Decidability and Undecidability
Major Ideas from Last Time

- Every TM can be converted into a string representation of itself.
  - The **encoding** of $M$ is denoted $\langle M \rangle$.
- The **universal Turing machine** $U(TM)$ accepts an encoding $\langle M, w \rangle$ of a TM $M$ and string $w$, then simulates the execution of $M$ on $w$.
- The language of $U(TM)$ is the language $A(TM)$:
  \[
  A(TM) = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w. \}
  \]
- Equivalently:
  \[
  A(TM) = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in \mathcal{L}(M) \}
  \]
Major Ideas from Last Time

- The universal Turing machine $U_{\text{TM}}$ can be used as a subroutine in other Turing machines.

\[ H = \text{“On input } (M)\text{, where } M \text{ is a Turing machine:} \]
- Run $M$ on $\varepsilon$.
- If $M$ accepts $\varepsilon$, then $H$ accepts $(M)$.
- If $M$ rejects $\varepsilon$, then $H$ rejects $(M)$.

\[ H = \text{“On input } (M)\text{, where } M \text{ is a Turing machine:} \]
- Nondeterministically guess a string $w$.
- Run $M$ on $w$.
- If $M$ accepts $w$, then $H$ accepts $(M)$.
- If $M$ rejects $w$, then $H$ rejects $(M)$.
Major Ideas from Last Time

- 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.

- **Theorem:** $L_D \notin \text{RE}$
Regular Languages (CFLs, DCFLs)

RE

All Languages

$L_D$
Outline for Today

- **More non-RE Languages**
  - We now know $L_D \notin \text{RE}$. Can we use this to find other non-RE languages?

- **Decidability and Class R**
  - How do we formalize the idea of an algorithm?

- **Undecidable Problems**
  - What problems admit no algorithmic solution?
Additional Unsolvable Problems
Finding Unsolvable Problems

- We can use the fact that $L_D \not\in \text{RE}$ to show that other languages are also not \text{RE}.

- General proof approach: to show that some language $L$ is not \text{RE}, we will do the following:
  - Assume for the sake of contradiction that $L \in \text{RE}$, meaning that there is some TM $M$ for it.
  - Show that we can build a TM that uses $M$ as a subroutine in order to recognize $L_D$.
  - Reach a contradiction, since no TM recognizes $L_D$.
  - Conclude, therefore, that $L \not\in \text{RE}$.
The Complement of $A_{TM}$

- Recall: the language $A_{TM}$ is the language of the universal Turing machine $U_{TM}$:

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

- The complement of $A_{TM}$ (denoted $\overline{A}_{TM}$) is the language of all strings not contained in $A_{TM}$.

- Questions:
  - What language is this?
  - Is this language RE?
\( A_{TM} \) and \( \overline{A}_{TM} \)

- The language \( A_{TM} \) is defined as
  \[
  \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \} 
  \]

- Equivalently:
  \[
  \{ x \mid x = \langle M, w \rangle \text{ for some TM } M \\
  \text{and string } w, \text{ and } M \text{ accepts } w \} 
  \]

- Thus \( \overline{A}_{TM} \) is
  \[
  \{ x \mid x \neq \langle M, w \rangle \text{ for any TM } M \text{ and string } w, \\
  \text{or } M \text{ is a TM that does not accept } w \} 
  \]
Cheating With Math

• As a mathematical simplification, we will assume the following:

  **Every string can be decoded into any collection of objects.**

• Every string is an encoding of some TM $M$.

• Every string is an encoding of some TM $M$ and string $w$.

• Can do this as follows:
  • If the string is a legal encoding, go with that encoding.
  • Otherwise, pretend the string decodes to some predetermined group of objects.
Cheating With Math

- Example: Every string will be a valid C++ program.
- If it's already a C++ program, just compile it.
- Otherwise, pretend it's this program:

```cpp
int main() {
    return 0;
}
```
$A_{TM}$ and $\overline{A}_{TM}$

- The language $A_{TM}$ is defined as
  \[
  \{\langle M, w \rangle \mid M \text{ is a TM that accepts } w\}\]
- Thus $\overline{A}_{TM}$ is the language
  \[
  \{\langle M, w \rangle \mid M \text{ is a TM that doesn't accept } w\}\]
$\overline{A_{TM}} \notin \text{RE}$

- Although the language $A_{TM} \in \text{RE}$ (since it's the language of $U_{TM}$), its complement $\overline{A_{TM}} \notin \text{RE}$.

- We will prove this as follows:
  - Assume, for contradiction, that $\overline{A_{TM}} \in \text{RE}$.
  - This means there is a TM $R$ for $\overline{A_{TM}}$.
  - Using $R$ as a subroutine, we will build a TM $H$ that will recognize $L_D$.
  - This is impossible, since $L_D \notin \text{RE}$.
  - Conclude, therefore, that $\overline{A_{TM}} \notin \text{RE}$. 
Comparing $L_D$ and $\overline{A_{TM}}$

- The languages $L_D$ and $\overline{A_{TM}}$ are closely related:
  - $L_D$: Does $M$ not accept $\langle M \rangle$?
  - $\overline{A_{TM}}$: Does $M$ not accept string $w$?
- Given this connection, we will show how to turn a hypothetical recognizer for $\overline{A_{TM}}$ into a hypothetical recognizer for $L_D$. 
$H = \text{"On input } \langle M \rangle \text{:}
\begin{itemize}
  \item Construct the string $\langle M, \langle M \rangle \rangle$.
  \item Run $R$ on $\langle M, \langle M \rangle \rangle$.
  \item If $R$ accepts $\langle M, \langle M \rangle \rangle$, then $H$ accepts $\langle M \rangle$.
  \item If $R$ rejects $\langle M, \langle M \rangle \rangle$, then $H$ rejects $\langle M \rangle$.
\end{itemize}$

What happens if...

$M$ does not accept $\langle M \rangle$?

- **Accept**

$M$ accepts $\langle M \rangle$?

- **Reject or Loop**

$H$ is a TM for $L_D$!
Theorem: $\overline{A}_{\text{TM}} \notin \text{RE}$.

Proof: By contradiction; assume that $\overline{A}_{\text{TM}} \in \text{RE}$. Then there must be a recognizer for $\overline{A}_{\text{TM}}$; call it $R$.

Consider the TM $H$ defined below:

$$H = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$
$$\text{Construct the string } \langle M, \langle M \rangle \rangle.$$  
$$\text{Run } R \text{ on } \langle M, \langle M \rangle \rangle.$$  
$$\text{If } R \text{ accepts } \langle M, \langle M \rangle \rangle, \text{ } H \text{ accepts } \langle M \rangle.$$  
$$\text{If } R \text{ rejects } \langle M, \langle M \rangle \rangle, \text{ } H \text{ rejects } \langle M \rangle."$$

We claim that $\mathcal{L}(H) = L_D$. We will prove this by showing that $\langle M \rangle \in L_D$ iff $H$ accepts $\langle M \rangle$.

By construction we have that $H$ accepts $\langle M \rangle$ iff $R$ accepts $\langle M, \langle M \rangle \rangle$. Since $R$ is a recognizer for $\overline{A}_{\text{TM}}$, $R$ accepts $\langle M, \langle M \rangle \rangle$ iff $M$ does not accept $\langle M \rangle$. Finally, note that $M$ does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Therefore, we have $H$ accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathcal{L}(H) = L_D$. But this is impossible, since $L_D \notin \text{RE}$.

We have reached a contradiction, so our assumption must have been incorrect. Thus $\overline{A}_{\text{TM}} \notin \text{RE}$, as required. ■
Regular Languages

DCFLs

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 $\mathcal{L}(M)$, is the set of all strings that $M$ accepts:
  \[ \mathcal{L}(M) = \{ w \in \Sigma^* | M \text{ accepts } w \} \]
- 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: $\text{RE}$ is the set of all recognizable languages.

\[ L \in \text{RE} \iff L \text{ is recognizable} \]
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, 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 $L$. 
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.

\[
\begin{align*}
\text{Accept} & \quad \text{halts (always)} \\
\text{Reject} & \\
\end{align*}
\]
Decidable Languages

- A language $L$ is called **decidable** iff 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 set $R$ is the set of all decidable languages.
  \[ L \in R \text{ iff } L \text{ is decidable} \]
**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.

- $\{0^n1^n \mid n \in \mathbb{N}\}$ is in $R$.
  - The TM we built last Wednesday is a decider.

- Multiplication is in $R$.
  - Can check if $m \times n = p$ by repeatedly subtracting out copies of $n$. If the equation balances, accept; if not, reject.
CFLs and \( \mathbb{R} \)

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

- If a language is in $\mathbb{R}$, 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 $\mathbb{R}$.
  - By the Church-Turing thesis, any effective model of computation is equivalent in power to a Turing machine.
  - Thus if there is any algorithm for deciding membership in the language, there must be a decider for it.
  - Thus the language is in $\mathbb{R}$.
- A language is in $\mathbb{R}$ iff there is an algorithm for deciding membership in that language.
\[ R \neq RE \]

- Every decider is a Turing machine, but not every Turing machine is a decider.
- Thus \( R \subseteq RE \).
- Hugely important theoretical question:

\[ \text{Is } R = RE? \]

- 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?

- Regular Languages
- DCFLs
- CFLs
- R
- RE
- All Languages
An Important Observation
**R is Closed Under Complementation**

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

**Decider for $L$**

$M' = \text{"On input } w:\n$  
* 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 $L \in R$. We will prove that $\overline{L} \in R$ by constructing a decider $M'$ such that $\mathcal{L}(M') = \overline{L}$.

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

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

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

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 $\mathcal{L}(M') = \overline{L}$, 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 $w \notin \mathcal{L}(M)$. Thus $M'$ accepts $w$ iff $w \in \Sigma^*$ and $w \notin \mathcal{L}(M)$, so $M'$ accepts $w$ iff $w \in \overline{L}$. Therefore, $\mathcal{L}(M') = \overline{L}$.

Since $M'$ is a decider with $\mathcal{L}(M') = \overline{L}$, we have $\overline{L} \in R$, as required. ■
We can now resolve the question of $R \neq RE$.

If $R = RE$, we need to show that if there is a recognizer for any $RE$ language $L$, there has to be a decider for $L$.

If $R \neq RE$, 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, \( A_{TM} \in RE \).
- Is \( A_{TM} \in R \)?
**Theorem:** $A_{TM} \notin R$.

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

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

- Regular Languages
- DCFLs
- CFLs
- $R$
- All Languages
- $A_{TM}$
- $A_{TM}^{-}$
- $L_D$
- $RE$

Diagram showing the relationships between different classes of languages.
What this Means

- 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't do either. For example, you cannot always recognize that a TM will not accept a string.
What this Means

- **Major result:** $\mathbb{R} \neq \text{RE}$.
- There are some problems where we can only give a “yes” answer when the answer is “yes” and cannot necessarily give a yes-or-no answer.
- Solving a problem is *fundamentally* harder than recognizing a correct answer.
Another Undecidable Problem
$L_D$ Revisited

- The diagonalization language $L_D$ is the language

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

- As we saw before, $L_D \notin \text{RE}$.

- But what about $\overline{L_D}$?
$L_D$

- The language $L_D$ is the language
  $$L_D = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathcal{L}(M)\}$$
- Therefore, $\overline{L}_D$ is the language
  $$L_D = \{\langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \in \mathcal{L}(M)\}$$
- Two questions:
  - What is this language?
  - Is this language RE?
<table>
<thead>
<tr>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle M_5 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>Acc</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Acc</td>
<td>Acc</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
</tbody>
</table>

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

This language is $\mathcal{I}_D$. 
\[ \overline{L_D} \in \text{RE} \]

- Here's an TM for \( \overline{L_D} \):

  \[
  R = \text{"On input } \langle M \rangle:\text{"}
  \]
  \[
  \text{Run } M \text{ on } \langle M \rangle.
  \]
  \[
  \text{If } M \text{ accepts } \langle M \rangle, \text{ accept.}
  \]
  \[
  \text{If } M \text{ rejects } \langle M \rangle, \text{ reject.}"
  \]

- Then \( R \) accepts \( \langle M \rangle \) iff \( \langle M \rangle \in \mathcal{L}(M) \) iff \( \langle M \rangle \in \overline{L_D} \), so \( \mathcal{L}(R) = \overline{L_D} \).
Is $\overline{L}_D$ Decidable?

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

- Regular Languages (CFLs)
- DCFLs
- ALL Languages

- $R$
- $A_{TM}$
- $L_{D}$

- $\overline{A}_{TM}$
- $\overline{L}_{D}$

- $RE$

All Languages
Finding Unsolvable Problems

Diagram:

- $L_D$: Not RE
- $A_{TM}$: Not RE
- $A_{TM}$: Not R
- $L_D$: Not R

Relationships:

- $L_D \rightarrow \overline{A_{TM}} \rightarrow A_{TM}$
- $\overline{L_D} \rightarrow \overline{A_{TM}} \rightarrow A_{TM}$