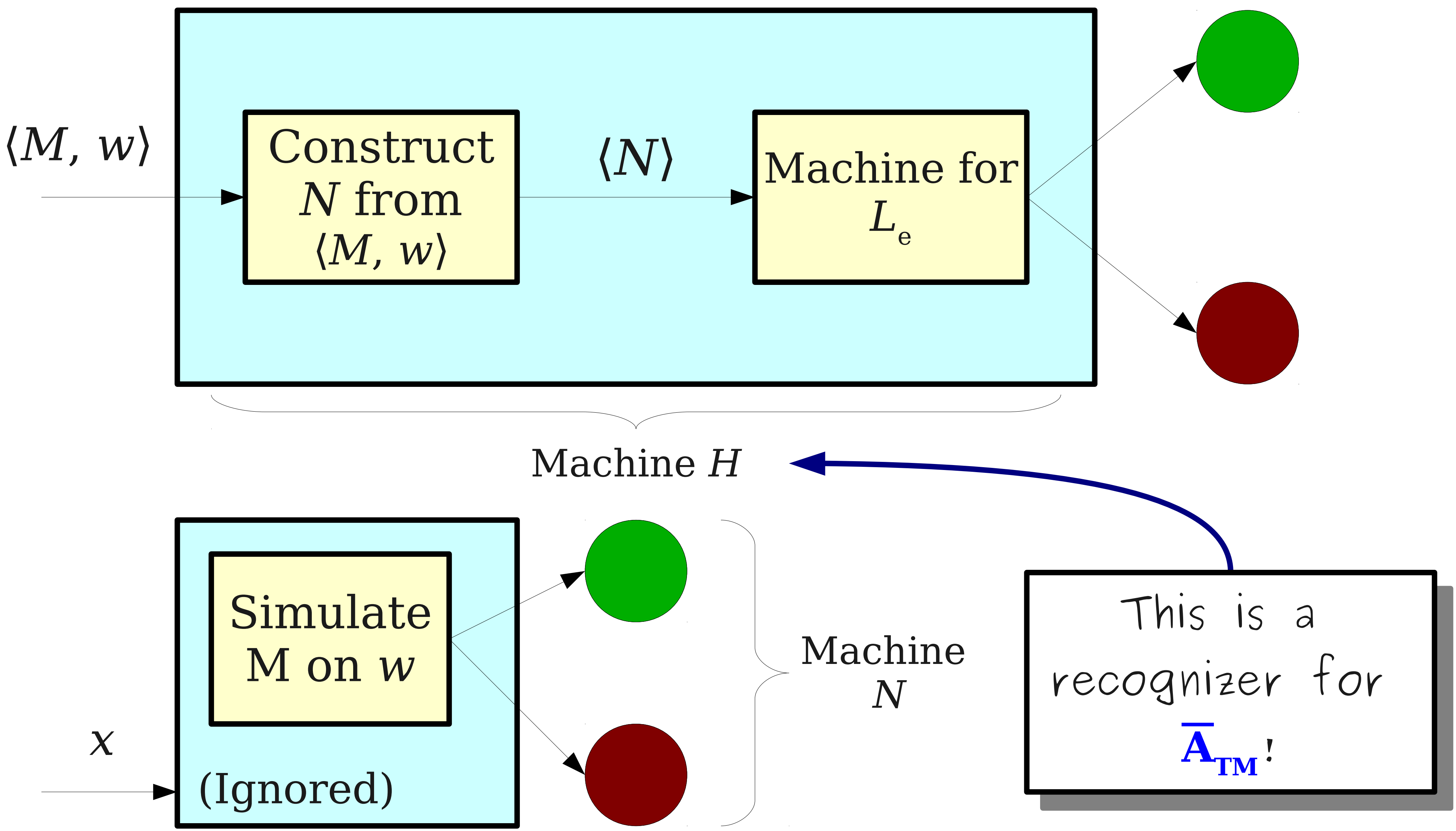
  Ignore x.

  Run M on w.

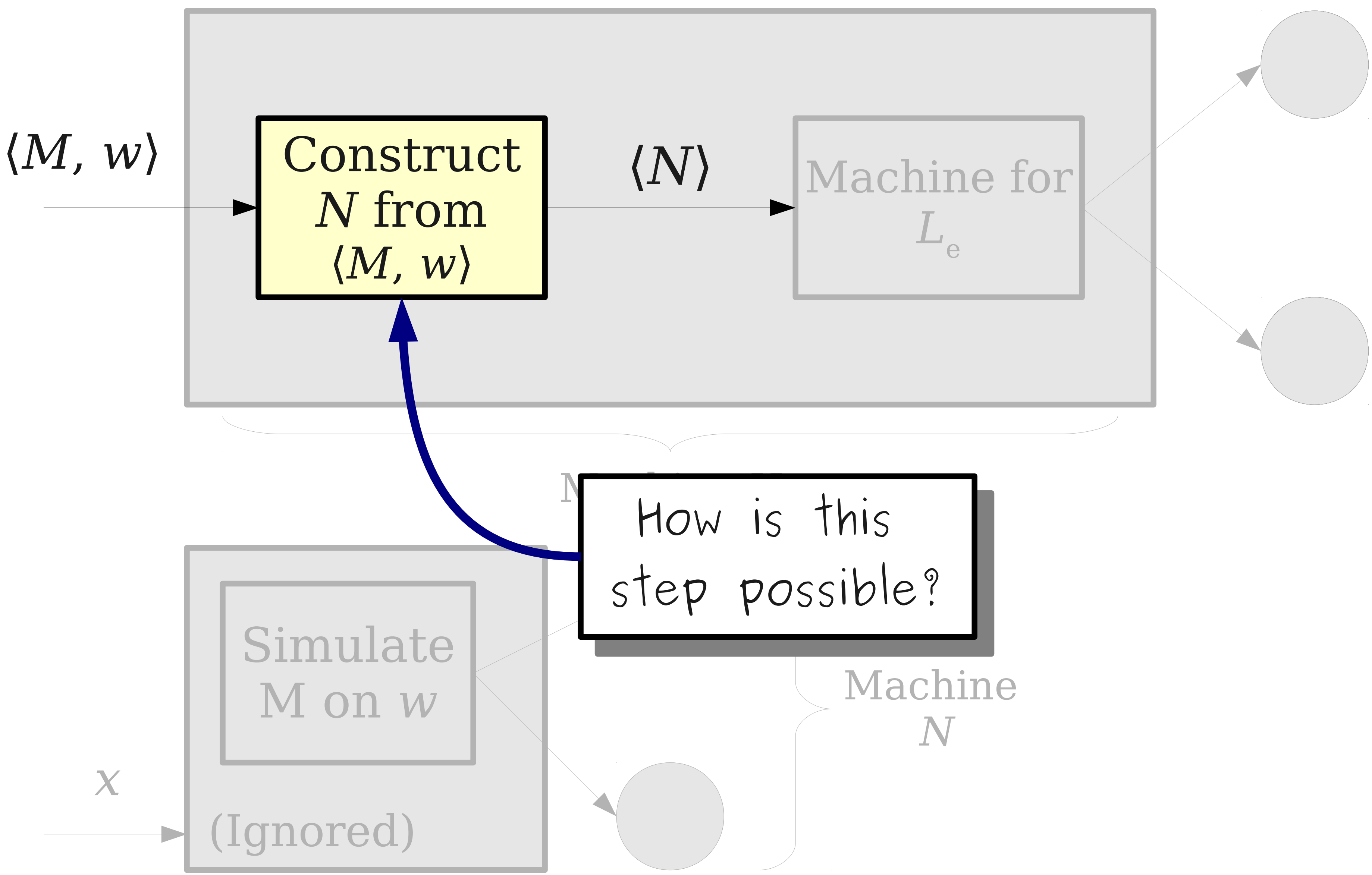
  If M accepts w, then N accepts x.

  If M rejects w, then N rejects x."
- Notice that N "amplifies" what M does on w:
  - If M does not accept w, N does not accept anything.
  - If M does accept w, N accepts everything.

The Reduction



The Reduction



Justifying N

- Notice that our machine N has the machine M and string w built into it!
- This is different from the machines we have constructed in the past.
- How do we justify that it's possible for some TM to construct a new TM at all?

N = "On input x:

Ignore x.

Run M on w.

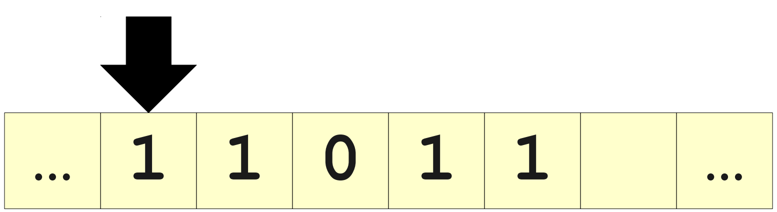
If M accepts w, accept.

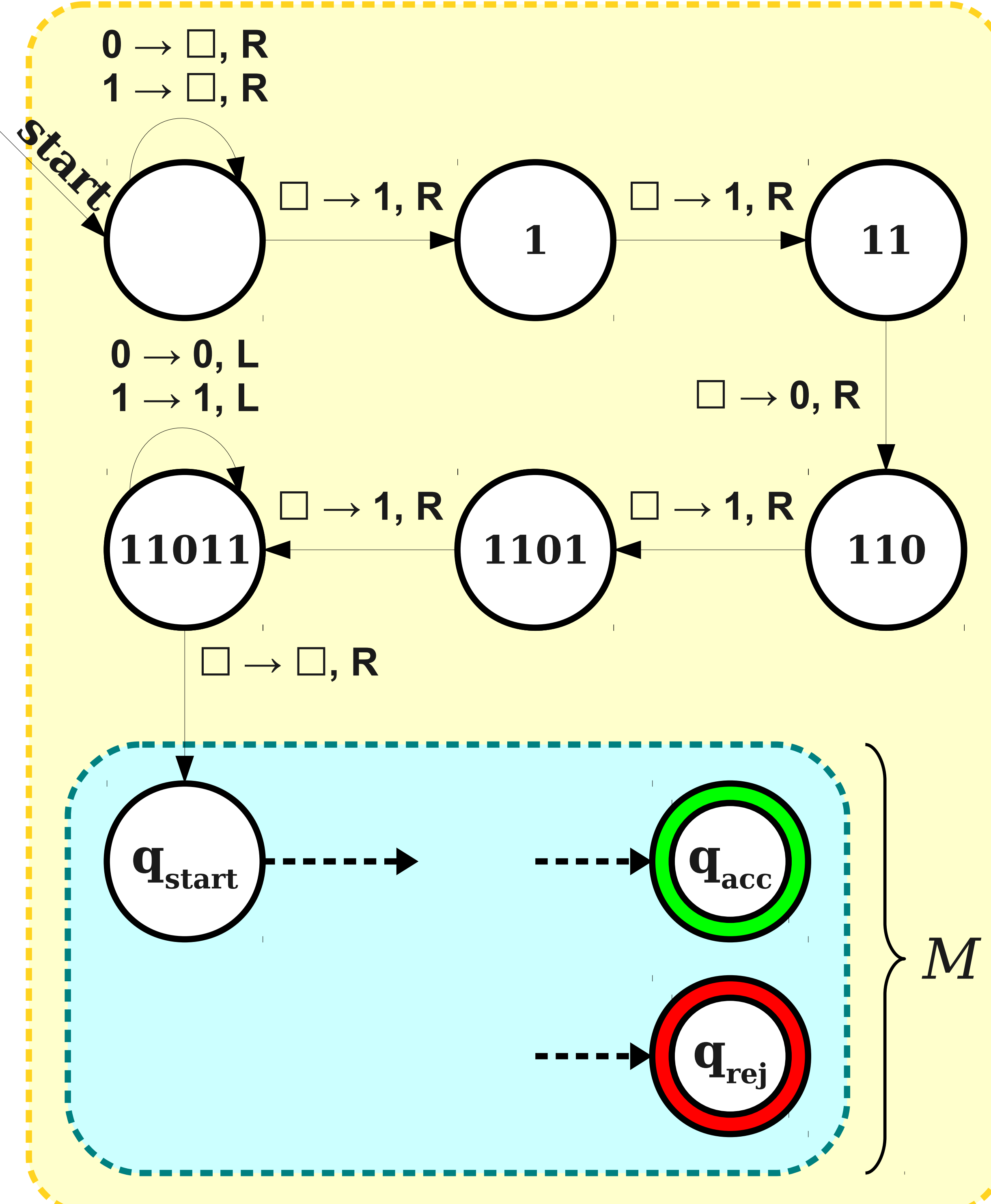
If M rejects w, reject."

N = "On input x:

- Ignore x.
- Run M on w.
- If M accepts w, then N accepts x.
- If M rejects w, then N rejects x."

Hypothetically, assume that w is the string **11011**.





The Takeaway Point

- Turing machines can embed TMs inside of other TMs.
- TMs of the following form are legal:

H = "On input $\langle M, w \rangle$, where M is a TM:
- Construct N = "On input x:
  - Do something with x.
  - Run M on w.
  - …"
- Do something with N."

Theorem: $\overline{A}_{TM} \leq_M L_e$.

Proof: We exhibit a mapping reduction from $\overline{A}_{TM}$ to L_e.

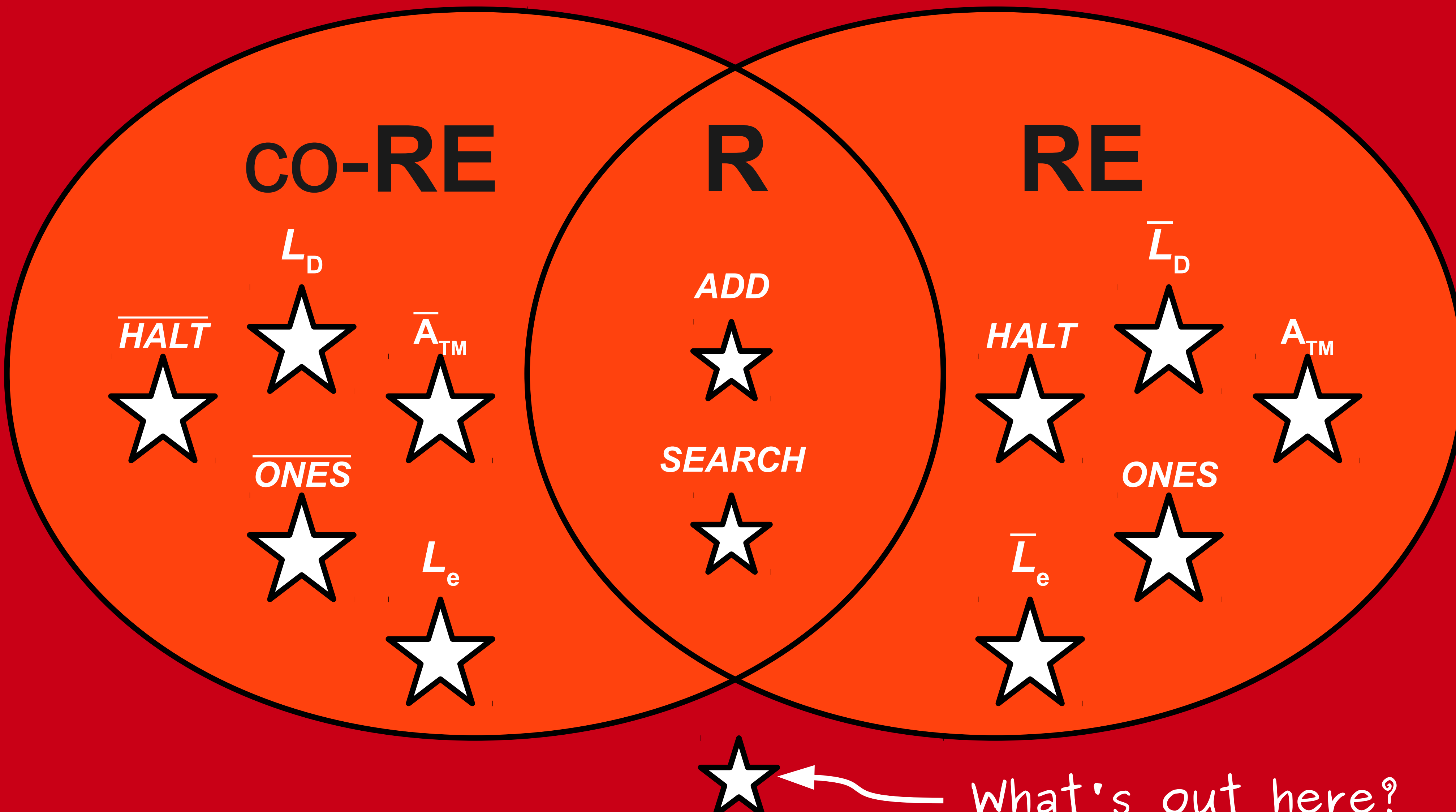
For any TM/string pair $\langle M, w \rangle$, let $f(\langle M, w \rangle) = \langle N \rangle$, where $\langle N \rangle$ is defined in terms of M and w as follows:

N = "On input x:
Ignore x.
Run M on w.
If M accepts w, then N accepts x.
If M rejects w, then N rejects x."

We state without proof that N is computable. We further claim that $\langle M, w \rangle \in \overline{A}_{TM}$ iff $f(\langle M, w \rangle) \in L_e$. To see this, note that $f(\langle M, w \rangle) = N \in L_e$ iff N does not accept any strings. We claim that N does not accept any strings iff M does not accept w. To see this, note that M does not accept w iff M loops on w or M rejects w. By construction, if M loops on w, then N loops on all strings, and if M rejects w, then N rejects all strings. Thus N does not accept any strings iff M does not accept w. Finally, M does not accept w iff $\langle M, w \rangle \in \overline{A}_{TM}$. Thus $\langle M, w \rangle \in \overline{A}_{TM}$ iff $f(\langle M, w \rangle) \in L_e$, so f is a mapping reduction from $\overline{A}_{TM}$ to L_e, and so $\overline{A}_{TM} \leq_M L_e$, as required. ■

Recitation Sections

The Limits of Computability



RE $\cup$ co-**RE** is Not Everything

- Using the same reasoning as the first day of lecture, we can show that there must be problems that are neither **RE** nor co-**RE**.
- There are more sets of strings than TMs.
- There are more sets of strings than twice the number of TMs.
- What do these languages look like?

An Extremely Hard Problem

- Recall: All regular languages are also **RE**.
- This means that some TMs accept regular languages and some TMs do not.
- Let $\text{REGULAR}_{\text{TM}}$ be the language of all TM descriptions that accept regular languages:

$$\textbf{REGULAR}_{\textbf{TM}} = \{ \langle M \rangle \mid \mathscr{L}(M) \text{ is regular } \}$$

- Is $\text{REGULAR}_{\text{TM}} \in$ **R**? How about **RE**? How about co-**RE**?

Building an Intuition

- If you were *convinced* that a TM had a regular language, how would you mechanically verify that?
- If you were *convinced* that a TM had a nonregular language, how would you mechanically verify that?
- Both of these seem difficult, if not impossible. Chances are $\text{REGULAR}_{\text{TM}}$ is neither **RE** nor co-**RE**.

$\text{REGULAR}_{\text{TM}} \notin \textbf{RE}$

- It turns out that $\text{REGULAR}_{\text{TM}}$ is unrecognizable, meaning that there is no computer program that can confirm that another TM's language is regular!
- To do this, we'll do a reduction from L_D and prove that $L_D \leq_M \text{REGULAR}_{\text{TM}}$.

$L_D \leq_M \text{REGULAR}_{TM}$

- We want to find a computable function f such that

$$\langle M \rangle \in L_D \quad \text{iff} \quad f(\langle M \rangle) \in \text{REGULAR}_{TM}.$$

- We need to choose N such that $f(\langle M \rangle) = \langle N \rangle$ for some TM N. Then

$$\langle M \rangle \in L_D \quad \text{iff} \quad f(\langle M \rangle) \in \text{REGULAR}_{TM}$$

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle N \rangle \in \text{REGULAR}_{TM}$$

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \mathscr{L}(N) \text{ is regular.}$$

- Question: How do we pick N?

$L_D \leq_M \text{REGULAR}_{TM}$

- We want to construct some N out of M such that
  - If $\langle M \rangle \in \mathscr{L}(M)$, then $\mathscr{L}(N)$ is not regular.
  - If $\langle M \rangle \notin \mathscr{L}(M)$, then $\mathscr{L}(N)$ is regular.
- One option: choose two languages, one regular and one nonregular, then construct N so its language switches from regular to nonregular based on whether $\langle M \rangle \notin \mathscr{L}(M)$.
  - If $\langle M \rangle \in \mathscr{L}(M)$, then $\mathscr{L}(N) = \{ \mathtt{0}^n\mathtt{1}^n \mid n \in \mathbb{N} \}$
  - If $\langle M \rangle \notin \mathscr{L}(M)$, then $\mathscr{L}(N) = \emptyset$

The Reduction

- We want to build N from M such that
  - If $\langle M \rangle \in \mathscr{L}(M)$, then $\mathscr{L}(N) = \{ \mathtt{0}^n\mathtt{1}^n \mid n \in \mathbb{N} \}$
  - If $\langle M \rangle \notin \mathscr{L}(M)$, then $\mathscr{L}(N) = \varnothing$
- Here is one way to do this:

N = "On input x:

If x does not have the form $\mathtt{0}^n\mathtt{1}^n$, reject.

Run M on $\langle M \rangle$.

If M accepts, accept x.

If M rejects, reject x."

Theorem: $L_D \leq_M \text{REGULAR}_{TM}$.

Proof: We exhibit a mapping reduction from L_D to REGULAR_{TM}.

For any TM M, let $f(\langle M \rangle) = \langle N \rangle$, where N is defined in terms of M as follows:

N = "On input x:

If x does not have the form 0^n1^n, then N rejects x.

Run M on $\langle M \rangle$.

If M accepts $\langle M \rangle$, then N accepts x.

If M rejects $\langle M \rangle$, then N rejects x."

We claim f is computable and omit the details from this proof. We further claim that $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \text{REGULAR}_{TM}$. To see this, note that $f(\langle M \rangle) = \langle N \rangle \in \text{REGULAR}_{TM}$ iff $\mathscr{L}(N)$ is regular. We claim that $\mathscr{L}(N)$ is regular iff $\langle M \rangle \notin \mathscr{L}(M)$. To see this, note that if $\langle M \rangle \notin \mathscr{L}(M)$, then N never accepts any strings. Thus $\mathscr{L}(N) = \emptyset$, which is regular. Otherwise, if $\langle M \rangle \in \mathscr{L}(M)$, then N accepts all strings of the form 0^n1^n, so we have that $\mathscr{L}(N) = \{ 0^n1^n \mid n \in \mathbb{N} \}$, which is not regular. Finally, $\langle M \rangle \notin \mathscr{L}(\langle M \rangle)$ iff $\langle M \rangle \in L_D$. Thus $\langle M \rangle \in L_D$ iff $f(\langle M \rangle) \in \text{REGULAR}_{TM}$, so f is a mapping reduction from L_D to REGULAR_{TM}. Therefore, $L_D \leq_M \text{REGULAR}_{TM}$. ■