Introduction to the Theory of Computation

Set 9 — Undecidability

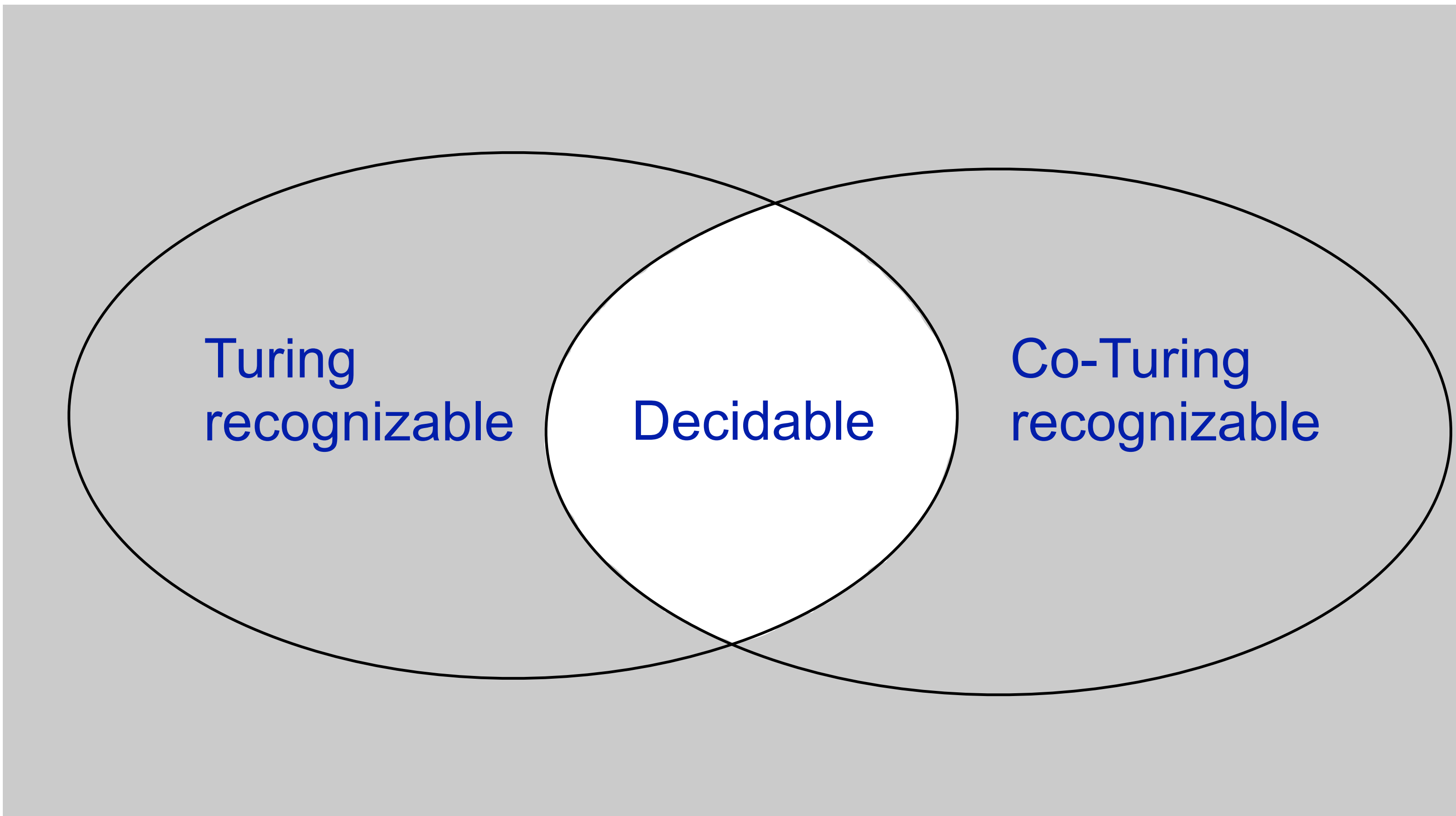
Classes of Languages

We have shown some language falls within each of the following classes

- Regular
- Context-free
- Decidable
- Turing recognizable

Here we review how to show that a language is undecidable using proof by contradiction.

Undecidable Languages



Undecidable Languages

We can prove a problem is undecidable by contradiction

- Assume the problem is decidable
- Show that this implies something impossible

The Halting Problem $HALT_{TM}$

$HALT_{TM}$ = {<M,w> | M is a TM and M halts on input w}

Theorem: $HALT_{TM}$ is undecidable

Proof: (by contradiction)

Show that if $HALT_{TM}$ is decidable then A_{TM} is also decidable

Proof (1)

Assume R decides $HALT_{TM}$

Let S be the following TM

S = “on input <M,w>

1. Run R on <M,w>
2. If R rejects, reject
3. If R accepts,
   simulate M on w until it halts
4. If M accepts, accept;
   if M rejects, reject”

Proof (2)

If $HALT_{TM}$ is decidable
then S decides A_{TM}

Since A_{TM} is not decidable,
$HALT_{TM}$ cannot be decidable

Proving Language L Is Undecidable

Assume L is decidable

- Let N be a TM that decides L

Show that a known undecidable language L' will be decidable if it can use N to make decisions

- This is called reducing problem L' to problem L

Conclude N cannot exist

- That is, the language L is not decidable

Reducibility

If we have two languages (or problems) A and B, then "A is reducible to B" means that we can use B to solve A.

- Measuring the area of a rectangle is reducible to measuring the lengths of its sides
- We showed that A_{NFA} is reducible to A_{DFA}

If A is reducible to B then

- solving B gives a solution to A

Reducibility

Why *"reduce"*

- When we reduce A to B, we show how to solve A by using B…

  …and can conclude that A is no harder than B

If A is reducible to B then

- solving B gives a solution to A
- B is easy → A is easy
- A is hard → B is hard

Undecidability of E_{TM}

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

Theorem: E_{TM} is undecidable

Proof: Assume E_{TM} is decidable with *decider* TM ***R***. Use ***R*** to decide A_{TM}

Recall $A_{TM} =$

$\{\langle M,w\rangle \mid M \text{ is a TM that accepts } w\}$

How can we use ***R*** (which takes <M> as input) to determine if M accepts w?

Make new TM, M_1, with $L(M_1) \neq \emptyset \Leftrightarrow$ M accepts w

Proof

New TM: Reject everything other than w, do whatever M does on input w.

M_1 = "On input x

1. **If $x \neq w$, reject**
2. **If $x = w$, run M on input w**
   - **Accept if M accepts"**

$L(M_1) \neq \emptyset \Leftrightarrow$ M accepts w

Make new TM, M_1, with $L(M_1) \neq \emptyset \Leftrightarrow$ M accepts w

Use *R* and M_1 to decide A_{TM}

Consider the following TM

S = “On input <M,w>

1. Construct M_1 that rejects all but w and simulates M on w
2. Run *R* on <M_1> $L(M_1) \neq \emptyset \Leftrightarrow$ M accepts w
3. If *R* accepts, reject; if *R* rejects, accept”

S decides A_{TM} — a contradiction

Therefore, E_{TM} is not decidable

Recap

Assume *R* decides E_{TM}

Create Turing machine M_1 such that

$L(M_1) \neq \varnothing \Leftrightarrow$ M accepts w

Create Turing machine S that decides A_{TM} by running *R* on input M_1

Conclude *R* cannot exist

- ☞ **E_{TM} cannot be decidable**

Another Undecidable Language

Let $REGULAR_{TM}$ = {<M> | M is a TM and L(M) is a regular language }

Theorem: $REGULAR_{TM}$ is undecidable

Proof: Assume R decides $REGULAR_{TM}$ and use R to decide A_{TM} (reduce the A_{TM} problem to the $REGULAR_{TM}$ problem).

As before, make a new TM, M_2, that accepts a regular language ⇔ M accepts w.

Proof *(continued)*

M_2 = "On input x

1. If $x = 0^n1^n$ for some n, accept
2. Otherwise, run M on w.
   If M accepts w, accept"

If M accepts w, then $L(M_2) = \Sigma^*$

- A regular language

Otherwise, $L(M_2) = 0^n1^n$

- Not a regular language

Proof *(continued)*

Assuming R decides $REGULAR_{TM}$ consider the following TM

S = "On input <M,w>

1. **Construct M_2 such that**
   $L(M_2)$ is regular $\Leftrightarrow$ M accepts w

2. **Run R on M_2**

3. **If R accepts, accept;**
   if R rejects, reject"

S decides A_{TM} $\Leftrightarrow$ R decides $REGULAR_{TM}$

Insight

TM M_2 is designed specifically so that

$L(M_2)$ is regular $\Leftrightarrow$ M accepts w

Run TM that decides $REGULAR_{TM}$ on M_2

Reducibility Recap

To prove some language L is undecidable, show that any known undecidable language (such as A_{TM}) is reducible to L

Having shown that
"A_{TM} is reducible to L"
we have shown that
L is undecidable

Course Recap — Goals

Explore the capabilities and limitations of computers

- Automata theory
  - How can we mathematically model computation?
- Computability theory
  - What problems can be solved by a computer?
- Complexity theory
  - What makes some problems computationally hard and others easy?

Course Recap

Automata Theory

- Introduced DFA, NFA, Regular Grammar, RE
  - Showed that they all accept the same class of languages
- Introduced CFG, PDA
  - PDA is essentially an NFA with a stack
  - PDAs and CFGs accept the same class of languages

Course Recap

Computability Theory

- Introduced TM
  - Like PDA's with more general memory model
- Importance of TM
  - Church-Turing Thesis
  - Any algorithm can be implemented on a TM
- Use the TM model and Church-Turing Thesis to understand and classify languages
  - Decidable languages
  - Undecidable languages
  - Recognizable languages
  - Unrecognizable languages

Coming Up

Complexity Theory

- Use TM model to determine how long an algorithm takes to run
  - Function of input length
- Classify algorithms according to their complexity