Countable sets

Consider the set of even numbers $E = \{0, 2, 4, 6, \ldots\}$.
Are there fewer or greater elements than in the set of natural numbers?

Countable sets

Consider the set of even numbers $E = \{0, 2, 4, 6, \ldots\}$.
Are there fewer or greater elements than in the set of natural numbers?

If a function is both one-to-one and onto, then we say it is *bijective*, or a *correspondence*.

Countable sets

Consider the set of even numbers $E = \{0, 2, 4, 6, \ldots\}$.
Are there fewer or greater elements than in the set of natural numbers?

If a function is both one-to-one and onto, then we say it is *bijective*, or a *correspondence*.

If a set S has a correspondence with the natural numbers, i.e. $f : \mathcal{N} \rightarrow S$, we say that the set is *countable*.

Countable sets

Consider the set of even numbers $E = \{0, 2, 4, 6, \ldots\}$.
Are there fewer or greater elements than in the set of natural numbers?

If a function is both one-to-one and onto, then we say it is *bijective*, or a *correspondence*.

If a set S has a correspondence with the natural numbers, i.e. $f : \mathcal{N} \to S$, we say that the set is *countable*.

$f(a) = 2a$ is a correspondence, $f : \mathcal{N} \to E$.

Countable sets

Consider the set of even numbers $E = \{0, 2, 4, 6, \ldots\}$.
Are there fewer or greater elements than in the set of natural numbers?

If a function is both one-to-one and onto, then we say it is *bijective*, or a *correspondence*.

If a set S has a correspondence with the natural numbers, i.e. $f : \mathcal{N} \rightarrow S$, we say that the set is *countable*.

$f(a) = 2a$ is a correspondence, $f : \mathcal{N} \rightarrow E$.

Any subset of $\mathcal{N}$ is countable: sort the subset, and map the ith number in $\mathcal{N}$ to the ith element in the sorting.

The set of all TMs is countable! Each one can be encoded as a unique integer.
Sort the TM descriptions, and map from the naturals.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
    use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
    The ith bit is 1 iff the string corresponding to integer i is in the language.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
  use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
  The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
  use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
  The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
 use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
 The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \rightarrow \mathcal{L}$.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
    use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
    The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
    use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
    The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000\cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
  use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
  The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.
Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$
$\bar{\omega}$ does not appear in this sorted list:

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
$[j \in \mathcal{N}]$

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
  use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
  The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
  $[j \in \mathcal{N}]$
    $[\bar{\omega}$ is the jth item in the list $]$

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
$[j \in \mathcal{N}]$
$[\bar{\omega} \text{ is the } j\text{th item in the list }]$
$\bar{\omega}$ differs from ω_j in the jth bit

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
$[j \in \mathcal{N}]$
$[\bar{\omega}$ is the jth item in the list $]$
$\bar{\omega}$ differs from ω_j in the jth bit
False

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \rightarrow \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
$[j \in \mathcal{N}]$
$[\bar{\omega}$ is the jth item in the list $]$
$\bar{\omega}$ differs from ω_j in the jth bit
False
$\bar{\omega}$ is not the jth item in the list.

The set of all languages is uncountable

ω-string is a string of infinite length over $\{0, 1\}$.
A finite length string can be regarded as a unique integer:
  use a binary representation of the same string.
A language can be seen as a set of integers.
Represent a language using an ω-string:
  The ith bit is 1 iff the string corresponding to integer i is in the language.
Example: for $L = \{1, 11, 111\}$, $\omega = 10100010000000 \cdots$

Claim

Let $\mathcal{L}$ be the set of all languages. $\mathcal{L}$ is uncountable

Suppose $\mathcal{L}$ were countable.
Then there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.
Sort the elements of $\mathcal{L}$ according to the correspondence.
Let ω_i by the ω-string representing the ith language in the sorted list.

Define $\bar{\omega}$ as follows. The ith bit of $\bar{\omega} = \begin{cases} 0 & \text{if the } i\text{th bit of } \omega_i = 1 \\ 1, & \text{if the } i\text{th bit of } \omega_i = 0 \end{cases}$

$\bar{\omega}$ does not appear in this sorted list:
  $[j \in \mathcal{N}]$
    $[\bar{\omega}$ is the jth item in the list $]$
    $\bar{\omega}$ differs from ω_j in the jth bit
    False
  $\bar{\omega}$ is not the jth item in the list.
$\forall j \in \mathcal{N} : \bar{\omega}$ is not the jth item in the list.

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

$[\forall L \in \mathcal{L} : L$ is recognized by some Turing Machine$]$

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

$[\forall L \in \mathcal{L} : L$ is recognized by some Turing Machine$]$

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

$[\forall L \in \mathcal{L} : L$ is recognized by some Turing Machine$]$

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Sort the resulting list of integers.

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

$[\forall L \in \mathcal{L} : L$ is recognized by some Turing Machine$]$

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Sort the resulting list of integers.

This yields a correspondence between $\mathcal{N} \to \mathcal{L}$

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

$[\forall L \in \mathcal{L} : L$ is recognized by some Turing Machine$]$

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Sort the resulting list of integers.

This yields a correspondence between $\mathcal{N} \rightarrow \mathcal{L}$

$\mathcal{L}$ is countable.

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

[$\forall L \in \mathcal{L}$: L is recognized by some Turing Machine]

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Sort the resulting list of integers.

This yields a correspondence between $\mathcal{N} \rightarrow \mathcal{L}$

$\mathcal{L}$ is countable.

False

Some languages are not recognizable

Theorem

There exists a language $L \in \mathcal{L}$ that is not recognized by any Turing Machine.

Suppose otherwise, towards a contradiction:

[$\forall L \in \mathcal{L}$: L is recognized by some Turing Machine]

Assign to each $L \in \mathcal{L}$ the smallest integer corresponding to a TM that recognizes it.

Sort the resulting list of integers.

This yields a correspondence between $\mathcal{N} \to \mathcal{L}$

$\mathcal{L}$ is countable.

False

$\exists L \in \mathcal{L}$: L is not recognized by any TM.

An unrecognizable language: Notation

Recall, every TM can be described using an integer:
Write the transition function out as a binary string.
Interpret this binary string as an integer.

An unrecognizable language: Notation

Recall, every TM can be described using an integer:
Write the transition function out as a binary string.
Interpret this binary string as an integer.

Does every integer represent a valid TM? No!
We can write it in binary, but it might not correctly encode δ.

An unrecognizable language: Notation

Recall, every TM can be described using an integer:
Write the transition function out as a binary string.
Interpret this binary string as an integer.

Does every integer represent a valid TM? No!
We can write it in binary, but it might not correctly encode δ.

Nevertheless, we will consider every integer as representing a TM.
If it does not correctly encode a TM, we will say the language of that TM is $\emptyset$.

An unrecognizable language: Notation

Recall, every TM can be described using an integer:
Write the transition function out as a binary string.
Interpret this binary string as an integer.

Does every integer represent a valid TM? No!
We can write it in binary, but it might not correctly encode δ.

Nevertheless, we will consider every integer as representing a TM.
If it does not correctly encode a TM, we will say the language of that TM is $\emptyset$.

Bin(i) denotes the binary representation of $i \in \mathcal{N}$.
M_i is the TM described by Bin(i).

An unrecognizable language: Notation

Recall, every TM can be described using an integer:
Write the transition function out as a binary string.
Interpret this binary string as an integer.

Does every integer represent a valid TM? No!
We can write it in binary, but it might not correctly encode δ.

Nevertheless, we will consider every integer as representing a TM.
If it does not correctly encode a TM, we will say the language of that TM is $\emptyset$.

Bin(i) denotes the binary representation of $i \in \mathcal{N}$.
M_i is the TM described by Bin(i).
Note: $\langle M_i \rangle =$ Bin(i).
Sometimes we want to refer to the string representing machine M without knowing i.
Sometimes we want to think of the set of all $i \in \mathcal{N}$ and the machines they represent.

An unrecognizable language

$$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$$

An unrecognizable language

$$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$$

Theorem

L_D is not recognizable.

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

An unrecognizable language

$L_D = \{\text{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \text{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$$[\exists M : M \text{ recognizes } L_D]$$

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$
$\quad [M \text{ accepts } \langle M \rangle]$

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$
  $[M \text{ accepts } \langle M \rangle]$
  $\langle M \rangle \in L_D$ (By definition of "accepts")

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$	
$\quad [M \text{ accepts } \langle M \rangle]$	
$\quad \langle M \rangle \in L_D$	(By definition of "accepts")
$\quad M \text{ does not accept } \langle M \rangle$	(By definition of L_D)

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$
  $[M \text{ accepts } \langle M \rangle]$
  $\langle M \rangle \in L_D$ (By definition of "accepts")
  $M \text{ does not accept } \langle M \rangle$ (By definition of L_D)
  False

An unrecognizable language

$L_D = \{\text{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \text{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$	
$\quad [M \text{ accepts } \langle M \rangle]$	
$\quad \langle M \rangle \in L_D$	(By definition of "accepts")
$\quad M \text{ does not accept } \langle M \rangle$	(By definition of L_D)
$\quad$ False	
$M \text{ does not accept } \langle M \rangle$	

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$

- $[M \text{ accepts } \langle M \rangle]$
- $\langle M \rangle \in L_D$ (By definition of "accepts")
- $M \text{ does not accept } \langle M \rangle$ (By definition of L_D)
- False

$M \text{ does not accept } \langle M \rangle$

$\langle M \rangle \notin L_D$ (By Definition of "not accept")

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$	
$\quad [M \text{ accepts } \langle M \rangle]$	
$\quad \langle M \rangle \in L_D$	(By definition of "accepts")
$\quad M \text{ does not accept } \langle M \rangle$	(By definition of L_D)
$\quad$ False	
$M \text{ does not accept } \langle M \rangle$	
$\langle M \rangle \notin L_D$	(By Definition of "not accept")
$M \text{ accepts } \langle M \rangle$	(By Definition of L_D)

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \wedge M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$	
$\quad [M \text{ accepts } \langle M \rangle]$	
$\quad \langle M \rangle \in L_D$	(By definition of "accepts")
$\quad M \text{ does not accept } \langle M \rangle$	(By definition of L_D)
$\quad$ False	
$M \text{ does not accept } \langle M \rangle$	
$\langle M \rangle \notin L_D$	(By Definition of "not accept")
$M \text{ accepts } \langle M \rangle$	(By Definition of L_D)
False	

An unrecognizable language

$L_D = \{\mathsf{Bin}(i) \mid i \in \mathcal{N} \land M_i \text{ does not accept } \mathsf{Bin}(i)\}$

Theorem

L_D is not recognizable.

Suppose M recognizes L_D. Consider whether M accepts $x = \langle M \rangle$
If it does, then $x \in L_D$, because M should only accept strings in the language.
But, if M accepts $x = \langle M \rangle$, then, by the definition of L_D, x is NOT in the language!

$[\exists M : M \text{ recognizes } L_D]$	
$\quad [M \text{ accepts } \langle M \rangle]$	
$\quad \langle M \rangle \in L_D$	(By definition of "accepts")
$\quad M \text{ does not accept } \langle M \rangle$	(By definition of L_D)
$\quad$ False	
$M \text{ does not accept } \langle M \rangle$	
$\langle M \rangle \notin L_D$	(By Definition of "not accept")
$M \text{ accepts } \langle M \rangle$	(By Definition of L_D)
False	

$\neg \exists M : M \text{ recognizes } L_D$

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.
$[M^*(\langle M^* \rangle) = 1]$

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U\big(\langle M \rangle 0 \langle M \rangle\big)$ and flips the output bit.
$[M^*\big(\langle M^* \rangle\big) = 1]$
$M_U\big(\langle M^* \rangle 0 \langle M^* \rangle\big) = 0$ (By definition of M^*)

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.

$[M^*(\langle M^* \rangle) = 1]$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 0$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 0$ (By definition of M_U)

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M\rangle$, runs $M_U(\langle M\rangle 0\langle M\rangle)$ and flips the output bit.

$[M^*(\langle M^*\rangle) = 1]$
$M_U(\langle M^*\rangle 0\langle M^*\rangle) = 0$ (By definition of M^*)
$M^*(\langle M^*\rangle) = 0$ (By definition of M_U)
False

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.

$[M^*(\langle M^* \rangle) = 1]$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 0$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 0$ (By definition of M_U)
False

$M^*(\langle M^* \rangle) = 0$

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M\rangle$, runs $M_U(\langle M\rangle 0\langle M\rangle)$ and flips the output bit.

$[M^*(\langle M^*\rangle) = 1]$
$M_U(\langle M^*\rangle 0\langle M^*\rangle) = 0$ (By definition of M^*)
$M^*(\langle M^*\rangle) = 0$ (By definition of M_U)
False

$M^*(\langle M^*\rangle) = 0$
$M_U(\langle M^*\rangle 0\langle M^*\rangle) = 1$ (By definition of M^*)

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M\rangle$, runs $M_U(\langle M\rangle 0\langle M\rangle)$ and flips the output bit.

$[M^*(\langle M^*\rangle) = 1]$
$M_U(\langle M^*\rangle 0\langle M^*\rangle) = 0$ (By definition of M^*)
$M^*(\langle M^*\rangle) = 0$ (By definition of M_U)
False

$M^*(\langle M^*\rangle) = 0$
$M_U(\langle M^*\rangle 0\langle M^*\rangle) = 1$ (By definition of M^*)
$M^*(\langle M^*\rangle) = 1$ (By definition of M_U)

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.

$[M^*(\langle M^* \rangle) = 1]$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 0$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 0$ (By definition of M_U)
False

$M^*(\langle M^* \rangle) = 0$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 1$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 1$ (By definition of M_U)
False

An undecidable language

Recall: We can build a universal TM that *recognizes* the following language:
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

Theorem

L_U is not decidable

$[\exists M_U : M_U \text{ decides } L_U]$
Define M^*, which on input $\langle M \rangle$, runs $M_U(\langle M \rangle 0 \langle M \rangle)$ and flips the output bit.

$[M^*(\langle M^* \rangle) = 1]$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 0$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 0$ (By definition of M_U)
False

$M^*(\langle M^* \rangle) = 0$
$M_U(\langle M^* \rangle 0 \langle M^* \rangle) = 1$ (By definition of M^*)
$M^*(\langle M^* \rangle) = 1$ (By definition of M_U)
False

$\neg \exists M_U : M_U \text{ decides } L_U$

An undecidable language

Behavior of M_U, if it were to exist:

$$\begin{pmatrix} & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \dots & \dots \\ M_1 : & \underline{accept} & accept & reject & reject & \dots & accept \\ M_2 : & accept & \underline{reject} & reject & reject & \dots & reject \\ M_3 : & reject & accept & \underline{reject} & accept & \dots & accept \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & reject \\ M^* : & reject & accept & accept & reject & \dots & \underline{???} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Reducing one computation to another

Consider the language $L_{\mathsf{halt}} = \{\langle M\rangle 0x \mid M(x) \text{ terminates } \}$

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

Reducing one computation to another

Consider the language $L_{\mathsf{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M\rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$
$\neg\exists M_U : M_U \text{ decides } L_U$

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.

$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

$\neg\exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$
$\neg \exists M_U : M_U \text{ decides } L_U$
$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$
Define M_U
$\underline{M_U(\langle M \rangle 0x) :}$

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M\rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$
$\neg\exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$
Define M_U
$M_U(\langle M\rangle 0x)$:

1. Simulate $M_{\text{halt}}(\langle M\rangle 0x)$.

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.

$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

$\neg \exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$

Define M_U

$\underline{M_U(\langle M \rangle 0x) :}$

1. Simulate $M_{\text{halt}}(\langle M \rangle 0x)$.
2. If it outputs 0, halt and output 0.

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.

$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

$\neg \exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$

Define M_U

$\underline{M_U(\langle M \rangle 0x) :}$

1. Simulate $M_{\text{halt}}(\langle M \rangle 0x)$.
2. If it outputs 0, halt and output 0.
3. If it outputs 1, simulate M on input x until it halts. Output whatever M outputs.

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M\rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.
$L_U = \{\langle M\rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$
$\neg\exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$
Define M_U
$\underline{M_U(\langle M\rangle 0x) :}$

1. Simulate $M_{\text{halt}}(\langle M\rangle 0x)$.
2. If it outputs 0, halt and output 0.
3. If it outputs 1, simulate M on input x until it halts. Output whatever M outputs.

$\exists M_U : M_U \text{ decides } L_U$

Reducing one computation to another

Consider the language $L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.
$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$
$\neg \exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}]$
Define M_U
$\underline{M_U(\langle M \rangle 0x) :}$

1. Simulate $M_{\text{halt}}(\langle M \rangle 0x)$.
2. If it outputs 0, halt and output 0.
3. If it outputs 1, simulate M on input x until it halts. Output whatever M outputs.

$\exists M_U : M_U \text{ decides } L_U$
False

Reducing one computation to another

Consider the language $L_{\mathsf{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

Theorem

The language L_{halt} is undecidable.

We reduce the problem of deciding L_U to the problem of deciding L_{halt}.

$L_U = \{\langle M \rangle 0x \mid \text{ TM } M \text{ accepts input } x\}$

$\neg \exists M_U : M_U \text{ decides } L_U$

$[\exists M_{\mathsf{halt}} : M_{\mathsf{halt}} \text{ decides } L_{\mathsf{halt}}]$

Define M_U

$\underline{M_U(\langle M \rangle 0x) :}$

1. Simulate $M_{\mathsf{halt}}(\langle M \rangle 0x)$.
2. If it outputs 0, halt and output 0.
3. If it outputs 1, simulate M on input x until it halts. Output whatever M outputs.

$\exists M_U : M_U \text{ decides } L_U$

False

$\neg \exists M_{\mathsf{halt}} : M_{\mathsf{halt}} \text{ decides } L_{\mathsf{halt}}$

Reducing one computation to another

Consider the language $L_{\emptyset} = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\mathsf{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg \exists M_{\mathsf{halt}} : M_{\mathsf{halt}} \text{ decides } L_{\mathsf{halt}}$
$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg \exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$

Define M_{halt}

$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\mathsf{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg \exists M_{\mathsf{halt}} : M_{\mathsf{halt}} \text{ decides } L_{\mathsf{halt}}$
$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\mathsf{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$

Define M_{halt}

$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.
   - On input $y = x$, run $M(y)$ and output whatever it outputs.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$
$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.
   - On input $y = x$, run $M(y)$ and output whatever it outputs.
2. Run $M_\emptyset(\langle M' \rangle)$. Reverse the value of its output.

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$
$\neg \exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.
   - On input $y = x$, run $M(y)$ and output whatever it outputs.
2. Run $M_\emptyset(\langle M' \rangle)$. Reverse the value of its output.

$M_{\text{halt}} \text{ decides } L_{\text{halt}}$

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M\rangle \mid M \text{ rejects all strings }\}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.
$L_{\text{halt}} = \{\langle M\rangle 0x \mid M(x) \text{ terminates }\}$
$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$
$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$
Define M_{halt}
$\underline{M_{\text{halt}}(\langle M\rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.
   - On input $y = x$, run $M(y)$ and output whatever it outputs.
2. Run $M_\emptyset(\langle M'\rangle)$. Reverse the value of its output.

$M_{\text{halt}} \text{ decides } L_{\text{halt}}$
False

Reducing one computation to another

Consider the language $L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

Theorem

The language $L_\emptyset$ is undecidable.

We reduce the problem of deciding L_{halt} to the problem of deciding $L_\emptyset$.

$L_{\text{halt}} = \{\langle M \rangle 0x \mid M(x) \text{ terminates } \}$

$\neg\exists M_{\text{halt}} : M_{\text{halt}} \text{ decides } L_{\text{halt}}$

$[\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset]$

Define M_{halt}

$\underline{M_{\text{halt}}(\langle M \rangle 0x) :}$

1. Write down a description of a TM M' that modifies the behavior of M as follows.
   $\underline{M' :}$
   - On input $y \neq x$, reject.
   - On input $y = x$, run $M(y)$ and output whatever it outputs.
2. Run $M_\emptyset(\langle M' \rangle)$. Reverse the value of its output.

$M_{\text{halt}} \text{ decides } L_{\text{halt}}$

False

$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Reducing one computation to another

Consider the language $L_{\text{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

Reducing one computation to another

Consider the language $L_{\text{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_{\emptyset}$ to the problem of deciding L_{EQ}.

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$
$\neg \exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings }\}$
$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$
$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.

$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$

$\neg \exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$

Define $M_\emptyset$

$\underline{M_\emptyset(\langle M \rangle) :}$

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$
$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$
Define $M_\emptyset$
$\underline{M_\emptyset(\langle M \rangle) :}$

1. Construct the description of a Turing machine M' that rejects all strings.

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1\rangle 0\langle M_2\rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M\rangle \mid M \text{ rejects all strings }\}$
$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$
$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$
Define $M_\emptyset$
$\underline{M_\emptyset(\langle M\rangle):}$

1. Construct the description of a Turing machine M' that rejects all strings.
2. Run $M_{\mathsf{EQ}}(\langle M\rangle, \langle M'\rangle)$, and output whatever it outputs.

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings } \}$
$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$
Define $M_\emptyset$
$\underline{M_\emptyset(\langle M \rangle) :}$

1. Construct the description of a Turing machine M' that rejects all strings.
2. Run $M_{\mathsf{EQ}}(\langle M \rangle, \langle M' \rangle)$, and output whatever it outputs.

$M_\emptyset \text{ decides } L_\emptyset$

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1 \rangle 0 \langle M_2 \rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.
$L_\emptyset = \{\langle M \rangle \mid M \text{ rejects all strings }\}$
$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$
$[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$
Define $M_\emptyset$
$\underline{M_\emptyset(\langle M \rangle) :}$

1. Construct the description of a Turing machine M' that rejects all strings.
2. Run $M_{\mathsf{EQ}}(\langle M \rangle, \langle M' \rangle)$, and output whatever it outputs.

$M_\emptyset \text{ decides } L_\emptyset$
False

Reducing one computation to another

Consider the language $L_{\mathsf{EQ}} = \{\langle M_1\rangle 0\langle M_2\rangle \mid L(M_1) = L(M_2)\}$

Theorem

The language L_{EQ} is undecidable.

We reduce the problem of deciding $L_\emptyset$ to the problem of deciding L_{EQ}.

$L_\emptyset = \{\langle M\rangle \mid M \text{ rejects all strings } \}$

$\neg\exists M_\emptyset : M_\emptyset \text{ decides } L_\emptyset$

  $[\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}]$

  Define $M_\emptyset$

  $\underline{M_\emptyset(\langle M\rangle):}$

  1. Construct the description of a Turing machine M' that rejects all strings.
  2. Run $M_{\mathsf{EQ}}(\langle M\rangle, \langle M'\rangle)$, and output whatever it outputs.

  $M_\emptyset \text{ decides } L_\emptyset$

  False

$\neg\exists M_{\mathsf{EQ}} : M_{\mathsf{EQ}} \text{ decides } L_{\mathsf{EQ}}$