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

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

Computable Sets

Def 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

Def 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

Def 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

Def The HALTING set is the set

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

The HALTING Problem

Def 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

Def 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

Def 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

Def 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

Def 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

Def 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

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

HALT is Undecidable

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

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

HALT is Undecidable

Thm 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.

HALT is Undecidable

Thm 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

HALT is Undecidable

Thm 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 $M(d,d)$

HALT is Undecidable

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

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

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.

HALT is Undecidable

Thm 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.

HALT is Undecidable

Thm 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.

$M_e(e)\downarrow \implies M(e,e) = Y \implies M_e(e)\uparrow$

HALT is Undecidable

Thm 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 I

Why we will not be doing reductions in computability theory I:

HALT and SAT I

Why we will not be doing reductions in computability theory I:

Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:

HALT and SAT I

Why we will not be doing reductions in computability theory I:

Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*

HALT and SAT I

Why we will not be doing reductions in computability theory I:

Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*
   Is this interesting?

HALT and SAT I

Why we will not be doing reductions in computability theory I:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*
   Is this interesting? **Yes** Formulas related to Graphs!

HALT and SAT I

Why we will not be doing reductions in computability theory I:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ maps to graph G: $\phi \in \mathrm{SAT}$ iff $G \in 3COL$.
   Is this interesting? **Yes** Formulas related to Graphs!
2. HALT undecidable. TOT is undecidable by a reduction:

HALT and SAT I

Why we will not be doing reductions in computability theory I:
Contrast

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

HALT and SAT I

Why we will not be doing reductions in computability theory I:
Contrast

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

HALT and SAT II

Why we will not be doing reductions in computability theory **II**:

HALT and SAT II

Why we will not be doing reductions in computability theory II:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:

HALT and SAT II

Why we will not be doing reductions in computability theory II:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*

HALT and SAT II

Why we will not be doing reductions in computability theory **II**:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.
   A **poly time alg** maps **formulas** to **graphs** .

HALT and SAT II

Why we will not be doing reductions in computability theory II:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$*.*
   A **poly time alg** maps **formulas** to **graphs** .
2. HALT undecidable. TOT is undecidable by a reduction:

HALT and SAT II

Why we will not be doing reductions in computability theory II:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ maps to graph G: $\phi \in \mathrm{SAT}$ iff $G \in 3COL$.
   A **poly time alg** maps **formulas** to **graphs** .
2. HALT undecidable. TOT is undecidable by a reduction:
   A **Turing Machine** maps **Turing Machines** to **Turing Machines** .

HALT and SAT II

Why we will not be doing reductions in computability theory II:
Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula ϕ *maps to graph* G*:* $\phi \in \mathrm{SAT}$ *iff* $G \in 3COL$.
   A **poly time alg** maps **formulas** to **graphs** .
2. HALT undecidable. TOT is undecidable by a reduction:
   A **Turing Machine** maps **Turing Machines** to **Turing Machines** .
   A pedagogical nightmare!

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:

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

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

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

What Sets of TMs Are Decidable?

Decidable sets:

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

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

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

Key Difference:

What Sets of TMs Are Decidable?

Decidable sets:

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

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

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

Key Difference:

- **Semantic Question** : What does M_e do? is usually undecidable.

What Sets of TMs Are Decidable?

Decidable sets:

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

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

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

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.

Def $A \in \Sigma_1$ 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.

Def $A \in \Sigma_1$ 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.

Def $A \in \Sigma_1$ 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

Compare NP to Σ_1

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

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

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

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 quant and then something easy. So the sets are difficult because of the quant.

Compare NP to Σ_1

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

Compare NP to Σ_1

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

Compare NP to Σ_1

1. Both use a quant and then something easy. So the sets are difficult because of the quant.
2. 2.1 For NP **easy** means P and the quant is over an exp size set.
   2.2 For Σ_1 **easy** means DEC and the quant 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 quant and then something easy. So the sets are difficult because of the quant.
2. 2.1 For NP **easy** means P and the quant is over an exp size set.
   2.2 For Σ_1 **easy** means DEC and the quant 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 quant and then something easy. So the sets are difficult because of the quant.
2. 2.1 For NP **easy** means P and the quant is over an exp size set.
   2.2 For Σ_1 **easy** means DEC and the quant 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 quant and then something easy. So the sets are difficult because of the quant.
2. 2.1 For NP **easy** means P and the quant is over an exp size set.
   2.2 For Σ_1 **easy** means DEC and the quant 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

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

More on Σ_1

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

(1) A is Σ_1.

More on Σ_1

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

(1) A is Σ_1.

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

More on Σ_1

Thm 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
$A = \{y : (\exists e, s)[M_{e,s}(x) \downarrow = y]\}$.

More on Σ_1

Thm 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

Def B is always a decidable set.

Beyond Σ_1

Def B is always a decidable set.

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

Beyond Σ_1

Def 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

Def 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

Def 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

Def 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

Def 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

Def 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.

More Examples of Σ_i and Π_i Sets

More Examples of Σ_i and Π_i Sets

Set of Turing Machines that compute increasing functions:

More Examples of Σ_i and Π_i Sets

Set of Turing Machines that compute increasing functions:

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

More Examples of Σ_i and Π_i Sets

Set of Turing Machines that compute increasing functions:

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

Set of Turing machines that halt on all but a finite number of inputs

$$\{e : (\exists x)(\forall y > x)(\exists s)[M_{e,s}(y) \downarrow].$$

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.

Natural Undecidable Sets

Are there any undecidable sets that are **not** about computation?
Yes—a few. we will discuss three.

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.

Def $\mathbb{Z}[x_1, \ldots, x_n]$ 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.

Def $\mathbb{Z}[x_1, \ldots, x_n]$ 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.

Def $\mathbb{Z}[x_1, \ldots, x_n]$ 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.

Def $\mathbb{Z}[x_1, \ldots, x_n]$ 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 (cont)

In 1959

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials**
the problem is **undecidable.**

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

1. Logician got little respect in mathematics.

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

1. Logician got little respect in mathematics.
2. Philosopher got no respect in mathematics.

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials**
the problem is **undecidable.**
Outsiders At the time

1. Logician got little respect in mathematics.
2. Philosopher got no respect in mathematics.
3. Women got little respect in mathematics.

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

1. Logician got little respect in mathematics.
2. Philosopher got no respect in mathematics.
3. Women got little respect in mathematics.
   (This was before the Kiersten Stasko presidency.)

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

1. Logician got little respect in mathematics.
2. Philosopher got no respect in mathematics.
3. Women got little respect in mathematics.
   (This was before the Kiersten Stasko presidency.)

It may have taken people outside of the mathemmatical mainstream to even think the problem was undecidable.

Hilbert's Tenth Problem (cont)

In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
Julia Robinson (a female logician)
worked together and showed that if you **also allow exponentials** the problem is **undecidable.**

Outsiders At the time

1. Logician got little respect in mathematics.
2. Philosopher got no respect in mathematics.
3. Women got little respect in mathematics.
   (This was before the Kiersten Stasko presidency.)

It may have taken people outside of the mathemmatical mainstream to even think the problem was undecidable.
But they didn't have Hilbert's Tenth Problem undecidable... yet.

Hilbert's Tenth Problem (cont)

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 (cont)

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

A young Russian Mathematician

Hilbert's Tenth Problem (cont)

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

A young Russian Mathematician

He was right!

Hilbert's Tenth Problem (cont)

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

A young Russian Mathematician

He was right!
In 1970 a young Russian named Yuri Matiyasevich finished the proof.

Hilbert's Tenth Problem (cont)

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

A young Russian Mathematician

He was right!
In 1970 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.

Hilbert's Tenth Problem (cont)

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

A young Russian Mathematician

He was right!
In 1970 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.

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.

Historical Aside

The history of H10 is **interesting** because it's **boring** .

Historical Aside

The history of H10 is **interesting** because it's **boring**.

1. Davis, Putnam, Robinson were **delighted** that the problem was solved.

Historical Aside

The history of H10 is **interesting** because it's **boring**.

1. Davis, Putnam, Robinson were **delighted** that the problem was solved.
2. Davis, Putnam, Robinson, Matiyasevich all get credit which is how it should be.

Historical Aside

The history of H10 is **interesting** because it's **boring** .

1. Davis, Putnam, Robinson were **delighted** that the problem was solved.
2. Davis, Putnam, Robinson, Matiyasevich all get credit which is how it should be.
3. There have been no duels over who deserves more credit, as their have been in the past.

Historical Aside

The history of H10 is **interesting** because it's **boring** .

1. Davis, Putnam, Robinson were **delighted** that the problem was solved.
2. Davis, Putnam, Robinson, Matiyasevich all get credit which is how it should be.
3. There have been no duels over who deserves more credit, as their have been in the past.
4. Various combinations of the four have had papers since then simplifying and modifying the proof.

Historical Aside

The history of H10 is **interesting** because it's **boring** .

1. Davis, Putnam, Robinson were **delighted** that the problem was solved.
2. Davis, Putnam, Robinson, Matiyasevich all get credit which is how it should be.
3. There have been no duels over who deserves more credit, as their have been in the past.
4. Various combinations of the four have had papers since then simplifying and modifying the proof.

Math (and the rest of life) is full of stories of jealousy and credit-claimers (e.g., Newton vs Leibnitz) so its interesting that this aspect is boring.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.
3. Undec with deg-some d; vars-11;

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.
3. Undec with deg-some d; vars-11;
4. Dec with deg-1, vars-∞. Easy.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.
3. Undec with deg-some d; vars-11;
4. Dec with deg-1, vars-∞. Easy.
5. Dec with deg-∞, vars-1. Easy.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.
3. Undec with deg-some d; vars-11;
4. Dec with deg-1, vars-∞. Easy.
5. Dec with deg-∞, vars-1. Easy.
6. Dec with deg-2, vars-2. Hard. Gauss.

Back to Math

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 $p(a_1, \ldots, a_n) = 0$.

We now know this is undeciable.
For which degrees d and number-of-vars n is it undec? Dec?
For a full account see Gasarch's survey `h10.pdf`
highlights

1. Undec with deg-8, vars-174.
2. Undec with deg-10^{45}, vars-20.
3. Undec with deg-some d; vars-11;
4. Dec with deg-1, vars-∞. Easy.
5. Dec with deg-∞, vars-1. Easy.
6. Dec with deg-2, vars-2. Hard. Gauss.
7. Dec with deg-2, vars-∞. Hard. Recent (1972).

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.
Vote

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**
- This is unknown but people think **no algorithm**

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**
- This is unknown but people think **no algorithm**
- This is unknown but people think **there is an algorithm**

A Simple Case

Consider the following problem: Given k, determine if
$(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**
- This is unknown but people think **no algorithm**
- This is unknown but people think **there is an algorithm**
- This is unknown but **there is no consensus**

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**
- This is unknown but people think **no algorithm**
- This is unknown but people think **there is an algorithm**
- This is unknown but **there is no consensus**
- This is a weird problem that **only Bill cares about**

A Simple Case

Consider the following problem: Given k, determine if $(\exists x, y, z \in \mathbb{Z})[x^3 + y^3 + z^3 = k]$.

Vote

- It has been proven that there **is no algorithm**
- It has been proven that there **is an algorithm**
- This is unknown but people think **no algorithm**
- This is unknown but people think **there is an algorithm**
- This is unknown but **there is no consensus**
- This is a weird problem that **only Bill cares about**

Answer on next slide.

Thought Decidable

Thought Decidable

1. Easy to show that if $k \equiv 4, 5 \pmod{9}$ then NO solution. All future items assume that restriction.

Thought Decidable

1. Easy to show that if $k \equiv 4, 5 \pmod 9$ then NO solution. All future items assume that restriction.
2. If $k \leq 1000$, k not on list below, and $\max\{|x|, |y|, |z|\} \leq 10^{15}$ then k is sum of three cubes.

$$114, 164, 390, 579, 627, 633, 732, 921, 975$$

Thought Decidable

1. Easy to show that if $k \equiv 4, 5 \pmod 9$ then NO solution. All future items assume that restriction.
2. If $k \le 1000$, k not on list below, and $\max\{|x|, |y|, |z|\} \le 10^{15}$ then k is sum of three cubes.

$$114, 164, 390, 579, 627, 633, 732, 921, 975$$

3. Number Theorists think that there is a solution iff $k \not\equiv 4, 5 \pmod 9$.

Thought Decidable

1. Easy to show that if $k \equiv 4, 5 \pmod 9$ then NO solution. All future items assume that restriction.
2. If $k \le 1000$, k not on list below, and $\max\{|x|, |y|, |z|\} \le 10^{15}$ then k is sum of three cubes.

$$114, 164, 390, 579, 627, 633, 732, 921, 975$$

3. Number Theorists think that there is a solution iff $k \not\equiv 4, 5 \pmod 9$.
4. Number Theorists think that this will be hard to prove.

Thought Decidable

1. Easy to show that if $k \equiv 4, 5 \pmod 9$ then NO solution. All future items assume that restriction.
2. If $k \le 1000$, k not on list below, and $\max\{|x|, |y|, |z|\} \le 10^{15}$ then k is sum of three cubes.

$$114, 164, 390, 579, 627, 633, 732, 921, 975$$

3. Number Theorists think that there is a solution iff $k \not\equiv 4, 5 \pmod 9$.
4. Number Theorists think that this will be hard to prove.
5. LARGE knowledge gap between decidable and undecidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.
4. For six 3×3 matrices, undecidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.
4. For six 3×3 matrices, undecidable.
5. For two 2×2 matrices, decidable.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.
4. For six 3×3 matrices, undecidable.
5. For two 2×2 matrices, decidable.

Everything elseis **unknown to science** . We pick out two:

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.
4. For six 3×3 matrices, undecidable.
5. For two 2×2 matrices, decidable.

Everything elseis **unknown to science** . We pick out two:

1. For two 3×3 matrices, unknown.

The Matrix Mortality Question

Input $n \in \mathbb{N}$ and a set $\{M_1, \ldots, M_m\}$ of $n \times n$ matrices over $\mathbb{Z}$.
Question Does some product of the matrices equal the ZERO matrix? (You can use a matrix more than once.)
This problem is undecidable. We refine this:

1. For two 15×15 matrices, undecidable.
2. For three 9×9 matrices, undecidable.
3. For four 5×5 matrices, undecidable.
4. For six 3×3 matrices, undecidable.
5. For two 2×2 matrices, decidable.

Everything elseis **unknown to science** . We pick out two:

1. For two 3×3 matrices, unknown.
2. For three 2×2 matrices, unknown.

Can you Compliment a Context Free Grammar

Can you Compliment a Context Free Grammar

No

Can you Compliment a Context Free Grammar

No Some math objects just don't like being complimented.

Can you Compliment a Context Free Grammar

No Some math objects just don't like being complimented.
Why?

Can you Compliment a Context Free Grammar

No Some math objects just don't like being complimented.
Why? Shy?

Can you Compliment a Context Free Grammar

No Some math objects just don't like being complimented.
Why? Shy? Modest?

Can you Complement a Context Free Grammar

Can you Complement a Context Free Grammar

Input A CFG G.

Can you Complement a Context Free Grammar

Input A CFG G.
Question Is $\overline{L(G)}$ a CFL?

Can you Complement a Context Free Grammar

Input A CFG G.
Question Is $\overline{L(G)}$ a CFL?

This problem is undecidable.

Can you Complement a Context Free Grammar

Input A CFG G.
Question Is $\overline{L(G)}$ a CFL?

This problem is undecidable.

Proof involves looking at the set of all accepting sequences of configurations.
(We will not be doing that, but the proof is here:
`https://www.cs.umd.edu/users/gasarch/COURSES/452/S20/notes/undcfg.pdf`

Are These Problem Natural?

For each of the following problems we will VOTE on if they are natural.

Are These Problem Natural?

For each of the following problems we will VOTE on if they are natural.

(1) Given $p \in \mathbb{Z}[x_1, \ldots, x_n]$ does p have an integer solution?

Are These Problem Natural?

For each of the following problems we will VOTE on if they are natural.

(1) Given $p \in \mathbb{Z}[x_1, \ldots, x_n]$ does p have an integer solution?

(2) Given Matrices $M_1, \ldots, M_m$, does some product $=$ ZERO?

Are These Problem Natural?

For each of the following problems we will VOTE on if they are natural.

(1) Given $p \in \mathbb{Z}[x_1, \ldots, x_n]$ does p have an integer solution?

(2) Given Matrices $M_1, \ldots, M_m$, does some product $=$ ZERO?

(3) Given a CFG G, is $\overline{L(G)}$ a CFL?