Something that started being about Hilbert's 6th Problem but ended up being about Computability.

Taylor Dupuy

July 2, 2009

Hilbert's Problems Dealing With Logic (1900)

Hilbert's First Problem

About Set Theory.

Hilbert's Problems Dealing With Logic (1900)

Hilbert's First Problem

About Set Theory.

Hilbert's Second Problem

Axiomatic Systems and Logic.

Hilbert's Problems Dealing With Logic (1900)

Hilbert's First Problem

About Set Theory.

Hilbert's Second Problem

Axiomatic Systems and Logic.

Hilbert's Sixth Problem

Axiomatic Approach to Physics.

Hilbert's Problems Dealing With Logic (1900)

Hilbert's First Problem

About Set Theory.

Hilbert's Second Problem

Axiomatic Systems and Logic.

Hilbert's Sixth Problem

Axiomatic Approach to Physics.

Hilbert's Tenth Problem

Algorithms for Solutions to Equations.

Hilbert's First Problem: Two types of infinities

Countable Infinity

Any set that can be put into 1-1 correspondence with the natural numbers $\mathbb{N} = \{1, 2, 3, 4, \ldots\}$ is said to be **countable**.

Hilbert's First Problem: Two types of infinities

Countable Infinity

Any set that can be put into 1-1 correspondence with the natural numbers $\mathbb{N} = \{1, 2, 3, 4, \ldots\}$ is said to be **countable**.

Hilbert's First Problem: Two types of infinities

Countable Infinity

Any set that can be put into 1-1 correspondence with the natural numbers $\mathbb{N} = \{1, 2, 3, 4, \dots\}$ is said to be **countable**.

Famous Example: the rational numbers are countable.

Hilbert's First Problem: Two types of infinities

Countable Infinity

Any set that can be put into 1-1 correspondence with the natural numbers $\mathbb{N} = \{1, 2, 3, 4, \ldots\}$ is said to be **countable**.

Famous Example: the rational numbers are countable.

1/1	2/1	3/1	4/1	5/1
1/2	2/2	3/2	4/2	5/2
1/3	2/3	3/3	4/3	5/3
1/4	2/4	3/4	4/4	5/4
1/5	2/5	3/5	4/5	5/5

Countable Infinity

Any set that can be put into 1-1 correspondence with the unit interval $[0,1]$ is said to be **uncountable**.

Real Numbers Aren't Countable

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237
  0.57721566490

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237
  0.57721566490
  ...

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237
  0.57721566490
  ...
- Consider the number by going across the diagonal
  0.11521...

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237
  0.57721566490
  ...
- Consider the number by going across the diagonal
  0.11521...

Real Numbers Aren't Countable

Famous Example: The real numbers are not countable.

- Suppose they were, start counting
  3.14159265353
  2.71828182845
  0.02500000000
  1.41421355237
  0.57721566490
  ...
- Consider the number by going across the diagonal
  0.11521...
- Change each digit
  0.22632...
  Get a real number that isn't in the sequence.

Hilbert's First Problem: Statement of Continuum Hypothesis

part a: Prove the Continuum Hypothesis

There is no cardinality between the natural numbers and the continuum.

Hilbert's First Problem: Statement of Continuum Hypothesis

part a: Prove the Continuum Hypothesis

There is no cardinality between the natural numbers and the continuum.

Hilbert's First Problem: Statement of Continuum Hypothesis

part a: Prove the Continuum Hypothesis

There is no cardinality between the natural numbers and the continuum.

(The answer to this problem is VERY weird)

Well-Ordering

A set is said to be **well-ordered** if every subset has a minimal element.

Well-Ordering

A set is said to be **well-ordered** if every subset has a minimal element.

Well-Ordering

A set is said to be **well-ordered** if every subset has a minimal element.

- Natural Numbers have this property: The set

$$\{6, 310, 25, 19999, 2345\}$$

Well-Ordering

A set is said to be **well-ordered** if every subset has a minimal element.

- Natural Numbers have this property: The set

$$\{6, 310, 25, 19999, 2345\}$$

has 6 as it's smallest element.

Well-Ordering

A set is said to be **well-ordered** if every subset has a minimal element.

- Natural Numbers have this property: The set

$$\{6, 310, 25, 19999, 2345\}$$

  has 6 as it's smallest element.
- The Real Numbers with their traditional ordering do not have this property: Take any open interval.

part b: Well-Ordering of the Reals

Find a fancy way to rearrange the real numbers to make a well ordering (make up a new ordering)

part b: Well-Ordering of the Reals

Find a fancy way to rearrange the real numbers to make a well ordering (make up a new ordering)

part b: Well-Ordering of the Reals

Find a fancy way to rearrange the real numbers to make a well ordering (make up a new ordering)

The Answer to this is also kind of weird.

Hilbert's Second Problem: Axioms

Hilbert's Second Problem: The Statement

Hilbert's Second Problem

Hilbert's Second Problem: The Statement

Hilbert's Second Problem

Hilbert's Second Problem: The Statement

Hilbert's Second Problem

1. Given a system of axioms determine if they are **redundant**.

Hilbert's Second Problem: The Statement

Hilbert's Second Problem

1. Given a system of axioms determine if they are **redundant**.
2. Find a way to determine if a given system of axioms is consistent.

Hilbert's Second Problem: The Statement

Hilbert's Second Problem

1. Given a system of axioms determine if they are **redundant**.
2. Find a way to determine if a given system of axioms is consistent.

This has a very weird answer.

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Weirdness 1: Gödel

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Incompleteness Theorem (1936)

For every consistent system of axioms in predicate logic there exists a statement which is true but not provable!

Weirdness 1: Gödel

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Incompleteness Theorem (1936)

For every consistent system of axioms in predicate logic there exists a statement which is true but not provable!

Weirdness 1: Gödel

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Incompleteness Theorem (1936)

For every consistent system of axioms in predicate logic there exists a statement which is true but not provable!

This theorem tells us one of two things is going on with math:

Weirdness 1: Gödel

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Incompleteness Theorem (1936)

For every consistent system of axioms in predicate logic there exists a statement which is true but not provable!

This theorem tells us one of two things is going on with math:

1. The set theory we use in everyday life is inconsistent.

Weirdness 1: Gödel

Completeness Theorem

We know how to prove things in propositional logic! (Seven basic rules for manipulating logic)

Incompleteness Theorem (1936)

For every consistent system of axioms in predicate logic there exists a statement which is true but not provable!

This theorem tells us one of two things is going on with math:

1. The set theory we use in everyday life is inconsistent.
2. There is something within that set theory that is true but can't be proved.

ZF vs ZFC

Two flavors of set theory. ZF and ZFC

Continuum Hypothesis

Gödel 1940

You will not run into a contradiction if you take the continuum hypothesis to be true.

Continuum Hypothesis

Gödel 1940

You will not run into a contradiction if you take the continuum hypothesis to be true.

Cohen 1963,1964

You will not run into a contradiction if you take the continuum hypothesis to be false.

Continuum Hypothesis

Gödel 1940

You will not run into a contradiction if you take the continuum hypothesis to be true.

Cohen 1963,1964

You will not run into a contradiction if you take the continuum hypothesis to be false.

Continuum Hypothesis

Gödel 1940

You will not run into a contradiction if you take the continuum hypothesis to be true.

Cohen 1963,1964

You will not run into a contradiction if you take the continuum hypothesis to be false.

This shows that the continuum hypothesis is independent of set theory. Analogy: Parallel Postulate in Geometry.

Euclid's Axioms ± Parallel Postulate

Hilbert's Sixth Problem

Hilbert's 6th Problem

Axiomatize Physics and Probability.

Hilbert's 6th Problem

Axiomatize Physics and Probability.

Hilbert's 6th Problem

Axiomatize Physics and Probability.

1. Deriving large scale laws from small scale laws.

Hilbert's Sixth Problem

Hilbert's 6th Problem

Axiomatize Physics and Probability.

1. Deriving large scale laws from small scale laws.
2. Spirit of the problem: No re-adjusting assumptions. Finding out if your assumptions are **consistent** is a big issue.

Hilbert probably didn't have any idea how rough the logic was going to be.

Hilbert's Sixth Problem

Hilbert's 6th Problem

Axiomatize Physics and Probability.

1. Deriving large scale laws from small scale laws.
2. Spirit of the problem: No re-adjusting assumptions. Finding out if your assumptions are **consistent** is a big issue.

Hilbert probably didn't have any idea how rough the logic was going to be.
Idea: Is it possible to interpret physical phenomena in terms of some new stuff having to deal with proofs.

Note on Probability: This is done

A **Probability Space** is a tuple $(\Omega, \mathcal{F}, p)$ where

- Ω is a set.
- $\mathcal{F}$ is a σ-algebra, making Ω into a measure space.
- p is a probability measure satisfying $p(\Omega) = 1$.

Note on Probability: This is done

A **Probability Space** is a tuple $(\Omega, \mathcal{F}, p)$ where

- Ω is a set.
- $\mathcal{F}$ is a σ-algebra, making Ω into a measure space.
- p is a probability measure satisfying $p(\Omega) = 1$.

There is a more general version of probability based on operators.

Hilbert's Tenth Problem

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

Hilbert's Tenth Problem

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

- The equation

$$X^2 + 1 = 0,$$

Hilbert's Tenth Problem

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

- The equation

  $$X^2 + 1 = 0,$$

  has **NO** solutions in the integers.

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

- The equation

$$X^2 + 1 = 0,$$

  has **NO** solutions in the integers.
- The equation

$$X^3Y^2 + 3X^2 - 13YZ^4 + 5053 = 0,$$

Diophantine Equations

Given a polynomial with integer coefficients $f(X_1, X_2, \ldots, X_n)$, determine if there are integers you can plug in to solve

$$f(X_1, X_2, \ldots, X_n) = 0.$$

- The equation
$$X^2 + 1 = 0,$$
has **NO** solutions in the integers.
- The equation
$$X^3Y^2 + 3X^2 - 13YZ^4 + 5053 = 0,$$
has $(X, Y, Z) = (2, 5, 3)$ as a solution in the integers.

Hilbert's Tenth Problem

Hilbert's Tenth Problem

Find an algorithm for deciding if there are integer solutions.

Hilbert's Tenth Problem

Find an algorithm for deciding if there are integer solutions.

Hilbert's Tenth Problem

Find an algorithm for deciding if there are integer solutions.

YOU CAN'T!

Hilbert's Tenth Problem

Hilbert's Tenth Problem

Find an algorithm for deciding if there are integer solutions.

YOU CAN'T!

- 1930s Turing, Post, Church (Developed the theory on Computation)
- 1960s Robinson (Did the Large Part)
- 1970s Matiyasevich (Finished what Robinson couldn't prove)

What Hilbert's Tenth Problem gives us

What Hilbert's Tenth Problem gives us

- A Model for computation.

What Hilbert's Tenth Problem gives us

- A Model for computation.
- Probablistic proofs: You give me an epsilon and I'll check your proof within that probability

What Hilbert's Tenth Problem gives us

- A Model for computation.
- Probablistic proofs: You give me an epsilon and I'll check your proof within that probability
- Ways to set up machines to check proofs (Interactive proof systems)

What Hilbert's Tenth Problem gives us

- A Model for computation.
- Probablistic proofs: You give me an epsilon and I'll check your proof within that probability
- Ways to set up machines to check proofs (Interactive proof systems)
- Long Proofs are Checkable using probability.

What Hilbert's Tenth Problem gives us

- A Model for computation.
- Probablistic proofs: You give me an epsilon and I'll check your proof within that probability
- Ways to set up machines to check proofs (Interactive proof systems)
- Long Proofs are Checkable using probability.
- Model for computation over the real numbers.

What Hilbert's Tenth Problem gives us

- A Model for computation.
- Probablistic proofs: You give me an epsilon and I'll check your proof within that probability
- Ways to set up machines to check proofs (Interactive proof systems)
- Long Proofs are Checkable using probability.
- Model for computation over the real numbers.

Big Curiosity: Is there a way to "hook-up" high precision experiments to this shit?

An Example of a Turing Machine.

RULES

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

$$\begin{array}{cccccccc} & & q_0 & & & & & \\ \text{x} & \text{x} & \text{x} & 0 & \_ & \_ & \_ & \ldots \end{array}$$

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

$$\begin{array}{cccccccc} & & & q_0 & & & & \\ x & x & x & 0 & \_ & \_ & \_ & \ldots \end{array}$$

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

$$\begin{array}{ccccccccc} & & & q_1 & & & & \\ x & x & x & x & \_ & \_ & \_ & \ldots \end{array}$$

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_1

x x x x _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

NO!

x x x x _ _ _ ...

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

Another Example: Input word 000

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated to the Turing machine.

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_0

0 0 0 _ _ _ _ ...

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_1

x 0 0 _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

$$\begin{array}{cccccccc} & q_1 & & & & & & \\ x & 0 & 0 & \_ & \_ & \_ & \_ & \dots \end{array}$$

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

	q_2						
x	x	0	_	_	_	_	...

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_2

x x 0 _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

$$\begin{array}{cccccccc} & & q_0 & & & & & \\ x & x & x & \_ & \_ & \_ & \_ & \dots \end{array}$$

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_0

x x x _ _ _ _ ...

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

YES!

x x x _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

Another Example: Input word 01

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated to the Turing machine.

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_0

0	1	_	_	_	_	_	...

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_1

x 1 _ _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

q_1

x 1 _ _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

An Example of a Turing Machine.

RULES

- If the head is in state q_0, q_1 or q_2 and it reads a 0, print x.
- If the head is in state q_0, q_1 or q_2 and it reads a x, move right.
- If the head is in state q_0 and reads a blank, go to the accept state "YES!".
- If the head is in state q_1, q_2 reads a blank, go to the reject state "NO!".
- If the head is in any state and reads a 1, go the the reject state "NO!".

NO!

x 1 _ _ _ _ _ . . .

This Machine will accept strings which consist only of zeros which occur in multiples of three. This is called the Language associated

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.
- The tape has a certain language which the head reads.

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.
- The tape has a certain language which the head reads.
- The head has a state space which determines how it moves along the tape.

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.
- The tape has a certain language which the head reads.
- The head has a state space which determines how it moves along the tape.
- INPUT: A finite string from the alphabet

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.
- The tape has a certain language which the head reads.
- The head has a state space which determines how it moves along the tape.
- INPUT: A finite string from the alphabet
- OUTPUT: If the head ever goes to its accept state we output YES. If the head ever goes to its reject state we output no.

Informal Definition of a Turing Machine

Turing Machine (Informal Version)

- Consists of an infinitely long tape on which the head sits.
- The tape has a certain language which the head reads.
- The head has a state space which determines how it moves along the tape.
- INPUT: A finite string from the alphabet
- OUTPUT: If the head ever goes to its accept state we output YES. If the head ever goes to its reject state we output no. Never getting to either YES or NO is ok.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.
3. Γ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.
3. Γ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.
3. Γ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
5. $q_0 \in Q$ the start state.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.
3. Γ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
5. $q_0 \in Q$ the start state.
6. $YES \in Q$ the accept state.

Formal Definition of a Turing Machine

Turing Machine (Formal Version)

A Turing Machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, YES, NO)$ consisting of

1. Q the set of states.
2. Σ the input alphabet not containing the blank symbol $\_$.
3. Γ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
5. $q_0 \in Q$ the start state.
6. $YES \in Q$ the accept state.
7. $NO \in Q$ the reject state.

Formal Definition of a Turing Machine: Example

In our example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.
3. $\Gamma = \{\_, 0, 1, x\}$ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.
3. $\Gamma = \{\_, 0, 1, x\}$ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is described by the rules printed above.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.
3. $\Gamma = \{\_, 0, 1, x\}$ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is described by the rules printed above.
5. $q_0 \in Q$ the start state.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.
3. $\Gamma = \{\_, 0, 1, x\}$ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is described by the rules printed above.
5. $q_0 \in Q$ the start state.
6. $YES \in Q$ the accept state.

Formal Definition of a Turing Machine: Example

In our example

1. $Q = \{q_0, q_1, q_2, YES, NO$ is the set of states.
2. $\Sigma = \{0, 1\}$ the input alphabet not containing the blank symbol $\_$.
3. $\Gamma = \{\_, 0, 1, x\}$ the tape alphabet, containing the blank symbol and $\Sigma \subset \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is described by the rules printed above.
5. $q_0 \in Q$ the start state.
6. $YES \in Q$ the accept state.
7. $NO \in Q$ the reject state.

Turing Machines Correspond to Languages

Language of a Turing Machine

The language L of a Turing Machine M consists of all words w such that M accepts w.

Turing Machines Correspond to Languages

Language of a Turing Machine

The language L of a Turing Machine M consists of all words w such that M accepts w.

Turing Machines Correspond to Languages

Language of a Turing Machine

The language L of a Turing Machine M consists of all words w such that M accepts w.

For example the Turing Machine in our example accepted the language

Turing Machines Correspond to Languages

Language of a Turing Machine

The language L of a Turing Machine M consists of all words w such that M accepts w.

For example the Turing Machine in our example accepted the language

$$\{\epsilon, 000, 000000, \ldots, 0^{3k}, \ldots\}$$

- If Σ is a finite set, Σ^* denotes the collection of all words made from those characters.
- When encoding problems involving polynomials or other complicated objects we don't write out a huge thing. Notation: $\langle Object \rangle$ for encoding of an object.
- Think of something like ASCII. For example the polynomial $x^2 + 1$ will be encoded by it's characters..
- You can end up doing all the computations you want with these things. This is what Turing Completeness is about.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Example: The language of polynomials which have integer solutions is recognizable.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Example: The language of polynomials which have integer solutions is recognizable.

1. Just have the machine plug in every possible combo of integers.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Example: The language of polynomials which have integer solutions is recognizable.

1. Just have the machine plug in every possible combo of integers.
2. If some combo is a solution, solutions exist and we should accept.

Turing Recognizable

Recognizable Language

The language L of is recognizable if there exists some Turing Machine that accepts every word in the Language.

Example: The language of polynomials which have integer solutions is recognizable.

1. Just have the machine plug in every possible combo of integers.
2. If some combo is a solution, solutions exist and we should accept.
3. If there isn't, the machine will go into an infinite loop and not accept.

Turing Decidable

Decidable Language

The language L of is decidable if there exists a Turing Machine that accepts every word in the language and rejects every word that is not in the language.

Aside: Classical and Quantum Computability

One can setup a quantum like model. For that we get the following:

Language

- The set of Turing Decidable Languages is Equal to the Set of Quantum Decidable Languages.
- The set of Turing Recognizable Languages is Equal to the set of Quantum Recognizable Languages.

Result of Hilbert's Tenth Problem

Halting Problem, and Matesevich's Proof

Graph Isomorphism Problem

Prover Verifier Systems

PCP

Real Turing Machines