Decidability

Problem set six is due in the box up front if using a late period.

The Language of U_{TM}

- Recall: For any TM M, the language of M, denoted $\mathscr{L}(M)$, is the set

$$\mathscr{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

- U_{TM} accepts $\langle M, w \rangle$ iff M is a TM that accepts w. Therefore:

$$\mathscr{L}(U_{TM}) = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$$

$$\mathscr{L}(U_{TM}) = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in \mathscr{L}(M) \}$$

- For simplicity, define $A_{TM} = \mathscr{L}(U_{TM})$. We will use the language A_{TM} extensively.

Diagonalization Revisited

- The **diagonalization language**, which we denote $\boldsymbol{L_D}$, is defined as

$$\boldsymbol{L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) }$$

- That is, L_D is the set of descriptions of Turing machines that do not accept themselves.
- **Theorem:** $L_D \notin$ **RE**.

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

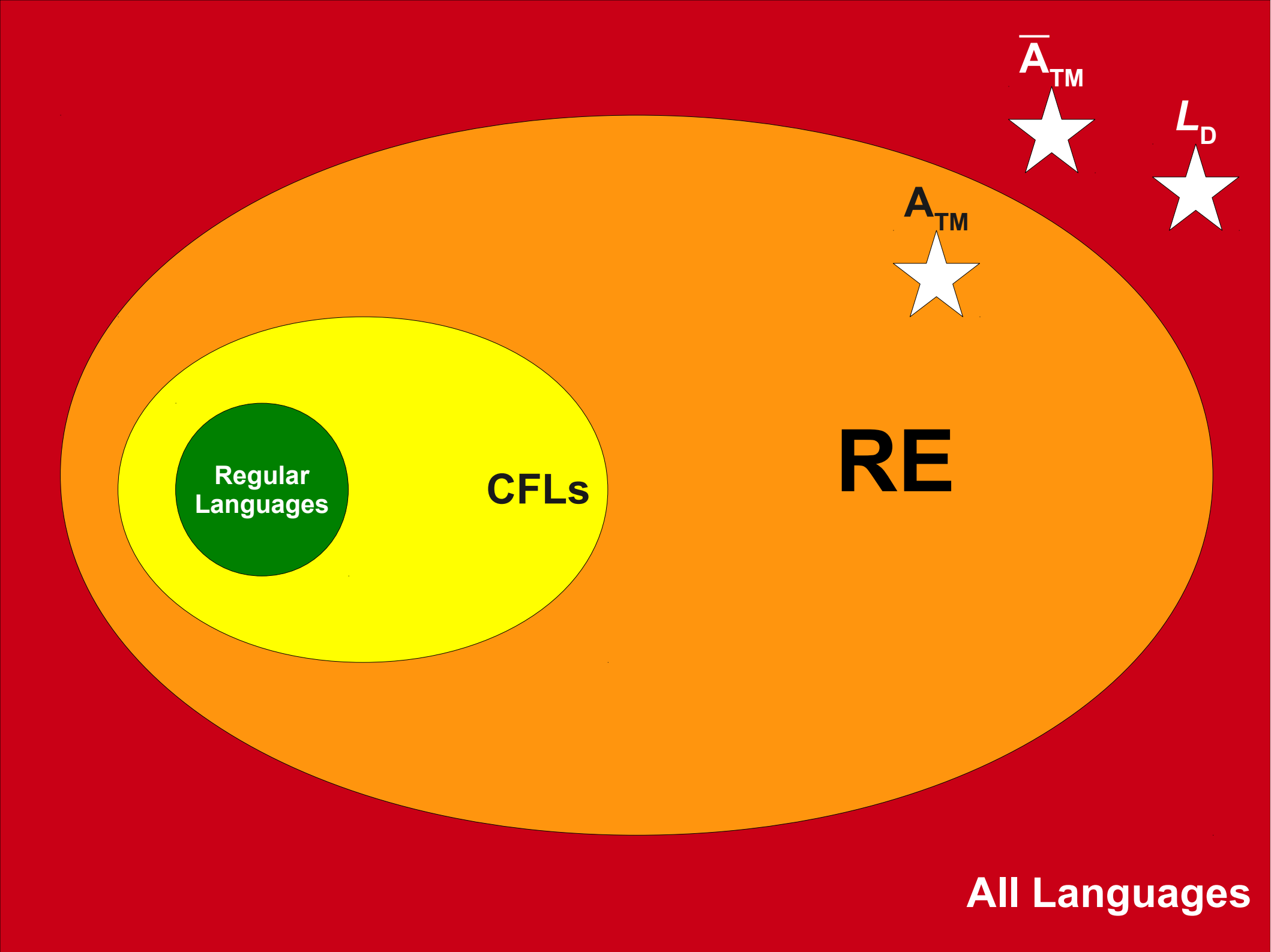
- The language A_{TM} is defined as

  $\{\langle M, w\rangle \mid M$ is a TM that accepts $w\}$

- Thus $\overline{A}_{TM}$ is the language

  $\{\langle M, w\rangle \mid M$ is a TM that doesn't accept $w\}$

- ***Theorem:*** $\overline{A}_{TM} \notin$ **RE**.


$\overline{A}_{TM}$
L_D
A_{TM}
Regular Languages
CFLs
RE
All Languages

Why All This Matters

- We *finally* have found concrete examples of unsolvable problems!
- We are starting to see a line of reasoning we can use to find unsolvable problems:
  - Start with a known unsolvable problem.
  - Try to show that the unsolvability of that problem entails the unsolvability of other problems.
- We will see this used extensively in the upcoming weeks.

Revisiting **RE**

Recall: Language of a TM

- The language of a Turing machine M, denoted $\mathscr{L}(M)$, is the set of all strings that M accepts:

$$\mathscr{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

- For any $w \in \mathscr{L}(M)$, M accepts w.
- For any $w \notin \mathscr{L}(M)$, M does not accept w.
  - It might loop forever, or it might explicitly reject.
- A language is called **recognizable** if it is the language of some TM.
- Notation: **RE** is the set of all recognizable languages.

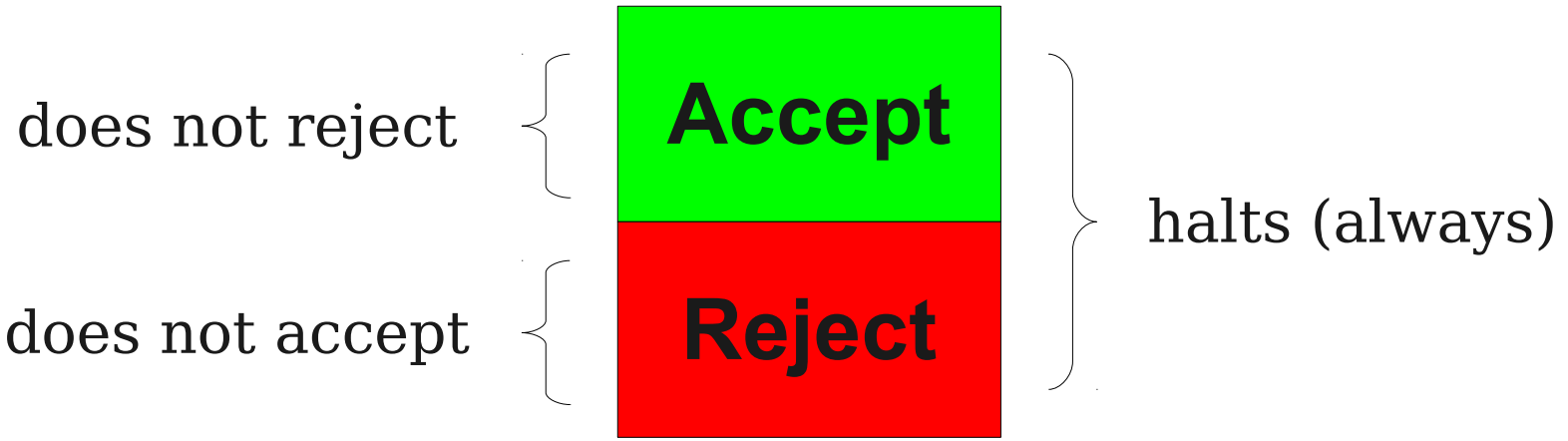
$L \in \mathbf{RE}$ iff L is recognizable

Why “Recognizable?”

- Given TM M with language $\mathscr{L}(M)$, running M on a string w will not necessarily tell you whether $w \in \mathscr{L}(M)$.
- If the machine is running, you can't tell whether
  - It is eventually going to halt, but just needs more time, or
  - It is never going to halt.
- However, if you know for a fact that $w \in \mathscr{L}(M)$, then the machine can confirm this (it eventually accepts).
- The machine can't *decide* whether or not $w \in \mathscr{L}(M)$, but it can *recognize* strings that are in the language.
- We sometimes call a TM for a language L a **recognizer** for L.

Deciders

- Some Turing machines always halt; they never go into an infinite loop.
- Turing machines of this sort are called **deciders**.
- For deciders, accepting is the same as not rejecting and rejecting is the same as not accepting.



Decidable Languages

- A language L is called **decidable** iff there is a decider M such that $\mathscr{L}(M) = L$.
- Given a decider M, you *can* learn whether or not a string $w \in \mathscr{L}(M)$.
  - Run M on w.
  - Although it might take a staggeringly long time, M will eventually accept or reject w.
- The set **R** is the set of all decidable languages.

$L \in \mathbf{R}$ iff L is decidable

R and **RE** Languages

- Intuitively, a language is in **RE** if there is some way that you could exhaustively search for a proof that $w \in L$.
  - If you find it, accept!
  - If you don't find one, keep looking!
- Intuitively, a language is in **R** if there is a concrete algorithm that can determine whether $w \in L$.
  - It tends to be *much* harder to show that a language is in **R** than in **RE**.

Examples of **R** Languages

- All regular languages are in **R**.
  - If L is regular, we can run the DFA for L on a string w and then either accept or reject w based on what state it ends in.
- $\{ \texttt{0}^n\texttt{1}^n \mid n \in \mathbb{N} \}$ is in **R**.
  - The TM we built last Monday is a decider.
- Verifying multiplication is in **R**.
  - The TM we built last Monday is a decider.

CFLs and **R**

- Using an NTM, we sketched a proof that all CFLs are in **RE**.
  - Nondeterministically guess a derivation, then deterministically check that derivation.
- Harder result: all CFLs are in **R**.
  - Read Sipser, Ch. 4.1 for details.
  - Or come talk to me after lecture!

Why **R** Matters

- If a language is in **R**, there is an algorithm that can decide membership in that language.
  - Run the decider and see what it says.
- If there is an algorithm that can decide membership in a language, that language is in **R**.
  - By the Church-Turing thesis, any effective model of computation is equivalent in power to a Turing machine.
  - Therefore, if there is *any* algorithm for deciding membership in the language, there is a decider for it.
  - Therefore, the language is in **R**.
- **A language is in R iff there is an algorithm for deciding membership in that language.**

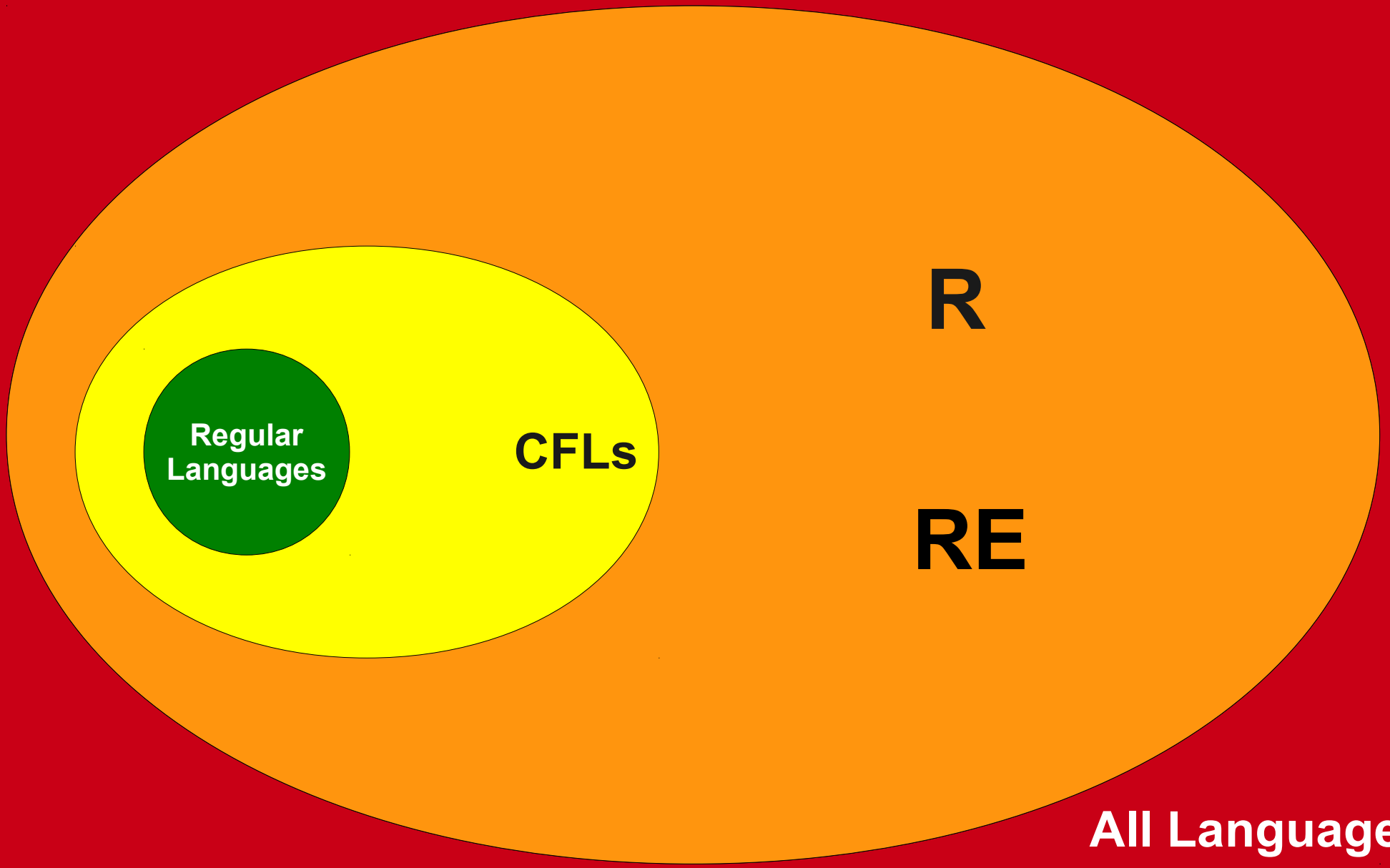
$\mathbf{R} \stackrel{?}{=} \mathbf{RE}$

- Every decider is a Turing machine, but not every Turing machine is a decider.
- Thus $\mathbf{R} \subseteq \mathbf{RE}$.
- Hugely important theoretical question:

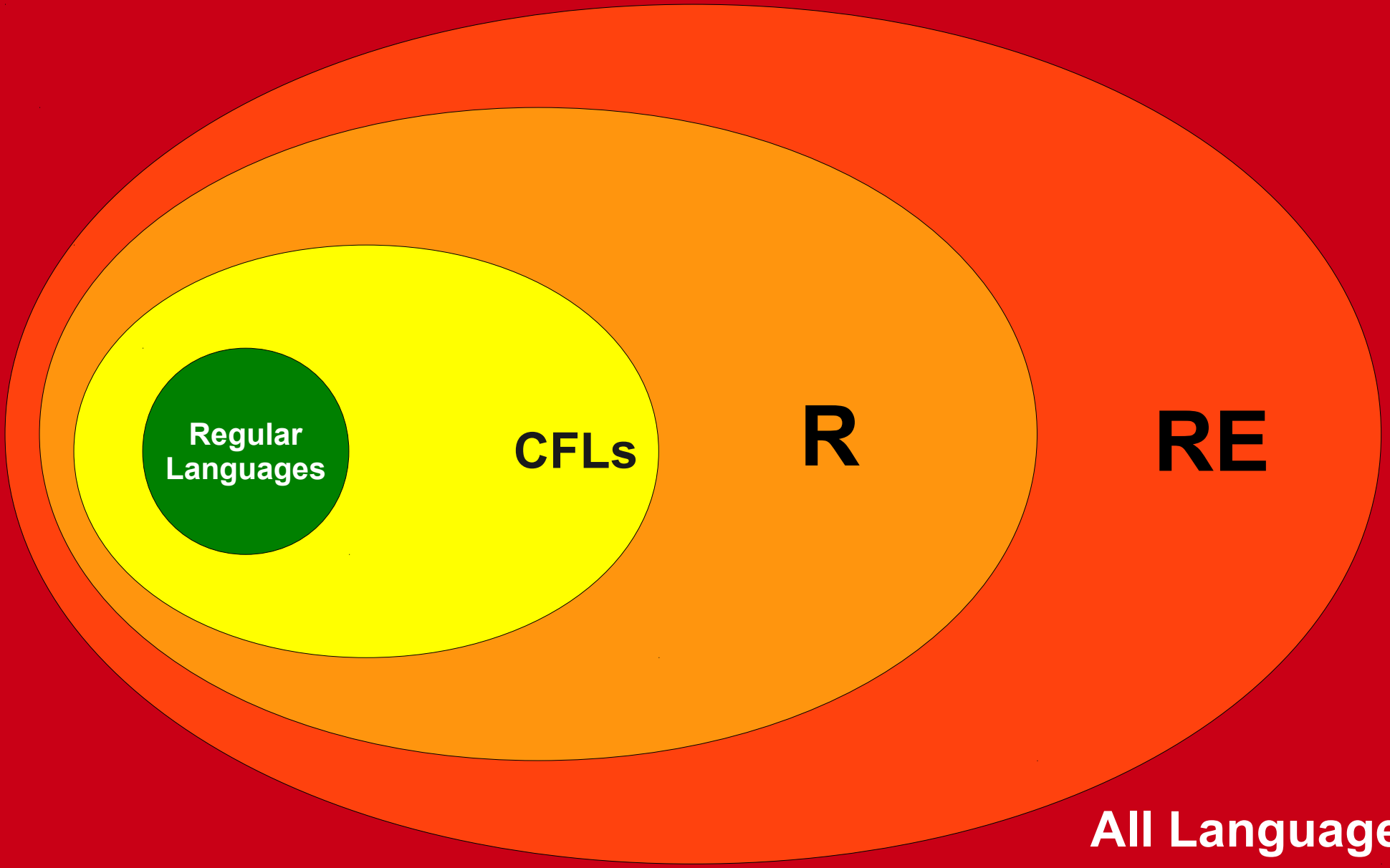
Is $\mathbf{R} = \mathbf{RE}$?

- That is, if we can *verify* that a string is in a language, can we *decide* whether that string is in the language?

Which Picture is Correct?



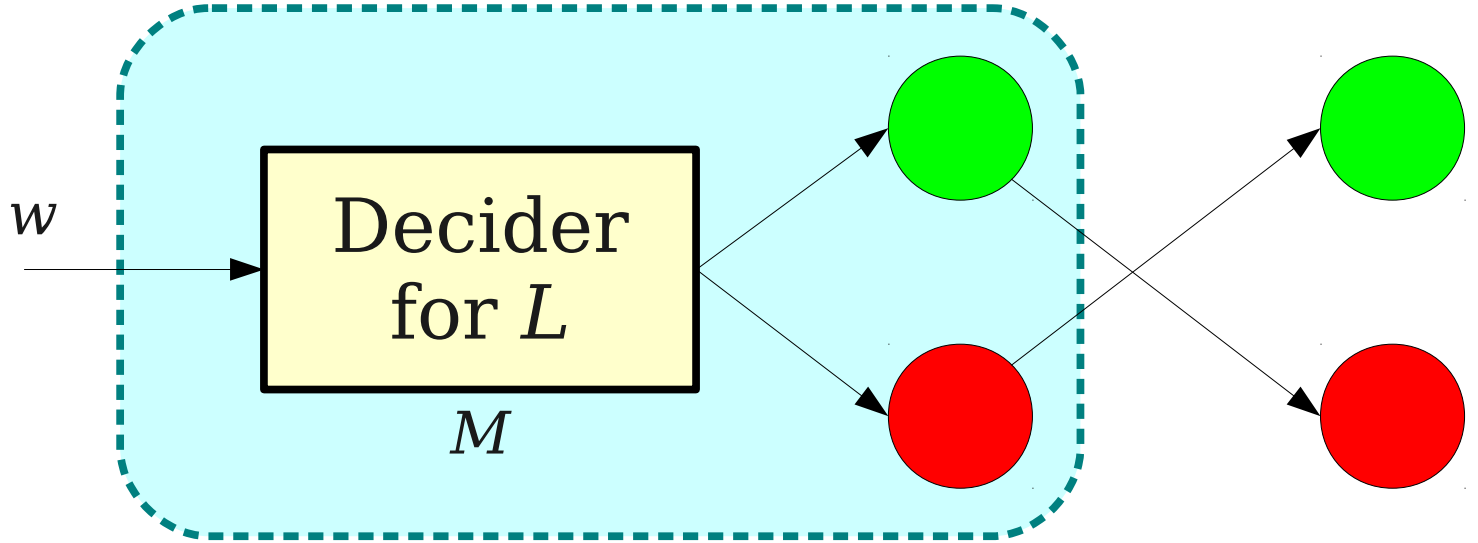
Which Picture is Correct?



An Important Observation

R is Closed Under Complementation

If $L \in \mathbf{R}$, then $\overline{L} \in \mathbf{R}$ as well.



M' = "On input w:
Run M on w.
If M accepts w, reject.
If M rejects w, accept."

Will this work if M is a **recognizer**, rather than a **decider**?

Theorem: **R** is closed under complementation.

Proof: Consider any $L \in \mathbf{R}$. We will prove that $\overline{L} \in \mathbf{R}$ by constructing a decider M' such that $\mathscr{L}(M') = \overline{L}$.

Let M be a decider for L. Then construct the machine M' as follows:

M' = "On input $w \in \Sigma^*$:
Run M on w.
If M accepts w, reject.
If M rejects w, accept."

We need to show that M' is a decider and that $\mathscr{L}(M') = \overline{L}$.

To show that M' is a decider, we will prove that it always halts. Consider what happens if we run M' on any input w. First, M' runs M on w. Since M is a decider, M either accepts w or rejects w. If M accepts w, M' rejects w. If M rejects w, M' accepts w. Thus M' always accepts or rejects, so M' is a decider.

To show that $\mathscr{L}(M') = \overline{L}$, we will prove that M' accepts w iff $w \in \overline{L}$. Note that M' accepts w iff $w \in \Sigma^*$ and M rejects w. Since M is a decider, M rejects w iff M does not accept w. M does not accept w iff $w \notin \mathscr{L}(M)$. Thus M' accepts w iff $w \in \Sigma^*$ and $w \notin \mathscr{L}(M)$, so M' accepts w iff $w \in \overline{L}$. Therefore, $\mathscr{L}(M') = \overline{L}$.

Since M' is a decider with $\mathscr{L}(M') = \overline{L}$, we have $\overline{L} \in \mathbf{R}$, as required. ■

$\mathbf{R} \stackrel{?}{=} \mathbf{RE}$

- We can now resolve the question of $\mathbf{R} \stackrel{?}{=} \mathbf{RE}$.
- If $\mathbf{R} = \mathbf{RE}$, we need to show that if there is a recognizer for *any* **RE** language L, there has to be a decider for L.
- If $\mathbf{R} \neq \mathbf{RE}$, we just need to find a single language in **RE** that is not in **R**.

A_{TM}

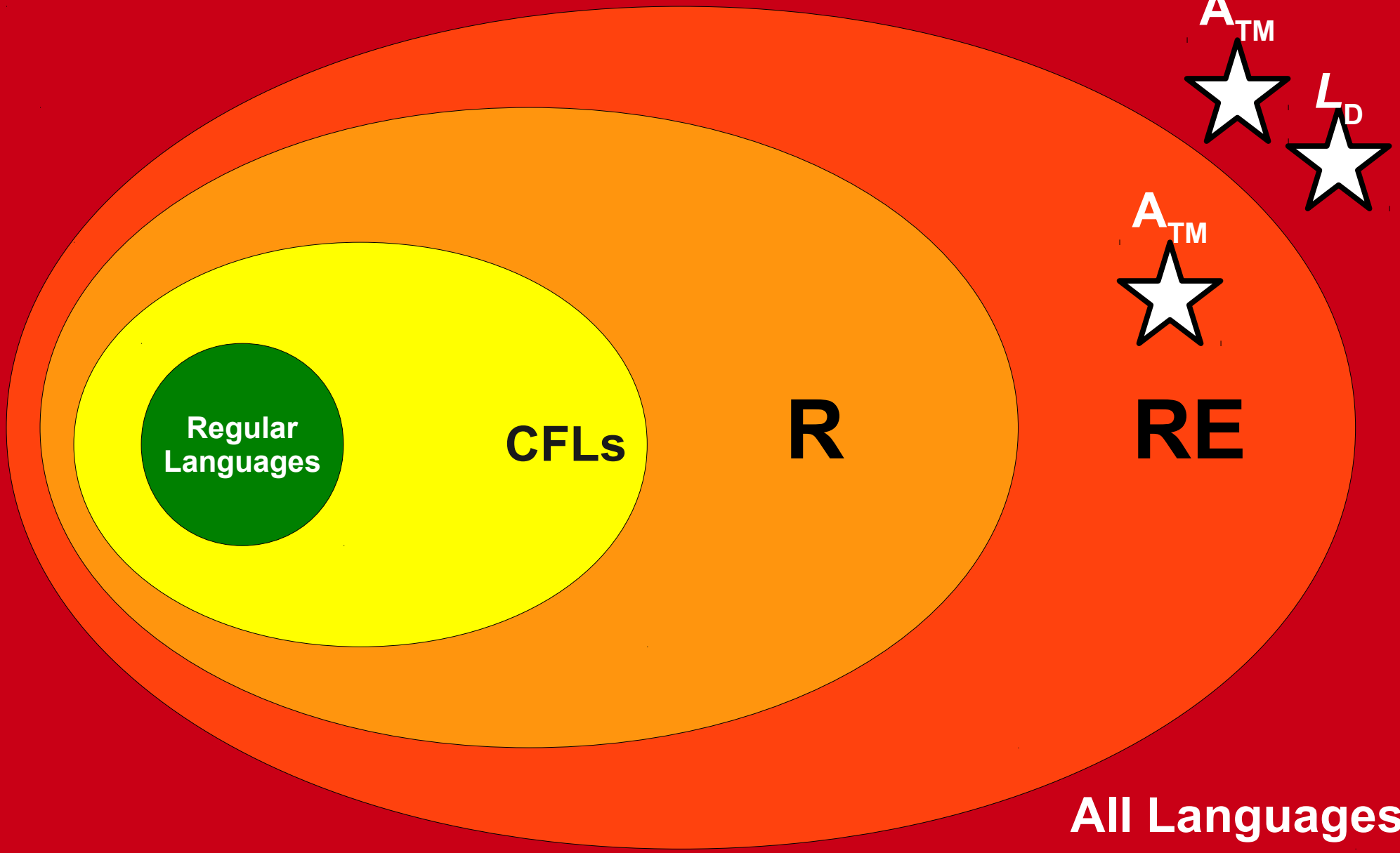
- Recall: the language A_{TM} is the language of the universal Turing machine U_{TM}.
- Consequently, $A_{TM} \in \mathbf{RE}$.
- Is $A_{TM} \in \mathbf{R}$?

Theorem: $A_{TM} \notin \mathbf{R}$.

Proof: By contradiction; assume $A_{TM} \in \mathbf{R}$. Since **R** is closed under complementation, this means that $\overline{A}_{TM} \in \mathbf{R}$. Since $\mathbf{R} \subseteq \mathbf{RE}$, this means that $\overline{A}_{TM} \in \mathbf{RE}$. But this is impossible, since we know $\overline{A}_{TM} \notin \mathbf{RE}$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $A_{TM} \notin \mathbf{R}$, as required. ■

The Limits of Computability



What this Means

- **Major result: R ≠ RE.**
- There are some problems where we can only *verify* yes answers but cannot *decide* whether the answer is yes or no.
- ***In a sense, it is fundamentally harder to solve a problem than it is to check a solution!***

What this Means, Part II

- The undecidability of A_{TM} means that we cannot "cheat" with Turing machines.
- We cannot necessarily build a TM to do an exhaustive search over a space (i.e. a recognizer), then decide whether it accepts without running it.
- **Intuition:** In most cases, you cannot *decide* what a TM will do without running it to see what happens.
  - In some cases, you can *recognize* when a TM has performed some task.
  - In some cases, you can do neither. For example, you cannot always recognize that a TM will not accept a string.

Time Out For Announcements!

Problem Set Five Graded

- We've just finished grading on-time Problem Set Five submissions; they'll be returned at the end of lecture today.
- Electronic submissions: Let us know if you don't hear back by 5PM today.

Your Questions

“Can we use known unsolvable problems as encryption and security mechanisms? If so, how?”

More on that in Week Ten!

“Is the human brain equivalent to a Turing Machine? If not, what makes it different?”

“Because we can program a Turing machine to pretty much do any computation, is there any list of steps human beings can do, which Turing machines cannot?”

“Now that we now how to find unsolvable problems, can we define the boundary between the solvable, and the unsolvable? Do we know with certainty what the limits are of the **RE** languages? If so, what are those problems?”

Another Undecidable Problem

L_D Revisited

- The diagonalization language L_D is the language

  $$\boldsymbol{L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}}$$

- As we saw before, $L_D \notin$ **RE**.
- But what about $\overline{L}_D$?

$\overline{L}_{\mathrm{D}}$

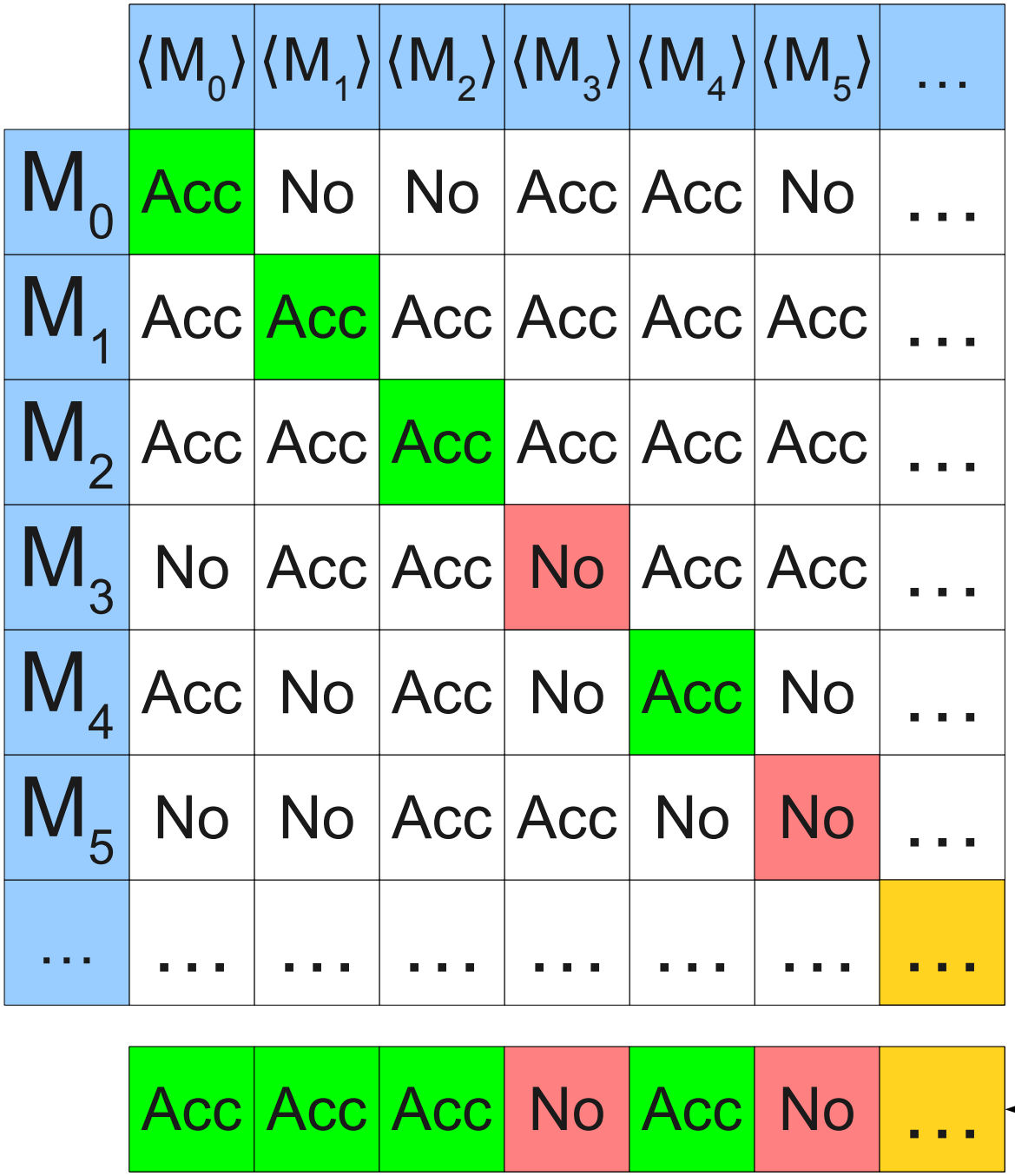
- The language L_{D} is the language

  $\boldsymbol{L_{\mathrm{D}} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M)\}}$

- Therefore, $\overline{L}_{\mathrm{D}}$ is the language

  $\boldsymbol{L_{\mathrm{D}} = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \in \mathscr{L}(M)\}}$

- Two questions:
  - What is this language?
  - Is this language **RE**?



{ ⟨*M*⟩ | *M* is a TM and ⟨*M*⟩ ∈ $\mathscr{L}$(M) }

This language is $\overline{L_D}$.

$\overline{L}_{\text{D}} \in \mathbf{RE}$

- Here's an TM for $\overline{L}_{\text{D}}$:

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

  Run M on $\langle M \rangle$.

  If M accepts $\langle M \rangle$, accept.

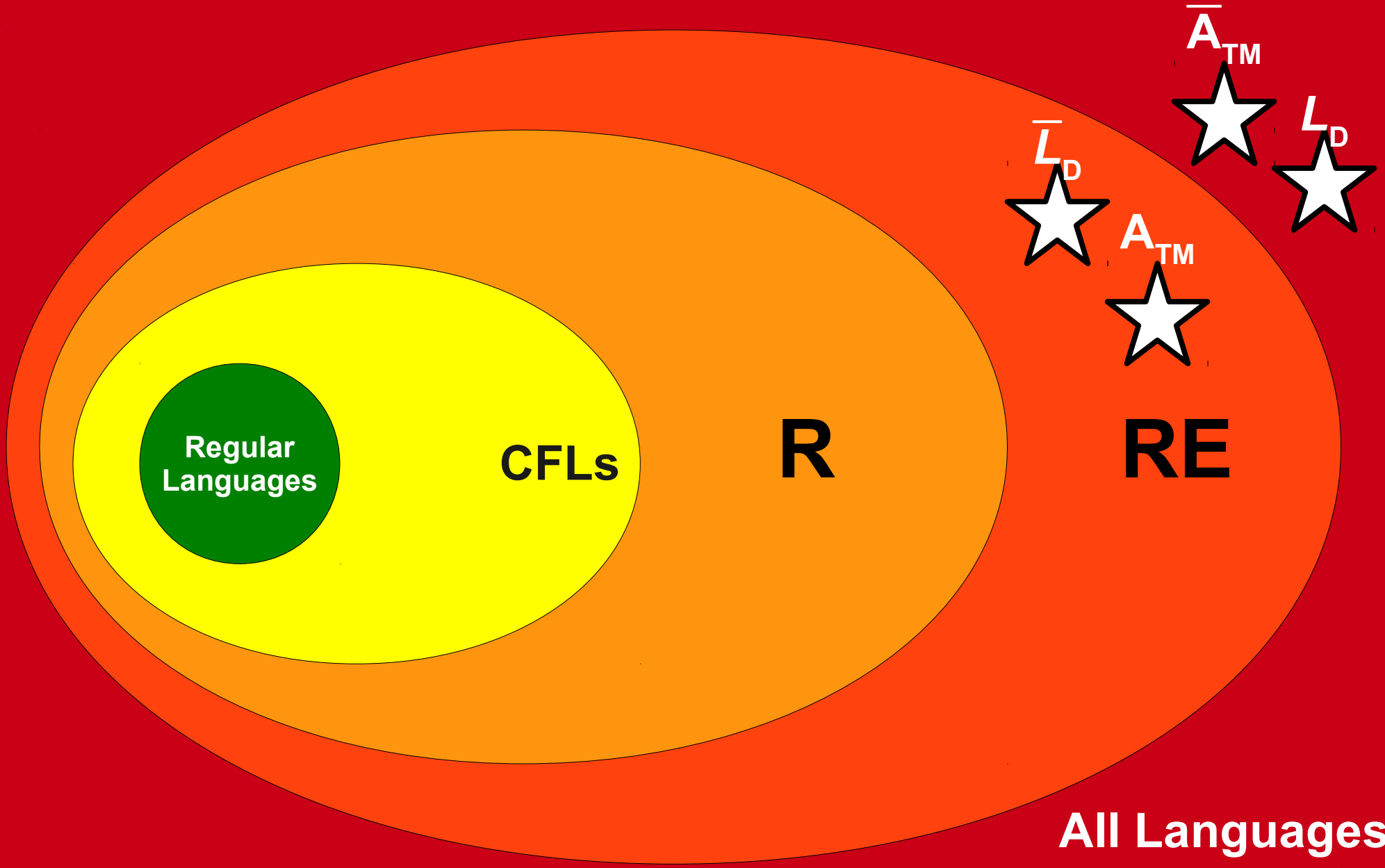
  If M rejects $\langle M \rangle$, reject."

- Then R accepts $\langle M \rangle$ iff $\langle M \rangle \in \mathscr{L}(M)$ iff $\langle M \rangle \in \overline{L}_{\text{D}}$, so $\mathscr{L}(R) = \overline{L}_{\text{D}}$.

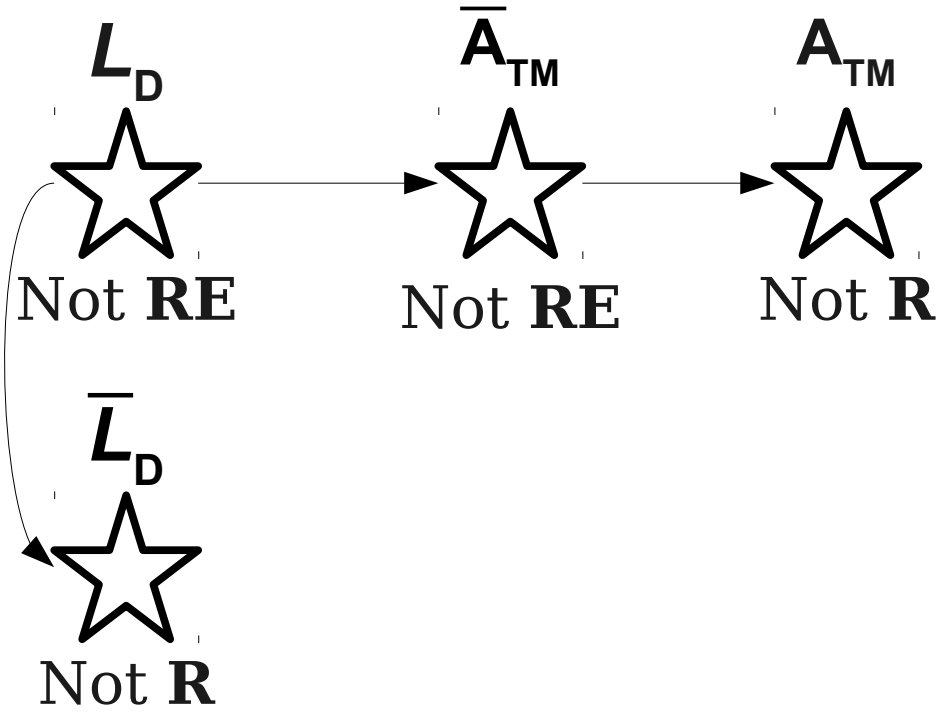
Is $\overline{L}_D$ Decidable?

- We know that $\overline{L}_D \in \mathbf{RE}$. Is $\overline{L}_D \in \mathbf{R}$?
- **No:** by a similar argument from before.
  - If $\overline{L}_D \in \mathbf{R}$, then $\overline{\overline{L}}_D = L_D \in \mathbf{R}$.
  - Since $\mathbf{R} \subset \mathbf{RE}$, this means that $L_D \in \mathbf{RE}$.
  - This contradicts that $L_D \notin \mathbf{RE}$.
  - So our assumption is wrong and $\overline{L}_D \notin \mathbf{R}$.

The Limits of Computability



Finding Unsolvable Problems



The Halting Problem

The Halting Problem

- The **halting problem** is the following problem:

  Given a TM *M* and string *w*,
  does *M* halt on *w*?

- Note that *M* doesn't have to *accept* *w*; it just has to *halt* on *w*.
- As a formal language:

  $$\textbf{\textit{HALT}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w. \}$$

- Is *HALT* ∈ **R**? Is *HALT* ∈ **RE**?

$HALT \in \mathbf{RE}$

- Consider this Turing machine:

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

  Run M on w.

  If M accepts, accept.

  If M rejects, accept."

- Then H accepts $\langle M, w \rangle$ iff M halts on w.
- Thus $\mathscr{L}(H) = HALT$, so $HALT \in \mathbf{RE}$.

Theorem: $HALT \notin \mathbf{R}$.

(The halting problem is undecidable)

Proving *HALT* $\notin$ **R**

- Our proof will work as follows:
  - Suppose that *HALT* $\in$ **R**.
  - Using a decider for *HALT*, construct a decider for A_{TM}.
  - Reach a contradiction, since there is no decider for A_{TM} ($A_{TM} \notin$ **R**).
  - Conclude, therefore, that *HALT* $\notin$ **R**.

Accepting, Rejecting, and Looping

- Suppose we have a TM M and a string w.
- Then M either
  - **Accepts**,
  - **Rejects**, or
  - **Loops**
- What if M never rejects?
- Then M either
  - **Accepts**, or
  - **Loops**.

The Key Insight

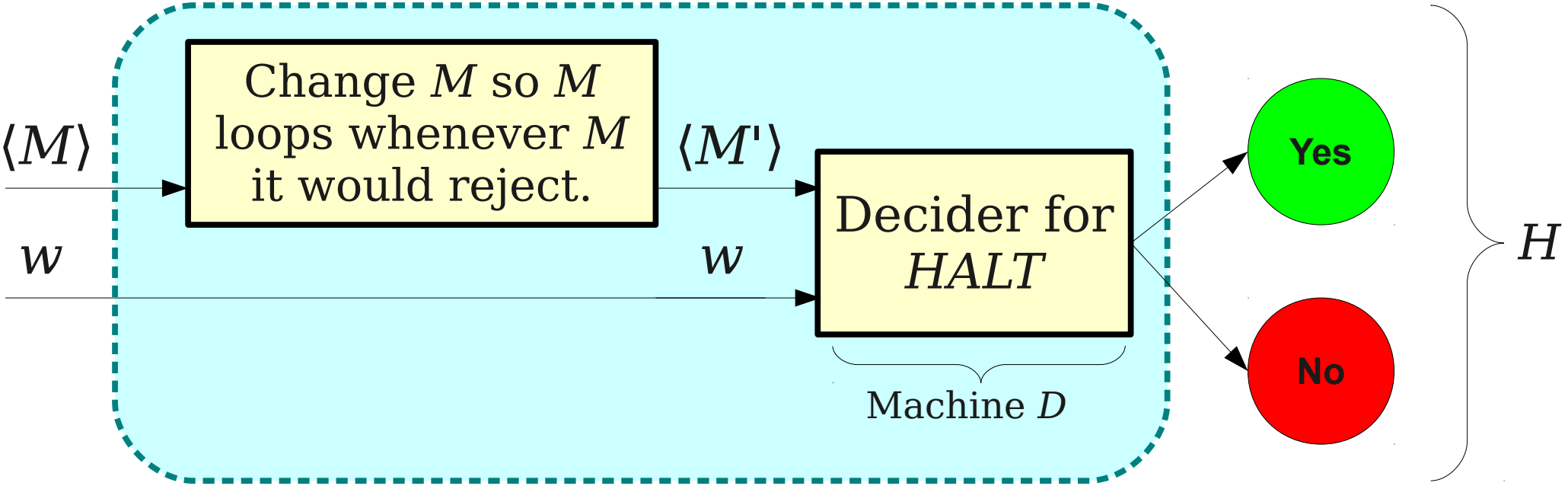
- If M never rejects, then

  M accepts w iff M halts on w

- In other words, if M never rejects, then

  $\langle M, w \rangle \in A_{TM}$ iff $\langle M, w \rangle \in HALT$

- If we can modify an arbitrary TM M so that M never rejects, then a decider for $HALT$ can be made to decide A_{TM}.
  - Since $A_{TM} \notin \mathbf{R}$, this is a contradiction!



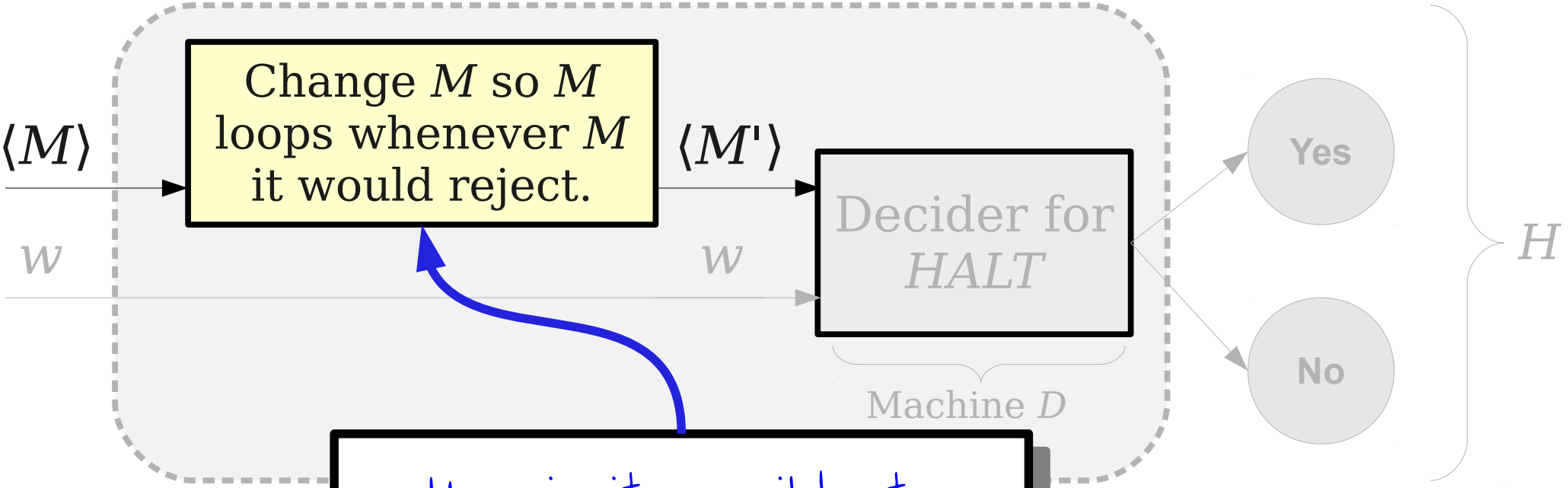
H = "On input $\langle M, w \rangle$:

- Transform M into M' by making M loop instead of rejecting.
- Run D on $\langle M', w \rangle$.
- If D accepts $\langle M', w \rangle$, then H accepts $\langle M, w \rangle$.
- If D rejects $\langle M', w \rangle$, then H rejects $\langle M, w \rangle$."

What happens if...

M accepts w? **Accept**
M loops on w? **Reject**
M rejects w? **Reject**

Machine H is a decider for A_{TM}!



How is it possible to build this part of the machine?

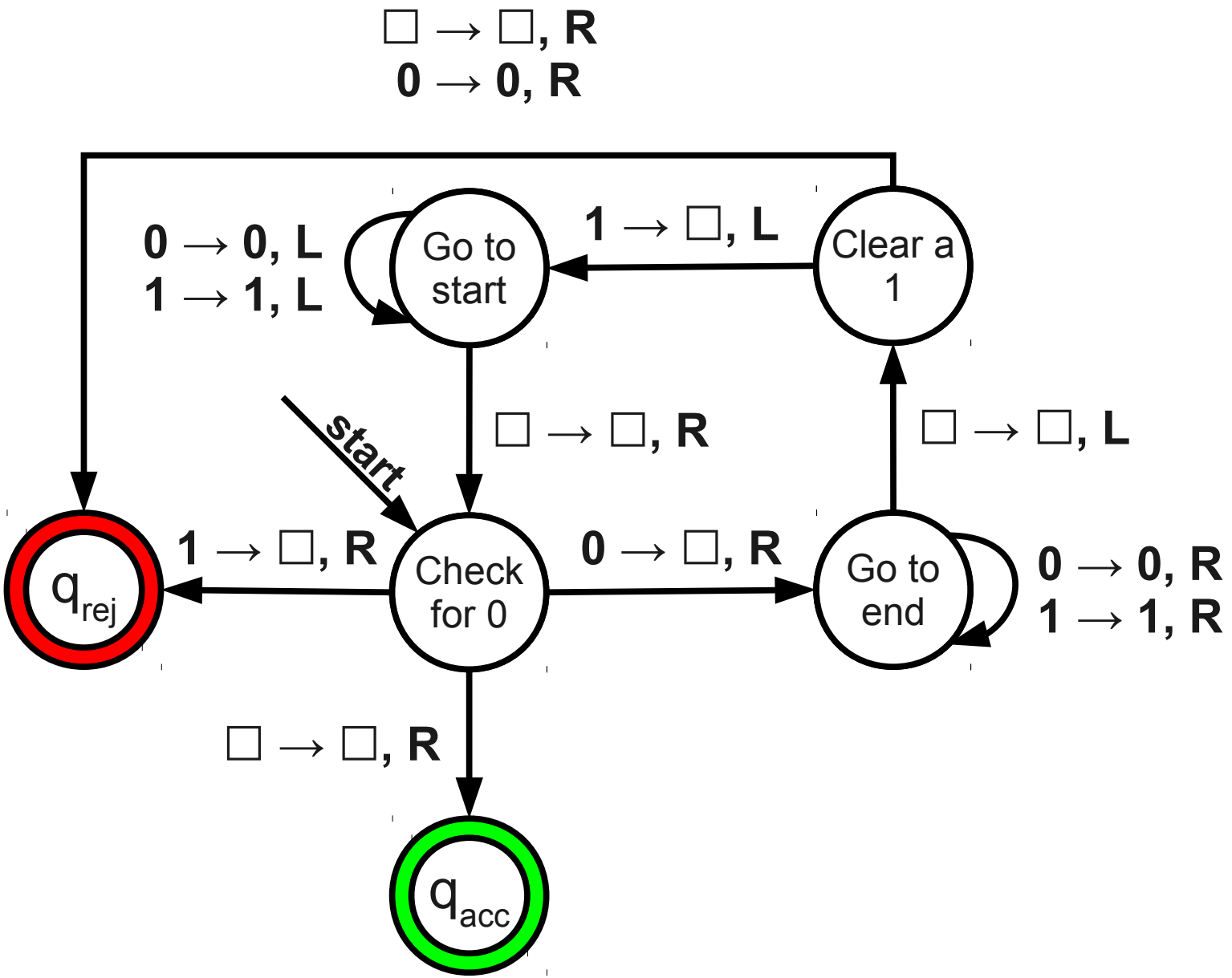
H = "On inpu

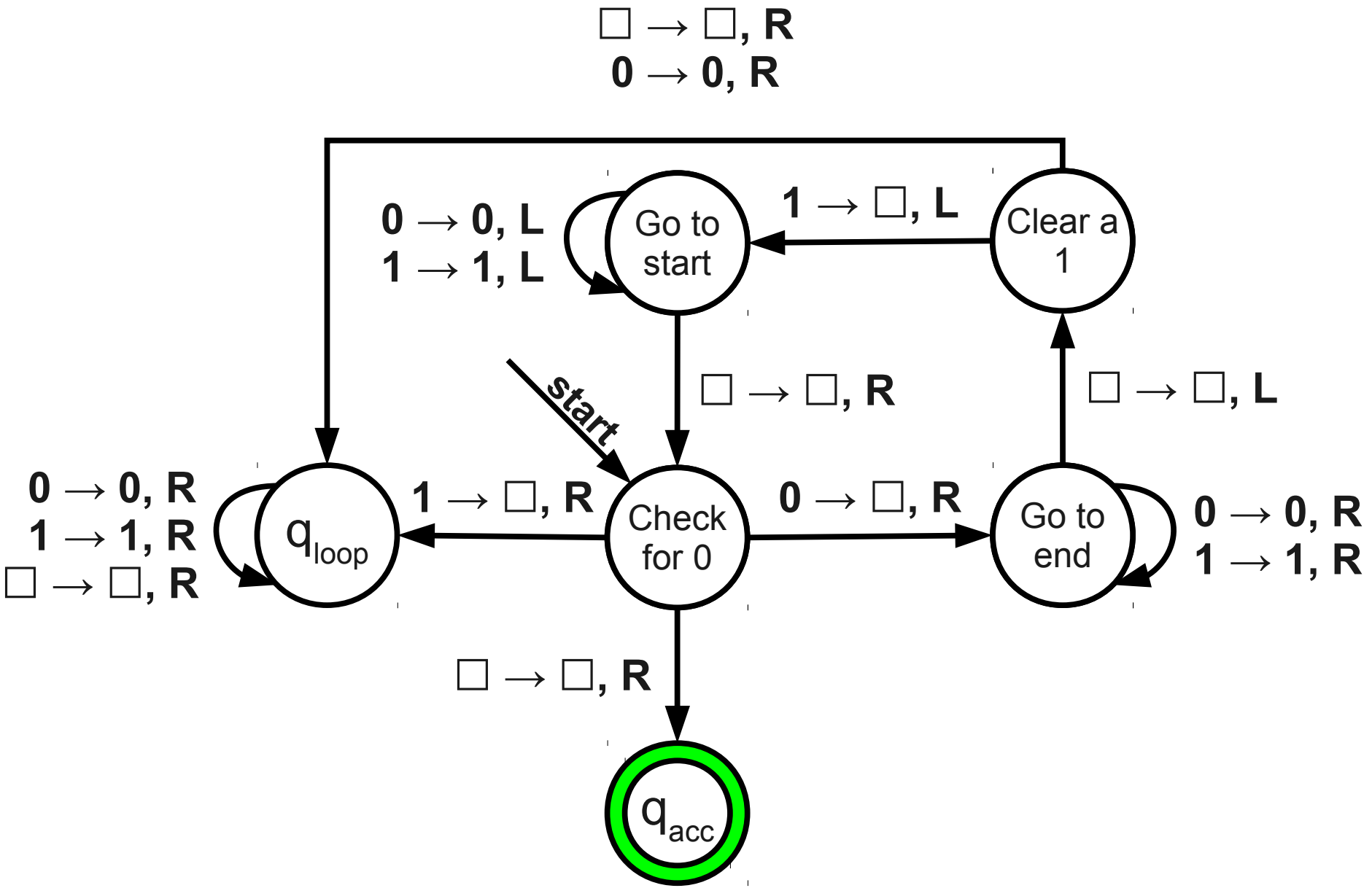
- Trans
  making M loop instead of rejecting.
- Run D on $\langle M', w \rangle$.
- If D accepts $\langle M', w \rangle$, then H accepts $\langle M, w \rangle$.
- If D rejects $\langle M', w \rangle$, then H rejects $\langle M, w \rangle$."

hat happens if...

ccepts w? **Accept**
M loops on w? **Reject**
M rejects w? **Reject**

Machine H is a decider for A_{TM}!


□ → □, R
0 → 0, R
0 → 0, L
1 → 1, L
Go to start
1 → □, L
Clear a 1
start
□ → □, R
□ → □, L
1 → □, R
Check for 0
0 → □, R
Go to end
0 → 0, R
1 → 1, R
q_rej
□ → □, R
q_acc


□ → □, R
0 → 0, R
0 → 0, L
1 → 1, L
Go to start
1 → □, L
Clear a 1
start
□ → □, R
□ → □, L
0 → 0, R
1 → 1, R
□ → □, R
q_loop
1 → □, R
Check for 0
0 → □, R
Go to end
0 → 0, R
1 → 1, R
□ → □, R
q_acc

Theorem: HALT $\notin$ **R**.

Proof: By contradiction; assume *HALT* $\in$ **R**. Then there must be some decider *D* for *HALT*. Consider the following TM *H*:

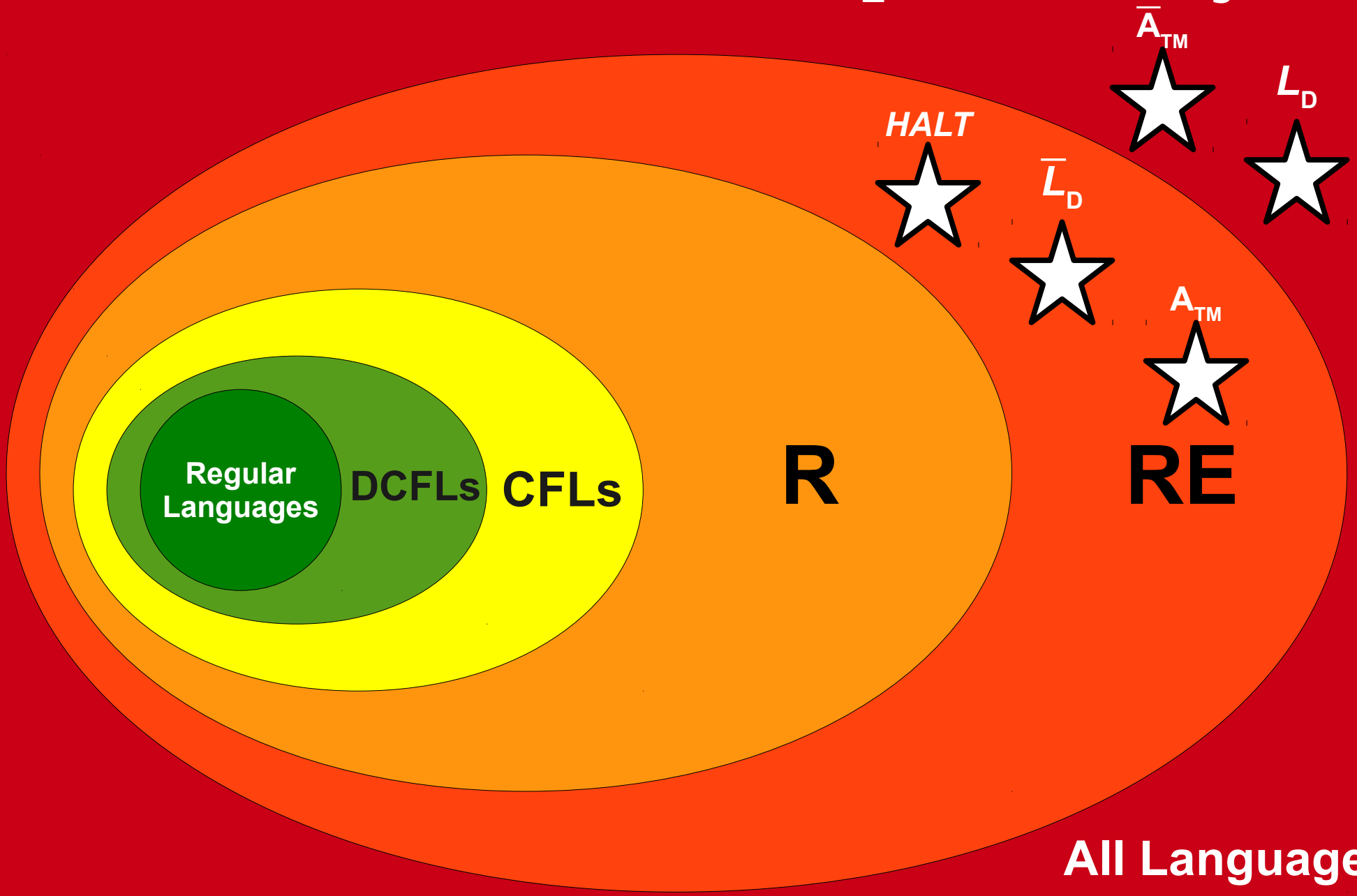
> *H* = "On input $\langle M, w \rangle$, where *M* is a TM and *w* is a string:
> Transform *M* into *M*' by making *M*' loop whenever *M* rejects.
> Run *D* on $\langle M', w \rangle$.
> If *D* accepts $\langle M', w \rangle$, then *H* accepts $\langle M, w \rangle$.
> If *D* rejects $\langle M', w \rangle$, then *H* rejects $\langle M, w \rangle$."

We claim that *H* is a decider for A_{TM}. This means that $A_{TM} \in$ **R**, which contradicts the fact that $A_{TM} \notin$ **R**. This means our assumption was wrong, and so *HALT* $\notin$ **R**, as required.

First, we prove *H* is a decider. Note that on any input $\langle M, w \rangle$, *H* constructs the machine *M*' (which can be done in finite time), then runs *D* on $\langle M', w \rangle$. Since *D* is a decider, *D* always halts. Since *H* halts as soon as *D* halts, we know *H* halts on $\langle M, w \rangle$. Since our choice of $\langle M, w \rangle$ was arbitrary, this means that *H* halts on all inputs, so *H* is a decider.

Next, we prove that $\mathscr{L}(H) = A_{TM}$. To see this, note that *H* accepts $\langle M, w \rangle$ iff *D* accepts $\langle M', w \rangle$. Since *D* decides *HALT*, *D* accepts $\langle M', w \rangle$ iff *M*' halts on *w*. By construction, *M*' halts iff it accepts, so *M*' halts on *w* iff *M*' accepts *w*. Again by construction, *M*' accepts *w* iff *M* accepts *w*. Finally, *M* accepts *w* iff $\langle M, w \rangle \in A_{TM}$. Thus *H* accepts $\langle M, w \rangle$ iff $\langle M, w \rangle \in A_{TM}$, and so $\mathscr{L}(H) = A_{TM}$, as required. ■

The Limits of Computability



A_{TM} and *HALT*

- Both A_{TM} and *HALT* are undecidable.
  - There is no way to decide whether a TM will accept or eventually terminate.
- However, both A_{TM} and *HALT* are recognizable.
  - We can always run a TM on a string *w* and accept if that TM accepts or halts.
- **Intuition:** The only general way to learn what a TM will do on a given string is to run it and see what happens.

Next Time

- **Co-recognizability**
  - Resolving an asymmetry with **R** and **RE**.
- **Mapping Reductions I**
  - A powerful tool for finding impossible problems.