

Unsolvable Problems


Outline for Today

- **Recap from Last Time**
 - Wow, we covered a lot. Where are we again?
- **R and RE Languages**
 - What does it mean to solve a problem?
- **Encodings**
 - Computing over general objects.
- **The Universal Turing Machine**
 - One machine to run them all.
- **Impossible Problems**
 - Discovering truly impossible problems.

Recap from Last Time

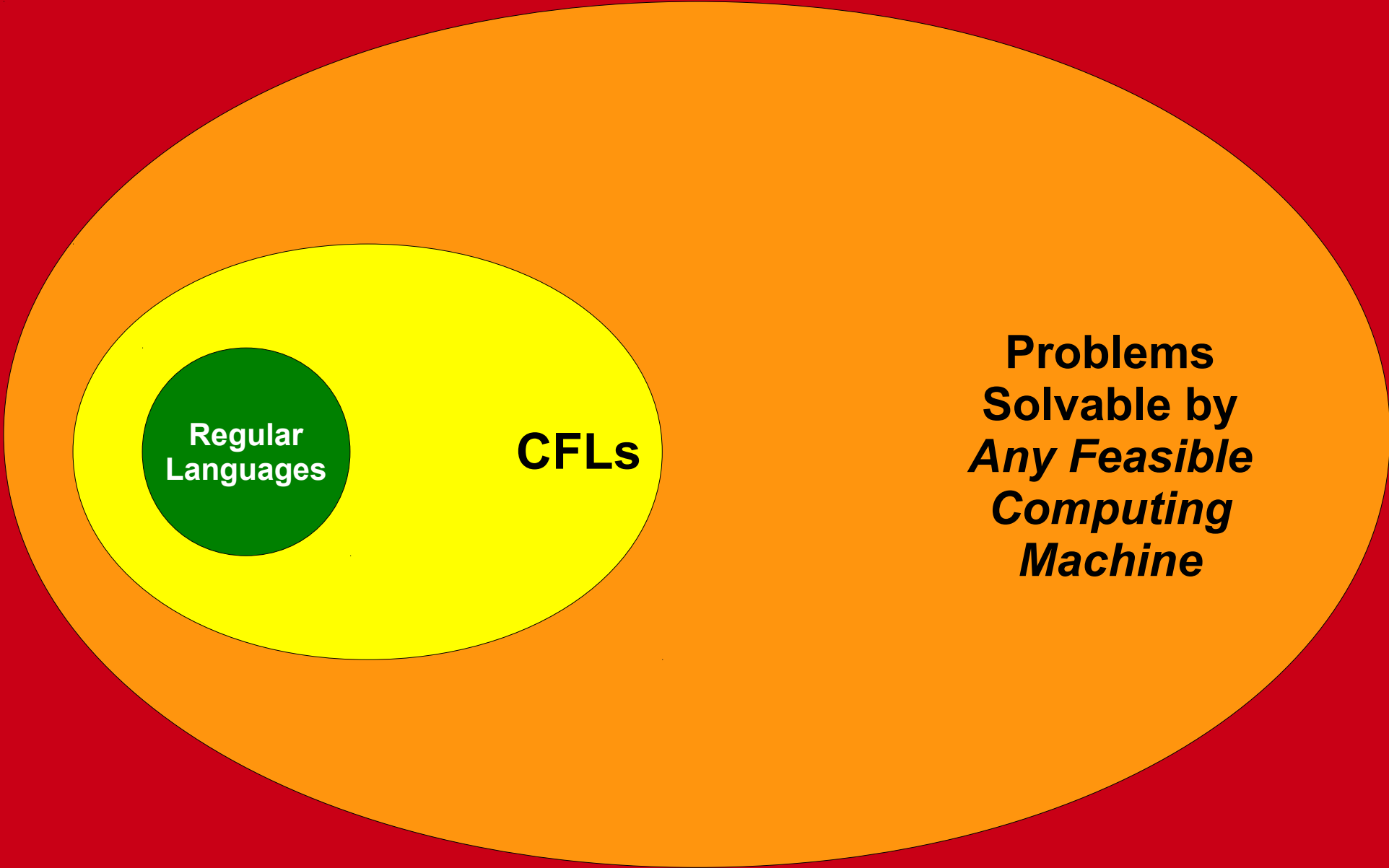
What problems can we solve with a computer?

What kind of
computer?



The *Church-Turing Thesis* claims that
*every effective method of computation
is either equivalent to or weaker than a
Turing machine.*

This is not a mathematical fact – it's a
hypothesis about the nature of
computation.



Regular
Languages

CFLs

**Problems
Solvable by
*Any Feasible
Computing
Machine***

All Languages

**Regular
Languages**

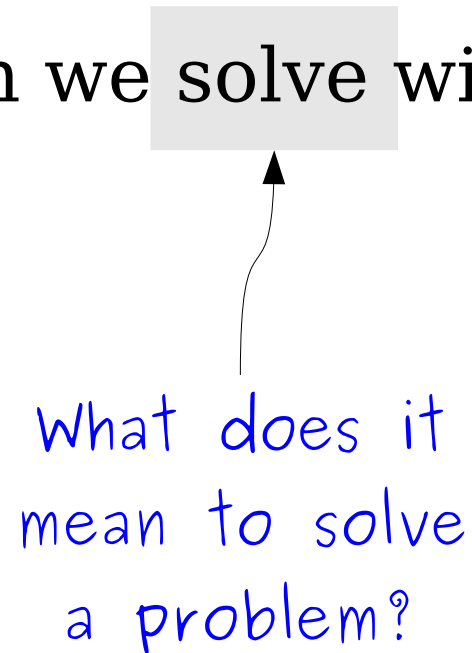
CFLs

**Languages
Recognized
by Turing
Machines**

All Languages

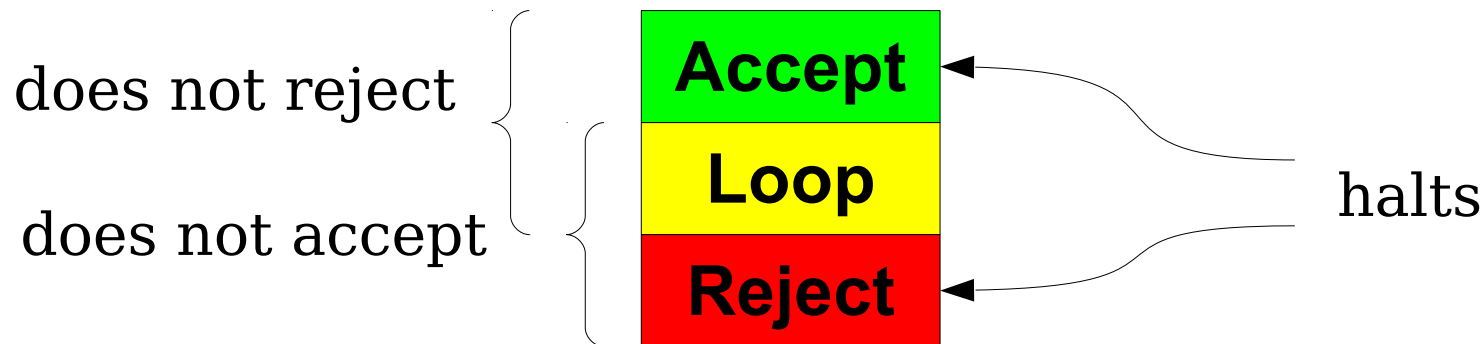
What problems can we solve with a computer?

what does it
mean to solve
a problem?



Some Important Terminology

- Let M be a Turing machine.
- M **accepts** a string w if it enters the accept state when run on w .
- M **rejects** a string w if it enters the reject state when run on w .
- M **loops infinitely** (or just **loops**) on a string w if when run on w it enters neither the accept or reject state.
- M **does not accept w** if it either rejects w or loops infinitely on w .
- M **does not reject w** if it either accepts w or loops on w .
- M **halts on w** if it accepts w or rejects w .



The Language of a TM

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

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

- For any $w \in \mathcal{L}(M)$, M accepts w .
- For any $w \notin \mathcal{L}(M)$, 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: the class **RE** is the set of all recognizable languages.

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

The Power of TMs

- Because TMs only need to accept strings in their languages, many problems can be formulated as **RE** languages.
 - Any context-free language: simulate all possible production rules and see if the target string can be derived.
 - Solving a maze – use the worklist to explore all paths of length 0, 1, 2, ... until a solution is found.
 - Determining whether a polynomial has an integer zeros: try 0, -1, +1, -2, +2, -3, +3, ... until a result is found.

Why “Recognizable?”

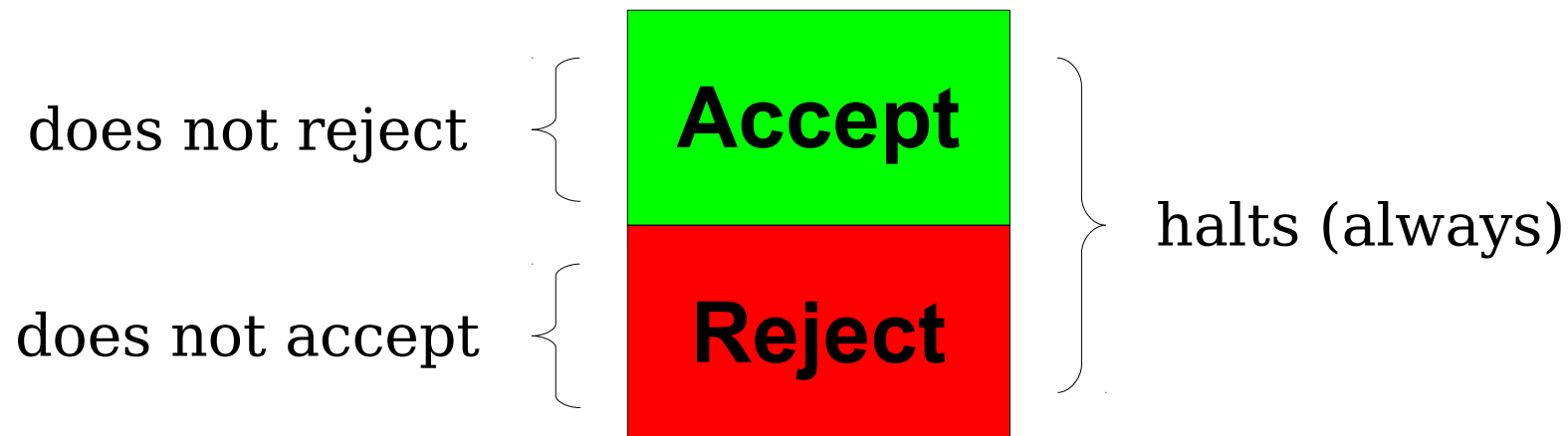
- Given TM M with language $\mathcal{L}(M)$, running M on a string w will not necessarily tell you whether $w \in \mathcal{L}(M)$.
- If the machine is running, as an observer, you can't tell whether
 - it is eventually going to halt, but just needs more time, or
 - it is never going to halt.
- However,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

Is this a satisfactory definition
of “solving” a problem?

New Stuff!

Deciders

- Some Turing machines always halt; they never go into an infinite loop.
- If M is a TM and M halts on every possible input, then we say that M is a **decider**.
- 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** if there is a decider M such that $\mathcal{L}(M) = L$.
- Given a decider M , you *can* learn whether or not a string $w \in \mathcal{L}(M)$.
 - Run M on w .
 - Although it might take a staggeringly long time, M will eventually accept or reject w .
- The class **R** is the set of all decidable languages.

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

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.
- $\{ 0^n 1^n \mid n \in \mathbb{N} \}$ is in **R**.
 - The TM we built on Wednesday is a decider.
- $\{ 1^n \mid n \in \mathbb{N} \text{ and } n \text{ is composite} \}$ is in **R**.
 - The TM we built on Friday is a decider.

CFLs and **R**

- With a worklist approach, we can build a recognizer that checks membership in any CFL.
- 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 if and only there is an algorithm for deciding membership in that language.***

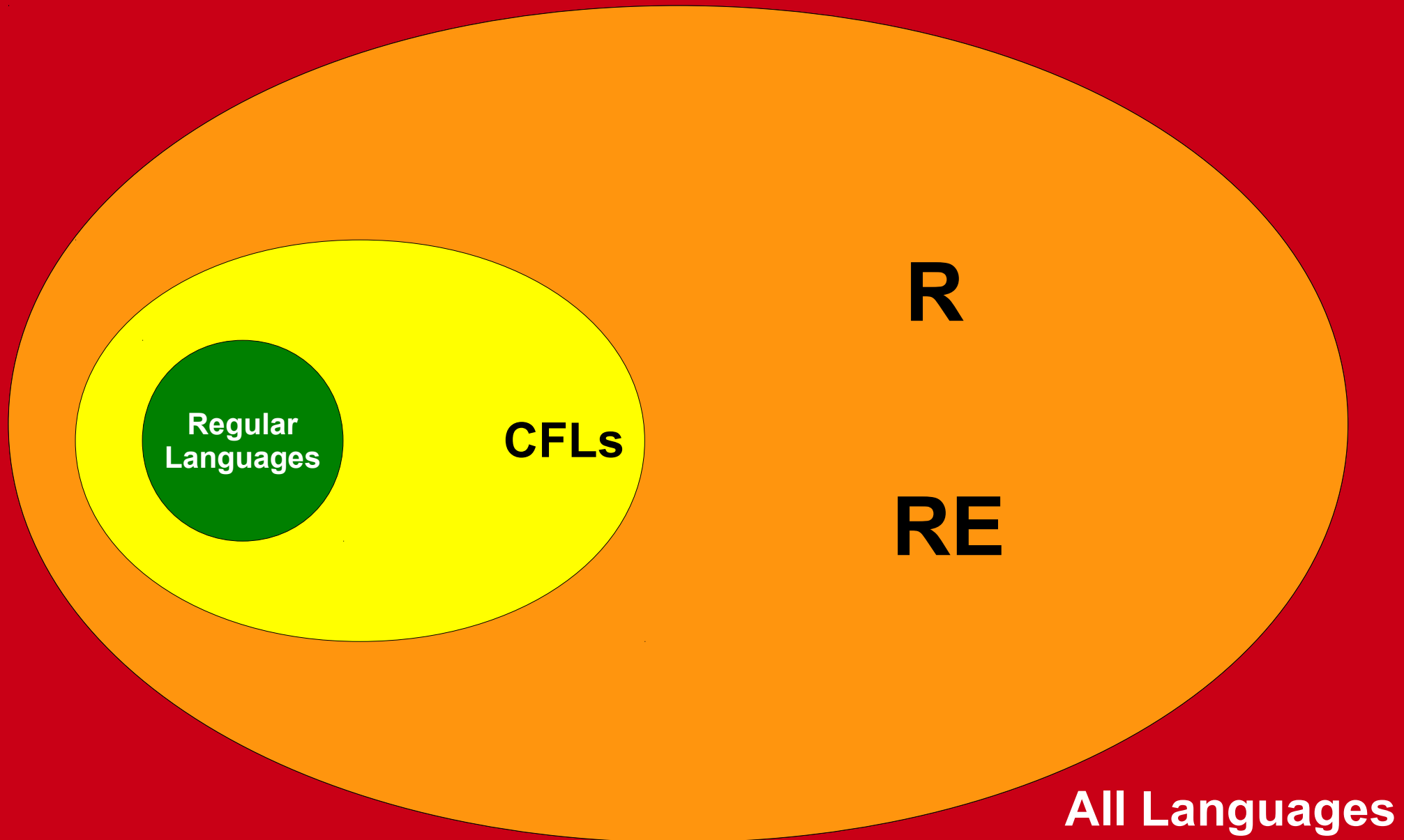
$$\mathbf{R} \stackrel{?}{=} \mathbf{RE}$$

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

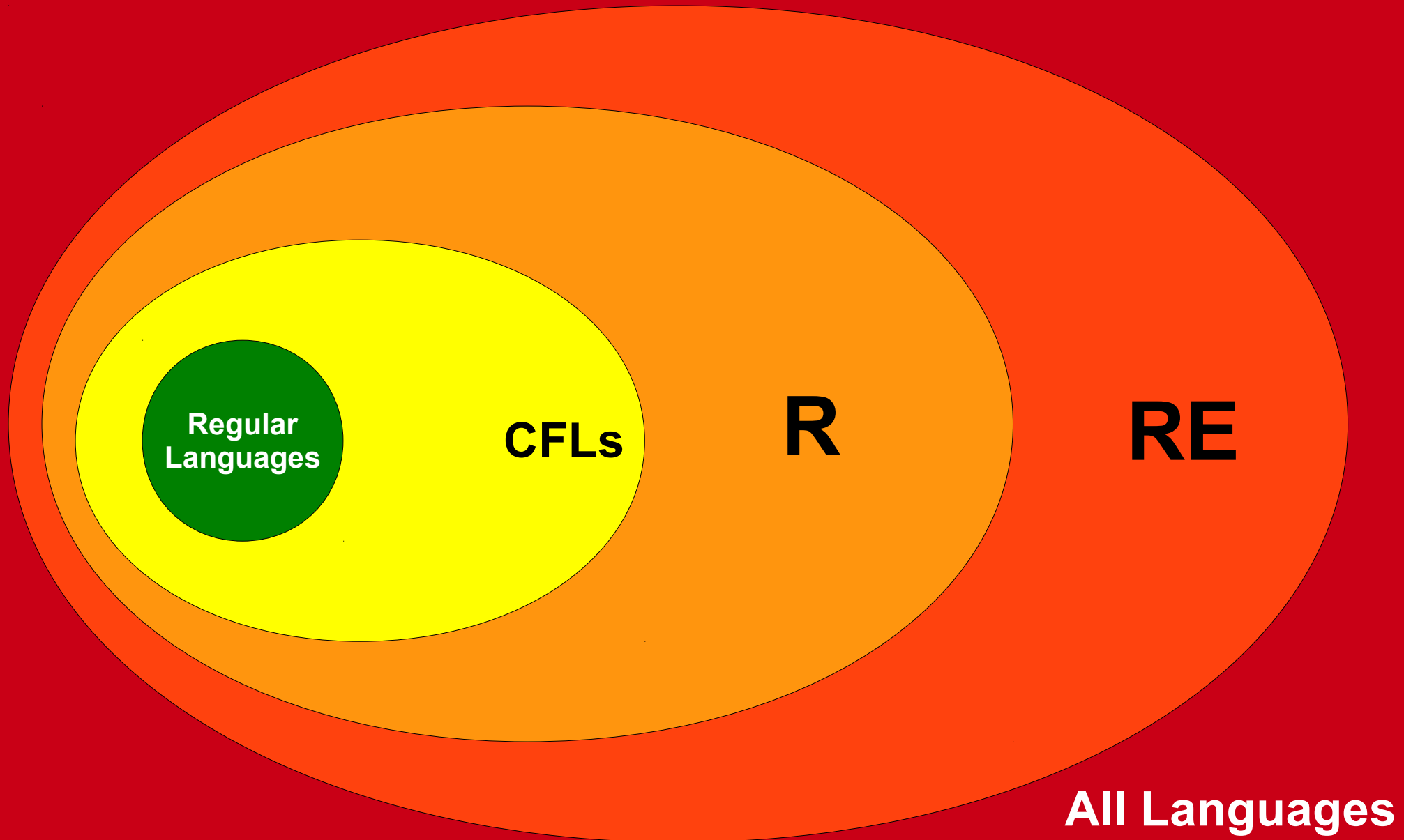
Is $\mathbf{R} = \mathbf{RE}$?

- 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?



Encodings

Computing over Objects

- Turing machines always compute over strings.
- We have seen examples of automata that can *essentially* compute over other objects:
 - Walking your Dog: Compute over paths.
 - Composite numbers: Compute over numbers.
- Up to this point, we have always said how we will encode objects:
 - e.g. $\{ 1^m + 1^n = 1^{mn} \mid m, n \in \mathbb{N} \}$

A Multitude of Encodings

- There can be many ways of encoding the same object.
- Example: the natural number 13 can be encoded
 - in unary: 11111111111111
 - in binary: 1101
 - in decimal: 13
 - in hexadecimal: D
 - in Roman numerals: XIII
 - ...
- **Claim:** Turing machines are sufficiently powerful to transform any one of these representations into any other of these representations.

An Abstract Idea of Encodings

- For simplicity, from this point forward we will make the following assumption:

For any finite, discrete object O , it is always possible to find some way of encoding O as a string.

- Think about how an actual computer works with mixed data – everything is 1's and 0's!
- When working with Turing machines, it really doesn't matter *how* we do the encoding. A TM can convert any reasonable encoding scheme into any other encoding scheme.

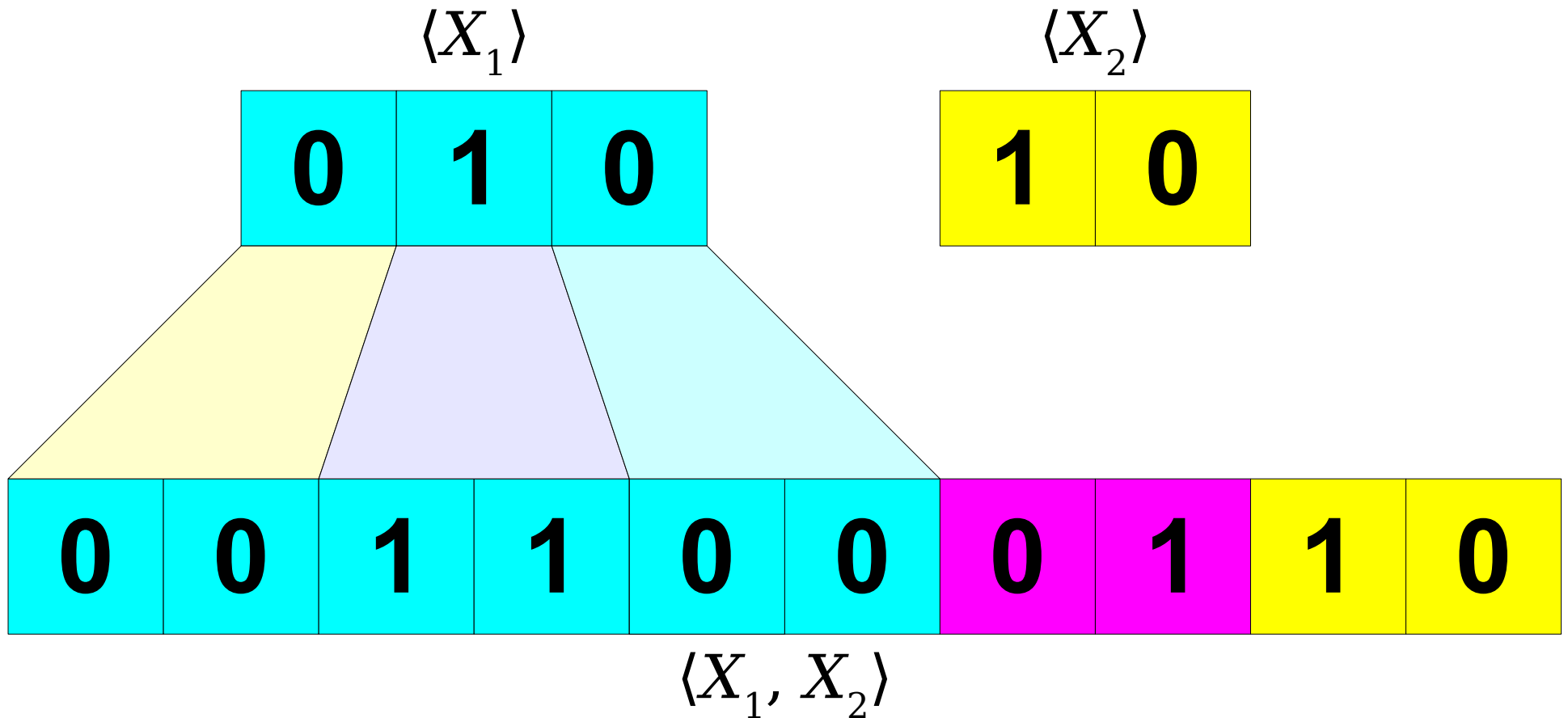
Notation for Encodings

- For any object O , we will denote a string encoding of O by writing O in angle brackets: O is encoded as $\langle O \rangle$.
- This makes it much easier to specify languages.
- Examples:
 - $\{ \langle R \rangle \mid R \text{ is a regular expression that matches } \varepsilon \}$
 - $\{ \langle n \rangle \mid n \in \mathbb{N} \text{ and the hailstone sequence terminates for } n. \}$
- The encoding scheme can make a difference when trying to determine whether a language is regular or context-free because of the relative weakness of DFAs and CFGs.

Encoding Multiple Objects

- Suppose that we want to provide an encoding of multiple objects.
 - Two natural numbers and their product.
 - A graph and a path in the graph.
 - “I just met you” and “this is crazy.”
- We can get encodings of each individual object.
- Can we make one string encoding all of these objects?

One Encoding Scheme



Encoding Multiple Objects

- Given several different objects O_1, \dots, O_n , we can represent the encoding of those n objects as $\langle \mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_n \rangle$.
- Examples:
 - $\{ \langle m, n, mn \rangle \mid m, n \in \mathbb{N} \}$
 - $\{ \langle G, w \rangle \mid G \text{ is a context-free grammar that generates } w \}$

Encoding Turing Machines

- ***Critically important fact***: Any Turing machine can be represented as a string.
- One way to do this: encode the TM as a transition table, then write it out one row at a time.
- Stronger claim: Any TM M can be represented as a string **in M 's own alphabet**.
- Analogy: program source code.
 - All data fed into a program is encoded using the binary alphabet $\Sigma = \{0, 1\}$.
 - A program's source code is itself represented in binary on disk.

We can now encode TMs as strings.

TMs accept strings as input.

What can we do with this knowledge?

Universal Machines

The Universal Turing Machine

- **Theorem:** There is a Turing machine U_{TM} called the **universal Turing machine** that, when run on $\langle M, w \rangle$, where M is a Turing machine and w is a string, simulates M running on w .
- Conceptually:

U_{TM} = “On input $\langle M, w \rangle$, where M is a TM and $w \in \Sigma^*$:

Set up the initial configuration of M running on w .

while (true) {

If M accepted w , then U_{TM} accepts $\langle M, w \rangle$.

If M rejected w , then U_{TM} rejects $\langle M, w \rangle$.

Otherwise, simulate one more step of M on w .

}”

The Universal Turing Machine

- **Theorem:** There is a Turing machine U_{TM} called the **universal Turing machine** that, when run on $\langle M, w \rangle$, where M is a Turing machine and w is a string, simulates M running on w .
- The observable behavior of U_{TM} is the following:
 - If M accepts w , then U_{TM} accepts $\langle M, w \rangle$.
 - If M rejects w , then U_{TM} rejects $\langle M, w \rangle$.
 - If M loops on w , then U_{TM} loops on $\langle M, w \rangle$.

An Intuition for U_{TM}

- You can think of U_{TM} as a general-purpose, programmable computer.
- Rather than purchasing one TM for each language, just purchase U_{TM} and program in the “software” corresponding to the TM you actually want.
- U_{TM} is a powerful machine: it can perform any computation that could be performed by any feasible computing device!

The Language of U_{TM}

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

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

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

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

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

- For simplicity, define $A_{\text{TM}} = \mathcal{L}(U_{\text{TM}})$. This is an important language and we'll see it a lot this week.

Regular
Languages

CFLs



A_{TM}

RE

All Languages

Time-Out for Announcements!

Midterm Logistics

- Second midterm is this Thursday, November 13, from 7PM – 10PM.
- Rooms divvied up by last name:
 - **Abr – Sad:** Go to **Cemex Auditorium**.
 - **Sal – Zie:** Go to **Cubberly Auditorium**.
- Exam is closed-computer, closed-book, open one page of notes (double-sided, 8.5" × 11").
- Cumulative exam, focus is PS4 – PS6.
- Alternate exams: if you've requested an alternate exam time, you should have heard back from Maesen with location information. Contact us ASAP if that isn't the case.

Practice Midterm Exam

- We're holding a practice midterm exam tonight in Annenberg Auditorium from 7PM – 10PM.
- Excellent way to get practice for the exam; anecdotally, this really, really seems to help.
- SCPD students: practice exam will be posted online later tonight.

Extra Practice Problems

- There are now three sets of extra practice problems up on the course website.
 - Solutions to EPP1 available in the filing cabinet.
 - Solutions to EPP2 will be available in the filing cabinet later tonight (and at the practice exam.)
 - Solutions to EPP3 will be released on Wednesday at the start of class.
- Have questions? Feel free to stop by office hours. We can take questions on these problems!
- Hope this helps!

Problem Set 7

- Problem Set 7 goes out today and is due next Monday at 2:15PM.
- Covers material up through and including today's discussion of L_D .
- Explore Turing machines and the limits of computational problem-solving!

Your Questions!

“Did you get my previous email?”

“In class, you mentioned that there is no physical analog to NFAs, and that quantum computers "do not really match" how an NFA works. I've read a bit about QFAs - could you give us a broad idea of how they can solve problems that we otherwise can't?”

“How does quantum computing relate to Turing Machines and DFAs? Do the same bounds apply?”

“I feel like I am spending the majority of my time and thought on CS related classes and projects. While they are super cool, this also means that I am exploring less. Do you have any advice?”

“Could you share an example of a nonregular finite language (if such a thing exists...)?”

“In proofs, why do you use
'we' instead of 'I'?”

Back to CS103!

The Story So Far

- We can now encode arbitrary objects, *including Turing machines*, as strings.
- Turing machines are capable of running other Turing machines specified through TM encodings.
- This has some deep consequences for what TMs can ever hope to accomplish...

Timeline of CS103

- **Lecture 00:** Unsolvable problems exist.
- **Lecture 11:** Proof by diagonalization.
- **Lecture 20:** TMs formalize computation.
- **Lecture 21:** TMs can be encoded as strings.

We are finally ready to start answering the following question:

What problems cannot be solved by a computer?

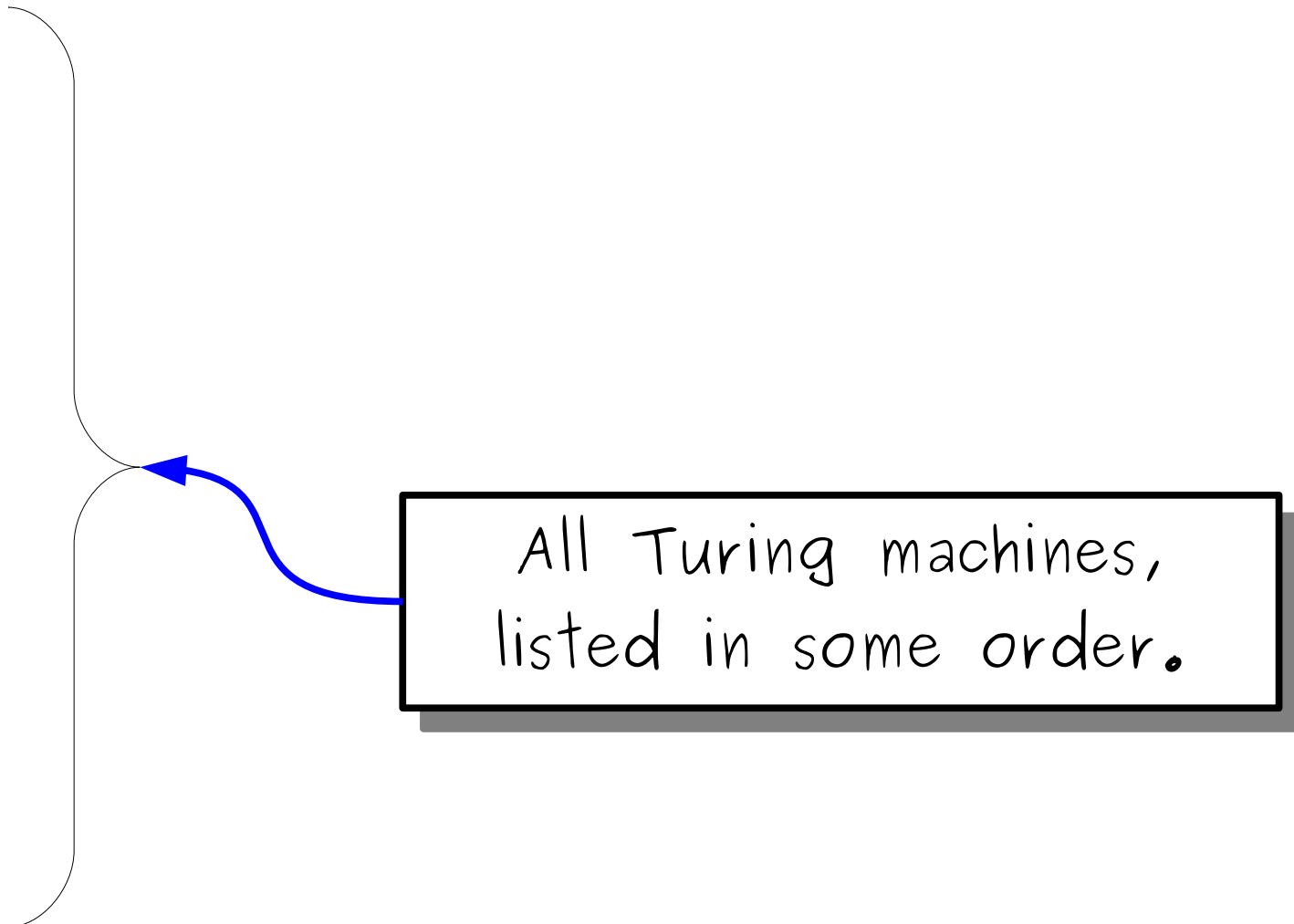
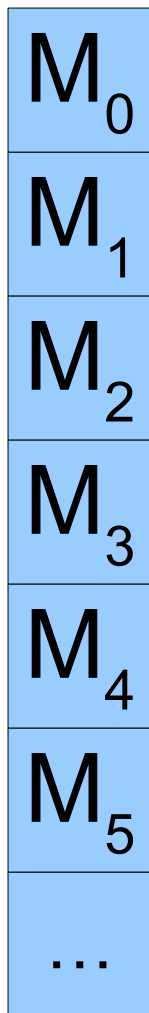
Languages, TMs, and TM Encodings

- Recall: The language of a TM M is the set

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

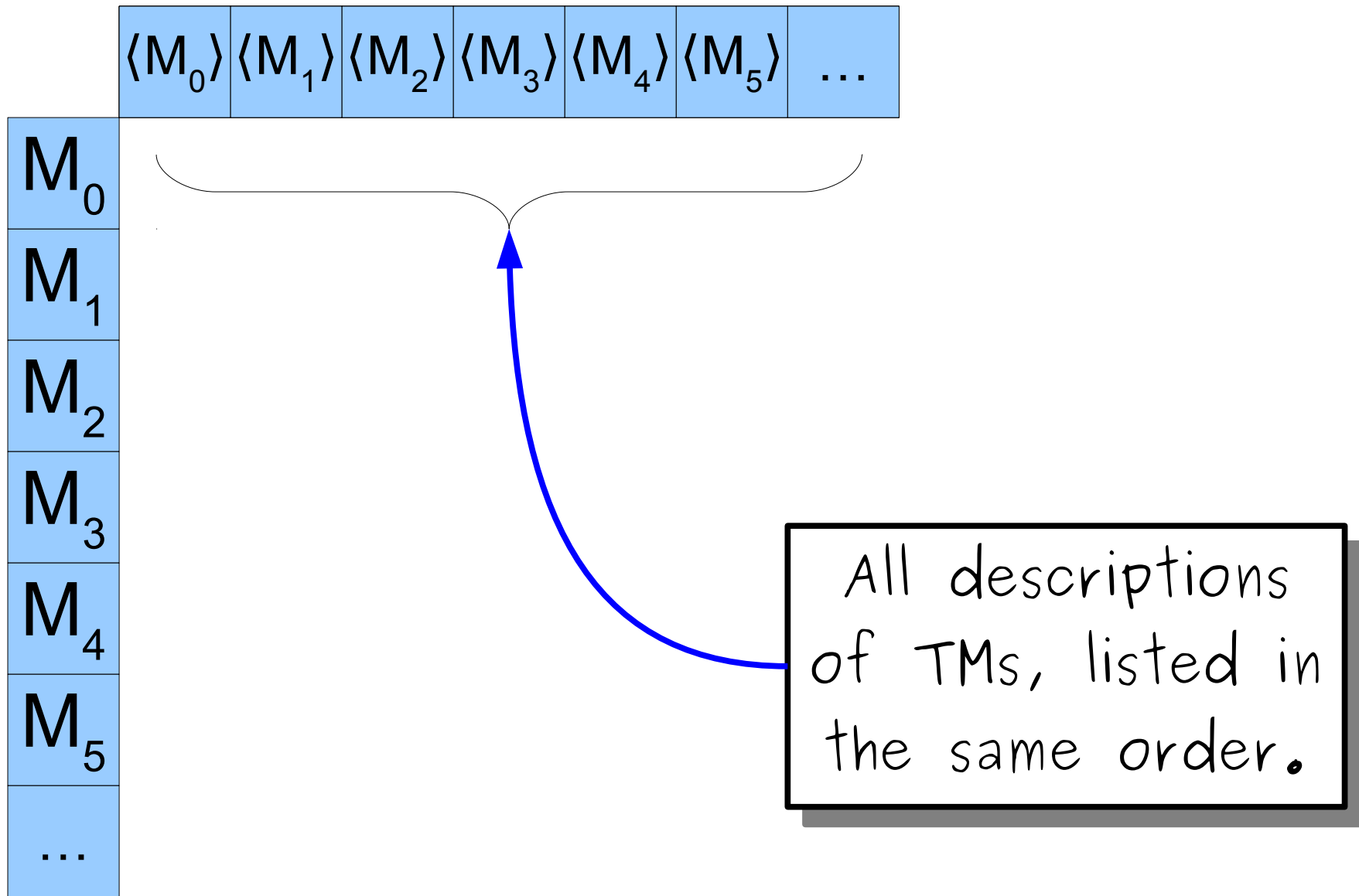
- Some of the strings in this set might be descriptions of TMs.
- What happens if we just focus on the set of strings that are legal TM descriptions?

M_0
M_1
M_2
M_3
M_4
M_5
...



$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----

M_0
M_1
M_2
M_3
M_4
M_5
...



	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1							
M_2							
M_3							
M_4							
M_5							
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2							
M_3							
M_4							
M_5							
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3							
M_4							
M_5							
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4							
M_5							
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5							
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

Acc	Acc	Acc	No	Acc	No	...
-----	-----	-----	----	-----	----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

Flip all "accept"
to "no" and
vice-versa

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

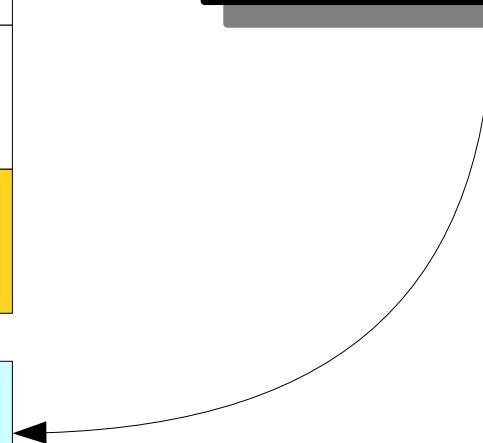
	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

What TM has this behavior?



	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

No TM has
this behavior!

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

“The language of all TMs that do not accept their own description.”

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

$\{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \}$

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$...
M_0	Acc	No	No	Acc	Acc	No	...
M_1	Acc	Acc	Acc	Acc	Acc	Acc	...
M_2	Acc	Acc	Acc	Acc	Acc	Acc	...
M_3	No	Acc	Acc	No	Acc	Acc	...
M_4	Acc	No	Acc	No	Acc	No	...
M_5	No	No	Acc	Acc	No	No	...
...

$\{ \langle M \rangle \mid M \text{ is a TM} \\ \text{and } \langle M \rangle \notin \mathcal{L}(M) \}$

No	No	No	Acc	No	Acc	...
----	----	----	-----	----	-----	-----

Diagonalization Revisited

- The ***diagonalization language***, which we denote L_D , is defined as

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

- That is, L_D is the set of descriptions of Turing machines that do not accept themselves.
- This is a hugely important language and we'll see it a lot this week.

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

Theorem: $L_D \notin \mathbf{RE}$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathcal{L}(R) \text{ iff } \langle R \rangle \in \mathcal{L}(R) \quad (3)$$

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathcal{L}(R) \text{ iff } \langle R \rangle \in \mathcal{L}(R) \quad (3)$$

This is clearly impossible.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathcal{L}(R) \text{ iff } \langle R \rangle \in \mathcal{L}(R) \quad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathcal{L}(R) \text{ iff } \langle R \rangle \in \mathcal{L}(R) \quad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong. Thus $L_D \notin \mathbf{RE}$.

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

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathcal{L}(R) = L_D$.

Since $\mathcal{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (1)$$

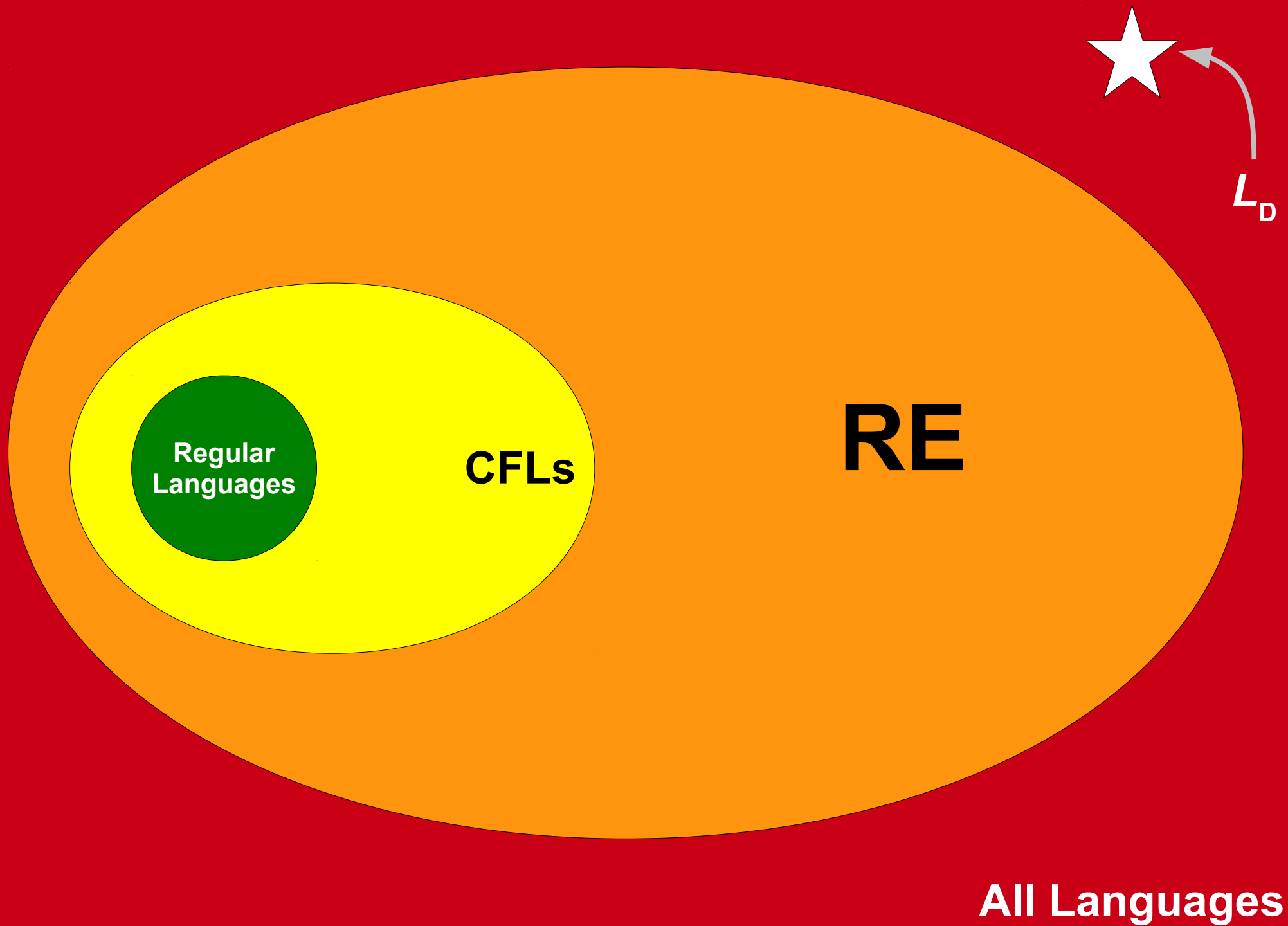
From the definition of L_D , we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathcal{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathcal{L}(M) \text{ iff } \langle M \rangle \in \mathcal{L}(R) \quad (2)$$

Statement (2) holds for any TM M , so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathcal{L}(R) \text{ iff } \langle R \rangle \in \mathcal{L}(R) \quad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong. Thus $L_D \notin \mathbf{RE}$. ■



Why is $L_D \notin \mathbf{RE}$?

- Intuitively, any TM for L_D would have to be wrong on itself.
- If the TM accepts itself, then it doesn't belong to L_D , so it shouldn't accept itself.
- If the TM doesn't accept itself, then it belongs to L_D , so it should accept itself.
- This is *indirect self-reference*: the language L_D isn't directly self-referential, but it still causes problems due to self-reference.