Today.

Today.

- Finish Countability.
- ..or uncountaility.
- Undecidability.

Countable.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Bijection to or from natural numbers implies countably infinite.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Subset of countable set is countable.

Countable.

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Subset of countable set is countable.

All countably infinite sets are the same cardinality as each other.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1

Examples

Countable infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: 0,

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1, -2,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.

- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1, -2, 2...$
  Where $sign(z) = 1$ if $z > 0$ and $sign(z) = 0$ otherwise.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\dots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\dots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative, second positive..

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b+1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative, second positive..
  Will eventually get to any rational.

Real numbers..

Real numbers are same size as integers?

The reals.

Are the set of reals countable?

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000...

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558...

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$
.345212312...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$
.345212312... Some real number

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$
.345212312... Some real number

Diagonalization.

If countable, there a listing, L contains all reals.

Diagonalization.

If countable, there a listing, L contains all reals. For example

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number:

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
⋮

Construct “diagonal” number: .7

Diagonalization.

If countable, there a listing, *L* contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
⋮

Construct "diagonal" number: .77

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .776

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .7767

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .77677

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .77677 ...

Diagonal Number:

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
⋮

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Contradiction!

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
⋮

Construct “diagonal” number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i’s ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0,1]$ is not countable!!

All reals?

Subset $[0,1]$ is not countable!!

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

Any subset of a countable set is countable.

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

Any subset of a countable set is countable.

If reals are countable then so is $[0, 1]$.

Diagonalization.

1. Assume that a set S can be enumerated.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.
5. Show that t is in S.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.
5. Show that t is in S.
6. Contradiction.

Another diagonalization.

The set of all subsets of N.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

Another diagonalization.

The set of all subsets of *N*.

Example subsets of *N*: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, *L*, that contains all subsets of *N*.

Define a diagonal set, *D*:
If *i*th set in *L* does not contain *i*, $i \in D$.
otherwise $i \notin D$.

D is different from *i*th set in *L* for every *i*.
$\Longrightarrow$ *D* is not in the listing.

D is a subset of *N*.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Contradiction.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Contradiction.

Theorem: The set of all subsets of N is not countable.

Another diagonalization.

The set of all subsets of *N*.

Example subsets of *N*: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, *L*, that contains all subsets of *N*.

Define a diagonal set, *D*:
If *i*th set in *L* does not contain *i*, $i \in D$.
otherwise $i \notin D$.

D is different from *i*th set in *L* for every *i*.
$\Longrightarrow$ *D* is not in the listing.

D is a subset of *N*.

L does not contain all subsets of *N*.

Contradiction.

Theorem: The set of all subsets of *N* is not countable.
(The set of all subsets of *S*, is the **powerset** of *N*.)

Diagonalize Natural Number.

Natural numbers have a listing, L.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number...

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Any natural number has a finite number of digits.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Any natural number has a finite number of digits.

"Construction" requires an infinite number of digits.

The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals.

The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals.

First of Hilbert's problems!

Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$,

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't,

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f: R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0,1]$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0,1]$. Bijection into (possible) subset.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0,1]$. Bijection into (possible) subset.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^{+} \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0,1]$. Bijection into (possible) subset.

Postive reals are not bigger than $[0,1]$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0,1]$. Bijection into (possible) subset.

Postive reals are not bigger than $[0,1]$.

Vice Versa.

Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \to [0, 1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0, 1/2]$ scale/reciprocate $\implies f(x) \neq f(y)$.
If one is in $[0, 1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Map into $[0, 1]$. Bijection into (possible) subset.

Postive reals are not bigger than $[0, 1]$.

Vice Versa.

$[0, 1]$ has same cardinality as nonnegative reals!

Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

The powerset of a set is the set of all subsets.

Resolution of hypothesis?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Next Topic: Undecidability.

- Undecidability.

Barber paradox.

Barber announces:

Barber paradox.

Barber announces:
"The barber shaves every person who does not shave themselves."

Barber paradox.

Barber announces:
"The barber shaves every person who does not shave themselves."

Who shaves the barber?

Barber paradox.

Barber announces:
"The barber shaves every person who does not shave themselves."

Who shaves the barber?

Get around paradox?

Barber paradox.

Barber announces:
"The barber shaves every person who does not shave themselves."

Who shaves the barber?

Get around paradox?
The barber lies.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops!

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops!

What type of object is a set that contain sets?

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops!

What type of object is a set that contain sets?

Axioms changed.

Changing Axioms?

Goedel:
Any set of axioms is either

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine...

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?

See Logicomix by Doxiaidis, Papadimitriou (professor here), Papadatos, Di Donna.

Is it actually useful?

Write me a program checker!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
 P - program
 I - input.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
    P - program
    I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Implementing HALT.

Implementing HALT.

$HALT(P, I)$

Implementing HALT.

$HALT(P, I)$
P - program

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Implementing HALT.

$HALT(P, I)$
    P - program
    I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

Implementing HALT.

$HALT(P, I)$

P - program

I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Something about infinity here, maybe?

Halt does not exist.

Halt does not exist.

$HALT(P, I)$

Halt does not exist.

$HALT(P, I)$
P - program

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No!

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! ..

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

Halt and Turing.

Proof:

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$

Another view of proof: diagonalization.

Any program is a fixed length string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & H & H & L & \cdots \\
P_2 & L & L & H & \cdots \\
P_3 & L & H & H & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \color{blue}{\mathrm{H}} & \mathrm{H} & \mathrm{L} & \cdots \\
P_2 & \mathrm{L} & \color{blue}{\mathrm{L}} & \mathrm{H} & \cdots \\
P_3 & \mathrm{L} & \mathrm{H} & \color{blue}{\mathrm{H}} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \color{blue}{\mathrm{H}} & \mathrm{H} & \mathrm{L} & \cdots \\
P_2 & \mathrm{L} & \color{blue}{\mathrm{L}} & \mathrm{H} & \cdots \\
P_3 & \mathrm{L} & \mathrm{H} & \color{blue}{\mathrm{H}} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$\begin{array}{c|cccc} & P_1 & P_2 & P_3 & \cdots \\ \hline & & & & \\ P_1 & \color{blue}{\text{H}} & \text{H} & \text{L} & \cdots \\ P_2 & \text{L} & \color{blue}{\text{L}} & \text{H} & \cdots \\ P_3 & \text{L} & \text{H} & \color{blue}{\text{H}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist!

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist! $\square$

Wow, that was easy!

Wow, that was easy!

We should be famous!

No computers for Turing!

In Turing's time.

No computers for Turing!

In Turing's time.

No computers.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.
e.g., Babbage (from table of logarithms) 1812.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.
e.g., Babbage (from table of logarithms) 1812.

Concept of program as data wasn't really there.

Turing machine.

Turing machine.

A Turing machine.
– an (infinite) tape with characters

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ...

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ... Turing machine!

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI,

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI, self modifying code,

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
–where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI, self modifying code, learning...

Turing and computing.

Just a mathematician?

Turing and computing.

Just a mathematician?

"Wrote" a chess program.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won!

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Involved with computing labs through the 40s.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is...

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! ! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! ! !
Today:

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! ! !
Today:Programs can be written in

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! ! !
Today:Programs can be written in ascii.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to an
....natural number! ! ! !
Today:Programs can be written in ascii.

Undecidable problems.

Does a program print "Hello World"?

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?

Undecidable problems.

Does a program print "Hello World"?
Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

Undecidable problems.

Does a program print "Hello World"?
Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no.

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations
$\implies$ no program can take any set of integer equations

Undecidable problems.

Does a program print "Hello World"?
Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations
$\implies$ no program can take any set of integer equations
and always correctly output whether it has an integer solution.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn't really work.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  - Almost dependent matrices.
- Seminal paper in mathematical biology.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
    Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
      Almost dependent matrices.
- Seminal paper in mathematical biology.
    Person: embryo is blob.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs,

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms,

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head....

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
      Almost dependent matrices.
- Seminal paper in mathematical biology.
    Person: embryo is blob. Legs, arms, head.... How?

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
    Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.
  Developed chemical reaction-diffusion networks that break symmetry.

Turing: personal.

Tragic ending...

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- suicided with cyanide at age 42.

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- suicided with cyanide at age 42.
  (A bite from the apple....)

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- suicided with cyanide at age 42.
  (A bite from the apple....) accident?

British Apology.

Gordon Brown. 2009.

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated.

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back,

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him.

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him. Alan and the many thousands of other gay men who were convicted as he was convicted under homophobic laws were treated terribly.

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him. Alan and the many thousands of other gay men who were convicted as he was convicted under homophobic laws were treated terribly. Over the years millions more lived in fear of conviction.

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him. Alan and the many thousands of other gay men who were convicted as he was convicted under homophobic laws were treated terribly. Over the years millions more lived in fear of conviction. ....

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him. Alan and the many thousands of other gay men who were convicted as he was convicted under homophobic laws were treated terribly. Over the years millions more lived in fear of conviction. ....
So on behalf of the British government, and all those who live freely thanks to Alan’s work I am very proud to say: we’re sorry, you deserved so much better.”

British Apology.

Gordon Brown. 2009. “Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can’t put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him. Alan and the many thousands of other gay men who were convicted as he was convicted under homophobic laws were treated terribly. Over the years millions more lived in fear of conviction. ....
So on behalf of the British government, and all those who live freely thanks to Alan’s work I am very proud to say: we’re sorry, you deserved so much better.”
2013. Granted Royal pardon.

Summary: decidability.

Computer Programs are an interesting thing.

Summary: decidability.

Computer Programs are an interesting thing.
Like Math.

Summary: decidability.

Computer Programs are an interesting thing.

Like Math.

Formal Systems.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea:

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.

Summary: decidability.

Computer Programs are an interesting thing.
 Like Math.
 Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
 Program: Turing (or DIAGONAL) takes P.
 Assume there is HALT.
  DIAGONAL flips answer.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
    DIAGONAL flips answer.
      Loops if P halts, halts if P loops.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
    DIAGONAL flips answer.
      Loops if P halts, halts if P loops.
  What does Turing do on turing?

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
   DIAGONAL flips answer.
    Loops if P halts, halts if P loops.
  What does Turing do on turing? Doesn’t loop or HALT.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
    DIAGONAL flips answer.
      Loops if P halts, halts if P loops.
  What does Turing do on turing? Doesn’t loop or HALT.
    HALT does not exist!

Summary: decidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
Program: Turing (or DIAGONAL) takes P.
Assume there is HALT.
DIAGONAL flips answer.
Loops if P halts, halts if P loops.
What does Turing do on turing? Doesn't loop or HALT.
HALT does not exist! $\square$

Summary: decidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
Program: Turing (or DIAGONAL) takes P.
Assume there is HALT.
DIAGONAL flips answer.
Loops if P halts, halts if P loops.
What does Turing do on turing? Doesn't loop or HALT.
HALT does not exist! □

Computation is a lens for other action in the world.