Computable Ramsey Theory

Exposition by William Gasarch

March 22, 2026

A Crash Course in Comp. Theory: Turing Machines

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

$M_{e,s}(x)$ means that we run M_e for s steps.

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

$M_{e,s}(x)$ means that we run M_e for s steps.

$M(x)\downarrow = a$ means that $M(x)$ halts and outputs a.

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

$M_{e,s}(x)$ means that we run M_e for s steps.

$M(x)\downarrow = a$ means that $M(x)$ halts and outputs a.

$M(x) = a$ means that $M(x)$ halts and outputs a

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

$M_{e,s}(x)$ means that we run M_e for s steps.

$M(x)\downarrow = a$ means that $M(x)$ halts and outputs a.
$M(x) = a$ means that $M(x)$ halts and outputs a
(we use the $\downarrow$ when we want to emphasize that $M(x)$ halts).

A Crash Course in Comp. Theory: Turing Machines

$M_1, M_2, \ldots$ is a standard list of Turing Machines (TMs). You can think of them as all Java programs.

We assume that from e we can extract the code for M_e.

$M_{e,s}(x)$ means that we run M_e for s steps.

$M(x)\downarrow = a$ means that $M(x)$ halts and outputs a.

$M(x) = a$ means that $M(x)$ halts and outputs a

(we use the $\downarrow$ when we want to emphasize that $M(x)$ halts).

$M(x)\uparrow$ means that $M(x)$ does not halt.

A Crash Course in Comp. Theory: Computable

A Crash Course in Comp. Theory: Computable

A set A is **computable** if there is a TM M such that

A Crash Course in Comp. Theory: Computable

A set A is **computable** if there is a TM M such that

$$\begin{aligned} x \in A &\implies M(x)\downarrow = 1 \\ x \notin A &\implies M(x)\downarrow = 0 \end{aligned}$$

A Crash Course in Comp. Theory: Computable

A set A is **computable** if there is a TM M such that

$$\begin{aligned} x \in A &\implies M(x)\downarrow = 1 \\ x \notin A &\implies M(x)\downarrow = 0 \end{aligned}$$

A function$f$ is **computable** if there is a TM M such that, for all x, $M(x)\downarrow = f(x)$.

A Crash Course in Comp. Theory: Arith Hier

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

$A \in \Sigma_1$ is there exists computable B such that $A = \{x : (\exists y)[(x, y) \in B]\}$.

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

$A \in \Sigma_1$ is there exists computable B such that $A = \{x : (\exists y)[(x, y) \in B]\}$.

$A \in \Pi_1$ is there exists computable B such that $A = \{x : (\forall y)[(x, y) \in B]\}$.

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

$A \in \Sigma_1$ is there exists computable B such that $A = \{x : (\exists y)[(x, y) \in B]\}$.

$A \in \Pi_1$ is there exists computable B such that $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ is there exists computable B such that $A = \{x : (\exists y)(\forall z)[(x, y, z) \in B]\}$.

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

$A \in \Sigma_1$ is there exists computable B such that $A = \{x : (\exists y)[(x, y) \in B]\}$.

$A \in \Pi_1$ is there exists computable B such that $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ is there exists computable B such that $A = \{x : (\exists y)(\forall z)[(x, y, z) \in B]\}$.

$A \in \Pi_2$ is there exists computable B such that $A = \{x : (\forall y)(\exists z)[(x, y, z) \in B]\}$.

A Crash Course in Comp. Theory: Arith Hier

Sets are classified in the Arithmetic hierarchy.

$A \in \Sigma_1$ is there exists computable B such that $A = \{x : (\exists y)[(x, y) \in B]\}$.

$A \in \Pi_1$ is there exists computable B such that $A = \{x : (\forall y)[(x, y) \in B]\}$.

$A \in \Sigma_2$ is there exists computable B such that $A = \{x : (\exists y)(\forall z)[(x, y, z) \in B]\}$.

$A \in \Pi_2$ is there exists computable B such that $A = \{x : (\forall y)(\exists z)[(x, y, z) \in B]\}$.

Σ_3, Π_3, Σ_4, Π_4, ..., you can guess

A Crash Course in Comp. Theory: Arith Hier

A Crash Course in Comp. Theory: Arith Hier

$$\mathrm{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$$

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e, x) : (\exists s)[M_{e,s}(x) \downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

The proof that $\text{FIN} \notin \Pi_2$ is out of scope.

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

The proof that $\text{FIN} \notin \Pi_2$ is out of scope.

INF is the set of all e where M_e halts on an ∞ numb of inputs.

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

The proof that $\text{FIN} \notin \Pi_2$ is out of scope.

INF is the set of all e where M_e halts on an ∞ numb of inputs.

$$\text{INF} = \{e\colon (\forall x)(\exists y,s)[y > x \wedge M_{e,s}(y)\downarrow\} \in \Pi_2 - \Sigma_2.$$

The proof that $\text{INF} \notin \Sigma_2$ is out of scope.

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

The proof that $\text{FIN} \notin \Pi_2$ is out of scope.

INF is the set of all e where M_e halts on an ∞ numb of inputs.

$$\text{INF} = \{e\colon (\forall x)(\exists y,s)[y > x \wedge M_{e,s}(y)\downarrow\} \in \Pi_2 - \Sigma_2.$$

The proof that $\text{INF} \notin \Sigma_2$ is out of scope.

$\Sigma_1 \subset \Sigma_2 \subset \cdots$.

A Crash Course in Comp. Theory: Arith Hier

$\text{HALT} = \{(e,x)\colon (\exists s)[M_{e,s}(x)\downarrow\} \in \Sigma_1 - \Sigma_0$

FIN is the set of all e where M_e halts on a finite numb of inputs.

$$\text{FIN} = \{e\colon (\exists x)(\forall y,s)[y > x \implies M_{e,s}(y)\uparrow\} \in \Sigma_2 - \Pi_2.$$

The proof that $\text{FIN} \notin \Pi_2$ is out of scope.

INF is the set of all e where M_e halts on an ∞ numb of inputs.

$$\text{INF} = \{e\colon (\forall x)(\exists y,s)[y > x \wedge M_{e,s}(y)\downarrow\} \in \Pi_2 - \Sigma_2.$$

The proof that $\text{INF} \notin \Sigma_2$ is out of scope.

$\Sigma_1 \subset \Sigma_2 \subset \cdots$.
$\Pi_1 \subset \Pi_2 \subset \cdots$.

A Natural Example of an Undecidable Set

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

A Natural Example of an Undecidable Set

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

By the combined efforts of Davis-Putnam-Robinson (1959) and Matiyasevich (1970) showed the following:

A Natural Example of an Undecidable Set

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

By the combined efforts of Davis-Putnam-Robinson (1959) and Matiyasevich (1970) showed the following:

Thm There is no such algorithm.

A Natural Example of an Undecidable Set

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

By the combined efforts of Davis-Putnam-Robinson (1959) and Matiyasevich (1970) showed the following:

Thm There is no such algorithm.

From this work we get the following:

A Natural Example of an Undecidable Set

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

By the combined efforts of Davis-Putnam-Robinson (1959) and Matiyasevich (1970) showed the following:

Thm There is no such algorithm.

From this work we get the following:

Thm The following set is undecidable:
$\{p(\vec{x}) \in \mathbb{Z}[\vec{x}] : (\exists \vec{a} \mathbb{Z}^*)[p(\vec{a}) = 0]\}$.

A Natural Example of an Undecidable Set

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

By the combined efforts of Davis-Putnam-Robinson (1959) and Matiyasevich (1970) showed the following:

Thm There is no such algorithm.

From this work we get the following:

Thm The following set is undecidable:
$\{p(\vec{x}) \in \mathbb{Z}[\vec{x}] : (\exists \vec{a} \mathbb{Z}^*)[p(\vec{a}) = 0]\}$.

I did a survey about when the number of vars and degree caused undecidability, and when they caused decidability. See `https://arxiv.org/abs/2104.07220`

Every Computable Coloring . . .

Every Computable Coloring . . .

Vote on which of the following is true:

Every Computable Coloring . . .

Vote on which of the following is true:
$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_0 homog set.

Every Computable Coloring . . .

Vote on which of the following is true:

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_0 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_1 homog set.

Every Computable Coloring . . .

Vote on which of the following is true:

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_0 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_1 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

Every Computable Coloring . . .

Vote on which of the following is true:

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_0 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_1 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

$\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_i set for any i.

Every Computable Coloring . . .

Vote on which of the following is true:

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_0 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_1 homog set.

$\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

$\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_i set for any i.

Answer on Next Page.

Every Computable Coloring . . .

Every Computable Coloring . . .

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

Every Computable Coloring . . .

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

Every Computable Coloring . . .

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

(3) $\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

Every Computable Coloring . . .

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

(3) $\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

We prove something weaker than (3).

Every Computable Coloring . . .

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

(3) $\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

We prove something weaker than (3).

We then make comments on what else is known.

Every Computable Coloring ...

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

(3) $\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

We prove something weaker than (3).

We then make comments on what else is known.

The proofs of (2) and (3) are possible in this class but

Every Computable Coloring ...

(1) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_0.

(2) $\exists$ a computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ that has no ∞ Π_1.

(3) $\forall$ computable $\mathrm{COL}\colon \binom{\mathbb{N}}{2} \to [2]$ has an ∞ Π_2 homog set.

We prove something weaker than (3).

We then make comments on what else is known.

The proofs of (2) and (3) are possible in this class but

We are busy people!

Every Computable Coloring has an ∞ Σ_3 Homog set

In the next slide packet we will redo the proof of Ramsey's Theorem keeping close attention to how complicated it gets.

Every Computable Coloring has an ∞ Σ_3 Homog set

In the next slide packet we will redo the proof of Ramsey's Theorem keeping close attention to how complicated it gets.

The proof will look familiar to you