Decidability and Undecidability

Exposition by William Gasarch—U of MD

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



I am not going to bother defining TM's again. Here is all you need to know:

▲□▶ ▲□▶ ▲目▶ ▲目▶ 二目 - のへで

I am not going to bother defining TM's again. Here is all you need to know:

1. TM's are Java Programs.

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

◆□▶ ◆□▶ ◆三▶ ◆三▶ ・三 ・ つへぐ

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.

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.

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 .

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

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

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

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 のへぐ

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.

<ロト < 置 > < 置 > < 置 > < 置 > の < @</p>

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

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.

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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 $M_{e,s}(d)$ 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. $\{e: M_e \text{ has a prime number of states }\}$.

Are there any noncomputable sets?



Are there any noncomputable sets?

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 二目 - のへで

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.

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.

▲ロ ▶ ▲周 ▶ ▲ ヨ ▶ ▲ ヨ ▶ → ヨ → の Q @

Def The HALTING set is the set

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

<□▶ < □▶ < □▶ < 三▶ < 三▶ = 三 のへぐ

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.

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.

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?

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.

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.

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

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

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

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ | 目 | のへの

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

1. Input d

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

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

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 - のへぐ

- 1. Input d
- 2. Run M(d, d)

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

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

ション ふゆ アメビア メロア しょうくしゃ

- 1. Input d
- 2. Run M(d, d)
- 3. If M(d, d) = Y then RUN FOREVER.

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

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

- 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.

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

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

ション ふゆ アメビア メロア しょうくしゃ

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.

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

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

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

ション ふゆ アメビア メロア しょうくしゃ

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.

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

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

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

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

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

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

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

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

- $\{e: M_e \text{ halts on an infinite number of numbers}\}$
- $\{e: M_e \text{ halts on a finite number of numbers}\}$

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

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

- $\{e: M_e \text{ halts on an infinite number of numbers}\}$
- $\{e: M_e \text{ halts on a finite number of numbers}\}$
- $\{e: M_e \text{ does the Hokey Pokey and turns itself around }\}$

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

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

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

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

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

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

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

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

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

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

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

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

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:

・ロト・母ト・ヨト・ヨト・ヨー つへぐ

Why we will not be doing reductions in computability theory I: $\ensuremath{\textbf{Contrast}}$

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



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

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

 SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 3COL. Is this interesting? Yes Formulas related to Graphs!

- SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 3COL. Is this interesting? Yes Formulas related to Graphs!
- 2. HALT undecidable. TOT is undecidable by a reduction:

- SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 3COL. Is this interesting? Yes Formulas related to Graphs!
- HALT undecidable. TOT is undecidable by a reduction: Given (e, d) we can find e' such that (e, d) ∈ HALT iff e' ∈ TOT

ション ふゆ アメビア メロア しょうくしゃ

Is this interesting?

- SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 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:

・ロト・母ト・ヨト・ヨト・ヨー つへぐ

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



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

 SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 3COL. A poly time alg maps formulas to graphs.

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

- SAT is proven NPC. 3COL NPC by a reduction: *Formula* φ maps to graph G: φ ∈ SAT iff G ∈ 3COL. A poly time alg maps formulas to graphs.
- HALT undecidable. TOT is undecidable by a reduction: A Turing Machine maps Turing Machines to Turing Machines.

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

A pedagogical nightmare!

Decidable sets:

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

Decidable sets:

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

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

Decidable sets:

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

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

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

Decidable sets:

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

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

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

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.

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.
$\Sigma_1 \,\, \text{Sets}$

HALT is undecidable.

▲□▶▲圖▶▲圖▶▲圖▶ 圖 のへで

$\Sigma_1 \,\, \text{Sets}$

HALT is undecidable. How undecidable?

・ロト ・ 理ト ・ ヨト ・ ヨー・ つへぐ

$\Sigma_1 \,\, \text{Sets}$

HALT is undecidable. How undecidable? Measure with quants:

・ロト・日本・ヨト・ヨト・日・ つへぐ

$\Sigma_1 \; \textbf{Sets}$

HALT is undecidable. How undecidable? Measure with quants:

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

・ロト・日本・ヨト・ヨト・日・ つへぐ

$\Sigma_1 \; \textbf{Sets}$

HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

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

◆□▶ ◆□▶ ◆ 臣▶ ◆ 臣▶ ○臣 ○ のへぐ

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 三日 - のへの

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

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

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 - のへぐ

$\Sigma_1 \text{ Sets}$

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

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

ション ふゆ アメリア メリア しょうくしゃ

Does this definition remind you of something?

$\Sigma_1 \text{ Sets}$

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

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

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 - のへぐ

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 目 のへの

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 目 のへの

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 二目 - のへで

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

 $A \in NP$ if there exists $B \in P$ 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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 二目 - のへで

Compare NP to $\boldsymbol{\Sigma}_1$

▲□▶▲圖▶▲≣▶▲≣▶ ■ のへの

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

- 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.

*ロ * * @ * * ミ * ミ * ・ ミ * の < や

- 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} .

- 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.

ション ふゆ アメリア メリア しょうくしゃ

- 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?

- 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.

ション ふゆ アメリア メリア しょうくしゃ

4. Are ideas from Computability theory useful in complexity theory?

Yes, to a limited extent.

- 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.



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

▲□▶▲圖▶▲≣▶▲≣▶ ≣ の�?

Thm Let A be any set. The following are equivalent: (1) A is Σ_1 .

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

Thm Let A be any set. The following are equivalent:
(1) A is Σ₁.
(2) There exists a TM such that A = {x : (∃s)[M_{e,s}(x) ↓]}.

<□ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

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

Thm Let A be any set. The following are equivalent:
(1) A is Σ₁.
(2) There exists a TM such that A = {x : (∃s)[M_{e,s}(x) ↓]}.
(3) There exists a total TM such that A = {y : (∃e, s)[M_{e,s}(x) ↓= y]}.
Because of (3) Σ₁ is often called recursively enumerable or

computably enumerable

Def *B* is always a decidable set.

Def B is always a decidable set. $A \in \Pi_1$ if $A = \{x : (\forall y) [(x, y) \in B]\}.$

▲□▶ ▲□▶ ▲目▶ ▲目▶ 三日 - のへの

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

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

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ ▲ 三 ● ● ●

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 $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$. : $TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ ▲ 三 ● ● ●

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 $A = \{x : (\forall y_1)(\exists y_2)[(x, y_1, y_2) \in B]\}$. : $TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2$. Known: $TOT \notin \Sigma_1 \cup \Pi_1$.

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 $A = \{x : (\forall y_1)(\exists y_2) | (x, y_1, y_2) \in B]\}.$: $TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2.$ Known: $TOT \notin \Sigma_1 \cup \Pi_1$. Known: $\Sigma_1 \subset \Sigma_2 \subset \Sigma_3 \cdots$ $\Pi_1 \subset \Pi_2 \subset \Pi_3 \cdots$

◆□▶ ◆御▶ ◆臣▶ ◆臣▶ 三臣 - のへで

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 $A = \{x : (\forall y_1)(\exists y_2) | (x, y_1, y_2) \in B]\}.$: $TOT = \{x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow]\} \in \Pi_2.$ 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.

・ロト・西・・日・・日・・日・

More Examples of Σ_i and Π_i Sets

▲□▶ ▲圖▶ ▲ 国▶ ▲ 国▶ ■ 目 ● の Q @
More Examples of Σ_i and Π_i Sets

Set of Turing Machines that compute increasing functions:

(ロト (個) (E) (E) (E) (E) のへの

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

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 三日 - のへの

Are there any undecidable sets that are **not** about computation?

Are there any undecidable sets that are $\displaystyle {\rm not}$ about computation? Yes—

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

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

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

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} .

*ロ * * @ * * ミ * ミ * ・ ミ * の < や

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} .

*ロ * * @ * * ミ * ミ * ・ ミ * の < や

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

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

ション ふゆ アメビア メロア しょうくしゃ

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.

<□▶ <□▶ < □▶ < □▶ < □▶ < □▶ < □ > ○ < ○

In 1959

In 1959 Martin Davis (a Logician)

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

▲□▶ ▲□▶ ▲目▶ ▲目▶ 二目 - のへで

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

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**.

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

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

ション ふぼう メリン メリン しょうくしゃ

1. Logician got little respect in mathematics.

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.

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.

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.
- Women got little respect in mathematics. (This was before the Eliza Furtak presidency, more on that later.)

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.
- Women got little respect in mathematics. (This was before the Eliza Furtak presidency, more on that later.)

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

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.
- Women got little respect in mathematics. (This was before the Eliza Furtak presidency, more on that later.)

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.

If Eliza Furtak becomes president she has promised two things

If **Eliza Furtak** becomes president she has promised two things She will fund my grant.

If **Eliza Furtak** becomes president she has promised two things She will fund my grant. She will be was sworn in by her nickname **Eliza**, not her given name Elizabeth.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

If **Eliza Furtak** becomes president she has promised two things She will fund my grant. She will be was sworn in by her nickname **Eliza**, not her given name Elizabeth.

Bill Clinton was sworn in as William Jefferson Clinton

If **Eliza Furtak** becomes president she has promised two things She will fund my grant. She will be was sworn in by her nickname **Eliza**, not her given name Elizabeth.

Bill Clinton was sworn in as **William Jefferson Clinton Joe Biden** was sworn in as **Joseph Robinette Biden** (Robinette?)

If **Eliza Furtak** becomes president she has promised two things She will fund my grant. She will be was sworn in by her nickname **Eliza**, not her given name Elizabeth.

Bill Clinton was sworn in as **William Jefferson Clinton Joe Biden** was sworn in as **Joseph Robinette Biden** (Robinette?)

Who is the only president so far who was sworn in by his nickname?

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

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*

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!

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.

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

ション ふぼう メリン メリン しょうくしゃ

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.

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 $p(x_1, ..., x_n) \in \mathbb{Z}[x_1, ..., x_n]$ does it have an integer solution is a natural question that is undecidable.
The history of H10 is interesting because it's boring .

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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

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

▲□▶ ▲□▶ ▲ 臣▶ ▲ 臣▶ ― 臣 … のへぐ

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.

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.

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.

ション ふぼう メリン メリン しょうくしゃ

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 For example Newton vs Leibnitz had a rap battle, see

see https://www.youtube.com/watch?v=COeKdP3EkXU
So its interesting that for H10 this aspect is boring.

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

ション ふゆ アメビア メロア しょうくしゃ

We now know this is undeciable.

For which degrees d and number-of-vars n is it undec? Dec?

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

ション ふゆ アメビア メロア しょうくしゃ

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

- イロト イボト イモト - モー のへぐ

Input A CFG G.

- イロト イロト イヨト イヨト ヨー のへぐ

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



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

This problem is undecidable.



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