Undecidability

Chanathip Namprempre

Faculty of Engineering
Thammasat University

Outline

1. Undecidability
2. Mapping Reducibility
    - Definitions
    - Properties

An undecidable language

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}\ .$$

This language is recognizable (can you see why?) BUT undecidable!

Proof.

We prove by contradiction. Suppose A_{TM} is decidable. Let H be the decider for A_{TM}. Let D be a TM that, on input a TM $\langle M\rangle$,

1. runs H as a sub-routine
2. if $H(\langle M, \langle M\rangle\rangle)$ rejects, then accepts
3. if $H(\langle M, \langle M\rangle\rangle)$ accepts, then rejects

We get a contradiction when we run $D(\langle D\rangle)$. (Can you see why?) $\square$

Showing that A_{TM} is undecidable

The contradiction

Case 1: Suppose $D(\langle D \rangle) =$ yes.
From the definition of D, this means that $H(\langle D, \langle D \rangle \rangle) =$ no, but since H decides A_{TM}, we see that H would only say no if $D(\langle D \rangle) =$ no. This is a contradiction.

Case 2: Suppose $D(\langle D \rangle) =$ no.
From the definition of D, this means that $H(\langle D, \langle D \rangle \rangle) =$ yes, but since H decides A_{TM}, we see that H would only say yes if $D(\langle D \rangle) =$ yes. This is a contradiction.

Diagonalization

The proof above is based on a technique called diagonalization.

We can see this if we draw a table whose entries tell whether the machine in the given row accepts the input in the given column as follows.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\dots$	$\langle D \rangle$	$\dots$
M_1	accept	reject	accept	$\dots$	accept	$\dots$
M_2	reject	accept	accept	$\dots$	accept	$\dots$
M_3	accept	accept	reject	$\dots$	accept	$\dots$
$\vdots$				$\ddots$		
D	reject	reject	accept	$\dots$	???	$\dots$
$\vdots$						$\ddots$

The cell ??? asks whether D accepts $\langle D \rangle$.

Undecidability and Unrecognizability

Since A_{TM} is not recursive but is r.e., we get the following corollary.

Corollary

A_{TM} *is not co-r.e.*

Languages we consider

Most of these are r.e. Some are not r.e. Some are even neither r.e. nor co-r.e. (Can you tell which are r.e.?)

Example

$$
\begin{aligned}
\mathrm{A}_{\mathrm{TM}} &= \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\} \\
\mathrm{Halt}_{\mathrm{TM}} &= \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\} \\
\mathrm{Hang}_{\mathrm{TM}} &= \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ hangs on input } w\} \\
\mathrm{E}_{\mathrm{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\} \\
\mathrm{BlankTape}_{\mathrm{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } M \text{ halts on } \varepsilon\} \\
\mathrm{AcceptsSome}_{\mathrm{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } M \text{ accepts some input }\} \\
\mathrm{AcceptsAll}_{\mathrm{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } M \text{ accepts all inputs }\} \\
\mathrm{Regular}_{\mathrm{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\} \\
\mathrm{EQ}_{\mathrm{TM}} &= \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\} \\
\mathrm{All}_{\mathrm{CFG}} &= \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\} \\
\mathrm{All}_{\mathrm{DFA}} &= \{\langle D\rangle \mid D \text{ is a DFA and } L(D) = \Sigma^*\}
\end{aligned}
$$

Many problems are undecidable!

We do not want to go through diagonalization every time we want to show that something is undecidable.

So we relate undecidable problems to other problems.
(This is a recurring theme in computability and complexity theory.)

We do this via reduction.

Halting Problem

Theorem

$\text{Halt}_{\text{TM}} = \{\langle M, w\rangle \mid M \textit{ is a TM and } M \textit{ halts on input } w\}$ *is undecidable.*

The reduction goes as follows:

Suppose toward a contradiction that Halt_{TM} is decidable. Let R be the decider for Halt_{TM}. We use R to construct a decider for A_{TM}. This is a contradiction. So Halt_{TM} is undecidable.

$S =$ "On input $\langle M, w\rangle$, an encoding of a TM M and a string w,

1. Run R on input $\langle M, w\rangle$.
2. If R rejects, reject.
3. If R accepts, simulate M on w until it halts.
4. If M has accepted, accept. Otherwise, reject."

Then, we show that, if R decides Halt_{TM}, then S decides A_{TM}.

One could try this approach to prove other problems undecidable.

For example, try E_{TM}.

Outline

1. Undecidability
2. Mapping Reducibility
   - Definitions
   - Properties

Mapping reducibility: Definitions

The reduction that we saw can be formalized explicitly as a mapping (also known as many-to-one) reduction.

Definition

Language A is mapping reducible to language B, written $A \leq_m B$ iff there is a mapping reduction of A to B.

Definition

A function $f : \Sigma^* \rightarrow \Sigma^*$ is a mapping reduction of a language A to a language B iff f is computable and, for every $w \in \Sigma^*$,

- If $w \in A$, then $f(w) \in B$, and
- If $w \notin A$, then $f(w) \notin B$.

Try drawing a picture to visualize this property of f.

Mapping reducibility: Definitions (cont.)

Definition

A function $f : \Sigma^* \to \Sigma^*$ is a computable function iff there is a TM M that computes it.

Definition

A TM M computes a function f iff, for any input $w \in \Sigma^*$, the machine M halts with just $f(w)$ on its tape.

Why do we care about mapping reductions?

A reduction helps us relate different problems to each other.

For example, suppose we know that $A \leq_m B$ and that B is decidable.

Then, we know that A is also decidable. WHY??

We can decide A like so.

$$\begin{tikzcd}
\text{A} & \text{B} \\
w \arrow[r, "f"] \arrow[d, dotted] & f(w) \arrow[d, "\text{decider for } B"] \\
\text{Y/N} & \text{Y/N} \arrow[l]
\end{tikzcd}$$

Properties we get from mapping reducibility

Let A, B be languages. Suppose $A \leq_m B$.

1. If B is recursive, then so is A.
2. If A is not recursive, then neither is B.
3. If B is recursively enumerable, then so is A.
4. If A is not recursively enumerable, then neither is B.

Also, these statements are true:

- $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$
- If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.
  (i.e. $\leq_m$ is a transitive relation.)

Questions: Is $\leq_m$ reflexive? Is it symmetric?

Applying mapping reducibility

The properties of mapping reducibility are useful.

For example, if you want to show that a language

- L is not r.e., then you can show that $\mathrm{A_{TM}} \leq_m \overline{L}$
- L is not co-r.e., then you can show that $\mathrm{A_{TM}} \leq_m L$
- L is undecidable, then you can show either
  (but choose carefully)

Try writing reductions for these

Example

- $A_{TM} \le_m \text{Halt}_{TM}$
- $A_{TM} \le_m \overline{E_{TM}}$
- $E_{TM} \le_m EQ_{TM}$
- EQ_{TM} is neither r.e. nor co-r.e.
  (Try proving that $A_{TM} \le_m \overline{EQ_{TM}}$ and $A_{TM} \le_m EQ_{TM}$)
- $\text{Halt}_{TM} \le_m \text{BlankTape}_{TM}$
- $\text{Halt}_{TM} \le_m \text{AcceptsSome}_{TM}$
- $\text{Halt}_{TM} \le_m \text{AcceptsAll}_{TM}$
- $\text{BlankTape}_{TM} \le_m \text{Halt}_{TM}$
- $\text{AcceptsSome}_{TM} \le_m \text{Halt}_{TM}$
- $\text{AcceptsAll}_{TM} \le_m \text{Halt}_{TM}$ (You will fail.)
- $\text{AcceptsAll}_{TM} \le_m EQ_{TM}$

What conclusions can you draw from each of these reductions?