Decidability and Undecidability

Exposition by William Gasarch—U of MD
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:

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.
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:

$$M(x) = \begin{cases} Y & \text{if } x \in A \\ N & \text{if } x \notin A \end{cases}$$  \hspace{1cm} (1)
**Def** A set $A$ is *computable* if there exists a Turing Machine $M$ that behaves as follows:

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

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

**Notation** $\text{DEC}$ is the set of Decidable Sets.
Computable Sets

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

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

(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

\( M_e, s(d) \) 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 halt.

\( M_e, s(d) \downarrow \) 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. \( \{ e : M_e \text{ has a prime number of states} \} \).
Notation and Examples

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

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. \( \{ e : M_e \text{ has a prime number of states} \} \).
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 halt.
\( M_{e,s}(d) \downarrow \) means \( M_e(d) \) halts 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\}. 
**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 halt.
- $M_{e,s}(d) \downarrow$ means $M_e(d)$ halts within $s$ steps.
- $M_{e,s}(d) \downarrow= z$ means $M_e(d)$ halts within $s$ steps and outputs $z$. 

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 \}. 


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  \( M_{e,s}(d) \) 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 halt.

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

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

**Def** The HALTING set is the set

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

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

*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

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

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

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

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

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

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 $M$. 

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.

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

We now have that $M_e(e)$ cannot $\downarrow$ and cannot $\uparrow$. Contradiction.
HALT is Undecidable

**Thm**  HALT is not computable.

**Proof**  Assume HALT computable via TM $M$.

$$M(e, d) = \begin{cases} Y & \text{if } M_e(d) \downarrow \\ N & \text{if } M_e(d) \uparrow \end{cases}$$  \hspace{1cm} (2)
HALT is Undecidable

**Thm**  HALT is not computable.

**Proof**  Assume HALT computable via TM $M$.

\[
M(e, d) = \begin{cases} 
Y & \text{if } M_e(d) \downarrow \\
N & \text{if } M_e(d) \uparrow 
\end{cases}
\]  \hspace{1cm} (2)

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

HALLT is Undecidable

**Thm** HALLT is not computable.

**Proof** Assume HALLT computable via TM $M$.

$$M(e, d) = \begin{cases} 
Y & \text{if } M_e(d) \downarrow \\
N & \text{if } M_e(d) \uparrow 
\end{cases}$$  \hspace{1cm} (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 $M$.

$$M(e, d) = \begin{cases} 
Y & \text{if } M_e(d) \downarrow \\
N & \text{if } M_e(d) \uparrow
\end{cases}$$  \hspace{1cm} (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 $M$.

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

(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 $M$.

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

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 $M$.

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

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 $M$.

$$M(e, d) = \begin{cases} Y & \text{if } M_e(d) \downarrow \\ N & \text{if } M_e(d) \uparrow \end{cases}$$  \hfill (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.**
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 \text{ halts on at least } 12 \text{ numbers } \} \ (\text{at most , exactly })
Other Undecidable Problems

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

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

\{ e : M_e \text{ halts on an infinite 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.
Other Undecidable Problems

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

\{ e : M_e \text{ halts on at least 12 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} \}
Other Undecidable Problems

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

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

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

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

\{ e : M_e \text{ 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 \text{ halts on at least 12 numbers} \} \ (\text{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} \}
Other Undecidable Problems

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

1. \{ e : M_e \text{ halts on at least 12 numbers} \} (at most, exactly)
2. \{ e : M_e \text{ halts on an infinite number of numbers} \}
3. \{ e : M_e \text{ halts on a finite number of numbers} \}
4. \{ 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:

Contrast

1. SAT is proven NPC. 3COL NPC by a reduction:
   - Formula \( \phi \) maps to graph \( G \):
     - \( \phi \in \text{SAT} \) iff \( G \in \text{3COL} \).
   - 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 \((e, d) \in \text{HALT} \) iff \(e' \in \text{TOT} \).
   - Is this interesting?
     - No
     - Machines related to other machines.
Why we will not be doing reductions in computability theory I:

**Contrast**

1. SAT is proven NPC. 3COL NPC by a reduction:
   - Formula $\phi$ maps to graph $G$: $\phi \in \text{SAT}$ iff $G \in \text{3COL}$.
   - 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 $(e, d) \in \text{HALT}$ iff $e' \in \text{TOT}$.
   - Is this interesting? No
   - Machines related to other machines.
Why we will not be doing reductions in computability theory I: 

**Contrast**

1. SAT is proven NPC. 3COL NPC by a reduction: 
   
   *Formula $\phi$ maps to graph $G$: $\phi \in \text{SAT}$ iff $G \in \text{3COL}$.*
Why we will not be doing reductions in computability theory I:

**Contrast**

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

   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* \( \phi \) *maps to graph* \( G \): \( \phi \in \text{SAT} \) *iff* \( G \in \text{3COL} \).
   - 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 $\phi$ maps to graph $G$: $\phi \in \text{SAT} \iff G \in 3\text{COL}$.*

   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 $(e, d) \in \text{HALT} \iff e' \in TOT$*

   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:
   \(\text{Formula } \phi \text{ maps to graph } G: \phi \in \text{SAT iff } G \in \text{3COL}.\)
   Is this interesting? Yes Formulas related to Graphs!

2. HALT undecidable. TOT is undecidable by a reduction:
   \(\text{Given } (e, d) \text{ we can find } e' \text{ such that } (e, d) \in \text{HALT iff } e' \in \text{TOT}\)
   Is this interesting? No Machines related to other machines.
HALT and SAT II

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

Contrast

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

Formula \( \phi \) maps to graph \( G \): \( \phi \in \text{SAT} \) iff \( G \in \text{3COL} \).

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!
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:

   \[ \phi \in \text{SAT} \iff G \in \text{3COL} \]
Why we will not be doing reductions in computability theory II: 

**Contrast**

1. SAT is proven NPC. 3COL NPC by a reduction: 
   *Formula $\phi$ maps to graph $G$: $\phi \in SAT$ 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:
   
   \[ \text{Formula } \phi \text{ maps to graph } G: \phi \in \text{SAT iff } G \in 3\text{COL}. \]
   
   A poly time alg maps formulas to graphs.
Why we will not be doing reductions in computability theory II:

**Contrast**

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

2. HALT undecidable. TOT is undecidable by a reduction:
Why we will not be doing reductions in computability theory II:

**Contrast**

1. SAT is proven NPC. 3COL NPC by a reduction:
   Formula $\phi$ 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:
   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* \( \phi \) maps to graph \( G \): \( \phi \in \text{SAT} \iff G \in 3\text{COL} \).
   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 : M_e \text{ no transition does a MOVE-L} \}
What Sets of TMs Are Decidable?

Decidable sets:

\{ e : M_e \text{ has a prime number of states} \} \\
\{ e : M_e \text{ has a square number of alphabet symbols} \} \\
\{ e : M_e \text{ no transition does a MOVE-L} \}

Key Difference:

- **Semantic Question**: What does $M_e$ do? is usually undecidable.
- **Syntactic Question**: What does $M_e$ look like? is usually decidable.
$\Sigma_1$ Sets

HALT is undecidable.
$\Sigma_1$ Sets

HALT is undecidable. How undecidable?
HALT is undecidable. How undecidable? Measure with quants:
HALT is undecidable. How undecidable? Measure with quants:

\[ \text{HALT} = \{ (e, d) : (\exists s)[M_{e,s}(d) \downarrow] \} \]
Σ₁ Sets

HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

$$HALT = \{(e, d) : (\exists s)[(e, d, s) \in B]\}$$
HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

\(B\) is decidable and

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

\(B\) is decidable. This inspires the following definition.
Σ1 Sets

HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

\( B \) is decidable and

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

\( B \) is decidable. This inspires the following definition.

**Def** \( A \in \Sigma_1 \) if there exists decidable \( B \) such that

\[ A = \{ x : (\exists y)[(x, y) \in B] \} \]
HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

\( B \) is decidable and

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

\( B \) 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?
Σ₁ Sets

HALT is undecidable. How undecidable? Measure with quants:

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

Let

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

\( B \) is decidable and

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

\( B \) is decidable. This inspires the following definition.

**Def** \( A \in \Sigma₁ \) 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 $\Sigma_1$

$A \in \text{NP}$ if there exists $B \in \text{P}$ and poly $p$ such that
Compare NP to $\Sigma_1$

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

$$A = \{x : (\exists y, |y| \leq p(|x|))[x, y] \in B\}$$
Compare NP to $\Sigma_1$

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

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

$A \in \Sigma_1$ if there exists $B \in \text{DEC}$ such that
Compare NP to $\Sigma_1$

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

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

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

$$A = \{x : (\exists y)[x, y] \in B\}$$
Compare NP to $\Sigma_1$.

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

2. For NP easy means P and the quant is over an exp size set.

2.1 For $\Sigma_1$ easy means DEC and the quant is over $\mathbb{N}$.

3. $\Sigma_1$ 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.
Compare NP to $\Sigma_1$

1. Both use a quant and then something easy. So the sets are difficult because of the quant.
Compare NP to $\Sigma_1$

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 $\Sigma_1$

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 $\Sigma_1$ easy means DEC and the quant is over $\mathbb{N}$. 
Compare NP to $\Sigma_1$

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 $\Sigma_1$ easy means DEC and the quant is over $\mathbb{N}$.
3. $\Sigma_1$ came first by several decades. Complexity theory borrowed ideas from Computability theory for the basic definitions.
Compare NP to $\Sigma_1$

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 $\Sigma_1$ easy means $\text{DEC}$ and the quant is over $\mathbb{N}$.

3. $\Sigma_1$ 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 $\Sigma_1$

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 $\Sigma_1$ **easy** means DEC and the quant is over $\mathbb{N}$.

3. $\Sigma_1$ 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 $\Sigma_1$

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 $\Sigma_1$ easy means DEC and the quant is over $\mathbb{N}$.

3. $\Sigma_1$ 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 $\Sigma_1$

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

1. $A$ is $\Sigma_1$.
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) $\Sigma_1$ is often called *recursively enumerable* or *computably enumerable*. 
More on $\Sigma_1$

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

1. $A$ is $\Sigma_1$. 

Because of (3) $\Sigma_1$ is often called *recursively enumerable* or *computably enumerable*. 
More on $\Sigma_1$

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

1. $A$ is $\Sigma_1$.
2. There exists a TM such that $A = \{x : (\exists s)[M_{e,s}(x) \downarrow]\}$.

Because of (3) $\Sigma_1$ is often called recursively enumerable or computably enumerable.
More on $\Sigma_1$

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

1. $A$ is $\Sigma_1$.
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) $\Sigma_1$ is often called recursively enumerable or computably enumerable.
More on $\Sigma_1$

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

1. $A$ is $\Sigma_1$.
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) $\Sigma_1$ is often called *recursively enumerable* or *computably enumerable*. 
Beyond $\Sigma_1$

**Def** $B$ is always a decidable set.
Beyond $\Sigma_1$

**Def** $B$ is always a decidable set.

$A \in \Pi_1$ if $A = \{x : (\forall y)[(x, y) \in B]\}$.
Beyond $\Sigma_1$

**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 $\Sigma_1$

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

$\vdots$

Known: $\text{TOT} / \in \Sigma_1 \cup \Pi_1$.

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

$\text{TOT}$ is harder than $\text{HALT}$. 
**Beyond \( \Sigma_1 \)**

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

\( TOT = \{ x : (\forall y)(\exists s)[M_{x,s}(y) \downarrow] \} \in \Pi_2 \).
Beyond $\Sigma_1$

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

$\vdash$

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

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

**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: $TOT \not\in \Sigma_1 \cup \Pi_1$.

Known:

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

$\Pi_1 \subset \Pi_2 \subset \Pi_3 \cdots$
Beyond $\Sigma_1$

**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:

$\Sigma_1 \subset \Sigma_2 \subset \Sigma_3 \ldots$

$\Pi_1 \subset \Pi_2 \subset \Pi_3 \ldots$

$TOT$ is **harder** than HALT.
More Examples of $\Sigma_i$ and $\Pi_i$ 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 are the least indexed machine computing what they compute.
\[ \{ e : (\forall i < e)(\exists x, s) (\forall t) [ (M_e, s(x) \downarrow \land M_i, s(x) \downarrow \land \text{they differ}) \lor (M_e, s(x) \downarrow \land M_i, t(x) \uparrow) \lor (M_e, t(x) \uparrow \land M_i, t(x) \downarrow) ] \} \in \Pi_3. \]

Can we get this lower in the Arithmetic Hierarchy? Vote Yes. The first quantifier is over a finite set. So better:
More Examples of $\Sigma_i$ and $\Pi_i$ 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 are the least indexed machine computing what they compute.

{ $e$: ($\forall i < e$)($\exists x, s$)($\forall t$)[($M_e, s(x) \downarrow \land M_i, s(x) \downarrow \land$ they differ) $\lor$ ($M_e, s(x) \downarrow \land M_i, t(x) \uparrow$) $\lor$ ($M_e, t(x) \uparrow \land M_i, t(x) \downarrow$)] $\in \Pi_3$.

Can we get this lower in the Arithmetic Hierarchy? Vote Yes. The first quantifier is over a finite set. So better:
More Examples of $\Sigma_i$ and $\Pi_i$ 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 $\Sigma_i$ and $\Pi_i$ 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 are the least indexed machine computing what they compute.

$$\{ e : (\forall i < e)(\exists x, s)(\forall t) \\
[ (M_{e,s}(x) \downarrow \land M_{i,s}(x) \downarrow \land \text{they differ} ) \lor \\
( M_{e,s}(x) \downarrow \land M_{i,t}(x) \uparrow ) \lor \\
( M_{e,t}(x) \uparrow \land M_{i,t}(x) \downarrow ) ] \in \Pi_3$$
More Examples of $\Sigma_i$ and $\Pi_i$ 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 are the least indexed machine computing what they compute.

$$\{ e : (\forall i < e)(\exists x, s)(\forall t)$$

$$[(M_{e,s}(x) \downarrow \land M_{i,s}(x) \downarrow \land \text{they differ}) \lor$$

$$(M_{e,s}(x) \downarrow \land M_{i,t}(x) \uparrow) \lor$$

$$(M_{e,t}(x) \uparrow \land M_{i,t}(x) \downarrow)] \in \Pi_3$$

Can we get this lower in the Arithmetic Hierarchy? Vote
More Examples of $\Sigma_i$ and $\Pi_i$ 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 are the least indexed machine computing what they compute.

$$\{ e : (\forall i < e)(\exists x, s)(\forall t)$$

$$[(M_{e,s}(x) \downarrow \land M_{i,s}(x) \downarrow \land \text{they differ }) \lor$$

$$(M_{e,s}(x) \downarrow \land M_{i,t}(x) \uparrow) \lor$$

$$(M_{e,t}(x) \uparrow \land M_{i,t}(x) \downarrow)] \in \Pi_3$$

Can we get this lower in the Arithmetic Hierarchy? Vote Yes. The first quantifier is over a finite set. So better:
Lower in the Arith Hierarchy

\[ \{ e : (\exists x_1, \ldots, x_{e-1}, s)(\forall t) \]

\[ e-1 \]

\[ \bigwedge_{i=0}^{e-1} [(M_{e,s}(x_i) \downarrow \land M_{i,s}(x_i) \downarrow \land \text{they differ}) \lor \]

\[ (M_{e,s}(x_i) \downarrow \land M_{i,t}(x_i) \uparrow) \lor \]

\[ (M_{e,t}(x_i) \uparrow \land M_{i,t}(x_i) \downarrow)] \in \Sigma_2 \]
Natural Undecidable Sets

Are there any undecidable sets that are not about computation?
Are there any undecidable sets that are not about computation? Yes—
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.
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 Tori Sauders presidency.)

It may have taken people outside of the mathematical mainstream to even think the problem was undecidable. But they didn't have Hilbert's Tenth Problem undecidable... yet.
In 1959
Martin Davis (a Logician)
In 1959
Martin Davis (a Logician)
Hillary Putnam (a philosopher who knew math)
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.
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

Logician got little respect in mathematics.
Philosopher got no respect in mathematics.
Women got little respect in mathematics.
(This was before the Tori Sauders presidency.)
It may have taken people outside of the mathematical
mainstream to even think the problem was undecidable.
... yet.
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.
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.
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 Tori Sauders presidency.)
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 Tori Sauders presidency.)
It may have taken people outside of the mathematical 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 Tori Sauders presidency.)

It may have taken people outside of the mathematical 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
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

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

\[ \text{A young Russian Mathematician} \]

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

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

We now know this is undeciable.
For which degrees \( d \) and number-of-vars \( n \) is it undeciable? Decidable?
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 undecidable.

For which degrees $d$ and number-of-vars $n$ is it undecidable?

Decidable?

For a full account see Gasarch’s survey h10.pdf

H10 with quants over $\mathbb{N}$ and $\mathbb{Z}$ are slightly different.

highlights
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 undecidable.
For which degrees \( d \) and number-of-vars \( n \) is it undecidable?
Decidable?
For a full account see Gasarch’s survey h10.pdf
H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.

highlights

1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.
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 undecidable.
For which degrees \( d \) and number-of-vars \( n \) is it undecidable?
Decidable?
For a full account see Gasarch’s survey h10.pdf
H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.

highlights

1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.
2. \( \mathbb{N} \): Undec with deg-\( 10^{45} \), vars-9; \( \mathbb{Z} \): Undec with deg-\( 10^{45} \), vars-20.
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 undecidable.

For which degrees \( d \) and number-of-vars \( n \) is it undeciable?
Decidable?
For a full account see Gasarch’s survey h10.pdf
H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.

highlights
1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.
2. \( \mathbb{N} \): Undec with deg-10^{45}, vars-9; \( \mathbb{Z} \): Undec with deg-10^{45}, vars-20.
3. \( \mathbb{Z} \): Undec with deg-some \( d \); vars-11;
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 undecidable.

For which degrees \( d \) and number-of-vars \( n \) is it undecidable?

Decidable?

For a full account see Gasarch’s survey h10.pdf

H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.

highlights

1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.
2. \( \mathbb{N} \): Undec with deg-\( 10^{45} \), vars-9; \( \mathbb{Z} \): Undec with deg-\( 10^{45} \), vars-20.
3. \( \mathbb{Z} \): Undec with deg-some \( d \); vars-11;
4. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-1, vars-\( \infty \). 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 undeciable?
Decidable?
For a full account see Gasarch's survey h10.pdf
H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.

highlights

1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.
2. \( \mathbb{N} \): Undec with deg-10^{45}, vars-9; \( \mathbb{Z} \): Undec with deg-10^{45}, vars-20.
3. \( \mathbb{Z} \): Undec with deg-some \( d \); vars-11;
4. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-1, vars-\( \infty \). Easy.
5. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-\( \infty \), vars-1. Easy.
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 undecidable.  
For which degrees \( d \) and number-of-vars \( n \) is it undecidable? Decidable?  
For a full account see Gasarch’s survey h10.pdf 
H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different. 
highlights  
1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.  
2. \( \mathbb{N} \): Undec with deg-\( 10^{45} \), vars-9; \( \mathbb{Z} \): Undec with deg-\( 10^{45} \), vars-20.  
3. \( \mathbb{Z} \): Undec with deg-some \( d \); vars-11;  
4. \( \mathbb{N},\mathbb{Z} \): Dec with deg-1, vars-\( \infty \). Easy.  
5. \( \mathbb{N},\mathbb{Z} \): Dec with deg-\( \infty \), vars-1. Easy.  
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 undecidable.  

For which degrees \( d \) and number-of-vars \( n \) is it undecidable?  

Decidable?  

For a full account see Gasarch’s survey \texttt{h10.pdf}  

H10 with quants over \( \mathbb{N} \) and \( \mathbb{Z} \) are slightly different.  

highlights  

1. \( \mathbb{N} \): Undec with deg-4, vars-58; \( \mathbb{Z} \): Undec with deg-8, vars-174.  
2. \( \mathbb{N} \): Undec with deg-10^{45}, vars-9; \( \mathbb{Z} \): Undec with deg-10^{45}, vars-20.  
3. \( \mathbb{Z} \): Undec with deg-some \( d \); vars-11;  
4. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-1, vars-\( \infty \). Easy.  
5. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-\( \infty \), vars-1. Easy.  
7. \( \mathbb{N}, \mathbb{Z} \): Dec with deg-2, vars-\( \infty \). Hard. Recent (1972).
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 2 15 \( \times \) 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 2 \( 15 \times 15 \) matrices, undecidable.
2. For 3 \( 9 \times 9 \) matrices, undecidable.

Everything that is not subsumed is unknown to science. We pick out two:

1. For 2 \( 3 \times 3 \) matrices, unknown.
2. For 3 \( 2 \times 2 \) 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 2 $15 \times 15$ matrices, undecidable.
2. For 3 $9 \times 9$ matrices, undecidable.
3. For 4 $5 \times 5$ matrices, undecidable.
4. For 6 $3 \times 3$ matrices, undecidable.
5. For 2 $2 \times 2$ matrices, decidable.

Everything that is not subsumed is unknown to science. We pick out two:

1. For 2 $3 \times 3$ matrices, unknown.
2. For 3 $2 \times 2$ 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 $2 \times 15$ matrices, undecidable.
2. For $3 \times 9$ matrices, undecidable.
3. For $4 \times 5$ matrices, undecidable.
4. For $6 \times 3$ matrices, undecidable.

5. For $2 \times 2$ matrices, decidable.

Everything that is not subsumed is unknown to science. We pick out two:

1. For $2 \times 3$ matrices, unknown.
2. For $3 \times 2$ 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 $2 \times 15$ matrices, undecidable.
2. For $3 \times 9$ matrices, undecidable.
3. For $4 \times 5$ matrices, undecidable.
4. For $6 \times 3$ matrices, undecidable.
5. For $2 \times 2$ matrices, decidable.

Everything that is not subsumed is unknown to science. We pick:

1. For $2 \times 3$ matrices, unknown.
2. For $3 \times 2$ 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 2 15 \( \times \) 15 matrices, undecidable.
2. For 3 9 \( \times \) 9 matrices, undecidable.
3. For 4 5 \( \times \) 5 matrices, undecidable.
4. For 6 3 \( \times \) 3 matrices, undecidable.
5. For 2 2 \( \times \) 2 matrices, decidable.

Everything that is not subsumed is 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 $2 \times 15$ matrices, undecidable.
2. For $3 \times 9$ matrices, undecidable.
3. For $4 \times 5$ matrices, undecidable.
4. For $6 \times 3$ matrices, undecidable.
5. For $2 \times 2$ matrices, decidable.

Everything that is not subsumed is unknown to science. We pick out two:

1. For $2 \times 3$ 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 2 $15 \times 15$ matrices, undecidable.
2. For 3 $9 \times 9$ matrices, undecidable.
3. For 4 $5 \times 5$ matrices, undecidable.
4. For 6 $3 \times 3$ matrices, undecidable.
5. For 2 $2 \times 2$ matrices, decidable.

Everything that is not subsumed is unknown to science. We pick out two:

1. For 2 $3 \times 3$ matrices, unknown.
2. For 3 $2 \times 2$ matrices, unknown.
Can you Compliment a Context Free Grammar

No

Some math objects just don’t like being complimented.

Why?

Shy?

Modest?
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

Input: A CFG $G$

Question: Is $L(G)$ 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)
Can you Complement a Context Free Grammar

**Input**  A CFG $G$.

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](https://www.cs.umd.edu/users/gasarch/COURSES/452/S20/notes/undcfg.pdf)
Can you Complement a Context Free Grammar

**Input**: A CFG $G$.

**Question**: Is $L(G)$ 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)
Can you Complement a Context Free Grammar

**Input**  A CFG $G$.

**Question**  Is $L(G)$ a CFL?

This problem is undecidable.
Can you Complement a Context Free Grammar

**Input** A CFG \( G \).

**Question** Is \( L(G) \) 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.

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 equal ZERO?

3. Given a CFG $G$, is $L(G)$ a CFL?
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 $= \text{ZERO}$?
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 $= \text{ZERO}$?
(3) Given a CFG $G$, is $L(G)$ a CFL?