جامعة
الملك سعود
King Saud University

Theory of Computation
CSC 339 – Spring 2021

Chapter-4: part2
Undecidability

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- Recap
- Introduction
- Undecidability

Recap

- **Decidability**
  - Language $A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$.

Recap

- **Decidability**
  - Language $A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$.
  - Language $A_{NFA} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$.

Recap

- **Decidability**
  - Language $A_{DFA} = \{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$.
  - Language $A_{NFA} = \{\langle B, w\rangle \mid B$ is an NFA that accepts input string $w\}$.
  - Language $A_{REX} = \{\langle R, w\rangle \mid R$ is a regular expression that generates string $w\}$.

Recap

- **Decidability**
  - Language $A_{DFA} = \{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$.
  - Language $A_{NFA} = \{\langle B, w\rangle \mid B$ is an NFA that accepts input string $w\}$.
  - Language $A_{REX} = \{\langle R, w\rangle \mid R$ is a regular expression that generates string $w\}$.

All of these are decidable languages.

Recap

- **Decidability**
  - Language A_{DFA} = {⟨B, *w*⟩| B is a DFA that accepts input string w}.
  - Language A_{NFA} = {⟨B, *w*⟩| B is an NFA that accepts input string w}.
  - Language A_{REX} = {⟨R, w⟩| R is a regular expression that generates string w}.

All of these are decidable languages. → There are TMs that decide these languages

Recap

- **Decidability**
  - Language A_{DFA} = {⟨B, *w*⟩| B is a DFA that accepts input string w}.
  - Language A_{NFA} = {⟨B, *w*⟩| B is an NFA that accepts input string w}.
  - Language A_{REX} = {⟨R, w⟩| R is a regular expression that generates string w}.

TM *M* that decides A_{DFA}

M = "On input ⟨B, w⟩, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, ***accept***. If it ends in a nonaccepting state, ***reject***."

Introduction

- **Computers are powerful in solving a wide range of problems.**

Introduction

- **Computers are powerful in solving a wide range of problems.**
- **But, computers do have some limits..**

Introduction

- **Computers are powerful in solving a wide range of problems.**
- **But, computers do have some limits..**
- **Some problems are unsolvable by computers, even though they may seem simple.**

Introduction

- **Computers are powerful in solving a wide range of problems.**

- **But, computers do have some limits..**

- **Some problems are unsolvable by computers, even though they may seem simple.**

Introduction

- **Computers are powerful in solving a wide range of problems.**

- **But, computers do have some limits..**

- **Some problems are unsolvable by computers, even though they may seem simple.**

- **Why do we need to study and prove undecidability?**

Undecidability

- **Undecidable languages**
  - **There aren’t TMs that can decide such languages on every input string.**

Undecidability

- **Undecidable languages**
  - **There aren’t TMs that can decide such languages on every input string.**
  - **The TMs could make decisions (halt) on some of the input strings, but cannot make such decisions on all input strings.**

Undecidability

- Undecidable languages
  - There aren’t TMs that can decide such languages on every input string.
  - The TMs could make decisions (halt) on some of the input strings, but cannot make such decisions on all input strings.
- Example of an undecidable problem
  - Membership problem

Undecidability

- **Undecidable languages**
  - **There aren't TMs that can decide such languages on every input string.**
  - **The TMs could make decisions (halt) on some of the input strings, but cannot make such decisions on all input strings.**
- **Example of an undecidable problem**
  - **Membership problem**
    - **Given a string w and a language L, we would like to check whether $w \in L$.**
    - **The outcome we wish to obtain is a *yes* or *no* answer.**

Undecidability: Membership Problem

- **Input:**
  - **Turing machine** M
  - **Input string** w

Undecidability: Membership Problem

- **Input:**
  - **Turing machine** M
  - **Input string** w
- **Does** M **accept** w**?**

Undecidability: Membership Problem

- **Input:**
  - **Turing machine** M
  - **Input string** w
- **Does** M **accept** w**?**

- **Corresponding language** $A_{TM} = \{\langle M,w \rangle$ M **is a Turing machine that accepts** $w\}$.

Undecidability: Membership Problem

- **Input:**
  - **Turing machine M**
  - **Input string w**
- **Does M accept w?**

- **Corresponding language $A_{TM} = \{\langle M,w \rangle\ M$ is a Turing machine that accepts $w\}$.**

- **A_{TM} is recognizable, but undecidable. Why?**

Undecidability: Membership Problem

TM U that recognizes A_{TM}

U = "On input ⟨M, *w*⟩, where M is a TM and *w* is a string:

1. Simulate M on input *w*.
2. If M enters its accept state, ***accept***. If it ends in a nonaccepting state, ***reject***."

Undecidability: Membership Problem

TM U that recognizes A_{TM}

U = "On input ⟨M, *w*⟩, where M is a TM and *w* is a string:

1. Simulate M on input *w*.
2. If M enters its accept state, ***accept***. If it ends in a nonaccepting state, ***reject***."

What happens if M does not halt?

Undecidability: Membership Problem

TM U that recognizes A_{TM}

U = "On input ⟨M, *w*⟩, where M is a TM and *w* is a string:

1. Simulate M on input *w*.
2. If M enters its accept state, ***accept***. If it ends in a nonaccepting state, ***reject***."

What happens if M does not halt?

U will not halt as well

Undecidability: Membership Problem

TM U that recognizes A_{TM}

U = "On input ⟨M, *w*⟩, where M is a TM and *w* is a string:

1. Simulate M on input *w*.
2. If M enters its accept state, ***accept***. If it ends in a nonaccepting state, ***reject***."

What happens if M does not halt?

U will not halt as well

U is not a decider

Undecidability: Examples

- **Undecidable problems**
  - **Does TM *M* accept any string?**

Undecidability: Examples

- **Undecidable problems**
  - **Does TM *M* accept any string?**
  - **Does TM *M* accept all strings?**

Undecidability: Examples

- **Undecidable problems**
  - **Does TM M accept any string?**
  - **Does TM M accept all strings?**
  - **Does TM_1 accept more strings than TM_2?**

Undecidability: Examples

- **Undecidable problems**
  - **Does TM M accept any string?**
  - **Does TM M accept all strings?**
  - **Does TM_1 accept more strings than TM_2?**
  - **Does TM_1 take more steps than TM_2 to process string w?**

Undecidability: Examples

- **Undecidable problems**
  - **Does TM *M* accept any string?**
  - **Does TM *M* accept all strings?**
  - **Does TM_1 accept more strings than TM_2?**
  - **Does TM_1 take more steps than TM_2 to process string *w*?**
- **Decidable problems**
  - **Does TM *M* take more than *100* steps to process string *w*?**

Undecidability: Proving Undecidability

- **Proving undecidability for a certain problem/language**

Undecidability: Proving Undecidability

- **Proving undecidability for a certain problem/language**
  - **Need to show that there is not a TM that decides the problem/language.**

Undecidability: Proving Undecidability

- **Proving undecidability for a certain problem/language**
  - **Need to show that there is not a TM that decides the problem/language.**
  - **This is difficult: can we examine all possible TMs for that problem?**

Undecidability: Diagonalization Method

- **We’re going to use the diagonalization method to prove undecidability.**

Undecidability: Diagonalization Method

- **We’re going to use the diagonalization method to prove undecidability.**

- **Basically, how can we measure and compare different infinite sets?**

Undecidability: Diagonalization Method

- **We're going to use the diagonalization method to prove undecidability.**

- **Basically, how can we measure and compare different infinite sets?**

  - **e.g., is the set of <u>even natural numbers</u> (*E*) larger than the set of <u>natural numbers</u> (*N*)?**

Undecidability: Diagonalization Method

- **We're going to use the diagonalization method to prove undecidability.**

- **Basically, how can we measure and compare different infinite sets?**

  - **e.g., is the set of <u>even natural numbers</u> (*E*) larger than the set of <u>natural numbers</u> (*N*)?**
  - **Both sets are infinite!**

Undecidability: Diagonalization Method

- **We're going to use the diagonalization method to prove undecidability.**
- **Basically, how can we measure and compare different infinite sets?**
  - **e.g., is the set of <u>even natural numbers</u> (*E*) larger than the set of <u>natural numbers</u> (*N*)?**
  - **Both sets are infinite!**
- **We can simply do this for finite sets.**
  - **Count the elements of each finite set.**

Undecidability: Diagonalization Method

- **Another approach for finite sets (e.g.,** A **and** B**) is to use a mapping function.**

Undecidability: Diagonalization Method

- **Another approach for finite sets (e.g., *A* and *B*) is to use a mapping function.**
  - **If we have a one-to-one mapping between elements of *A* and elements of *B*, then we can say *A* and *B* have the same size.**

Undecidability: Diagonalization Method

- **Another approach for finite sets (e.g., *A* and *B*) is to use a mapping function.**
  - **If we have a one-to-one mapping between elements of *A* and elements of *B*, then we can say *A* and *B* have the same size.**
  - **<u>One-to-one</u> mapping is when each element of *A* has a unique corresponding element of *B*, and vice versa.**

Undecidability: Diagonalization Method

- **Another approach for finite sets (e.g., A and B) is to use a mapping function.**
  - **If we have a one-to-one mapping between elements of A and elements of B, then we can say A and B have the same size.**
  - **<u>One-to-one</u> mapping is when each element of A has a unique corresponding element of B, and vice versa.**

Mapping Function

Assume $a \in A$ and $b \in B$

$f(a) = b$

And

$f(b) = a$

See definition 4.12 on p. 203

Undecidability: Diagonalization Method

- **Another approach for finite sets (e.g., A and B) is to use a mapping function.**
  - **If we have a one-to-one mapping between elements of A and elements of B, then we can say A and B have the same size.**
  - **<u>One-to-one</u> mapping is when each element of A has a unique corresponding element of B, and vice versa.**

Mapping Function

Assume $a \in A$ and $b \in B$
$f(a) = b$
And
$f(b) = a$

We can use this 1-to-1 mapping for infinite sets as well.

See definition 4.12 on p. 203

Undecidability: Diagonalization Method

- **Countable sets**

A set A is ***countable*** if either:
it is *finite*
or
it has the same size as N.

Undecidability: Diagonalization Method

- **Countable sets**
- **All finite sets are countable**
- **Infinite sets could be countable or uncountable**

A set A is ***countable*** if either:
it is *finite*
or
it has the same size as N.

Undecidability: Diagonalization Method

- **Countable sets**

- **All finite sets are countable**

- **Infinite sets could be countable or uncountable**

    - **We say an infinite set is countable if it has a 1-to-1 mapping with elements of the set of natural numbers (*N*).**

A set A is ***countable*** if either:
it is *finite*
or
it has the same size as *N*.

Undecidability: Diagonalization Method

- **Countable sets**
- **All finite sets are countable**
- **Infinite sets could be countable or uncountable**
  - **We say an infinite set is countable if it has a 1-to-1 mapping with elements of the set of natural numbers (*N*).**

A set A is ***countable*** if either:
it is *finite*
or
it has the same size as *N*.

Set of even integers (*E*)	0, 2, 4, 6, 8, 10, …
	↓ ↓ ↓ ↓ ↓ ↓
Set of natural numbers (*N*)	1, 2, 3, 4, 5, 6, …

Undecidability: Diagonalization Method

- **Countable sets**
- **All finite sets are countable**
- **Infinite sets could be countable or uncountable**
  - **We say an infinite set is countable if it has a 1-to-1 mapping with elements of the set of natural numbers (*N*).**

A set A is ***countable*** if either:
it is *finite*
or
it has the same size as *N*.

(*E*) is countable because it has a 1-to-1 mapping with elements from *N*

Set of even integers (*E*)	0,	2,	4,	6,	8,	10, …
	↓	↓	↓	↓	↓	↓
Set of natural numbers (*N*)	1,	2,	3,	4,	5,	6, …

Undecidability: Diagonalization Method

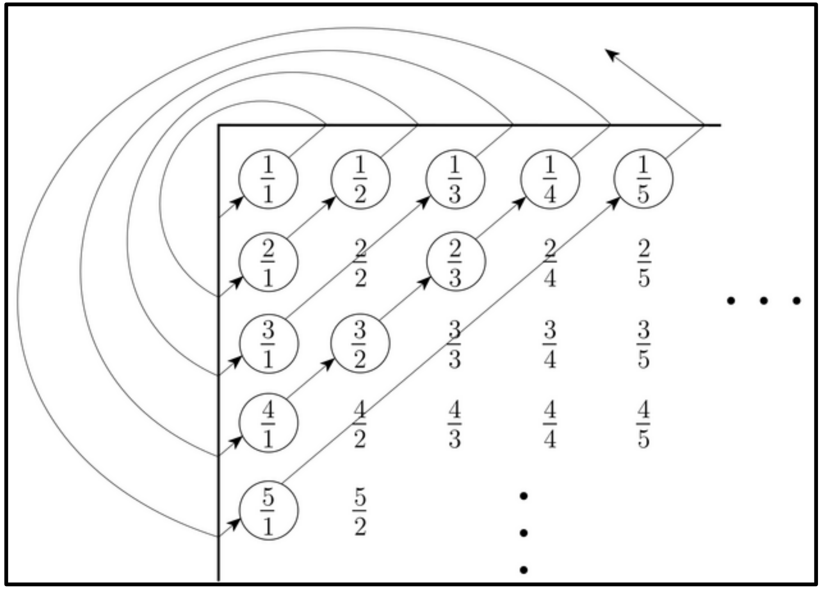
- **Is the set of positive rational numbers (*Q*) the same size as the set of natural numbers (*N*)?**

Undecidability: Diagonalization Method

- **Is the set of positive rational numbers (Q) the same size as the set of natural numbers (N)?**
  - **Positive rational numbers**
    - $Q = \{\, m/n \mid m, n \in N \,\}$

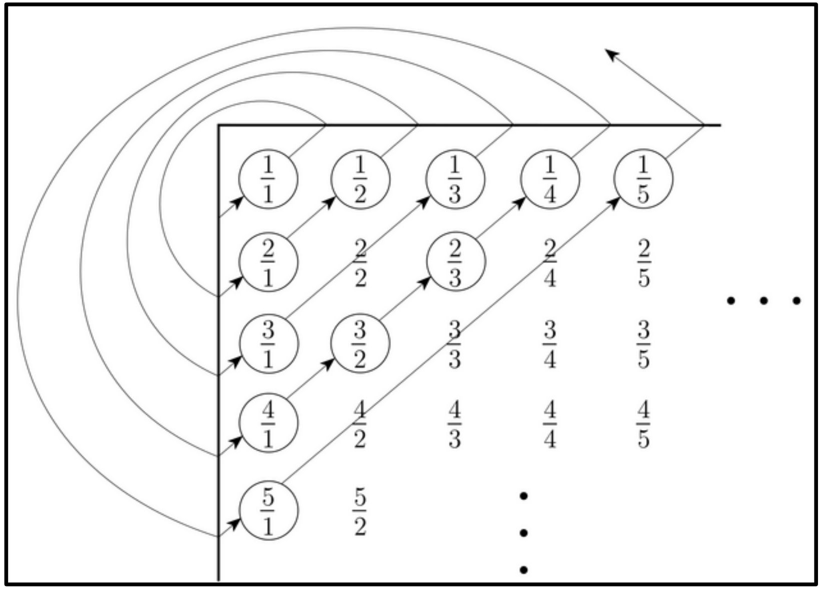
Undecidability: Diagonalization Method

- **Is the set of positive rational numbers (Q) the same size as the set of natural numbers (N)?**
  - **Positive rational numbers**
    - $Q = \{\ m/n \mid m, n \in N\ \}$
  - **If we could find a way to map each element in N to each element in Q, then we can say the two sets are the same size.**



Undecidability: Diagonalization Method

- **There can be cases where**
  - $f(x) = m/n$, for $x, m, n \in N$ and $m/n \in Q$
  - $f(y) = 2m/2n$ for $x \in N$

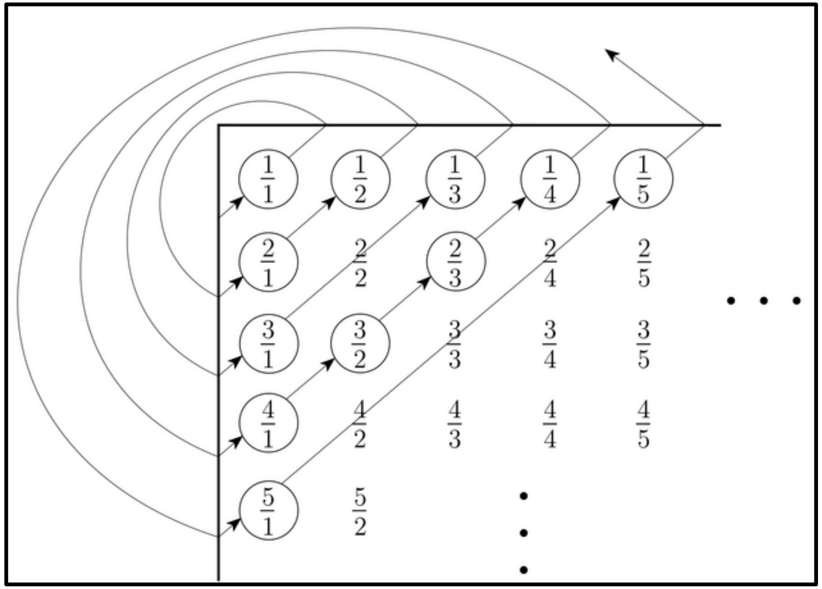


Undecidability: Diagonalization Method

- **There can be cases where**
  - $f(x) = m/n$, for $x,m,n \in N$ and $m/n \in Q$
  - $f(y) = 2m/2n$ for $x \in N$

They are basically
the same number

For example: 3/2 and 6/4

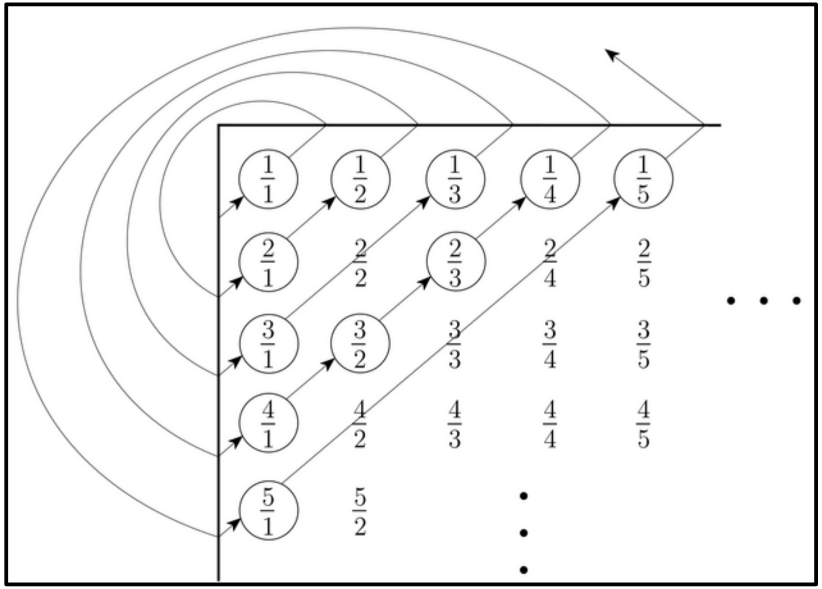


Undecidability: Diagonalization Method

- **There can be cases where**
  - f(x) = m/n, for x,m,n ∈ N and m/n ∈ Q
  - f(y) = 2m/2n for x ∈ N

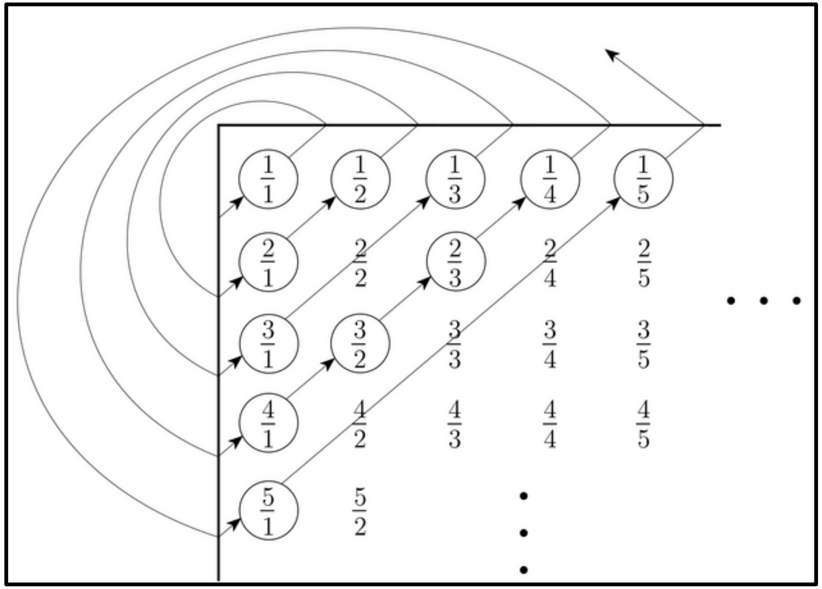
They are basically
the same number

But, they have different mappings.
This is not one-to-one mapping!



Undecidability: Diagonalization Method

- **There can be cases where**
  - $f(x) = m/n$, for $x,m,n \in N$ and $m/n \in Q$
  - $f(y) = 2m/2n$ for $x \in N$
- **To avoid this, we include numbers in *Q* in their most basic form.**
  - 1/2 is the same as 2/4, 3/6, and so on.
  - So, we only include 1/2



Undecidability: Diagonalization Method

- **Is the set of real numbers (R) the same size as the set of natural numbers (N)?**

Undecidability: Diagonalization Method

- **Is the set of real numbers (R) the same size as the set of natural numbers (N)?**
  - **Real numbers are those with a decimal representation.**
    - **π = 3.1415926 …**
    - **$\sqrt{2}$ = 1.4142135 …**

Undecidability: Diagonalization Method

- **Is the set of real numbers (R) the same size as the set of natural numbers (N)?**
  - **Real numbers are those with a decimal representation.**
    - **π = 3.1415926 …**
    - **$\sqrt{2}$ = 1.4142135 …**

R is uncountable

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (x) in R that is not paired with anything in N.**
    - **Then, we can say R is uncountable**

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
$\vdots$	$\vdots$

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

numbers from *N*

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
$\vdots$	$\vdots$

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

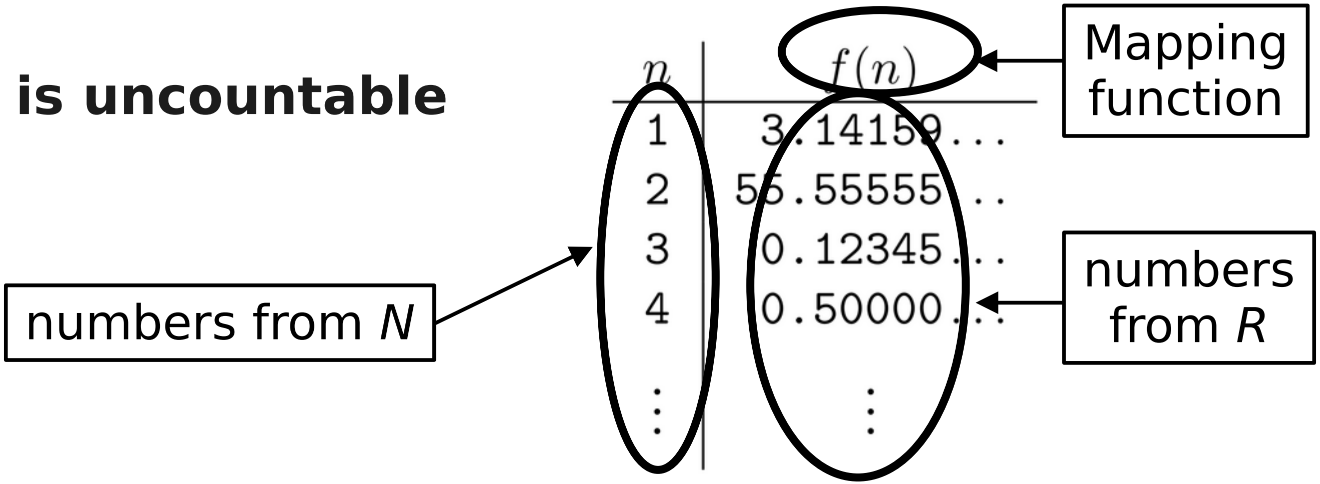
Mapping function

n	$f(n)$
1	$3.14159\ldots$
2	$55.55555\ldots$
3	$0.12345\ldots$
4	$0.50000\ldots$
$\vdots$	$\vdots$

numbers from *N*

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**



Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

<u>Construct x:</u>

To ensure *x* is unique select the i^{th} decimal (+1) from each i^{th} element

$X = 0.2$

n	$f(n)$
1	$3.\boxed{1}4159\ldots$
2	$55.55555\ldots$
3	$0.12345\ldots$
4	$0.50000\ldots$
$\vdots$	$\vdots$

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

n	$f(n)$
1	3.14159...
2	55.$\boxed{5}$5555...
3	0.12345...
4	0.50000...
$\vdots$	$\vdots$

<u>Construct x:</u>

To ensure *x* is unique select the i^{th} decimal (+1) from each i^{th} element

$X = 0.26$

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12$\boxed{3}$45...
4	0.50000...
$\vdots$	$\vdots$

<u>Construct x:</u>

To ensure *x* is unique select the i^{th} decimal (+1) from each i^{th} element

$X = 0.264$

Undecidability: Diagonalization Method

- **Proof that R is uncountable**
  - **We must show that the mapping function does not work as it should.**
    - **If we could find a number (*x*) in *R* that is not paired with anything in *N*.**
    - **Then, we can say R is uncountable**

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.500$\boxed{0}$0...
$\vdots$	$\vdots$

<u>Construct x:</u>

To ensure x is unique select the i^{th} decimal (+1) from each i^{th} element

$X = 0.2641$

Undecidability: Diagonalization Method

- Proof that R is uncountable
  - We must show that the mapping function does not work as it should.
    - If
      an
    - Th

Constru

Continuing this way, we observe that we can keep constructing different x's that won't have a mapping to a number in N

To ensure x is unique select the i^{th} decimal (+1) from each i^{th} element

$X = 0.2641$

3	0.12345...
4	0.500$\boxed{0}$0...
⋮	⋮

Non-recognizable Languages

- **Some languages are not Turing-recognizable**
- **The set of ALL Turing machines is countable.**
  - **Each TM has an encoding into a string.**
  - **Omitting those strings that are not legal encoding of TMs, we can obtain a list of all TMs.**
  - **The set of ALL Turing machines has the same size as N (set of natural numbers).**
- **The set of ALL languages is <u>un</u>countable.**
  - **It has a correspondence with the set of all infinite binary sequences (which is uncountable)**

Undecidable Languages

- $A_{TM} = \{\langle M, w\rangle \mid M$ is a TM and M accepts w$\}$
- ~~A_{TM} is the language of strings accepted by all Turing-machines~~.
- A_{TM} is the language of strings accepted by any TM.
- We need to prove that A_{TM} is not undecidable.
- Use a proof by contradiction
  - Assume A_{TM} is decidable and obtain a contradiction.

Undecidable Languages

- First, let’s establish some facts
  - The set of all Turing machines is countable
    - Each TM can be described as a string encoding
    - We can certainly enumerate (count) all those strings
    - Therefore, we can enumerate (count) the set of all TMs.

Table ***T*** describes the behavior of all enumerated TMs

T	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***T*** describes the behavior of all enumerated TMs

Enumeration of all TMs

T	<M$_1$>	<M$_2$>	…	<M$_k$>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***T*** describes the behavior of all enumerated TMs

String description of TMs

This is the input to each TM

Enumeration of all TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***T*** describes the behavior of all enumerated TMs

String description of TMs

This is the input to each TM

Enumeration of all TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Outcome of machine M_2 on input $<M_k>$

Table ***T*** describes the behavior of all enumerated TMs

String description of TMs

This is the input to each TM

Enumeration of all TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Outcome of machine M_2 on input $<M_k>$

Three possible outcomes:

1- Accept

2- Reject

3- Doesn't halt (loop)

Table ***T*** describes the behavior of all enumerated TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Now, let's assume $\boldsymbol{A_{TM}}$ is decidable. And suppose TM ***H*** is a decider for $\boldsymbol{A_{TM}}$.

Table ***T*** describes the behavior of all enumerated TMs

T	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Now, let's assume $\boldsymbol{A_{TM}}$ is decidable. And suppose TM ***H*** is a decider for $\boldsymbol{A_{TM}}$.

Goal is to have a contradiction.

Table $\boldsymbol{T}$ describes the behavior of all enumerated TMs

$\boldsymbol{T}$	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***T*** describes the behavior of all enumerated TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1					
M_2					
…					
M_k					
…					

Table ***T*** describes the behavior of all enumerated TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1					
M_2					
…					
M_k					
…					

Table ***T*** describes the behavior of all enumerated TMs

T

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject				
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Corresponds to first component of the input

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle \rangle$

H

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1					
M_2					
…					
M_k					
…					

Table ***T*** describes the behavior of all enumerated TMs

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject				
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Corresponds to second component of the input

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1					
M_2					
…					
M_k					
…					

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form <M_i, <M_j>>

T	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…			
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H simulates each TM M_i on input <M_j>

H	<M_1>	<M_2>	…	<M_k>	…
M_1					
M_2					
…					
M_k					
…					

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept		reject	
…	…				
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Since ***H*** is a decider for ***A_{TM}***, it has to always halt and decide!

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form <M_i, <M_j>>

T	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept		reject	
…	…				
M_k	accept	doesn't halt	…	reject	…
…	…				

H	<M_1>	<M_2>	…	<M_k>	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Since ***H*** is a decider for ***A_{TM}***, it has to always halt and decide!

Whenever A given TM doesn't halt, ***H*** should ***reject***.

Table ***T*** describes the behavior of all enumerated TMs

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle \rangle$

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

H is the same as ***T***, except ***H*** says ***reject*** whenever ***T*** says ***doesn't halt***

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…			

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
				…	…

Assuming that ***H*** exists, we can describe another TM ***D*** that takes the description of any TM as input $<M_j>$

What ***D*** does

1) Run H as a subroutine
2) Reverse the outcome of H

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$<M_1>$	$<M_2>$	…	$<M_k>$	…
D					

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
			…	…	…

The values for ***D***'s table will be the reverse values of the ones in the diagonal of ***H***'s table

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	…	$\langle M_k \rangle$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…				
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

If ***H*** exists, then ***D*** certainly exists

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

D	$<M_1>$	$<M_2>$	…	$<M_k>$	…
	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	…	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
…	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

If ***D*** exists, then it must be in ***T***

Recall, ***T*** describes behavior of all TMs

	$<M_1>$	$<M_2>$	…	$<M_k>$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	…	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle \rangle$

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	***D***	$\langle M_k \rangle$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle \rangle$

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	…	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	***D***	$\langle M_k \rangle$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

T	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle$***D***$\rangle$	$\langle M_k\rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle$***D***$\rangle$	$\langle M_k\rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$\langle M_1\rangle$	$\langle M_2\rangle$	***D***	$\langle M_k\rangle$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form <M_i, <M_j>>

T	<M_1>	<M_2>	<***D***>	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	<M_1>	<M_2>	<***D***>	<M_k>	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	<M_1>	<M_2>	***D***	<M_k>	…
D	reject	reject	**reject**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

T	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

H	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**accept**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Should be the same, but it's not

	$<M_1>$	$<M_2>$	***D***	$<M_k>$	…
D	reject	reject	**reject**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$<M_1>$	$<M_2>$	***D***	$<M_k>$	…
D	reject	reject	…	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	$<\boldsymbol{D}>$	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	$<\boldsymbol{D}>$	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$<M_1>$	$<M_2>$	***D***	$<M_k>$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

T	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle \boldsymbol{D}\rangle$	$\langle M_k\rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle \boldsymbol{D}\rangle$	$\langle M_k\rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Should be the same, but it's not

	$\langle M_1\rangle$	$\langle M_2\rangle$	***D***	$\langle M_k\rangle$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Should be the same, but it's not

Same thing for ***"doesn't halt"***

	$<M_1>$	$<M_2>$	***D***	$<M_k>$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn’t halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn’t halt	…	reject	…
…	…	…	…	…	…

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle \boldsymbol{D} \rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2		accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Clearly, in each case we will have a contradiction.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	***D***	$\langle M_k \rangle$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form <M_i, <M_j>>

T	<M_1>	<M_2>	<***D***>	<M_k>	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	<M_1>	<M_2>	<***D***>	<M_k>	…
M_1	accept	reject	…	**reject**	…
M_2		accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Clearly, in each case we will have a contradiction.

Therefore, ***D*** cannot exist

	<M_1>	<M_2>	***D***	<M_k>	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle$***D***$\rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Clearly, in each case we will have a contradiction.

Therefore, ***D*** cannot exist

And, ***H*** cannot exist

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle$***D***$\rangle$	$\langle M_k \rangle$	…
[illegible]	accept	reject	…	**reject**	…
[illegible]	[illegible]	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	***D***	$\langle M_k \rangle$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $<M_i, <M_j>>$

T	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

Clearly, in each case we will have a contradiction.

Therefore, ***D*** cannot exist

And, ***H*** cannot exist

H	$<M_1>$	$<M_2>$	$<$***D***$>$	$<M_k>$	…
M_1	accept	reject	…	**reject**	…
M_2		accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
	…	…	…	…	…

We don't have a decider TM for $\boldsymbol{A_{TM}}$.

	$<M_1>$	$<M_2>$	***D***	$<M_k>$	…
D	reject	reject	**accept**	accept	…

Table ***T*** describes the behavior of all enumerated TMs

Table ***H*** describes the behavior of TM H on input of the form $\langle M_i, \langle M_j \rangle\rangle$

T	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle$***D***$\rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	doesn't halt	…
M_2	reject	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	doesn't halt	…	reject	…
…	…	…	…	…	…

H	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle$***D***$\rangle$	$\langle M_k \rangle$	…
M_1	accept	reject	…	**reject**	…
M_2	[illegible]	accept	…	reject	…
D	…	…	**reject**	…	…
M_k	accept	**reject**	…	reject	…
…	…	…	…	…	…

Clearly, in each case we will have a contradiction.

Therefore, ***D*** cannot exist

And, ***H*** cannot exist

We don't have a decider TM for $\boldsymbol{A_{TM}}$.

Therefore, $\boldsymbol{A_{TM}}$ is undecidable.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	***D***	$\langle M_k \rangle$	…
D	reject	reject	**accept**	accept	…

Turing Unrecognizable Languages

- **Some languages are Turing-unrecognizable.**

Turing Unrecognizable Languages

- Some languages are Turing-unrecognizable.

- A complement of a language consists of all strings not in the language.

Turing Unrecognizable Languages

- **Some languages are Turing-unrecognizable.**
- **A complement of a language consists of all strings not in the language.**
- **A language L is <u>*co-Turing-recognizable*</u> if it is the complement of a Turing-recognizable language.**
- **$\overline{A}$ is a complement of A.**

Turing Unrecognizable Languages

Theorem 4.22

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

Turing Unrecognizable Languages

Theorem 4.22

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

- **If A is decidable, then A and $\overline{A}$ are Turing-recognizable.**

Turing Unrecognizable Languages

Theorem 4.22

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

- **If A is decidable, then A and $\overline{A}$ are Turing-recognizable.**
- **If both A and $\overline{A}$ are Turing-recognizable, then we can describe a TM M that does the following.**

Turing Unrecognizable Languages

Theorem 4.22

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

- **If A is decidable, then A and $\overline{A}$ are Turing-recognizable.**
- **If both A and $\overline{A}$ are Turing-recognizable, then we can describe a TM M that does the following.**
- **Let M_1 be a recognizer for A and M_2 a recognizer for $\overline{A}$.**

M = “On input w:
1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, *accept*; if M_2 accepts, *reject*.”

Turing Unrecognizable Languages

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