Lecture 15:

- Undecidable and Unrecognizable Languages
- Reductions

Reading:

Sipser Ch 4.2, 5.1

Mark Bun
March 23, 2020

- Ask questions about HW after class today
- HW 6 released, due Mon. March 30
- Midterm 2: 24-hr take-home exam, released 2:30 4/1, due (on Gradescope) @ 2:30 4/2
How can we compare sizes of infinite sets?

**Definition:** Two sets have the same size if there is a correspondence (bijection) between them.

A set is **countable** if

- it is a finite set, or
- it has the same size as \( \mathbb{N} \), the set of natural numbers.
A general theorem about set sizes

**Theorem:** Let $X$ be a set. Then the power set $P(X)$ does **not** have the same size as $X$.

**Proof:** Assume for the sake of contradiction that there is a correspondence $f : X \rightarrow P(X)$

**Goal:** Use diagonalization to construct a set $S \in P(X)$ that cannot be the output $f(x)$ for any $x \in X$
Undecidable Languages
Problems in language theory

<table>
<thead>
<tr>
<th>Acceptance problem</th>
<th>$A_{DFA}$ decidable</th>
<th>$A_{CFG}$ decidable</th>
<th>$A_{TM}$ ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emptiness testing</td>
<td>$E_{DFA}$ decidable</td>
<td>$E_{CFG}$ decidable</td>
<td>$E_{TM}$ ?</td>
</tr>
<tr>
<td>Equality</td>
<td>$EQ_{DFA}$ decidable</td>
<td>$EQ_{CFG}$ ?</td>
<td>$EQ_{TM}$ ?</td>
</tr>
</tbody>
</table>
Undecidability

These natural computational questions about computational models are undecidable

I.e., computers cannot solve these problems no matter how much time they are given
An existential proof

Theorem: There exists an undecidable language over \{0, 1\}

Proof: "Counting argument"

A simplifying assumption: Every string in \{0, 1\}^* is the encoding \langle M \rangle of some Turing machine \( M \)

Set of all Turing machines: \( X = \{0, 1\}^* \)

Set of all languages over \{0, 1\}: all subsets of \( \{0, 1\}^* \)

\(|P(\{0, 1\}^*)| = 2^{2^{2^{\cdots}}} = P(X)\)

There are more languages than there are TM deciders!
An existential proof

**Theorem:** There exists an *unrecognizable* language over \( \{0, 1\} \)

**Proof:**

A simplifying assumption: Every string in \( \{0, 1\}^* \) is the encoding \( \langle M \rangle \) of some Turing machine \( M \)

Set of all Turing machines: \( X = \{0, 1\}^* \)

Set of all languages over \( \{0, 1\} \): all subsets of \( \{0, 1\}^* \)

\[ = P(X) \]

There are more languages than there are TM *recognizers!*
An explicit undecidable language

<table>
<thead>
<tr>
<th>TM $M$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An explicit undecidable language

<table>
<thead>
<tr>
<th>TM M</th>
<th>M(⟨M₁⟩)?</th>
<th>M(⟨M₂⟩)?</th>
<th>M(⟨M₃⟩)?</th>
<th>M(⟨M₄⟩)?</th>
<th>D(⟨D⟩)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁</td>
<td>Y N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>...</td>
</tr>
<tr>
<td>M₂</td>
<td>N</td>
<td>N Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>M₃</td>
<td>Y</td>
<td>Y</td>
<td>Y N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>M₄</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N Y</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y N</td>
</tr>
</tbody>
</table>

Cell in row i, column j: Y if Mᵢ accepts an input ⟨Mⱼ⟩; N if Mᵢ rejects or loops forever on input ⟨Mⱼ⟩.

\[ L = \{ \langle M \rangle \mid M \text{ is a TM that does not accept on input } \langle M \rangle \} \]

Suppose D decides L \( \neq \)
An explicit undecidable language

**Theorem:** $L = \{\langle M \rangle \mid M$ is a TM that does not accept on input $\langle M \rangle \}$ is undecidable

**Proof:** Suppose for contradiction, that $D$ decides $L$

1. $\langle M \rangle \in L \implies D(\langle M \rangle)$ accepts
2. $D(\langle O \rangle)$ is not well-defined

Claim: $D(\langle O \rangle)$ does not accept $\implies \langle O \rangle \notin L$ (3) $\implies D(\langle O \rangle)$ does not accept (1)

"Self-acceptance problem"

**Corollary:** $SA_{TM} = \bar{L} = \{\langle M \rangle \mid M$ is a TM that accepts on input $\langle M \rangle \}$ is undecidable
A more useful undecidable language

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

**Theorem:** \( A_{TM} \) is undecidable

**But first:** \( A_{TM} \) *is* Turing-recognizable

The following “universal TM” \( U \) recognizes \( A_{TM} \)

On input \( \langle M, w \rangle \):
1. Simulate running \( M \) on input \( w \)
2. If \( M \) accepts, **accept**. If \( M \) rejects, **reject**.

*Not a decider because \( M \) may loop forever on \( w \)*
More on the Universal TM

"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $U$ is supplied with a tape on the beginning of which is written the S.D ["standard description"] of some computing machine $M$, then $U$ will compute the same sequence as $M$.”

- Turing, “On Computable Numbers...” 1936

• Foreshadowed general-purpose programmable computers
• No need for specialized hardware: Virtual machines as software

Harvard architecture: Separate instruction and data pathways
von Neumann architecture: Programs can be treated as data
A more useful undecidable language

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

Theorem: \( A_{TM} \) is undecidable

Proof: Assume for the sake of contradiction that TM \( H \) decides \( A_{TM} \):

\[
H(\langle M, w \rangle) = \begin{cases} 
\text{accept} & \text{if } M \text{ accepts } w \\
\text{reject} & \text{if } M \text{ does not accept } w 
\end{cases}
\]

Idea: Show that \( H \) can be used to decide the (undecidable) language \( SA_{TM} \) -- a contradiction.
A more useful undecidable language

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

Suppose \( H \) decides \( A_{TM} \)

Consider the following TM \( D \).

On input \( \langle M \rangle \) where \( M \) is a TM:
1. Run \( H \) on input \( \langle M, \langle M \rangle \rangle \)
2. If \( H \) accepts, accept. If \( H \) rejects, reject.

Claim: \( D \) decides
\[ SA_{TM} = \{ \langle M \rangle \mid M \text{ is a TM that accepts on input } \langle M \rangle \} \]
...but this language is undecidable
Unrecognizable Languages

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

**Proof:**

$\Rightarrow$ 

1. $L$ decidable $\implies L$ recognizable
2. $\overline{L}$ decidable (closure) $\implies \overline{L}$ recognizable

$\Leftarrow$

1. $L, \overline{L}$ recognizable, recognized by TMs $M_1, M_2$ respectively

Construct decider for $L$.

On input $w$:

1. Run $M_1$ and $M_2$ in parallel on $w$
2. If $M_1$ accepts, accept
   If $M_2$ accepts, reject
Classes of Languages

- Recognizable
- Decidable
- Context Free
- Regular

\( A_{TM} \)

\( \Sigma^* \)

\( L(0^*) \)

\( \Sigma^* \)
Reductions
A more useful undecidable language

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

**Theorem:** \( A_{TM} \) is undecidable

**Proof:** Assume for the sake of contradiction that TM \( H \) decides \( A_{TM} \):

\[
H(\langle M, w \rangle) = \begin{cases} 
\text{accept} & \text{if } M \text{ accepts } w \\
\text{reject} & \text{if } M \text{ does not accept } w
\end{cases}
\]

**Idea:** Show that \( H \) can be used to decide the (undecidable) language \( SA_{TM} \) -- a contradiction.

“A reduction from \( SA_{TM} \) to \( A_{TM} \)”
Scientists vs. Engineers

A computer scientist and an engineer are stranded on a desert island. They find two palm trees with one coconut on each. The engineer climbs a tree, picks a coconut and eats.

The computer scientist climbs the second tree, picks a coconut, climbs down, climbs up the first tree and places it there, declaring success.

“Now we’ve reduced the problem to one we’ve already solved.”
Reductions

A reduction from problem $A$ to problem $B$ is an algorithm for problem $A$ which uses an algorithm for problem $B$ as a subroutine.

If such a reduction exists, we say “$A$ reduces to $B$”
Two uses of reductions

Positive uses: If $A$ reduces to $B$ and $B$ is decidable, then $A$ is also decidable

$E_{DFA} = \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$

Theorem: $E_{DFA}$ is decidable

Proof: The following TM decides $E_{DFA}$

On input $\langle D_1, D_2 \rangle$, where $\langle D_1, D_2 \rangle$ are DFAs:

1. Construct a DFA $D$ that recognizes the symmetric difference $L(D_1) \triangle L(D_2)$
2. Run the decider for $E_{DFA}$ on $\langle D \rangle$ and return its output
Two uses of reductions

Negative uses: If \( A \) reduces to \( B \) and \( A \) is undecidable, then \( B \) is also undecidable

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

Suppose \( H \) decides \( A_{TM} \)

Consider the following TM \( D \).
On input \( \langle M \rangle \) where \( M \) is a TM:
1. Run \( H \) on input \( \langle M, \langle M \rangle \rangle \)
2. If \( H \) accepts, accept. If \( H \) rejects, reject.

Claim: \( D \) decides

\[ A_{TM} = \{ \langle M \rangle \mid M \text{ is a TM that accepts on input } \langle M \rangle \} \]
Two uses of reductions

Negative uses: If $A$ reduces to $B$ and $A$ is undecidable, then $B$ is also undecidable

Proof template:
1. Suppose to the contrary that $B$ is decidable
2. Using $B$ as a subroutine, construct an algorithm deciding $A$
3. But $A$ is undecidable. Contradiction! (conclude is not decidable)
Halting Problem

\[ \text{HALT}_\text{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \} \]

**Theorem:** \( \text{HALT}_\text{TM} \) is undecidable

**Proof:** Suppose for contradiction that there exists a decider \( H \) for \( \text{HALT}_\text{TM} \). We construct a decider for \( A_{\text{TM}} \) as follows:

On input \( \langle M, w \rangle \):

1. Run \( H \) on input \( \langle M, w \rangle \)
2. If \( H \) rejects, reject
3. If \( H \) accepts, simulate \( M \) on \( w \)
4. If \( M \) accepts, accept. Otherwise, reject

\[ \because \ \text{because } A_{\text{TM}} \text{ undecidable} \]

This is a reduction from \( A_{\text{TM}} \) to \( \text{HALT}_{\text{TM}} \)
Empty language testing for TMs

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

**Theorem:** \( E_{TM} \) is undecidable

**Proof:** Suppose for contradiction that there exists a decider \( R \) for \( E_{TM} \). We construct a decider for \( A_{TM} \) as follows:

On input \( \langle M, w \rangle \):

1. Run \( R \) on input \( \langle M, w \rangle \).

This is a reduction from \( A_{TM} \) to \( E_{TM} \).
Empty language testing for TMs

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

**Theorem:** \( E_{TM} \) is undecidable

**Proof:** Suppose for contradiction that there exists a decider \( R \) for \( E_{TM} \). We construct a decider for \( A_{TM} \) as follows:

On input \( \langle M, w \rangle \):

1. **Construct a TM \( M' \) as follows:**
   - \( M' \): On input \( x \):
     1) Ignore \( x \)
     2) Run \( M \) on \( w \)
     3) If \( M \) accepts, accept. If \( M \) rejects, reject

2. Run \( R \) on input \( \langle M' \rangle \)

3. If \( R \) rejects, accept. Otherwise, reject

This TM decides \( A_{TM} \).

This is a reduction from \( A_{TM} \) to \( E_{TM} \).