Undecidable languages, Reducibility

204213 Theory of Computation

Jittat Fakcharoenphol

Kasetsart University

January 20, 2009

Outline

Comparing the size of two infinite sets

- A function $f : A \rightarrow B$ is **one-to-one** if for $x, y \in A$, $x \neq y$ implies that $f(x) = f(y)$.
- f is called **onto** if for all $y \in B$, there exists $x \in A$ such that $f(x) = y$.
- If f is both one-to-one and onto, f is called a **correspondence**.

Comparing the size of two infinite sets

- A function $f : A \to B$ is **one-to-one** if for $x, y \in A$, $x \neq y$ implies that $f(x) = f(y)$.
- f is called **onto** if for all $y \in B$, there exists $x \in A$ such that $f(x) = y$.
- If f is both one-to-one and onto, f is called a **correspondence**.
- If a correspondence exists between two sets, we say that they have the same size.

Definition

A set A is **countable** if either it is finite or it has the same size as $\mathcal{N}$.

Countable sets

- Natural numbers $\mathcal{N}$, Even numbers $\mathcal{E}$, Integers $\mathcal{R}$, Rational numbers $\mathcal{Q}$

Countable sets

- Natural numbers $\mathcal{N}$, Even numbers $\mathcal{E}$, Integers $\mathcal{R}$, Rational numbers $\mathcal{Q}$
- A set of all finite strings Σ^*

Countable sets

- Natural numbers $\mathcal{N}$, Even numbers $\mathcal{E}$, Integers $\mathcal{R}$, Rational numbers $\mathcal{Q}$
- A set of all finite strings Σ^*
- A set of all Turing machines.

Countable sets

- Natural numbers $\mathcal{N}$, Even numbers $\mathcal{E}$, Integers $\mathcal{R}$, Rational numbers $\mathcal{Q}$
- A set of all finite strings Σ^*
- A set of all Turing machines.
  - Each TM can be encoded as a string,
  - All strings are countable,
  - List all strings, skip those which are not encodings of TM's.

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \rightarrow \mathcal{P}(N)$.

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \to \mathcal{P}(N)$.
- Construct a set X as follows: for any natural number i, $i \in X$ iff $i \notin F(i)$.

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \to \mathcal{P}(N)$.
- Construct a set X as follows: for any natural number i, $i \in X$ iff $i \notin F(i)$.
- We can show that there is no $i \in \mathcal{N}$ such that $X = F(i)$.

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \rightarrow \mathcal{P}(N)$.
- Construct a set X as follows: for any natural number i, $i \in X$ iff $i \notin F(i)$.
- We can show that there is no $i \in \mathcal{N}$ such that $X = F(i)$.
  - (Proof by contradiction.)

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \rightarrow \mathcal{P}(N)$.
- Construct a set X as follows: for any natural number i, $i \in X$ iff $i \notin F(i)$.
- We can show that there is no $i \in \mathcal{N}$ such that $X = F(i)$.
  - (Proof by contradiction.)
- Thus, F is not onto. (Because $X \in \mathcal{P}(N)$, but noone maps to X.)

Diagonalization

Theorem 1

$\mathcal{N}$ and $\mathcal{P}(N)$ have different sizes, i.e., $\mathcal{P}(N)$ is uncountable.

Proof.

- We'll prove by contradiction. Thus, assume that $\mathcal{P}(N)$ is countable, i.e., there is a correspondence $F : \mathcal{N} \rightarrow \mathcal{P}(N)$.
- Construct a set X as follows: for any natural number i, $i \in X$ iff $i \notin F(i)$.
- We can show that there is no $i \in \mathcal{N}$ such that $X = F(i)$.
  - (Proof by contradiction.)
- Thus, F is not onto. (Because $X \in \mathcal{P}(N)$, but noone maps to X.)
- This leads to the contradiction as required.

$\square$

Previous proof in table form

Some languages is not Turing-recognizable

- Note that $\mathcal{P}(N)$ is of the same size as the set all of languages.

Some languages is not Turing-recognizable

- Note that $\mathcal{P}(N)$ is of the same size as the set all of languages. Thus, the set of all languages is uncountable.

Some languages is not Turing-recognizable

- Note that $\mathcal{P}(N)$ is of the same size as the set all of languages. Thus, the set of all languages is uncountable.
- However, the set of all Turing machines is countable.
- Since each TM recognizes only one language, there's some language that no TM's recognize.

A_{TM}

Recall the language

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

A_{TM}

Recall the language

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

Recall the definition: a language is **decidable** is some TM decides it.

Review: A_{TM} is recognizable

Just simulate M on w. Accept when M accepts, reject when M rejects.

Theorem 2

A_{TM} is undecidable

Proof (1)

- Assume that A_{TM} is decidable. We'll try to obtain a contradiction.

Proof (1)

- Assume that A_{TM} is decidable. We'll try to obtain a contradiction.
- Since A_{TM} is decidable, there exists a TM H that decides it.

Proof (1)

- Assume that A_{TM} is decidable. We'll try to obtain a contradiction.
- Since A_{TM} is decidable, there exists a TM H that decides it. For TM M and input w, we have that

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

Proof (2)

- Given H, we can construct another TM E with H as a subroutine.

Proof (2)

- Given H, we can construct another TM E with H as a subroutine. Given $\langle M, w \rangle$, E calls H to check if M accepts w, then acts the opposite.

Proof (2)

- Given H, we can construct another TM E with H as a subroutine. Given $\langle M, w \rangle$, E calls H to check if M accepts w, then acts the opposite.
- $E =$ "On input $\langle M, w \rangle$ where M is a TM:

Proof (2)

- Given H, we can construct another TM E with H as a subroutine. Given $\langle M, w\rangle$, E calls H to check if M accepts w, then acts the opposite.
- $E =$ "On input $\langle M, w\rangle$ where M is a TM:
  1. Run H on input $\langle M, w\rangle$.

Proof (2)

- Given H, we can construct another TM E with H as a subroutine. Given $\langle M, w\rangle$, E calls H to check if M accepts w, then acts the opposite.
- $E =$ "On input $\langle M, w\rangle$ where M is a TM:
  1. Run H on input $\langle M, w\rangle$.
  2. If H accepts, reject. If H rejects, accept."

Proof (3)

- We'll use a variant of E to get to the contradiction.

Proof (3)

- We'll use a variant of E to get to the contradiction.
- Given Construct another TM D with H as a subroutine.

Proof (3)

- We'll use a variant of E to get to the contradiction.
- Given Construct another TM D with H as a subroutine. D's input is an encoding to a TM.

Proof (3)

- We'll use a variant of E to get to the contradiction.
- Given Construct another TM D with H as a subroutine. D's input is an encoding to a TM. Given $\langle M \rangle$, D calls H to check if M accepts **its own description** $\langle M \rangle$, then acts the opposite.

Proof (3)

- We'll use a variant of E to get to the contradiction.
- Given Construct another TM D with H as a subroutine. D's input is an encoding to a TM. Given $\langle M \rangle$, D calls H to check if M accepts **its own description** $\langle M \rangle$, then acts the opposite.
- Think of $\langle M \rangle$ as a source code of M. Running M on $\langle M \rangle$ is just like running M on its source code.

Proof (4)

- Formally, $D =$ "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, reject. If H rejects, accept."

Proof (4)

- Formally, $D =$ "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, reject. If H rejects, accept."
- Thus,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accept } \langle M \rangle \end{cases}$$

Proof (4)

- Formally, $D =$ "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, reject. If H rejects, accept."
- Thus,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accept } \langle M \rangle \end{cases}$$

- What is $D(\langle D \rangle)$? (Run D on its description)

Proof (4)

- Formally, $D =$ "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, reject. If H rejects, accept."
- Thus,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accept } \langle M \rangle \end{cases}$$

- What is $D(\langle D \rangle)$? (Run D on its description)

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \text{reject} & \text{if } D \text{ accept } \langle D \rangle \end{cases}$$

Proof (4)

- Formally, $D =$ "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, reject. If H rejects, accept."
- Thus,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accept } \langle M \rangle \end{cases}$$

- What is $D(\langle D \rangle)$? (Run D on its description)

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \text{reject} & \text{if } D \text{ accept } \langle D \rangle \end{cases}$$

- Is this possible? (Try each case.)

Diagonalization Proof

- Was that a diagonalization proof?

Diagonalization Proof

- Was that a diagonalization proof?
- Try it. Rows are all TM's. Columns are all encodings of TM's.

Turing Unrecognizable

- We know that there exists some language that is not recognizable by any TM's.

Turing Unrecognizable

- We know that there exists some language that is not recognizable by any TM's. What are they?

Turing Unrecognizable

- We know that there exists some language that is not recognizable by any TM's. What are they? Are they crazy, unthinkable languages?

Turing Unrecognizable

- We know that there exists some language that is not recognizable by any TM's. What are they? Are they crazy, unthinkable languages?
- Not really. We'll demonstrate one such language.

Decidable/recognizable

A language is called **co-Turing recognizable** if its complement is Turing recognizable.

Theorem 3

A language is decidable iff it is Turing recognizable and co-Turing recognizable.

Theorem 4

$\overline{A_{TM}}$ is not Turing-recognizable.

Theorem 4

$\overline{A_{TM}}$ is not Turing-recognizable.

Proof.

- We know that A_{TM} is recognizable.
- If $\overline{A_{TM}}$ is also recognizable, from previous theorem, we have that A_{TM} must also be decidable.
- But A_{TM} is undecidable; thus, $\bar{A}_{TM}$ is not recognizable.

$\square$

More undecidable languages

- Now that we know that A_{TM} is undecidable.
- We want to show that other problems are also undecidable.
- A general technique for this is **reduction**.

Reduction

- We use reduction all the time.

Reduction

- We use reduction all the time.
- To solve A, maybe it's easy to "reduce" it to another problem B.

Reduction

- We use reduction all the time.
- To solve A, maybe it's easy to "reduce" it to another problem B.
  - Thus, if we can solve B then we can solve A.

Reduction

- We use reduction all the time.
- To solve A, maybe it's easy to "reduce" it to another problem B.
    - Thus, if we can solve B then we can solve A.
- Think another way: what if A is unsolvable?

Halting problem

Let $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

Theorem 5

$HALT_{TM}$ is undecidable.

Emptiness

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

Theorem 6

E_{TM} is undecidable.

Regular languages

Let $REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$

Theorem 7

$REGULAR_{TM}$ is undecidable.

Equivalence

Let
$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TM's and } L(M_1) = L(M_2)\}$

Theorem 8

EQ_{TM} is undecidable.