CS 383

Lecture 19 – Diagonalization and undecidable languages

Stephen Checkoway

Spring 2024

Sizes of sets

Two sets X and Y have the same size if there is a bijection between them, $f : X \to Y$

What's a bijection?

Sizes of sets

Two sets X and Y have the same size if there is a bijection between them, $f : X \to Y$
What's a bijection?

Recall $f : X \to Y$ is a bijection if

1. for all $a, b \in X$, $f(a) = f(b)$ implies $a = b$ (injective)
2. for all $y \in Y$, there exists $x \in X$ such that $y = f(x)$ (surjective)

Example

The natural numbers and the integers have the same size

$$f : \mathbb{Z} \to \mathbb{N}$$

$$f(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}$$

Example

The natural numbers and the integers have the same size

$$f : \mathbb{Z} \to \mathbb{N}$$

$$f(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}$$

$$\begin{aligned} &\vdots \\ -2 &\mapsto 3 \\ -1 &\mapsto 1 \\ 0 &\mapsto 0 \\ 1 &\mapsto 2 \\ 2 &\mapsto 4 \\ &\vdots \end{aligned}$$

Example

The integers and the rational numbers have the same size

Example

The integers and the rational numbers have the same size

The fundamental theorem of arithmetic tells us that every positive integer can be expressed uniquely as a product of prime powers

$$p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots$$

where p_i are the primes in order (2, 3, 5, 7, etc.) and
$n_i \in \mathbb{N}$ and finitely many n_i are nonzero

Example

The integers and the rational numbers have the same size

The fundamental theorem of arithmetic tells us that every positive integer can be expressed uniquely as a product of prime powers

$$p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots$$

where p_i are the primes in order (2, 3, 5, 7, etc.) and
$n_i \in \mathbb{N}$ and finitely many n_i are nonzero

Similarly, every positive rational number can be expressed uniquely as a product of prime powers

$$p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots$$

where p_i are the primes in order and
$n_i \in \mathbb{Z}$ and finitely many n_i are nonzero

Example continued

Let $f : \mathbb{Z} \to \mathbb{N}$ be our bijection from before
Define $g : \mathbb{Q}^+ \to \mathbb{Z}^+$ by

$$g(p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots) = p_1^{f(n_1)} p_2^{f(n_2)} p_3^{f(n_3)} \cdots$$

Note that we're mapping the integer exponents to natural number exponents and the (infinitely many) 0 exponents remain 0 because $f(0) = 0$

Example continued

Let $f : \mathbb{Z} \to \mathbb{N}$ be our bijection from before
Define $g : \mathbb{Q}^+ \to \mathbb{Z}^+$ by

$$g(p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots) = p_1^{f(n_1)} p_2^{f(n_2)} p_3^{f(n_3)} \cdots$$

Note that we're mapping the integer exponents to natural number exponents and the (infinitely many) 0 exponents remain 0 because $f(0) = 0$

Since f is a bijection, g is a bijection (this isn't hard to show)

Example continued

Let $f : \mathbb{Z} \to \mathbb{N}$ be our bijection from before
Define $g : \mathbb{Q}^+ \to \mathbb{Z}^+$ by

$$g(p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots) = p_1^{f(n_1)} p_2^{f(n_2)} p_3^{f(n_3)} \cdots$$

Note that we're mapping the integer exponents to natural number exponents and the (infinitely many) 0 exponents remain 0 because $f(0) = 0$

Since f is a bijection, g is a bijection (this isn't hard to show)

Finally, let's define our bijection $h : \mathbb{Q} \to \mathbb{Z}$

$$h(x) = \begin{cases} g(x) & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -g(-x) & \text{if } x < 0 \end{cases}$$

Example continued

Let $f : \mathbb{Z} \to \mathbb{N}$ be our bijection from before
Define $g : \mathbb{Q}^+ \to \mathbb{Z}^+$ by

$$g(p_1^{n_1} p_2^{n_2} p_3^{n_3} \cdots) = p_1^{f(n_1)} p_2^{f(n_2)} p_3^{f(n_3)} \cdots$$

Note that we're mapping the integer exponents to natural number exponents and the (infinitely many) 0 exponents remain 0 because $f(0) = 0$

Since f is a bijection, g is a bijection (this isn't hard to show)

Finally, let's define our bijection $h : \mathbb{Q} \to \mathbb{Z}$

$$h(x) = \begin{cases} g(x) & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -g(-x) & \text{if } x < 0 \end{cases}$$

And just for fun, $f \circ h : \mathbb{Q} \to \mathbb{N}$ is a bijection

Countable

A set X is countable if it is finite or it has the same size as $\mathbb{N}$

Countable

A set X is countable if it is finite or it has the same size as $\mathbb{N}$

Countably infinite sets include $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$

Countable

A set X is countable if it is finite or it has the same size as $\mathbb{N}$

Countably infinite sets include $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$

Subsets of countable sets are countable (intuitively true but a hassle to prove without some additional math or an alternative, but equivalent definition of countability)

Each language is a countable set

Given an alphabet Σ, the language Σ^* is countably infinite. How do we show this?

Each language is a countable set

Given an alphabet Σ, the language Σ^* is countably infinite. How do we show this?

List the strings in lexicographic order to construct the mapping
E.g., $f : \{0,1\}^* \to \mathbb{N}$ given by

$$
\begin{aligned}
\varepsilon &\mapsto 0 \\
0 &\mapsto 1 \\
1 &\mapsto 2 \\
00 &\mapsto 3 \\
01 &\mapsto 4 \\
10 &\mapsto 5 \\
11 &\mapsto 6 \\
000 &\mapsto 7 \\
&\vdots
\end{aligned}
$$

Each language is a countable set

Given an alphabet Σ, the language Σ^* is countably infinite. How do we show this?

List the strings in lexicographic order to construct the mapping
E.g., $f : \{0, 1\}^* \to \mathbb{N}$ given by

$$
\begin{aligned}
\varepsilon &\mapsto 0 \\
0 &\mapsto 1 \\
1 &\mapsto 2 \\
00 &\mapsto 3 \\
01 &\mapsto 4 \\
10 &\mapsto 5 \\
11 &\mapsto 6 \\
000 &\mapsto 7 \\
&\vdots
\end{aligned}
$$

Every language $L \subseteq \Sigma^*$ is thus countable

Diagonalization: infinite sequences over $\{0, 1\}$

Theorem

The set S of all infinite sequences over $\{0, 1\}$ is uncountable

Diagonalization: infinite sequences over $\{0, 1\}$

Theorem

The set S of all infinite sequences over $\{0, 1\}$ is uncountable

Proof.

Assume S is countable so there's a bijection $f : \mathbb{N} \to S$

We can construct a new infinite sequence $\mathbf{b} = b_0, b_1, \ldots$ that differs from every sequence in S.

n	$f(n)$
0	0 0 1 0 1 $\cdots$
1	1 0 0 0 1 $\cdots$
2	0 1 1 0 0 $\cdots$
3	1 1 0 1 0 $\cdots$
$\vdots$	$\vdots$

Diagonalization: infinite sequences over $\{0, 1\}$

Theorem

The set S of all infinite sequences over $\{0, 1\}$ is uncountable

Proof.

Assume S is countable so there's a bijection $f : \mathbb{N} \to S$

We can construct a new infinite sequence $\mathbf{b} = b_0, b_1, \ldots$ that differs from every sequence in S.

In particular, b_i will differ from $f(i)$ in position i

$$b_i = \begin{cases} 0 & \text{if the } i\text{th element of } f(i) \text{ is 1} \\ 1 & \text{if the } i\text{th element of } f(i) \text{ is 0} \end{cases}$$

n	$f(n)$
0	0 0 1 0 1 $\cdots$
1	1 0 0 0 1 $\cdots$
2	0 1 1 0 0 $\cdots$
3	1 1 0 1 0 $\cdots$
$\vdots$	$\vdots$

$\mathbf{b} = 1100\cdots$

Diagonalization: infinite sequences over $\{0, 1\}$

Theorem

The set S of all infinite sequences over $\{0, 1\}$ is uncountable

Proof.

Assume S is countable so there's a bijection $f : \mathbb{N} \to S$

We can construct a new infinite sequence $\mathbf{b} = b_0, b_1, \ldots$ that differs from every sequence in S.

In particular, b_i will differ from $f(i)$ in position i

$$b_i = \begin{cases} 0 & \text{if the } i\text{th element of } f(i) \text{ is 1} \\ 1 & \text{if the } i\text{th element of } f(i) \text{ is 0} \end{cases}$$

Now $\mathbf{b} \in S$ but for all i, $f(i) \neq \mathbf{b}$ which is a contradiction so S must not be countable □

n	$f(n)$
0	0 0 1 0 1 ⋯
1	1 0 0 0 1 ⋯
2	0 1 1 0 0 ⋯
3	1 1 0 1 0 ⋯
⋮	⋮

$\mathbf{b} = 1100\cdots$

There are a countable number of Turing machines

Consider any fixed binary representation of a TM

E.g., given

$$Q = \{1, 2, \ldots, k\}$$
$$\Sigma = \{1, 2, \ldots, m\}$$
$$\Gamma = \{1, 2, \ldots, n\}$$
$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{1, 2\}$$ where $1 = \mathrm{L}$ and $2 = \mathrm{R}$
$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

here's one possible representation

$$\langle \delta(q, a) \rangle = 0^r 1 0^b 1 0^d$$ where $\delta(q, a) = (r, b, d)$
$$\langle \delta \rangle = \langle \delta(1,1) \rangle \; 11 \; \langle \delta(1,2) \rangle \; 11 \cdots 11 \; \langle \delta(k,n) \rangle$$
$$\langle M \rangle = 0^k \; 111 \; 0^m \; 111 \; 0^n \; 111 \; \langle \delta \rangle \; 111 \; 0^{q_{\text{accept}}} \; 111 \; 0^{q_{\text{reject}}}$$

Thus $\langle M \rangle$ is an element of $\{0, 1\}^*$

There are a countable number of Turing machines continued

For simplicity, for all $x \in \{0,1\}^*$ such that x is not a valid encoding of a TM, define x to be a TM with $q_0 = q_{\text{reject}}$

There are a countable number of Turing machines continued

For simplicity, for all $x \in \{0, 1\}^*$ such that x is not a valid encoding of a TM, define x to be a TM with $q_0 = q_{\text{reject}}$

Now every binary string is a valid encoding of a TM, i.e.,

$$\{0, 1\}^* = \{\langle M \rangle \mid \langle M \rangle \text{ is is a TM}\}$$

There are a countable number of Turing machines continued

For simplicity, for all $x \in \{0,1\}^*$ such that x is not a valid encoding of a TM, define x to be a TM with $q_0 = q_{\text{reject}}$

Now every binary string is a valid encoding of a TM, i.e.,

$$\{0,1\}^* = \{\langle M \rangle \mid \langle M \rangle \text{ is is a TM}\}$$

Since $\{0,1\}^*$ is countable, there are a countable number of Turing machines

There are an uncountable number of languages

Theorem

For every alphabet Σ, the set of all languages over Σ is uncountable

There are an uncountable number of languages

Theorem

For every alphabet Σ, the set of all languages over Σ is uncountable

Proof.

We proved that Σ^* is countably infinite; let $f : \mathbb{N} \to \Sigma^*$ be a bijection

For each language L over Σ, define an infinite sequence $\mathbf{b} = b_0, b_1, \ldots$ over $\{0, 1\}$ where

$$b_i = \begin{cases} 0 & \text{if } f(i) \notin L \\ 1 & \text{if } f(i) \in L \end{cases}$$

$\mathbf{b}$ is called the characteristic sequence of L

There are an uncountable number of languages

Theorem

For every alphabet Σ, the set of all languages over Σ is uncountable

Proof.

We proved that Σ^* is countably infinite; let $f : \mathbb{N} \to \Sigma^*$ be a bijection

For each language L over Σ, define an infinite sequence $\mathbf{b} = b_0, b_1, \ldots$ over $\{0, 1\}$ where

$$b_i = \begin{cases} 0 & \text{if } f(i) \notin L \\ 1 & \text{if } f(i) \in L \end{cases}$$

$\mathbf{b}$ is called the characteristic sequence of L

Each characteristic sequence defines a language and each language has a unique characteristic sequence

We proved that there are uncountably many infinite binary sequences so there are uncountably many languages over Σ □

A simple corollary

There are (uncountably many) languages that are not Turing-recognizable (and thus not decidable)

An explicit undecidable language

Theorem

The language $\textsc{Diag} = \{\langle M \rangle \mid M$ *is a TM and does not accept* $\langle M \rangle\}$ *is undecidable*

An explicit undecidable language

Theorem

The language $\textsc{Diag} = \{\langle M \rangle \mid M$ *is a TM and does not accept* $\langle M \rangle\}$ *is undecidable*

Proof.

Assume that D is a TM that decides $\textsc{Diag}$

Is $\langle D \rangle \in \textsc{Diag}$?

An explicit undecidable language

Theorem

The language $\textsc{Diag} = \{\langle M \rangle \mid M$ *is a TM and does not accept* $\langle M \rangle\}$ *is undecidable*

Proof.

Assume that D is a TM that decides $\textsc{Diag}$

Is $\langle D \rangle \in \textsc{Diag}$?

Two options

- If $\langle D \rangle \in \textsc{Diag}$, then since D decides $\textsc{Diag}$, D must accept $\langle D \rangle$ but then by definition of $\textsc{Diag}$, $\langle D \rangle \notin \textsc{Diag}$

An explicit undecidable language

Theorem

The language $\textsc{Diag} = \{\langle M \rangle \mid M$ *is a TM and does not accept* $\langle M \rangle\}$ *is undecidable*

Proof.

Assume that D is a TM that decides $\textsc{Diag}$

Is $\langle D \rangle \in \textsc{Diag}$?

Two options

- If $\langle D \rangle \in \textsc{Diag}$, then since D decides $\textsc{Diag}$, D must accept $\langle D \rangle$ but then by definition of $\textsc{Diag}$, $\langle D \rangle \notin \textsc{Diag}$
- If $\langle D \rangle \notin \textsc{Diag}$, then since D decides $\textsc{Diag}$, D must reject $\langle D \rangle$ but if D rejects $\langle D \rangle$, then by definition, $\langle D \rangle \in \textsc{Diag}$

Either option leads to a contradiction so $\textsc{Diag}$ must not be decidable $\square$

An explicit undecidable language

Theorem

The language $\textsc{Diag} = \{\langle M \rangle \mid M$ *is a TM and does not accept* $\langle M \rangle\}$ *is undecidable*

Proof.

Assume that D is a TM that decides $\textsc{Diag}$

Is $\langle D \rangle \in \textsc{Diag}$?

Two options

- If $\langle D \rangle \in \textsc{Diag}$, then since D decides $\textsc{Diag}$, D must accept $\langle D \rangle$ but then by definition of $\textsc{Diag}$, $\langle D \rangle \notin \textsc{Diag}$
- If $\langle D \rangle \notin \textsc{Diag}$, then since D decides $\textsc{Diag}$, D must reject $\langle D \rangle$ but if D rejects $\langle D \rangle$, then by definition, $\langle D \rangle \in \textsc{Diag}$

Either option leads to a contradiction so $\textsc{Diag}$ must not be decidable □

Replacing “reject” with “does not accept” in the proof shows that $\textsc{Diag}$ is not only not decidable, it’s not even Turing-recognizable!

Acceptance problem for TMs

Theorem

The language $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}$ *is undecidable*

How should we approach problems like this?

Proving that a language is not decidable

To prove that a language A is undecidable,

1. Assume that A is decidable and let R be a TM that decides A
2. Select an undecidable language B
3. Construct a new TM D that decides B and that uses R as a subroutine
4. Since B is undecidable but D is a decider, this is a contradiction and our assumption in step 1 must be wrong so A is undecidable

Steps 2 and 3 are the hard steps that require some cleverness

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

D = "On input $\langle M \rangle$,

1. Run R on $\langle M, \langle M \rangle \rangle$
2. If R accepts, *reject*; otherwise *accept*."

We need to show that $L(D) = \textsc{Diag}$ and that D is a decider.

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

D = "On input $\langle M \rangle$,

1. Run R on $\langle M, \langle M \rangle \rangle$
2. If R accepts, *reject*; otherwise *accept*."

We need to show that $L(D) = \textsc{Diag}$ and that D is a decider.

By assumption, R is a decider so it halts on $\langle M, \langle M \rangle \rangle$ and thus D halts on all input so it is a decider

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

D = "On input $\langle M \rangle$,

1. Run R on $\langle M, \langle M \rangle \rangle$
2. If R accepts, *reject*; otherwise *accept*."

We need to show that $L(D) = \textsc{Diag}$ and that D is a decider.

By assumption, R is a decider so it halts on $\langle M, \langle M \rangle \rangle$ and thus D halts on all input so it is a decider

If $\langle M \rangle \in \textsc{Diag}$, then $\langle M \rangle \notin L(M)$ so R rejects and D accepts so $\langle M \rangle \in L(D)$.

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

D = "On input $\langle M \rangle$,

1. Run R on $\langle M, \langle M \rangle \rangle$
2. If R accepts, *reject*; otherwise *accept*."

We need to show that $L(D) = \textsc{Diag}$ and that D is a decider.

By assumption, R is a decider so it halts on $\langle M, \langle M \rangle \rangle$ and thus D halts on all input so it is a decider

If $\langle M \rangle \in \textsc{Diag}$, then $\langle M \rangle \notin L(M)$ so R rejects and D accepts so $\langle M \rangle \in L(D)$.

If $\langle M \rangle \notin \textsc{Diag}$, then $\langle M \rangle \in L(M)$ so R accepts and D rejects so $\langle M \rangle \notin L(D)$.

Proof

Proof that A_{TM} is undecidable.

Assume that A_{TM} is decidable with decider R.

Let's build a TM D that decides $\textsc{Diag}$.

D = "On input $\langle M \rangle$,

1. Run R on $\langle M, \langle M \rangle \rangle$
2. If R accepts, *reject*; otherwise *accept*."

We need to show that $L(D) = \textsc{Diag}$ and that D is a decider.

By assumption, R is a decider so it halts on $\langle M, \langle M \rangle \rangle$ and thus D halts on all input so it is a decider

If $\langle M \rangle \in \textsc{Diag}$, then $\langle M \rangle \notin L(M)$ so R rejects and D accepts so $\langle M \rangle \in L(D)$.

If $\langle M \rangle \notin \textsc{Diag}$, then $\langle M \rangle \in L(M)$ so R accepts and D rejects so $\langle M \rangle \notin L(D)$.

Thus D decides $\textsc{Diag}$. This is a contradiction so A_{TM} must not be decidable. □

Halting problem for TMs

Theorem
The language $\textsc{Halt}_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ *is a TM and* M *halts when run on* $w\}$ *is undecidable*

Assume that $\textsc{Halt}_{\mathsf{TM}}$ is decided by TM H. How do we use H to construct a decider D for A_{TM}?

Proof

Proof.

Assume H is a decider for $\text{HALT}_{\mathsf{TM}}$ and build a decider D for A_{TM}.
D = "On input $\langle M, w \rangle$,

1. Run H on $\langle M, w \rangle$ and if H rejects, *reject*.
2. Run M on w and if M accepts, *accept*; otherwise *reject*."

D is a decider because if M loops on w, then H and D will reject. Otherwise, M will halt on w so D will halt.

If $w \in L(M)$, then M halts on w so H will accept and then D will accept.

If $w \notin L(M)$, then there are two options. If M loops on w, then H and thus D will reject. If M rejects w, then H will accept but D will reject. □

Co-Turing-recognizable (CoRE)

A language L is co-Turing-recognizable (coRE) if $\overline{L}$ is Turing-recognizable (RE)

Co-Turing-recognizable (CoRE)

A language L is co-Turing-recognizable (coRE) if $\overline{L}$ is Turing-recognizable (RE)

Theorem

A language L is decidable $\iff$ L is RE and L is coRE

Co-Turing-recognizable (CoRE)

A language L is co-Turing-recognizable (coRE) if $\overline{L}$ is Turing-recognizable (RE)

Theorem

A language L is decidable $\iff$ L is RE and L is coRE

To prove this, we need to prove three things

1. If L is decidable, then L is RE
2. If L is decidable, then L is coRE
3. If L is RE and coRE, then L is decidable

Parts 1 and 2 together show the $\implies$ direction and part 3 shows the $\impliedby$ direction

Proof

Proof.

$\Longrightarrow$:

If L is decidable, then there is some decider M such that $L(M) = L$. Thus L is RE.

Proof

Proof.

$\implies$:

If L is decidable, then there is some decider M such that $L(M) = L$. Thus L is RE.

By swapping the accept and reject states of M, we get a new decider M' that decides $\overline{L}$. Thus L is coRE.

Proof

Proof.

$\Longrightarrow$:

If L is decidable, then there is some decider M such that $L(M) = L$. Thus L is RE.

By swapping the accept and reject states of M, we get a new decider M' that decides $\overline{L}$. Thus L is coRE.

$\Longleftarrow$:

If L is RE, then there is some TM M_1 that recognizes it
If L is coRE, then there is some TM M_2 that recognizes $\overline{L}$

Build M = "On input w,

1. Run M_1 and M_2 on w simultaneously (e.g., with 2 tapes)
2. If M_1 accepts, *accept*. If M_2 accepts, *reject*."

One of M_1 or M_2 must accept, so M will halt on any input and thus decides L. □

A_{TM} is RE but not coRE

Theorem

A_{TM} is RE but not coRE

Proof.

Since A_{TM} is not decidable, if we show that it is RE, then it can't be coRE because then it would be decidable.

We can build R to recognize A_{TM} as follows.

R = "On input $\langle M, w \rangle$,

1. Run M on w.
2. If M accepts, *accept*; if M rejects, *reject*."

A_{TM} is RE but not coRE

Theorem

A_{TM} is RE but not coRE

Proof.

Since A_{TM} is not decidable, if we show that it is RE, then it can't be coRE because then it would be decidable.

We can build R to recognize A_{TM} as follows.

R = "On input $\langle M, w \rangle$,

1. Run M on w.
2. If M accepts, *accept*; if M rejects, *reject*."

Note that if M loops on w, then R will loop, but this is okay because R just needs to recognize A_{TM}, not decide it

Proof continued

There are three cases

1. $\langle M, w \rangle \in A_{\mathsf{TM}}$. M will accept w so R will accept.

Proof continued

There are three cases

1. $\langle M, w\rangle \in A_{\mathsf{TM}}$. M will accept w so R will accept.
2. $\langle M, w\rangle \notin A_{\mathsf{TM}}$. M will either loop on w or reject and R will do the same.

Proof continued

There are three cases

1. $\langle M, w \rangle \in A_{\mathsf{TM}}$. M will accept w so R will accept.
2. $\langle M, w \rangle \notin A_{\mathsf{TM}}$. M will either loop on w or reject and R will do the same.
3. The input isn't a valid encoding of $\langle M, w \rangle$. R will reject before step 1.

Proof continued

There are three cases

1. $\langle M, w \rangle \in A_{\mathsf{TM}}$. M will accept w so R will accept.
2. $\langle M, w \rangle \notin A_{\mathsf{TM}}$. M will either loop on w or reject and R will do the same.
3. The input isn't a valid encoding of $\langle M, w \rangle$. R will reject before step 1.

Thus $L(R) = A_{\mathsf{TM}}$ so A_{TM} is RE. □

Emptiness problem for TMs

Theorem

The language $E_{\mathsf{TM}} = \{\langle M \rangle \mid M$ *is a TM and* $L(M) = \varnothing\}$ *is coRE.*

To prove this, we need only give a TM that recognizes $\overline{E_{\mathsf{TM}}}$

Emptiness problem for TMs

Theorem

The language $E_{\mathsf{TM}} = \{\langle M \rangle \mid M$ *is a TM and* $L(M) = \varnothing\}$ *is coRE.*

To prove this, we need only give a TM that recognizes $\overline{E_{\mathsf{TM}}}$

Proof.

Let R = "On input w,

1. If $w \neq \langle M \rangle$ for some TM M, *accept*.
2. For $n = 0$ up to ∞
3. For each string $w \in \Sigma^*$ of length at most n
4. Simulate M on w for at most n steps.
5. If M accepts w, *accept*."

Emptiness problem for TMs

Theorem

The language $E_{\mathsf{TM}} = \{\langle M \rangle \mid M$ *is a TM and* $L(M) = \varnothing\}$ *is coRE.*

To prove this, we need only give a TM that recognizes $\overline{E_{\mathsf{TM}}}$

Proof.

Let R = "On input w,

1. If $w \neq \langle M \rangle$ for some TM M, *accept*.
2. For $n = 0$ up to ∞
3.   For each string $w \in \Sigma^*$ of length at most n
4.   Simulate M on w for at most n steps.
5.   If M accepts w, *accept*."

If $L(M) \neq \varnothing$, then there is some w that M will accept so R will accept $\langle M \rangle$.

Emptiness problem for TMs

Theorem

The language $E_{\mathsf{TM}} = \{\langle M \rangle \mid M$ is a TM and $L(M) = \varnothing\}$ is coRE.

To prove this, we need only give a TM that recognizes $\overline{E_{\mathsf{TM}}}$

Proof.

Let R = "On input w,

1. If $w \neq \langle M \rangle$ for some TM M, *accept*.
2. For $n = 0$ up to ∞
3. For each string $w \in \Sigma^*$ of length at most n
4. Simulate M on w for at most n steps.
5. If M accepts w, *accept*."

If $L(M) \neq \varnothing$, then there is some w that M will accept so R will accept $\langle M \rangle$.

If $L(M) = \varnothing$, then M will never accept so R will loop on $\langle M \rangle$.

Thus $L(R) = \overline{E_{\mathsf{TM}}}$ so E_{TM} is coRE. □

Emptiness problem for TMs is undecidable

Theorem
The language E_{TM} is undecidable.

Emptiness problem for TMs is undecidable

Theorem

The language E_{TM} is undecidable.

Corollary

The language E_{TM} is not RE.

Emptiness problem for TMs is undecidable

Theorem

The language E_{TM} is undecidable.

Corollary

The language E_{TM} is not RE.

Proof of the corollary.

Since E_{TM} is coRE, if it were RE, then it would be decidable, contradicting the theorem. □

Proof idea for showing E_{TM} is undecidable

- Assume E decides E_{TM}
- Build a decider for A_{TM} using E
- Along the way, we're going to construct an entirely new TM M_w and we're going to run E on $\langle M_w \rangle$

We'll use the idea of constructing new TMs in a bunch of different proofs

Proof

Proof.

Assume that E decides E_{TM}. Build D to decide A_{TM}.

D = "On input $\langle M, w\rangle$,

1. Construct a new TM M_w = 'On any input x,
   1. Replace x on the tape with w and run M on w.
   2. If M accepts, *accept*; if M rejects, *reject*.'
2. Run E on $\langle M_w\rangle$.
3. If E accepts, *reject*; otherwise *accept*."

Note that M_w is never run. It is only constructed so that $\langle M_w\rangle$ can be given as input to decider E.

Proof

Proof.

Assume that E decides E_{TM}. Build D to decide A_{TM}.
D = "On input $\langle M, w\rangle$,

1. Construct a new TM M_w = 'On any input x,
    1. Replace x on the tape with w and run M on w.
    2. If M accepts, *accept*; if M rejects, *reject*.'
2. Run E on $\langle M_w\rangle$.
3. If E accepts, *reject*; otherwise *accept*."

Note that M_w is never run. It is only constructed so that $\langle M_w\rangle$ can be given as input to decider E.

If $w \in L(M)$, then $L(M_w) = \Sigma^* \neq \emptyset$ so E rejects and D accepts.

Proof

Proof.

Assume that E decides E_{TM}. Build D to decide A_{TM}.
D = "On input $\langle M, w\rangle$,

1. Construct a new TM M_w = 'On any input x,
   1. Replace x on the tape with w and run M on w.
   2. If M accepts, *accept*; if M rejects, *reject*.'
2. Run E on $\langle M_w\rangle$.
3. If E accepts, *reject*; otherwise *accept*."

Note that M_w is never run. It is only constructed so that $\langle M_w\rangle$ can be given as input to decider E.

If $w \in L(M)$, then $L(M_w) = \Sigma^* \neq \varnothing$ so E rejects and D accepts.

If $w \notin L(M)$, then $L(M_w) = \varnothing$ so E accepts and D rejects. Thus $L(D) = E_{\mathsf{TM}}$.

Proof

Proof.

Assume that E decides E_{TM}. Build D to decide A_{TM}.

D = "On input $\langle M, w \rangle$,

1. Construct a new TM M_w = 'On any input x,
   1. Replace x on the tape with w and run M on w.
   2. If M accepts, *accept*; if M rejects, *reject*.'
2. Run E on $\langle M_w \rangle$.
3. If E accepts, *reject*; otherwise *accept*."

Note that M_w is never run. It is only constructed so that $\langle M_w \rangle$ can be given as input to decider E.

If $w \in L(M)$, then $L(M_w) = \Sigma^* \neq \varnothing$ so E rejects and D accepts.

If $w \notin L(M)$, then $L(M_w) = \varnothing$ so E accepts and D rejects. Thus $L(D) = E_{\mathsf{TM}}$.

Constructing M_w can't loop and E is a decider so D is a decider. □