Decidability and Undecidability

Exposition by William Gasarch—U of MD

Recall Turing Machines

I am not going to bother defining TM's again.

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

1. TM's are Java Programs.

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

1. TM's are Java Programs.
2. We have a listing of them $M_1, M_2, \ldots$.

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

1. TM's are Java Programs.
2. We have a listing of them $M_1, M_2, \ldots$.
3. If you run $M_e(d)$ it might not halt.

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

1. TM's are Java Programs.
2. We have a listing of them $M_1, M_2, \ldots$.
3. If you run $M_e(d)$ it might not halt.
4. Everything computable is computable by some TM.

Recall Turing Machines

I am not going to bother defining TM's again.

Here is all you need to know:

1. TM's are Java Programs.
2. We have a listing of them $M_1, M_2, \ldots$.
3. If you run $M_e(d)$ it might not halt.
4. Everything computable is computable by some TM.
5. A TM that halts on all inputs is called **total**.

Computable Sets

Definition A set A is *computable* if there exists a Turing Machine M that behaves as follows:

$$M(x) = \begin{cases} Y & \text{if } x \in A \\ N & \text{if } x \notin A \end{cases} \tag{1}$$

Computable Sets

Definition A set A is *computable* if there exists a Turing Machine M that behaves as follows:

$$M(x) = \begin{cases} Y & \text{if } x \in A \\ N & \text{if } x \notin A \end{cases} \tag{1}$$

Computable sets are also called decidable or solvable. A machine such as M above is said to decide A.

Computable Sets

Definition A set A is *computable* if there exists a Turing Machine M that behaves as follows:

$$M(x) = \begin{cases} Y & \text{if } x \in A \\ N & \text{if } x \notin A \end{cases} \tag{1}$$

Computable sets are also called decidable or solvable. A machine such as M above is said to decide A.

Notation DEC is the set of Decidable Sets.

Notation and Examples

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d)\downarrow= z$ means $M_e(d)$ halts within s steps and outputs z.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d)\downarrow = z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d)\uparrow$ means $M_e(d)$ has not halted within s steps.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d)\downarrow= z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d)\uparrow$ means $M_e(d)$ has not halted within s steps.
Some examples of computable sets.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d) \downarrow$ means $M_e(d)$ halts.
$M_e(d) \uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d) \downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d) \downarrow = z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d) \uparrow$ means $M_e(d)$ has not halted within s steps.
Some examples of computable sets.

1. Primes, Evens, Fibonacci numbers, most sets that you know.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d)\downarrow= z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d)\uparrow$ means $M_e(d)$ has not halted within s steps.
Some examples of computable sets.

1. Primes, Evens, Fibonacci numbers, most sets that you know.
2. $\{(e,d,s) : M_{e,s}(d)\downarrow\}$.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d)\downarrow$ means $M_e(d)$ halts.
$M_e(d)\uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d)\downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d)\downarrow= z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d)\uparrow$ means $M_e(d)$ has not halted within s steps.
Some examples of computable sets.

1. Primes, Evens, Fibonacci numbers, most sets that you know.
2. $\{(e,d,s) : M_{e,s}(d)\downarrow\}$.
3. $\{(e,d,s) : M_{e,s}(d)\uparrow\}$.

Notation and Examples

Notation $M_{e,s}(d)$ is the result of running $M_e(d)$ for s steps.
$M_e(d) \downarrow$ means $M_e(d)$ halts.
$M_e(d) \uparrow$ means $M_e(d)$ does not halts.
$M_{e,s}(d) \downarrow$ means $M_e(d)$ halts within s steps.
$M_{e,s}(d) \downarrow = z$ means $M_e(d)$ halts within s steps and outputs z.
$M_{e,s}(d) \uparrow$ means $M_e(d)$ has not halted within s steps.
Some examples of computable sets.

1. Primes, Evens, Fibonacci numbers, most sets that you know.
2. $\{(e, d, s) : M_{e,s}(d) \downarrow\}$.
3. $\{(e, d, s) : M_{e,s}(d) \uparrow\}$.
4. $\{e : M_e$ has a prime number of states $\}$.

Noncomputable Sets

Are there any noncomputable sets?

Noncomputable Sets

Are there any noncomputable sets?

1. Yes—if not then my PhD thesis would have been a lot shorter.

Noncomputable Sets

Are there any noncomputable sets?

1. Yes—if not then my PhD thesis would have been a lot shorter.
2. Yes—ALL SETS: uncountable. DEC Sets: countable, hence there exists an uncountable number of noncomputable sets.

Noncomputable Sets

Are there any noncomputable sets?

1. Yes—if not then my PhD thesis would have been a lot shorter.
2. Yes—ALL SETS: uncountable. DEC Sets: countable, hence there exists an uncountable number of noncomputable sets.
3. That last answer is true but unsatisfying. We want an actual example of an noncomputable set.

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e, d) \mid M_e(d) \text{ halts } \}.$$

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e, d) \mid M_e(d) \text{ halts } \}.$$

Thought Experiment Here is one way you might want to determine if $(e, d) \in HALT$.

Given (e, d) run $M_e(d)$. If it halts say YES.

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e,d) \mid M_e(d) \text{ halts } \}.$$

Thought Experiment Here is one way you might want to determine if $(e,d) \in HALT$.

Given (e,d) run $M_e(d)$. If it halts say YES.

Does not work since do not know when to stop running it.

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e, d) \mid M_e(d) \text{ halts } \}.$$

Thought Experiment Here is one way you might want to determine if $(e, d) \in HALT$.

Given (e, d) run $M_e(d)$. If it halts say YES.

Does not work since do not know when to stop running it.
Is there *some* way to solve this?

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e, d) \mid M_e(d) \text{ halts }\}.$$

Thought Experiment Here is one way you might want to determine if $(e, d) \in HALT$.

Given (e, d) run $M_e(d)$. If it halts say YES.

Does not work since do not know when to stop running it.

Is there *some* way to solve this? No.

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e,d) \mid M_e(d) \text{ halts } \}.$$

Thought Experiment Here is one way you might want to determine if $(e,d) \in HALT$.

Given (e,d) run $M_e(d)$. If it halts say YES.

Does not work since do not know when to stop running it.
Is there *some* way to solve this? No.

We need to prove this. We must show that it is NOT the case that some clever person can look at the code and figure out that its NOT going to halt.

The HALTING Problem

Definition The HALTING set is the set

$$HALT = \{(e, d) \mid M_e(d) \text{ halts } \}.$$

Thought Experiment Here is one way you might want to determine if $(e, d) \in HALT$.

Given (e, d) run $M_e(d)$. If it halts say YES.

Does not work since do not know when to stop running it.

Is there *some* way to solve this? No.

We need to prove this. We must show that it is NOT the case that some clever person can look at the code and figure out that its NOT going to halt.

Recall You all thought there was no small NFA for $\{a^i : i \neq n\}$ and were wrong. Hence lower bounds need proof.

HALT is Undecidable

Theorem HALT is not computable.
Proof Assume HALT computable via TM M.

$$M(e,d) = \begin{cases} Y & \text{if } M_e(d)\downarrow \\ N & \text{if } M_e(d)\uparrow \end{cases} \tag{2}$$

We use M to create the following machine which is M_e.

1. Input d
2. Run $M(d,d)$
3. If $M(d,d) = Y$ then RUN FOREVER.
4. If $M(d,d) = N$ then HALT.

$M_e(e)\downarrow \implies M(e,e) = Y \implies M_e(e)\uparrow$
$M_e(e)\uparrow \implies M(e,e) = N \implies M_e(e)\downarrow$
We now have that $M_e(e)$ cannot $\downarrow$ and cannot $\uparrow$. Contradiction.

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most,exactly)

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most, exactly)

$\{e : M_e$ halts on an infinite number of numbers$\}$

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most,exactly)

$\{e : M_e$ halts on an infinite number of numbers$\}$

$\{e : M_e$ halts on a finite number of numbers$\}$

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most,exactly)

$\{e : M_e$ halts on an infinite number of numbers$\}$

$\{e : M_e$ halts on a finite number of numbers$\}$

$\{e : M_e$ does the Hokey Pokey and turns itself around $\}$

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most,exactly)

$\{e : M_e$ halts on an infinite number of numbers$\}$

$\{e : M_e$ halts on a finite number of numbers$\}$

$\{e : M_e$ does the Hokey Pokey and turns itself around $\}$

$TOT = \{e : M_e$ halts on all inputs$\}$

Other Undecidable Problems

Using that HALT is undecidable we can prove the following undecidable:

$\{e : M_e$ halts on at least 12 numbers $\}$ (at most,exactly)

$\{e : M_e$ halts on an infinite number of numbers$\}$

$\{e : M_e$ halts on a finite number of numbers$\}$

$\{e : M_e$ does the Hokey Pokey and turns itself around $\}$

$TOT = \{e : M_e$ halts on all inputs$\}$

Proofs by reductions. Similar to NPC. We will not do that.

HALT and SAT

Why will we not be doing reductions in computability theory?

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ *we can find a graph* G *such that* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ *we can find a graph G such that* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.
   Is this interesting?

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ *we can find a graph G such that* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*
   Is this interesting? Yes Formulas related to Graphs!
2. Once HALT is proven undecidable we can show TOT is undecidable by a reduction:

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ *we can find a graph* G *such that* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.
   Is this interesting? Yes Formulas related to Graphs!
2. Once HALT is proven undecidable we can show TOT is undecidable by a reduction:
   Given (e, d) *we can* e' *such that* $(e, d) \in HALT$ *iff* $e' \in TOT$
   Is this interesting?

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ *we can find a graph* G *such that* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.
   Is this interesting? Yes Formulas related to Graphs!
2. Once HALT is proven undecidable we can show TOT is undecidable by a reduction:
   Given (e, d) *we can* e' *such that* $(e, d) \in HALT$ *iff* $e' \in TOT$
   Is this interesting? No Machines related to other machines.

HALT and SAT

Why will we not be doing reductions in computability theory?

Contrast

1. Once SAT is proven NPC we can show 3COL NPC by a reduction:
   Given a formula ϕ we can find a graph G such that $\phi \in \mathrm{SAT}$ iff $G \in 3COL$.
   Is this interesting? Yes Formulas related to Graphs!
2. Once HALT is proven undecidable we can show TOT is undecidable by a reduction:
   Given (e, d) we can e' such that $(e, d) \in HALT$ iff $e' \in TOT$
   Is this interesting? No Machines related to other machines.

Reductions in Computability theory came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.

What Sets of TMs Are Decidable?

Decidable sets:

$$\{e : M_e \text{ has a prime number of states } \}$$

What Sets of TMs Are Decidable?

Decidable sets:

$$\{e : M_e \text{ has a prime number of states }\}$$

$$\{e : M_e \text{ has a square number of alphabet symbols}\}$$

What Sets of TMs Are Decidable?

Decidable sets:

$$\{e : M_e \text{ has a prime number of states }\}$$

$$\{e : M_e \text{ has a square number of alphabet symbols}\}$$

$$\{e : M_e \text{ no transition does a MOVE-L}\}$$

What Sets of TMs Are Decidable?

Decidable sets:

$$\{e : M_e \text{ has a prime number of states }\}$$

$$\{e : M_e \text{ has a square number of alphabet symbols}\}$$

$$\{e : M_e \text{ no transition does a MOVE-L}\}$$

Key Difference:

- **Semantic Question**: What does M_e do? is usually undecidable.
- **Syntactic Question**: What does M_e look like? is usually decidable.

Σ_1 Sets

HALT is undecidable.

Σ_1 Sets

HALT is undecidable. How undecidable?

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d) \downarrow]\}$$

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d)\downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d)\downarrow\}$$

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d)\downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d)\downarrow\}$$

B is decidable and

$$HALT = \{(e,d) : (\exists s)[(e,d,s) \in B]\}$$

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d) \downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d) \downarrow\}$$

B is decidable and

$$HALT = \{(e,d) : (\exists s)[(e,d,s) \in B]\}$$

B is decidable. This inspires the following definition.

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d)\downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d)\downarrow\}$$

B is decidable and

$$HALT = \{(e,d) : (\exists s)[(e,d,s) \in B]\}$$

B is decidable. This inspires the following definition.

Definition $A \in \Sigma_1$ if there exists decidable B such that

$$A = \{x : (\exists y)[(x,y) \in B]\}$$

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d)\downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d)\downarrow\}$$

B is decidable and

$$HALT = \{(e,d) : (\exists s)[(e,d,s) \in B]\}$$

B is decidable. This inspires the following definition.

Definition $A \in \Sigma_1$ if there exists decidable B such that

$$A = \{x : (\exists y)[(x,y) \in B]\}$$

Does this definition remind you of something?

Σ_1 Sets

HALT is undecidable. How undecidable? Measure with quants:

$$HALT = \{(e,d) : (\exists s)[M_{e,s}(d)\downarrow]\}$$

Let

$$B = \{(e,d,s) : M_{e,s}(d)\downarrow\}$$

B is decidable and

$$HALT = \{(e,d) : (\exists s)[(e,d,s) \in B]\}$$

B is decidable. This inspires the following definition.

Definition $A \in \Sigma_1$ if there exists decidable B such that

$$A = \{x : (\exists y)[(x,y) \in B]\}$$

Does this definition remind you of something? YES- NP.

Compare NP to Σ_1

$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ and poly p such that

$$A = \{x : (\exists y, |y| \leq p(|x|))[(x, y) \in B]\}$$

Compare NP to Σ_1

$A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ and poly p such that

$$A = \{x : (\exists y, |y| \le p(|x|))[(x, y) \in B]\}$$

$A \in \Sigma_1$ if there exists $B \in \mathrm{DEC}$ such that

$$A = \{x : (\exists y)[(x, y) \in B]\}$$

Compare NP to Σ_1

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2. 2.1 For NP easy means P and the quantifier is over an exp size set.

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2.
   2.1 For NP easy means P and the quantifier is over an exp size set.
   2.2 For Σ_1 easy means DEC and the quantifier is over $\mathbb{N}$.

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2. 2.1 For NP easy means P and the quantifier is over an exp size set.
   2.2 For Σ_1 easy means DEC and the quantifier is over $\mathbb{N}$.
3. Σ_1 came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2. 2.1 For NP easy means P and the quantifier is over an exp size set.
   2.2 For Σ_1 easy means DEC and the quantifier is over $\mathbb{N}$.
3. Σ_1 came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.
4. Are ideas from Computability theory useful in complexity theory?

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2. 2.1 For NP easy means P and the quantifier is over an exp size set.
   2.2 For Σ_1 easy means DEC and the quantifier is over $\mathbb{N}$.
3. Σ_1 came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.
4. Are ideas from Computability theory useful in complexity theory? Yes, to a limited extent.

Compare NP to Σ_1

1. Both use a quantifier and then something easy. So the sets are difficult because of the quantifier.
2. 2.1 For NP easy means P and the quantifier is over an exp size set.
   2.2 For Σ_1 easy means DEC and the quantifier is over $\mathbb{N}$.
3. Σ_1 came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.
4. Are ideas from Computability theory useful in complexity theory? Yes, to a limited extent. My thesis was on showing some of those limits.

More on Σ_1

Theorem Let A be any set. The following are equivalent:

(1) A is Σ_1.

(2) There exists a TM such that $A = \{x : (\exists s)[M_{e,s}(x)\downarrow]\}$.

(3) There exists a total TM such that
$A = \{y : (\exists e, s)[M_{e,s}(x)\downarrow = y]\}$.

Because of (3) Σ_1 is often called recursively enumerable or computably enumerable.

Beyond Σ_1

Definition B is always a decidable set.

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

$A \in \Pi_2$ if $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$.

$\vdots$

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

$A \in \Pi_2$ if $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$.

$\vdots$

$TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$.

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

$A \in \Pi_2$ if $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$.

$\vdots$

$TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$.

Known: $TOT \notin \Sigma_1 \cup \Pi_1$.

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

$A \in \Pi_2$ if $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$.

$\vdots$

$TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$.

Known: $TOT \notin \Sigma_1 \cup \Pi_1$.

Known:

$\Sigma_1 \subset \Sigma_2 \subset \Sigma_3 \cdots$

$\Pi_1 \subset \Pi_2 \subset \Pi_3 \cdots$

Beyond Σ_1

Definition B is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ if $A = \{x : (\exists y_1)(\forall y_2)[(x, y_1, y_2) \in B]\}$.

$A \in \Pi_2$ if $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$.

$\vdots$

$TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$.

Known: $TOT \notin \Sigma_1 \cup \Pi_1$.

Known:
$\Sigma_1 \subset \Sigma_2 \subset \Sigma_3 \cdots$
$\Pi_1 \subset \Pi_2 \subset \Pi_3 \cdots$

TOT is harder than HALT.

Natural Undecidable Sets

Are there any undecidable sets that are not about computation?

Natural Undecidable Sets

Are there any undecidable sets that are not about computation?

Yes—

Natural Undecidable Sets

Are there any undecidable sets that are not about computation?
Yes—a few.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's 10th problem (in modern language) Give an algorithm that will, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ determine if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's 10th problem (in modern language) Give an algorithm that will, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ determine if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.
Hilbert thought this would inspire interesting Number Theory.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's 10th problem (in modern language) Give an algorithm that will, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ determine if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.
Hilbert thought this would inspire interesting Number Theory.
In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher, though he knew quite a lot of math)
Julia Robinson (a Logician and, unusual for the time, a woman)
worked together and showed that if you also allow exponentials the problem is undecidable.

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's 10th problem (in modern language) Give an algorithm that will, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ determine if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.
Hilbert thought this would inspire interesting Number Theory.
In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher, though he knew quite a lot of math)
Julia Robinson (a Logician and, unusual for the time, a woman)
worked together and showed that if you also allow exponentials the problem is undecidable.

Martin Davis was asked who might take their work and extend it to get that H10 cannot be solved. He said

Hilbert's Tenth Problem

In the year 1900 David Hilbert proposed 23 problems for Mathematicians to work.

Definition $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of all polys in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Z}$.

Example $13x^7 + 8x^5 - 19x^2 + 19$

Hilbert's 10th problem (in modern language) Give an algorithm that will, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ determine if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.
Hilbert thought this would inspire interesting Number Theory.
In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher, though he knew quite a lot of math)
Julia Robinson (a Logician and, unusual for the time, a woman)
worked together and showed that if you also allow exponentials the problem is undecidable.

Martin Davis was asked who might take their work and extend it to get that H10 cannot be solved. He said

Yuri Matiyasevich

In 1979 a young Russian named Yuri Matiyasevich finished the proof.

Yuri Matiyasevich

In 1979 a young Russian named Yuri Matiyasevich finished the proof.
It is often said
H10 was proven undecidable by
Martin Davis, Hillary Putnam, Julia Robinson, and Yuri Matiyasevich.

Yuri Matiyasevich

In 1979 a young Russian named Yuri Matiyasevich finished the proof.

It is often said

H10 was proven undecidable by
Martin Davis, Hillary Putnam, Julia Robinson, and Yuri Matiyasevich.

Since then various combinations of the four of them have had papers simplifying the proof.

Yuri Matiyasevich

In 1979 a young Russian named Yuri Matiyasevich finished the proof.
It is often said
H10 was proven undecidable by
Martin Davis, Hillary Putnam, Julia Robinson, and Yuri Matiyasevich.

Since then various combinations of the four of them have had papers simplifying the proof.

The proof involved coding Turing Machines into Polynomials.

Upshot This problem of, given $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ does it have an integer solution is a natural question that is undecidable.