Undecidability

Lecture 21
Today

Undecidable Problems

Proving undecidability

Using reductions to prove more undecidability
Language of Universal TM

Language recognized by $U$:

$$L(U) = \{ (z,w) | U \text{ accepts } (z,w) \}$$

$$= \{ (z,w) | M_z \text{ accepts } w \}$$

We will call $L(U) = \text{ACCEPT}$

Today:

$\text{ACCEPT}$ is undecidable!

pair of binary strings encoded as a binary string

$M_z$ is the TM encoded by the string $M_z$

No matter what encoding schemes are used
Cantor’s Diagonal Slash

Is the set of all infinitely long binary strings countable?

Suppose it was: consider *enumerating* them in a table.

Consider the string corresponding to the “flipped diagonal”

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
<th>S₅</th>
<th>S₆</th>
<th>S₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁ =</td>
<td>1 0 0 1 0 0 0 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₂ =</td>
<td>0 0 1 0 1 0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₃ =</td>
<td>1 1 1 1 1 1 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₄ =</td>
<td>1 1 0 1 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₅ =</td>
<td>1 1 0 0 0 0 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₆ =</td>
<td>0 0 0 0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₇ =</td>
<td>0 1 0 1 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It doesn’t appear in this table!
Undecidability

Table of languages recognized by TMs

\[ T(z, w) = 1 \text{ iff } M_z \text{ accepts } w \]

\[ D = \text{“diagonal language”} \]
\[ = \{ w \mid M_w \text{ accepts } w \} \]

\[ \overline{D} = \{ w \mid M_w \text{ doesn’t accept } w \} \]

\[ \overline{D} \] does not appear as a row in this table. Hence not recognizable!
Undecidability

Table of languages recognized by TMs

\[ T(z,w) = 1 \text{ iff } M_z \text{ accepts } w \]

If \textit{ACCEPT} decidable, can compute \( T(z,w) \) using a TM that halts on every input

Then \( \bar{D} \) would be decidable too:
On input \( w \), compute \( T(w,w) \) and accept iff it is 0

Hence \textit{ACCEPT} undecidable!
Reduction

We just saw how a “reduction” can show impossibility

1. Showed that if $\text{ACCEPT}$ is decidable, then $\overline{D}$ decidable (using a “reduction” from $\overline{D}$ to $\text{ACCEPT}$)

2. We already saw $\overline{D}$ not decidable

3. Hence $\text{ACCEPT}$ not decidable
Reduction

Reduction from $L_1$ to $L_2$ ($L_1 \leq L_2$):

Any instance of $L_1$ can be solved by solving an instance of $L_2$ (and there is an algorithm to change the $L_1$-instance to the $L_2$-instance)

The task of solving $L_1$ is reduced to the task of solving $L_2$

**Positive implication**: If we can solve $L_2$, then we can solve $L_1$

**Negative implication**: If we can’t solve $L_1$, then we can’t solve $L_2
We use a simple notion of reduction (for most part). Algorithm for solving $L_1$ should behave as follows:

A (mapping) reduction from $L_1$ to $L_2$:

a computable function $f$ s.t. $\forall w, w \in L_1 \iff f(w) \in L_2$
Reduction

A (mapping) reduction from $L_1$ to $L_2$: a computable function $f$ s.t. $\forall w, w \in L_1 \Leftrightarrow f(w) \in L_2$

Note: a reduction from $L_1$ to $L_2$ is also a reduction from $\overline{L}_1$ to $\overline{L}_2$

$L_1 \leq L_2 \Leftrightarrow \overline{L}_1 \leq \overline{L}_2$
Reduction

A (mapping) reduction from $L_1$ to $L_2$: a computable function $f$ s.t. $\forall w, w \in L_1 \iff f(w) \in L_2$

On input $w$, compute $f(w)$
Accept iff $f(w) \in L_2$

Positive implication:
If $L_1 \leq L_2$ then: can “solve” $L_2 \Rightarrow$ can “solve” $L_1$

$L_2$ decidable $\Rightarrow L_1$ decidable
$L_2$ recognizable $\Rightarrow L_1$ recognizable

Negative implication: If $L_1 \leq L_2$ then:
$L_1$ undecidable $\Rightarrow L_2$ undecidable
$L_1$ unrecognizable $\Rightarrow L_2$ unrecognizable
Halting Problem

\[ \text{HALT} = \{ (z,w) | M_z \text{ halts on input } w \} \]

Claim: \( \text{ACCEPT} \leq \text{HALT} \)

\[ f(z,w) = (z',w) \text{ where } M_{z'} \text{ behaves as follows:} \]

On input \( x \), run \( M_z \) on \( x \).

- If \( M_z \) halts rejecting \( x \), go into an infinite loop.
- If \( M_z \) halts accepting \( x \), halt (and say, accept).

\[ (z',w) \in \text{HALT} \iff (z,w) \in \text{ACCEPT} \]

\( \text{ACCEPT} \) undecidable \( \Rightarrow \) \( \text{HALT} \) undecidable
Map

\[ \text{R.E.} \]

\[ D \]

\[ \tilde{D} \]

\[ \text{ACCEPT} \]

\[ \text{HALT} \]

\[ R \]
Complement & Undecidability

\textit{ACCEPT} is undecidable, but is recognizable (why?)

\textit{ACCEPT}^c is undecidable too (why?)

Is \textit{ACCEPT}^c recognizable?

Claim: \textit{ACCEPT}^c is not recognizable

If not, \textit{ACCEPT} and \textit{ACCEPT}^c both recognizable, Then \textit{ACCEPT} would be decidable! (why?)
Map

R.E.  

\[ D \]

ACCEPT

HALT

\[ \tilde{D} \]

ACCEPT^c

HALT^c

R

ACCEPT
Empty Language Problem

\[ EMPTY = \{ z | L(M_z) = \emptyset \} \]

Claim: \( ACCEPT^C \leq EMPTY \)

\( f(z,w) = z' \) where \( M_{z'} \) behaves as follows:

On input \( x \), run \( M_z \) on \( w \).
If \( M_z \) halts rejecting \( w \), reject \( x \).
If \( M_z \) halts accepting \( w \), accept \( x \).

\[ z' \in EMPTY \iff (z,w) \notin ACCEPT \]

\( ACCEPT^C \) unrecognizable \( \Rightarrow \) \( EMPTY \) is unrecognizable
Dovetailing

Claim: $EMPTYC = \{ z \mid L(M_z) \neq \emptyset \}$ is recognizable

$EMPTYC = \{ z \mid \exists w \ M_z \text{ accepts } w \}$.

Given $z$, how to check if there is some $w$ that $M_z$ accepts?

Run $M_z$ on all $w$, and if it accepts any, accept (if not keep trying)

In “parallel”? Can’t run infinitely many executions in parallel!

Solution: increasingly more executions in parallel
Exploring the ID Graph

Sequential Simulation: Depth first

ID0(w0) → ID1(w0) → ID2(w0) → ID3(w0) → ID4(w0)

ID0(w1) → ID1(w1) → ID2(w1) → ID3(w1) → ID4(w1) → ID5(w1)

ID0(w2) → ID1(w2) → ID2(w2) → ID3(w2)

Never gets here!

Goes on forever
Exploring the ID Graph

Parallel Simulation: Breadth first

ID₀(w₀) → ID₁(w₀) → ID₂(w₀) → ID₃(w₀) → ID₄(w₀)

ID₀(w₁) → ID₁(w₁) → ID₂(w₁) → ID₃(w₁) → ID₄(w₁) → ID₅(w₁)

ID₀(w₂) → ID₁(w₂) → ID₂(w₂) → ID₃(w₂)

Never gets here!

Goes on forever
Dovetailing

Explore increasingly more executions for increasingly more steps

Will discover an accepting execution if one exists
Language Equality Problem

\[
EQUAL = \{ \ (z, z') \mid L(M_z) = L(M_{z'}) \ \} 
\]

Claim: \( EMPTY \leq EQUAL \)

\[
f(z) = (z, z') \text{ where } M_{z'} \text{ rejects all inputs}
\]

\[
(z, z') \in EQUAL \iff z \in EMPTY
\]

\( EMPTY \) unrecognizable \( \implies \) \( EQUAL \) unrecognizable
Language Equality Problem

\[ \text{EQUAL} = \{ (z, z') \mid L(M_z) = L(M_{z'}) \} \]

Claim: ACCEPT \(\leq\) EQUAL

\[ f(z, w) = (z_1, z_2) \text{ where } M_{z_1} \text{ & } M_{z_2} \text{ behave as follows:} \]

- \(M_{z_1}\) accepts all strings. i.e., \(L(M_{z_1}) = \Sigma^*\)
- \(M_{z_2}\) runs \(M_z\) on \(w\) and if it accepts, accepts its input

\[(z_1, z_2) \in \text{EQUAL} \iff (z, w) \in \text{ACCEPT}\]

Hence \(\text{EQUAL}\) is not decidable.

Also, \(\text{EQUAL}^c\) is not recognizable. (Why?)
Map

R.E.

D
ACCEPT
HALT
EMPTY

EQUAL

coR.E.

D
ACCEPT^c
HALT^c
EMPTY

R
Post Correspondence Problem

**Theorem** [Post’46]: *HALT* reduces to *PostCP* — a “combinatorial” problem

*PostCP* is undecidable.

Given: Dominoes, each with a top-word and a bottom-word

Can one arrange them (using **any number of copies of each type**) so that the top and bottom strings are identical?

![Diagram of dominoes]

*abb* *ba* *abb* *abb* *a* *baa* *ab* *bbb* *a* *bb* *bba*
Recap

- If $L_1 \leq L_2$ then:
  - If $L_1$ is undecidable, so is $L_2$
  - If $L_1$ is unrecognizable, so is $L_2$
  - $\overline{L_1} \leq \overline{L_2}$

- $L$ and $\overline{L}$ recognizable $\iff$ $L$ and $\overline{L}$ decidable $\iff$ $L$ decidable

  - Corollary: If $L$ recognizable but undecidable, then $\overline{L}$ not recognizable
    - e.g., $\text{ACCEPT}^c$ is not recognizable
  - e.g.: If $\text{ACCEPT} \leq L$, then $\overline{L}$ not recognizable (Why?)

- If $L$ is recognizable, then so is $L' = \{ x \mid \exists w, (x,w) \in L \}$ (via dovetailing)