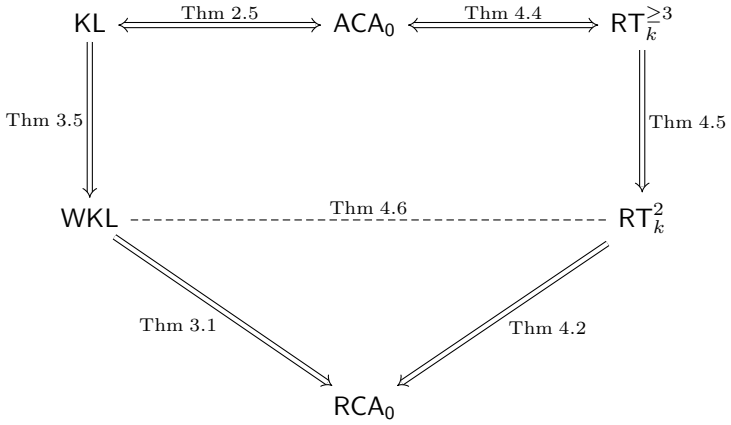
THE REVERSE MATHEMATICS OF TWO COMBINATORIAL PROBLEMS

JALEN CHRYSOS

Abstract. Reverse mathematics studies the precise strengths of theorems, defined by their ability to prove axioms (over a weak base theory). This paper discusses the strengths of two theorems in infinitary combinatorics: König's Lemma and Ramsey's Theorem. We compare weakened cases of each to theories of second-order arithmetic, to theorems from real analysis, and to one another. The method of forcing features prominently.



Contents

Current draft as of September 27, 2025.

Introduction

Mathematics aims to determine the truth through deduction. We begin with statements that we assume to be true—axioms—and combine them using basic logical rules to generate new statements. A statement that can be reached this way is "proven true." Of course, the notion of provability depends on one's foundations.

In the late 19th and early 20th centuries, when foundational questions seemed a pressing concern even for regular mathematicians, it was hoped that in some sufficiently strong base theory, all true statements could be proven. Gödel's work showed this was impossible for all but the weakest formal languages. In simple settings like Euclidean geometry, all statements could be decided. But in the areas most relevant to working mathematicians, like Arithmetic or Set Theory, there would always be unprovable true statements—or, in the case that the theory was contradictory, provable false statements. Gödel had revealed an inherent relativity within logic; knowing that all theories are incomplete, why should we privilege one over the others?

Mathematics adapted around this discovery. Later work by Paul Cohen gave the first natural examples of undecidability in Zermelo-Fraenkel Set Theory: the Axiom of Choice and the Continuum Hypothesis. Since that time, Choice has been mostly accepted as an axiom of set theory (though it is still common to point out where it is relied upon) and much mathematics has been built on top of it. The legitimacy of the continuum hypothesis, on the other hand, is still debated among logicians and is of little relevance to anyone else. Even in the absence of completeness, the mathematical community has settled on a satisfactory foundation of set theory, and alternate foundations are seldom discussed.

Nevertheless, the particulars of the foundations remain deeply involved in the results they support. Just as some theorems depend on the axiom of choice, others rely on (for example) the formation of sets whose elements satisfy a given arithmetically-complex property, though the latter cases are almost never noted. Suffice it to say, the relationships between theorems and their constituent axioms are complex and not fully understood. The field of *Reverse Mathematics* studies these relationships.

The study of reverse mathematics began in the 1970s through the work of Harvey Friedman. As a young professor at Stanford working in mathematical logic, Friedman wanted to impress upon his colleagues the influence of the foundations of mathematics on their work in analysis and algebra, and so began to precisely map out this influence.[1] The subject has since developed beyond Friedman's original conception, but the fundamental pieces have remained constant.

Reverse mathematics seeks to compare the strengths of different theories. The notion of strength is a partial order based on the space of sentences which can be proven; if T_1 proves everything that T_2 does, then T_1 is at least as strong as T_2. It is called "reverse" because it assesses the strength of *theorems* from ordinary mathematics as if they were axioms. In ideal cases, one can show that within a given theory, some subset of axioms can be equivalently replaced by a certain theorem, resulting in a precise classification of the theorem's power—a *reversal*.

We mostly work within second-order arithmetic, a language whose objects are only numbers and sets of numbers (there are no sets of sets, e.g.). Second-order arithmetic is expressive enough to state most theorems of interest but simpler to work with than full set theory. The theories of greatest interest all share certain axioms about the arithmetical properties of $+, \cdot$ and $<$, but vary especially in their axioms concerning the existence of sets. One weak set-existence axiom allows for the construction of all computable sets, which are considered to be the simplest. Another guarantees the existence of any set that can be arithmetically defined, which allows for much more complex sets. The set existence principles of most interest here fall in between these extremes.

[1] See [3].

This expository paper focuses on two of these in-between theorems in second-order arithmetic, each originating from infinitary combinatorics. Both have played essential roles in the development of reverse mathematics:

- *König's Lemma*: Every finitely-branching infinite tree has an infinite path.
- *Ramsey's Theorem*: If one assigns every n-tuple in $\mathbb{N}$ to one of k colors, then there is an infinite subset of $\mathbb{N}$ within which every n-tuple is the same color.

We will explore the differences in strength between special cases of each of these theorems, comparing each to theories of second-order arithmetic of varying complexities and also to several well-known theorems in real analysis.

In Section One, we'll use some ideas from computability theory and formal logic in order to define exactly what we mean by "complexity" in sets and functions.

Section Two will introduce second-order arithmetic and use the established notions of complexity to define the weakest and strongest theories we will consider: RCA_0 and ACA_0. We'll see what regular mathematics looks like in these theories by showing some formal proofs in real analysis. We'll also characterize the precise strengths of König's Lemma and a restricted version, Weak König's Lemma, and equate them to real-analytic principles.

Section Three will introduce *forcing*, a set-construction framework originally developed by Paul Cohen to separate the Axiom of Choice and the Continuum Hypothesis from ZF set theory. We'll use forcing to separate König's Lemma from Weak König's Lemma.

Finally, in Section Four, we will bring all of the developed techniques from the previous three sections to the study of Ramsey's Theorem, culminating in a proof of separation between the $n = 2$ and $n = 3$ cases, first proven by Seetapun.

1. Measures of Set Complexity

In this first section, we discuss the idea of *complexity* in $\mathbb{N} \to \mathbb{N}$ functions—particularly in subsets of $\mathbb{N}$. We'll see two fairly natural ways of formalizing this notion: first from computability theory, and second from syntactic logic. Finally, we will show that our two types of complexity interact well and share a common structure.

1.1. Algorithms and Computability.

The fundamental concept behind all of computability theory is the *partial-computable function* (alternatively, *algorithm* or *Turing machine*). We think of p.c. functions intuitively as those whose values can be determined by carrying out a finite list of instructions. They are "partial" in the sense that they may fail to return any output (in the case where they run forever), in contrast to "total" functions which always return an output.

Partial computable functions can naturally arise in several ways. First, one can define them as a subset of all partial functions via a precise characterization of their expressive power: they are the unique subset of partial functions containing

- *Zero*: $f(x) = 0$ is p.c.
- *Successor*: $f(x) = x + 1$ is p.c.
- *Projection*: $f(x_1, \dots, x_k) = x_j$ is p.c.

and generated by (finitely-many) applications of

- *Composition*: If $f(x_1, \dots, x_k), g_1(\vec{y_1}), g_2(\vec{y_2}), \dots, g_k(\vec{y_k})$ are p.c., then
$$f(g_1(\vec{y_1}), g_2(\vec{y_2}), \dots, g_k(\vec{y_k}))$$
is p.c.
- *Primitive Recursion*: If g, h are p.c., then the function $f(x, \vec{y})$ defined inductively by
$$f(0, \vec{y}) = g(\vec{y}) \quad \text{and} \quad f(x, \vec{y}) = h(f(x-1, \vec{y}), x, \vec{y})$$
is p.c.
- *Unbounded Search*: If $g(x, \vec{y})$ is p.c. then
$$f(\vec{y}) = (\mu x) g(x, \vec{y}) := \text{"the least } x \text{ such that } g(x, \vec{y}) = 0\text{"}$$
is p.c.

Note that of these generating rules, unbounded search is the only one that can initially produce non-total functions. If there is no x such that $g(x, \vec{y}) = 0$, then $(\mu x) g(x, \vec{y}) = \uparrow$ (read *undefined*). After the genesis of non-total functions, however, it is possible that undefined functions are used as inputs to compositions, or might need to be checked during an unbounded search. In either of these cases, we say that *the resulting function is always undefined on that input.*

For any partial function A (not necessarily p.c.), we can also define the p.c. functions *relative to* A by taking the set of functions containing zero, successor, projection, and A, generated by the same operations. We can also do this with any *collection* of partial functions.

Another way of looking at p.c. functions is as step-by-step computations carried out by *Turing machines*. I will not give a precise definition of Turing machines,[2] but instead I'll give an intuitive description: a Turing machine is a finitely-specified "program" that can be carried out in steps, and which may reach a step in which it "halts," reporting an output in $\mathbb{N}$. Given a Turing machine Φ and $x, t \in \mathbb{N}$, we use the notation

$$\Phi(x)[t] := \text{ "Output of running } \Phi \text{ on input } x \text{ for } t \text{ steps."}$$

If $\Phi(x)$ has not halted by time t, we write $\Phi(x)[t] = \uparrow$, and otherwise we write $\Phi(x)[t] \downarrow$. Thus, the outputs of $\Phi(x)[t]$ for increasing values of t will look something like

$$\uparrow, \uparrow, \uparrow, \uparrow, \uparrow, 19, 19, 19, 19, \dots$$

or, if $\Phi(x)[t]$ does not halt for any t,

$$\uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \dots$$

[2] See [9] for a rigorous definition of Turing machine.

We denote by $\Phi(x)$ the limit of $\Phi(x)[t]$ as $t \to \infty$. So in the first case above, $\Phi(x) = 19$, and in the second case, $\Phi(x) = \uparrow$. Though $\Phi(x)[t]$ can be computed for any x, t, $\Phi(x)$ can not—informally, this is because when $\Phi(x) \uparrow$, one must check infinitely many values of t in order to be sure.

In this paper, for partial functions f, g, I will occasionally reference the following relations:

$$f \perp g := \exists x : f(x) \downarrow \neq g(x) \downarrow, \quad \text{and} \quad f \parallel g := \neg f \perp g.$$

Note that $f \perp g$ is stronger than $f \neq g$, though if f and g are both total then it is equivalent. In particular, if σ is a partial function with finite domain $\{0, 1, 2, \dots, k\}$ such that $\sigma \parallel f$ we say

$$\sigma \text{ is an } \textit{initial segment} \text{ of } f, \text{ denoted } \sigma \prec f.$$

The set of such partial functions corresponds to the set of finite sequences, $2^{\mathbb{N}}$.

One additional type of instruction that we can allow algorithms to use is to query an outside source of information called the "*oracle*," which takes the form of a partial function. If Φ is an algorithm and A a partial function, then Φ^A denotes the function Φ with A treated as the oracle. The output of an algorithm may differ depending on which A plays the role of the oracle.

By convention, if $\Phi(x)$ queries $A(y)$ and $A(y) = \uparrow$, then $\Phi^A(y) = \uparrow$ as well. Thus, if $\Phi^A(x) \downarrow$, and B is a total function such that $A \parallel B$, then $\Phi^A(x) = \Phi^B(x)$. This is because $\Phi^A(x)$ never queries $A(y)$ for any y where A and B differ. We can also say that any oracle computation $\Phi^A(x)$ which halts must query only finitely-many of the inputs of A. Thus, there is some initial segment of A sufficient (and necessary) to cause halting:

$$\Phi^A(x) \downarrow \iff \exists \sigma \in 2^{\mathbb{N}} : (\sigma \prec A) \wedge (\Phi^\sigma(x) \downarrow)$$

Note that for $\sigma \in 2^{\mathbb{N}}$, the function Φ^σ is *partial computable*, since σ only contains a finite amount of information, thus an algorithm can be written for Φ^σ which hard-codes the values of σ rather than querying the oracle.

Likewise, if A is computable, then Φ^A is computable as well, since the algorithm for computing A can be grafted into Φ in place of querying A. In general, Turing machines Φ^A are exactly the p.c. functions relative to A. If a function f is equal to Φ^A for some algorithm Φ, we say that

$$\text{"}f \text{ is } \textit{Turing reducible} \text{ to } A\text{"} \text{ or } \text{"}f \text{ is } A\textit{-computable,}\text{"} \text{ denoted } f \leq_T A.$$

Turing reducibility is a partial order on functions, and its equivalence classes are called "Turing degrees." As we have seen, p.c. functions are in the minimum class, $[\emptyset]$, which we will usually write without brackets.

One might naturally consider Turing machines $\Phi^{A,B}$ with multiple oracles, which could express p.c. functions relative to $\{A, B\}$. But this notion can already be captured with single-oracle machines: for any A, B, there exists a unique minimum Turing degree among all degrees above both A and B, called the *effective join* of A and B:

$$\deg(A) \oplus \deg(B) = \min_{\leq_T} \{\mathbf{D} : (\deg(A) \leq_T \mathbf{D}) \wedge (\deg(B) \leq_T \mathbf{D})\}.$$

This degree has a canonical representative $A \oplus B$, defined

$$A = (a_j)_j, \quad B = (b_j)_j \quad \implies \quad A \oplus B = (a_0, b_0, a_1, b_1, a_2, b_2, \dots).$$

One can easily verify that $A \oplus B \leq_T C$ iff $A \leq C$ and $B \leq C$. Making use of the effective join operation, we can see that functions computable from A, B are exactly the ones computable from the single partial function $A \oplus B$.

As each Turing machine is specified by a finite "program," the collection of all Turing machines is countable. Thus, they can be enumerated

$$\Phi_1, \Phi_2, \Phi_3, \dots$$

One characterizing fact about Turing machines is that they can be *computably* enumerated; that is, there is a single partial computable function

$$\Omega : (e, x, t) \mapsto \Phi_e(x)[t].$$

In particular, the fact that the behavior of Φ_e can be computed from the index e is the key thing. We call Ω the *universal Turing machine*.

This is a characteristic advantage of working with p.c. functions rather than restricting our attention to the fully computable functions: we can easily list all possible algorithms, but we have no computable way of only listing the total ones. To prove that there is no such listing, we can apply a Cantor-style diagonal argument. For any proposed computable function

$$\Omega^* : (e, x) \mapsto \Psi_e(x)$$

which would act as a universal Turing machine for total computable functions *specifically*, we have the computable function

$$\Psi : x \mapsto \Omega^*(x, x) + 1.$$

By design, $\Psi(x) \neq \Psi_x(x)$, thus $\Psi \neq \Psi_x$ for every x. Yet Ψ can clearly be computed from Ω^*, which is itself computable, so it should be equal to Ψ_x for some x, given that Ω^* enumerates *all* computable functions. This is a contradiction. The same issue does not occur with Ω because $\Phi_x(x)$ might not halt, making it impossible to computably choose a different value for $\Phi(x)$.

A related example is the Halting Problem, which is the function H defined by

$$H : e \mapsto \begin{cases} 0 & \Phi_e(e) \uparrow \\ 1 & \Phi_e(e) \downarrow \end{cases}$$

One can prove that H is not computable by using H to compute an off-diagonal function:

$$D : e \mapsto \begin{cases} \Phi_e(e) + 1 & H(e) = 1 \\ 0 & H(e) = 0 \end{cases}$$

D is not computable because it differs from every computable function, yet $D \leq_T H$ (because $\Phi_e(e)$ *is* computable when one knows that it will halt), so H is not computable either.

Though H is not computable, it is "half computable" in the sense that one could verify $H(e) = 1$ computably, by finding a t such that $\Phi_e(e)[t] \downarrow$, but could not do the same if $H(e) = 0$. In general, for any algorithm Φ, the set

$$W_\Phi := \{e : \Phi(e) \downarrow\}$$

is called *computably enumerable* (or *c.e.*). This terminology comes from the fact that every c.e. set is the range of an injective computable function f, so that f enumerates the set, computably. To see this for W_Φ, let (e_j, t_j) be a computable enumeration of $\mathbb{N} \times \mathbb{N}$, and define f as the algorithm

$A \leftarrow \emptyset$
$j \leftarrow 0$
while $|A| < n$ **do**
  if $\Phi(e_j)[t_j] \downarrow$ **then**
    $A \leftarrow A \cup \{e_j\}$
  end if
  $j \leftarrow j + 1$
end while
Return e_j.

Conversely, if such an f exists for W, then W is the domain of

$$\Phi(x) := (\mu n)\big(f(n) = x\big).$$

Returning to the case of the halting problem, H is the domain of the function

$$x \mapsto (\mu t)\big(\Phi_e(e)[t] \downarrow\big),$$

and thus c.e., but it is not *co-c.e.*; that is, its complement $\overline{H}$ is not c.e. If a set A is computable then it is both c.e. and co-c.e., and in fact the converse is also true: if

$$x \in A \iff \Phi(x) \downarrow \iff \Psi(x) \uparrow,$$

then A can be computed by the function

$$A(x) = \Phi(x)\big[(\mu t)\big(\Phi(x)[t] \downarrow \vee\, \Psi(x)[t] \downarrow\big)\big] \downarrow.$$

Since it is guaranteed that either $\Phi(x)\downarrow$ or $\Psi(x)\downarrow$, this function is defined on all inputs.

The notion of a set being c.e. or co-c.e. can also be naturally relativized: we say A is c.e. *in* B if it is equivalent to the halting of a B-computable algorithm Φ^B.

Because H is non-computable, it follows that the Turing degree of H is above $\emptyset$. We will denote $\deg(H)$ by $\emptyset'$, which we call the *Turing jump* of $\emptyset$. In general, for any partial function f, the Turing jump f' is the function defined

$$f' : e \mapsto \begin{cases} 0 & \Phi_e^f(e)\uparrow \\ 1 & \Phi_e^f(e)\downarrow \end{cases}$$

and again $f' \not\leq_T f$. But the "jump" is in a sense bounded: while f' is not computable in f, it is still c.e. in f. Thus, the jump can be thought of as a standard increment of Turing complexity.

By repeatedly applying the Turing jump to $\emptyset$, we have an increasing chain of Turing degrees

$$\emptyset <_T \emptyset' <_T \emptyset'' <_T \emptyset^{(3)} <_T \dots$$

Thus, one can see that there are infinitely many different gradations of complexity that a function can have. In fact, there are far more than these.

1.2. **The Arithmetical Hierarchy.** So far, we have considered functions in general, but now we will focus in on $\{0,1\}$-valued functions, which we treat interchangeably with *subsets*[3] of $\mathbb{N}$. As we've seen, sets can be ordered and classified from the viewpoint of computability, using algorithms as the fundamental objects. Now, we will consider them from the viewpoint of arithmetic, whose central objects are *formulas*.

The language of first-order arithmetic, which we call $\mathcal{L}_1$, consists of the following symbols:

$$\text{Constants: } \{0,1\}, \quad \text{Functions: } \{+,\cdot\}, \quad \text{Relations: } \{<,=\},$$

along with symbols common to all languages:

$$\text{Quantifiers: } \{\forall,\exists\}, \quad \text{Logical Operators: } \{\neg,\wedge,\vee,\Rightarrow\}, \quad \text{and Variables: } \{a,b,c,\dots\}.$$

A *formula* in $\mathcal{L}_1$ is a finite string of these symbols which satisfies some simple syntactic rules (e.g. "$0 = +$" is not a sentence). We will also use parenthesis and commas for clarity. Here is an example of a formula:

$$\texttt{Prime}(x) := [\forall a, b]((a < x \wedge b < x) \Rightarrow (a \cdot b \neq x)) \wedge (x > 1).$$

Within a formula, a variable is called "bound" if it is quantified over. Otherwise it is called "free." In $\texttt{Prime}(x)$, as it is states above, x is a free variable and a, b are bound. A formula with no free variables is called a *sentence*.

Formulas and sentences are not inherently true or false—their truth values must be decided by an *interpretation* $\mathcal{M}$ of first-order arithmetic, which consists of a set M, called the "universe" of $\mathcal{M}$, and a specification of how the constants, functions, and relations of $\mathcal{L}_1$ act within M. These choices determine the truth values of all sentences. For example, one could define a model in which the result of all multiplication or addition is 0, and in this model $\texttt{Prime}(4)$ would be true. In our discussion, we'll stick with the *standard model* of first-order arithmetic, denoted $\mathbb{N}$, whose universe is $\{0,1,2,\dots\}$, and which interprets arithmetic in the expected way (i.e. satisfying the axioms of Peano Arithmetic).

With a model chosen, all formulas of $\mathcal{L}_1$ have corresponding sets. For example, in $\mathbb{N}$, the formula $\texttt{Prime}(x)$ given above acts as

$$\texttt{Prime}(0) = 0,\ \texttt{Prime}(1) = 0,\ \texttt{Prime}(2) = 1,\ \texttt{Prime}(3) = 1,\ \texttt{Prime}(4) = 0,\ \dots$$

and corresponds, naturally, to the set of prime numbers. The sets that can be expressed by formulas of $\mathcal{L}_1$ are called *arithmetically definable*, or just *arithmetical*.

Now, just as all sets can be classified along computability-theoretic lines by their Turing degrees, they can also be divided by their quantifier-complexity as formulas: this is called the *Arithmetical*

[3]In particular, for a set $A \subseteq \mathbb{N}$, we define $A : \mathbb{N} \to \{0,1\}$ such that $A(x) = 1$ if $x \in A$, and $A(x) = 0$ if $x \notin A$.

Hierarchy. Its lowest level, Δ_0, consists of the *bounded-quantifier* formulas—those whose quantifiers are all "bounded," as is the case with the $\forall y$ in the following formula:

$$\texttt{NotSquare}(x) := \forall y : (y < x) \Rightarrow (y \cdot y \neq x).$$

For any given x, $\texttt{NotSquare}(x)$ can be equivalently expressed as a finite conjunction

$$(0 \cdot 0 \neq x) \wedge (1 \cdot 1 \neq x) \wedge (2 \cdot 2 \neq x) \wedge \ldots \wedge (x \cdot x \neq x).$$

Of course, the number of terms in this conjunction depends on x, so $\texttt{NotSquare}(x)$ cannot readily be expressed without the use of the bounded quantifier, but we nevertheless consider such quantifiers to be fundamentally less complex than unbounded ones because they are "finitely-verifiable," roughly speaking. This is somewhat motivated by computability, as bounded-quantifier formulas can be computably checked (more on this later).

On the foundation of the quantifier-free formulas, we have the class Σ_1, which consists of formulas whose only (unbounded) quantifiers are existential, and similarly the class Π_1 whose only quantifiers are universal. For general n,

$$\Sigma_n = \{\exists y : \psi(y) \mid \psi \in \Pi_{n-1} \cup \Sigma_{n-1}\}$$
$$\Pi_n = \{\forall y : \psi(y) \mid \psi \in \Pi_{n-1} \cup \Sigma_{n-1}\}$$

so that for any $\psi \in \Delta_0$,

$$(\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)\ldots(Qx_n) : \psi(\vec{x}) \quad \in \Sigma_n$$

(whether Q is $\forall$ or $\exists$ depends on whether n is odd or even) and likewise for Π_n. Note that the negation of a Σ_n formula is Π_n and vice versa. Every formula of $\mathcal{L}_1$ has an equivalent expression in Σ_n or Π_n for some n, or Δ_0.

Now, given any definable set $A \subseteq \mathbb{N}$, we say that A is Σ_n if it is represented by *any* Σ_n formula. There will be many ways to express A as an $\mathcal{L}_1$-formula. For instance, one can always add on quantifiers with dummy variables to a sentence without affecting its truth value. Thus, if A is Σ_n, then it is Σ_{n+1} and Π_{n+1} as well.

Finally, if A is definable in *both* Σ_n and Π_n, we say that it is Δ_n. The arithmetical hierarchy consists of the classes $\Delta_n, \Sigma_n, \Pi_n$ for $n \in \mathbb{N}$ (we consider $\Delta_0 = \Sigma_0 = \Pi_0$ by convention). Viewed as a whole, we have the following diagram, ordered bottom-to-top by inclusion:

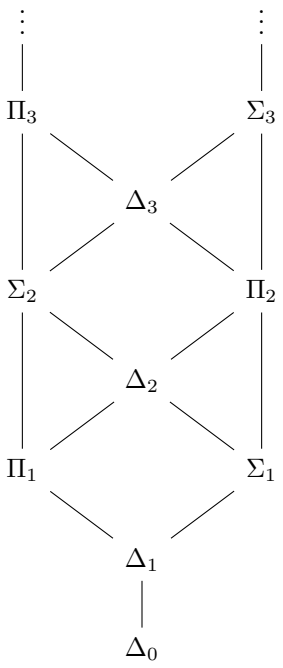


FIGURE 1. The Arithmetical Hierarchy.

An essential feature of the arithmetical hierarchy is its distinction between opposites; unlike in the Turing hierarchy, A and $\overline{A}$ are not necessarily of the same class. In the case[4] $A \in \Sigma_n \setminus \Pi_n$, this has the consequence that the *graph* of $A(x)$ may be more complex than either A or $\overline{A}$:

$$\text{“}A(x) = y\text{”} \;=\; \underbrace{(x \in A \wedge y = 1)}_{\Sigma_n} \vee \underbrace{(x \notin A \wedge y = 0)}_{\Pi_n} \;\in \Delta_{n+1}.$$

Finally, as with the Turing hierarchy, we can also define the arithmetical hierarchy Σ_n^B, Π_n^B, etc., relative to a base set B. These are defined in the same way, except that we expand the language $\mathcal{L}_1$ to include relation “$\in B$,” so that for example,

$$\text{“}x \in B\text{”} \;\in \Delta_0^B \quad \text{and} \quad \text{“}\forall y : x \cdot y \in B\text{”} \;\in \Pi_1^B.$$

1.3. **Mixing Arithmetic with Algorithmic.** At this point, we've established two separate notions of the complexity of sets: one originating from computability, and another from syntax. Now, we'll see that the two are closely linked.

There are two natural questions that we will answer in this section:

1 What are the computability properties of each arithmetic complexity class?
2 What are the arithmetical properties of each Turing degree?

In answer to question 1, we have the following lemma:

Lemma 1.1:

(i) Quantifier-free formulas are computable.
(ii) Δ_0 sets are computable.
(iii) Σ_1 sets are c.e., and Π_1 sets are co-c.e.
(iv) Δ_1 sets are computable.
(v) Σ_n sets are c.e. in $\emptyset^{(n-1)}$ and Π_n sets are co-c.e. in $\emptyset^{(n-1)}$.
(vi) Δ_n sets are $\emptyset^{(n-1)}$-computable.

Proof. (i) We can show this by induction on the number of symbols in the formula. The base case is the constants 0 and 1, which are clearly computable. For the inductive step, we must show that if a, b are computable terms, then

$$a + b, \;\; a \cdot b, \;\; a < b, \;\; a = b$$

are all computable, and that if φ, ψ are computable formulas, then

$$\varphi \wedge \psi, \;\; \varphi \vee \psi, \;\; \varphi \Rightarrow \psi, \;\; \neg\varphi$$

are all computable. These are all fairly straightforward, though not entirely trivial either, so I will show explicitly that $a = b$ is computable and leave the rest as simple exercises.

First, we can see that $|a - b|$ is computable by using primitive recursion:

$$|a - b| = \begin{cases} a & b = 0 \\ b & a = 0 \\ |(a-1) - (b-1)| & a, b \neq 0 \end{cases}$$

(here we use the fact that checking equality with 0 is computable by definition). Now,

$$a = b \iff |a - b| = 0$$

thus $a = b$ is computable.

(ii) Having shown that the computable sets include all quantifier-free definable sets, it suffices to show that they are closed under bounded quantification. If t is some computable term and $\varphi(x, y)$ is computable, then $(\exists y \leq t) : \varphi(x, y)$ can be defined recursively in t:

$$(\exists y \leq 0) : \varphi(x, y) \iff \varphi(x, 0).$$
$$(\exists y \leq t) : \varphi(x, y) \iff \varphi(x, t) \vee (\exists y < t - 1) : \varphi(x, y).$$

[4]Of course, we have not yet proven that Σ_n and Π_n are actually distinct classes, but that will come in the next section.

And bounded $\forall$ can be handled similarly.[5]

(iii) We know that the c.e. sets include all Δ_0 sets, so it suffices to show that the space of c.e. sets is closed under existential quantification. If $A(x,y)$ is $\{0,1\}$-valued and c.e. via $A(x,y) \iff \Phi(x,y)\downarrow$, then

$$\exists y : A(x,y) \iff \Big((\mu y)\Phi(x,y)\Big)\downarrow$$

thus $\exists y : A(x,y)$ is c.e. as desired. The statement for Π_1 formulas follows by negation.

(iv) Δ_1 sets are both Σ_1 and Π_1, so they are c.e. and co-c.e., hence computable.

(v) We'll use induction. The base case $n = 1$ is established by (iii). Thus, assume that the result is true for Σ_{n-1}, Π_{n-1}. For the inductive step, suppose

$$\varphi(x) = (\exists y) : \psi(x,y)$$

where $\psi \in \Pi_{n-1}$ or $\psi \in \Sigma_{n-1}$. In either case, by the inductive hypothesis, ψ is co-c.e. or c.e. in $\emptyset^{(n-1)}$ and hence computable in $\emptyset^{(n)}$. Thus, φ is c.e. relative to $\emptyset^{(n)}$ by (iii) relativized.

(vi) Δ_n sets are Σ_n and Π_n, hence c.e. and co-c.e. relative to $\emptyset^{(n)}$, hence computable in $\emptyset^{(n)}$. □

Thus, question 1 is resolved. As for question 2, we can now show converses to several of the results from the previous lemma.

Lemma 1.2:

(i) C.e. sets are Σ_1.
(ii) Computable sets are Δ_1.
(iii) $\emptyset^{(n)}$-c.e. sets are Σ_{n+1}, and $\emptyset^{(n)}$-co-c.e. sets are Π_{n+1}.
(iv) $\emptyset^{(n)}$-computable sets are Δ_{n+1}.

Proof. (i): We'll leverage the inductive definition of p.c. functions to show that for any algorithm Φ, the formula

$$\Phi(x) = y$$

in x, y (the graph of Φ) is Σ_1. Since

$$\Phi(x)\downarrow \iff (\exists y) : \Phi(x) = y,$$

this will imply that halting of all p.c. functions is Σ_1, as desired.

The collection of functions Φ whose graphs are Σ_1 clearly includes zero, successor, and projection, thus it suffices to show that this collection is closed under composition, primitive recursion, and unbounded search.

- *Composition*: Suppose that
$$f(\vec{x_1}, \vec{x_2}, \ldots, \vec{x_k}) = h(g_1(\vec{x_1}), g_2(\vec{x_2}), \ldots, g_k(\vec{x_k}))$$
where the graphs of g_j and h are all Σ_1. Then $f(\vec{x_1}, \ldots, \vec{x_k}) = z$ is equivalent to
$$\exists y_1, y_2, \ldots, y_k : (g_1(\vec{x_1}) = y_1) \wedge \ldots \wedge (g_k(\vec{x_k}) = y_k) \wedge (h(y_1, \ldots, y_k) = z)$$
which is Σ_1.
- *Primitive Recursion*: Suppose that $f(x,y)$ is recursively defined such that
$$f(0, \vec{y}) = g(\vec{y}) \quad \text{and} \quad f(x, \vec{y}) = h(f(x-1, \vec{y}), x, \vec{y})$$
and that the graphs of g, h are each Σ_1. Then $f(x, \vec{y}) = z$ is equivalent to
$$(\exists \sigma \in 2^{\mathbb{N}}) : (|\sigma| = x) \wedge \big(g(\sigma(\vec{y})) = \sigma(0)\big) \wedge \big((\forall 1 \leq j \leq x) : \sigma(j) = h(\sigma(j-1), j, \vec{y})\big)$$
which is Σ_1.

[5]Note that Δ_0 is not only computable but *primitive recursive*, i.e. all Δ_0 formulas can be computed *without unbounded search*. The converse is not true; one can define a primitive-recursive function that contradicts every Δ_0 formula by using a diagonal argument.

- *Unbounded Search*: Suppose that $f(x, \vec{y})$ is defined
$$f(\vec{y}) = (\mu x) g(x, \vec{y})$$
where g has Σ_1 graph. Then $f(x, \vec{y}) = z$ is equivalent to
$$(\exists \sigma \in 2^{\mathbb{N}}) : \big(g(z, \vec{y}) = 0\big) \wedge \big((\forall j < z) : g(j, \vec{y}) = \sigma(j) \wedge \sigma(j) \neq 0\big)$$
which is Σ_1.

Thus, the graph of every p.c. function is Σ_1, as desired.

(ii) Computable implies c.e. and co-c.e., which implies Σ_1 and Π_1 by (i), which implies Δ_1.

(iii) If A is c.e. in $\emptyset^{(n)}$, then it can be written
$$A(x) := \Phi^{\emptyset^{(n)}}(x) \downarrow$$
for some algorithm Φ. To get at the oracle $\emptyset^{(n)}$, we'll use the fact that halting with an oracle is equivalent to the halting with a finite initial segment of that oracle:
$$\Phi^{\emptyset^{(n)}}(x) \downarrow \iff \exists \sigma \in 2^{\mathbb{N}} : (\sigma \prec \emptyset^{(n)}) \wedge (\Phi^{\sigma}(x) \downarrow)$$
Now we check the complexities of $\sigma \prec \emptyset^{(n)}$ and $\Phi^{\sigma}(x) \downarrow$:

- $\Phi^{\sigma}(x)$ is computable (because σ is finite), so we have $\Phi^{\sigma}(x) \downarrow \in \Sigma_1$ by (i).
- $\sigma \prec \emptyset^{(n)}$ can be expressed
$$\sigma \prec \emptyset^{(n)} \iff \forall j < |\sigma| : \sigma(j) = \emptyset^{(n)}(j).$$
By the inductive hypothesis, $\emptyset^{(n)} \in \Sigma_n$, so its graph is Δ_{n+1}. Thus, $\sigma \prec \emptyset^{(n)} \in \Sigma_{n+1}$.

Putting these together,
$$A(x) = \exists \sigma : \Sigma_{n+1} \wedge \Sigma_1 \ \in \Sigma_{n+1}$$
as desired.

(iv) $\emptyset^{(n)}$-Computable implies c.e. and co-c.e. in $\emptyset^{(n)}$, which implies Σ_{n+1} and Π_{n+1} by (iii), which implies Δ_{n+1}. □

The results of both lemmas together amount to the statement of Post's Theorem:

Theorem 1.3 (Post):

- $A \in \Sigma_{n+1} \iff A$ is c.e. relative to $\emptyset^{(n)}$.
- $A \in \Pi_{n+1} \iff A$ is co-c.e. relative to $\emptyset^{(n)}$.
- $A \in \Delta_{n+1} \iff A \leq_T \emptyset^{(n)}$.

With Post's Theorem, we can now deduce that $\emptyset^{(n)} \in \Sigma_n \setminus \Pi_n$ and $\overline{\emptyset^{(n)}} \in \Pi_n \setminus \Sigma_n$, finally establishing that Σ_n and Π_n are truly distinct.

2. Theories of Second Order Arithmetic

In order to achieve the aim of reverse mathematics—to classify theorems by their proof-theoretic strength—first-order arithmetic is insufficient. Many important theorems state the existence of a certain set or function; the statement that two infinite fields k_1 and k_2 are isomorphic is of this form. Even real numbers, which we think of as fairly tame, cannot be put in correspondence with $\mathbb{N}$, and thus can only be expressed as sets or functions on $\mathbb{N}$. Therefore, to state most theorems requires a language which also takes sets and functions as its objects: *Second-Order Arithmetic*.

In first-order arithmetic, the objects under description are numbers. In second-order arithmetic, we also consider *sets* of numbers[6]. With the addition of sets as objects of study, we distinguish between variables representing first- and second-order objects, and add in a new relation, $\in$, denoting membership. Thus the language $\mathcal{L}_2$ consists of the following symbols:

- Constants: $\{0, 1\}$,
- Functions: $\{+, \cdot\}$,
- Relations: $\{<, =, \in\}$,
- Quantifiers: $\{\forall, \exists\}$,
- Logical Operators: $\{\neg, \wedge, \vee, \Rightarrow\}$,
- Number Variables: $\{a, b, c, \dots\}$,
- Set Variables: $\{A, B, C, \dots\}$.

Formulas in $\mathcal{L}_2$ can likewise be classified according to the arithmetical hierarchy, but we distinguish between quantifying over numbers and quantifying over sets: the complexity classes are denoted $\Delta^0_n, \Sigma^0_n, \Pi^0_n$ when referring to number-quantifiers, and $\Delta^1_n, \Sigma^1_n, \Pi^1_n$ for set-quantifiers[7]. Formulas without set-quantifiers are called *arithmetic formulas*.

An interpretation of second-order arithmetic, $\mathcal{M}$, has a universe of elements M and specifications of how the symbols $\{0, 1, +, \cdot, <\}$ act within M. The axioms determining the behavior of these symbols in M are PA^-, which is Peano Arithmetic without the axioms scheme of induction—we will reintroduce induction, but it may be weaker than the full arithmetical induction that PA enjoys. $\mathcal{M}$ must also specify a class $\mathcal{S} \subseteq 2^M$ collecting the *sets* of $\mathcal{M}$. The exact composition of $\mathcal{S}$ determines many properties of interest in $\mathcal{M}$; most theorems in mathematics are claims about the existence of sets with certain properties, after all.

The class of sets $\mathcal{S}$ must also satisfy certain axioms in relation to the first-order elements. The *full* theory of second order arithmetic consists of PA^- along with two more axioms: the first is *Set Induction*, which states

$$\forall X : (0 \in X) \wedge \Big((\forall n) : n \in X \Rightarrow n + 1 \in X\Big) \implies (\forall n) : n \in X.$$

That is, the property of membership in a set can be inducted upon.

The second is the axiom scheme of *Full Comprehension*, which consists of the statement

$$\exists X : \Big((\forall n) : n \in X \iff \varphi(n)\Big)$$

for each formula $\varphi \in \mathcal{L}_2$—this includes formulas φ with free variables (parameters), in which case the axiom is *universal* over all possible parameters. In other words, the collection of numbers satisfying a given arithmetical property form a set. Along with the set induction axiom, this automatically proves the axiom scheme of *Full Formula Induction*, which consists of

$$(\varphi(0)) \wedge \Big((\forall n) : \varphi(n) \Rightarrow \varphi(n + 1)\Big) \implies (\forall n) : \varphi(n)$$

for $\varphi \in \mathcal{L}_2$. Thus, all definable properties can be inducted upon.

Full second-order arithmetic also has a standard or "intended" model, where the first-order part is the standard model of first-order arithmetic, and $\mathcal{S} = \mathcal{P}(\mathbb{N})$. As far as most areas of mathematics are concerned, the axioms of full second-order arithmetic are simply True, and the intended model reflects the numbers and sets as they actually are.

[6]Similarly, nth order logic treats nested chains of sets of length at most n. In the language of ZFC, in contrast, chains of arbitrary (finite) depth are permitted, and they're all considered the same type of object.

[7]If the superscript is omitted, assume that it is 0.

Nevertheless, it is perfectly possible to develop the various branches of mathematics on more limited arithmetical foundations. One can do this by weakening the comprehension and induction axioms, restricting the formulas to which they apply, and thus reducing the range of sets which $\mathcal{S}$ must recognize. For example, for each arithmetical class $\Gamma \in \{\Sigma^0_n, \Pi^0_n\}$, we can consider the Γ-comprehension scheme, which only applies to $\varphi \in \Gamma$. Or we can restrict attention to Δ^0_n sets via the Δ^0_n-comprehension scheme, which applies to equivalent pairs $\varphi, \psi \in \Sigma^0_n, \Pi^0_n$:

$$\Big((\forall n) : \big(\varphi(n) \iff \psi(n)\big)\Big) \implies \Big((\exists X)(\forall n) : \big(n \in X \iff \varphi(n) \iff \psi(n)\big)\Big).$$

Likewise, we can consider Γ-induction.

By varying the complexity of comprehensible sets and inductible formulas, the truth of statements changes in response. Two formerly isomorphic fields would be separated if their underlying isomorphism lost its status as a set. Statements claiming the existence of a particular real number—the limit of a sequence or the zero of a polynomial, for instance—rely on the comprehension axiom to some extent. Every theorem of full second-order arithmetic has a precise level of axiom-complexity upon which it depends.

Reverse mathematics is interested in this interplay between theories of second-order arithmetic and the theorems they support. In this section, we will begin to define some of the most essential of these principles. At the same time, we will develop a formal study of $\mathbb{R}$ in second-order arithmetic, and compare various theories on what they are able and unable to prove about $\mathbb{R}$.

2.1. **The Baseline: RCA_0.** The most basic theory of second-order arithmetic is called RCA_0, or *Recursive Comprehension Axiom*. RCA_0 consists of the axioms of PA^- applied to its first-order elements, and the second-order axiom schemes of Δ^0_1-comprehension and Σ^0_1-induction.

By Post's Theorem, Δ^0_1-comprehension is equivalent to the existence of all computable functions. Thus, we can think of RCA_0 as a theory in which computing a set is sufficient and necessary to prove that it exists.

If $A, B \in \mathcal{S}$, then RCA_0 proves that the set $A \oplus B$ exists, via the computation

$$A \oplus B = \Big\{n : \Big((n \text{ odd}) \wedge ((n-1)/2) \in A)\Big) \vee \Big((n \text{ even}) \wedge ((n-2)/2) \in B)\Big)\Big\}$$

(note that "$\in A$" and "$\in B$" are part of the language because $A, B \in \mathcal{S}$). Similarly, $\mathcal{S}$ is closed under the effective join of any finite number of sets. Moreover, Δ^0_1-comprehension implies that $\mathcal{S}$ is downward-closed by $\leq_T$; if $A \leq_T B$ by $A(n) = \Phi^B(n)$ for some algorithm Φ, then $A \in \Delta^B_1 = \Delta^0_1$.

A collection of sets closed under $\oplus$ and $\leq_T$ is called a *Turing ideal*, thus we have just shown that RCA_0 proves $\mathcal{S}$ to be a Turing ideal. Equivalently, we can show that $\mathcal{S}$ is closed under composition, primitive recursion, and unbounded search. This is perhaps a natural place to start, since there isn't much room below the computable sets to prove anything; Δ^0_0-comprehension is too weak to really get anywhere.

Why Σ^0_1-induction, then? Δ^0_1-comprehension does prove that if a set is computable then it is in $\mathcal{S}$, but it is not actually strong enough to prove that all the functions we think of as computable actually are! In particular, it fails to show that even the primitive recursive functions are computable.

Suppose we define an algorithm Φ using primitive recursion, so that $\Phi(0)$ is computable and $\Phi(n+1)$ is computably determined from $\Phi(n)$ for each x. To show that such Φ are actually total requires the axiom

$$\Big(\Phi(0)\downarrow\Big) \wedge \Big((\forall n) : \Phi(n)\downarrow \Rightarrow \Phi(n+1)\downarrow\Big) \implies (\forall n) : \Phi(n)\downarrow,$$

which is exactly φ-induction for $\varphi(x) := \Phi(x)\downarrow$. Since $\Phi(x)\downarrow$ is Σ_1 by Post's Theorem, the axiom scheme of Σ_1-induction is sufficient to prove this fact. We can easily show (without the need for Σ_1-induction) that the collection of total functions is closed under composition, proving that all primitive recursive functions are total. Moreover, we can also show that this collection is closed under unbounded search (assuming that the goal of the search exists). Thus, Σ^0_1-induction is just the right level to allow for the use of all algorithms we might come up with.

As we've shown, every model of RCA_0 has a Turing ideal for its universe of sets. Thus, the minimal ω-model (i.e. model whose universe of numbers is $\{0, 1, 2, \dots\}$) of RCA_0 is one in which $\mathcal{S} = \Delta^0_1$. Because it is possible to model RCA_0 without any non-computable sets, without the existence of Turing jumps, etc., we immediately see that RCA_0 cannot prove the existence of any non-computable set. But RCA_0 doesn't prove that any non-computable sets *don't* exist, as any individual set can generate a Turing ideal and thus exist in a model of RCA_0.

RCA_0 is strong enough to prove many basic mathematical facts about the finite numbers and their arithmetical properties. When reasoning about objects containing infinite information, RCA_0 is limited in that it can only talk about computable things. This can come up even in seemingly tame settings—for example, each real number contains infinite information. The study of $\mathbb{R}$ provides a striking example of both the surprising strength and severe limitations of RCA_0[8].

We define $\mathbb{Q}$ as $\mathbb{Z} \times \mathbb{N}^+$ endowed with the operations and relations $+, \cdot, \leq, =$ given by

$$\begin{aligned}
(a, b) + (c, d) &\mapsto (ad + bc, bd) \\
(a, b) \cdot (c, d) &\mapsto (ac, bd) \\
(a, b) \leq (c, d) &\iff a \cdot d \leq b \cdot c \\
(a, b) = (c, d) &\iff a \cdot d = b \cdot c.
\end{aligned}$$

Note that $+, \cdot, \leq, =$ are all computable.

In defining $\mathbb{R}$, the classical way is to identify real numbers with Cauchy sequences in $\mathbb{Q}$, with two sequences being equivalent if they have the same limit. In the interest of making this definition more RCA_0-friendly, we'd like a computable way to get a rational approximation of any accuracy, which is not possible if we don't know anything about the rate of convergence. Thus, we define $\mathbb{R}$ as the set of "rapidly-converging" sequences in $\mathbb{Q}$,

$$\{(q_j) \in \mathbb{Q}^{\mathbb{N}} : (\forall i, j \in \mathbb{N}) : i < j \Rightarrow |q_i - q_j| < 2^{-i}\}.$$

We can think of a real number as a sequence of rational approximations whose error is bounded in a consistent way. Likewise, addition and multiplication are operations which take two sequences of approximations and produce one which approximates the sum or product.

To add two real numbers (a_j) and (b_j), the sequence $(a_j + b_j)$ will not work as their sum because the error of each approximation potentially doubles, losing the property of rapid convergence. Thus, we instead define addition by

$$(a_j) + (b_j) \mapsto (a_{j+1} + b_{j+1})$$

which ensures that the sum is still a real number.

In defining multiplication, the issue becomes even worse, as the error of $(a_j \cdot b_j)$ can be as high as $(a_j + b_j)2^{-j} + 2^{-2j}$. This error is bounded for all j by some $N \cdot 2^{-j}$, as $|a_j - a_0|, |b_j - b_0| \leq 1$ (specifically one can take $N = |a_0| + |b_0| + 3$). Thus, we can define multiplication by

$$(a_j) \cdot (b_j) \mapsto (a_{j+n} \cdot b_{j+n})$$

where n is chosen so that $N \cdot 2^{-j-n} < 2^{-j}$.

The most important thing about these somewhat awkward definitions of $+$ and $\cdot$ is that they are computable. The relations $\leq$ and $=$, on the other hand, cannot be made computable. We define them as follows:

$$\begin{aligned}
(a_j) \leq (b_j) &\iff (\forall j) : a_j - b_j \leq 2^{-j+1} \\
(a_j) = (b_j) &\iff (\forall j) : |a_j - b_j| \leq 2^{-j+1}.
\end{aligned}$$

The relations $(a_j) \leq (b_j)$ and $(a_j) = (b_j)$ are co-c.e. though not computable. If two real numbers really are equal, then one can never conclude this from only looking at finitely-many of their approximations. However, if one knows that two real numbers are different, then their order is computable.

One nice property of this definition of $\mathbb{R}$ is that from each *irrational* $x \in \mathbb{R}$, it is possible to compute the unique binary representation of x. The digits can be derived from comparisons with

[8]The formalization of $\mathbb{R}$ described here roughly follows [8].

finitely-many dyadic rationals, which we know to be computable. Moreover, if x is rational, then it may have multiple binary representations (e.g. $0.0\overline{111} = 0.1000$) but all of them are computable. In either case, the real number corresponding to any binary string can always compute that string.

We can leverage this fact to show a key weakness of computable mathematics: that RCA_0 fails to prove the completeness of $\mathbb{R}$.

Anti-Lemma 2.1 [**over** RCA_0]: $\mathbb{R}$ is complete, i.e. every Cauchy sequence has a limit.

Anti-Proof. The limit is unique if it exists, so it suffices to exhibit a computable sequence whose limit is a non-computable real number. RCA_0 can never prove the existence of any non-computable real number, therefore the statement must be unprovable in RCA_0.

In particular, we can construct a computable sequence whose limit corresponds to the binary expansion of $\emptyset'$, which is non-computable: let

$$c_j := \sum_{k=0}^{j} \Phi(k)[j]\downarrow \cdot\, 2^{-k}, \quad \text{and} \quad c := \lim_j c_j.$$

c_j is clearly Cauchy. But if c existed, then it could be used to compute its binary representation, which is $\emptyset'$. This would imply that $\emptyset'$ exists. However, RCA_0 does not prove this. Contradiction. □

We can conclude from this example that taking general limits an essentially non-computable problem. The limit of a given sequence is definable, but not computable. In order to prove that a limit exists, we need more sets.

2.2. **The Ceiling: ACA_0.** In extending RCA_0, one natural choice is to expand the class of formulas in the comprehension axiom scheme. But if comprehension is extended to even Σ^0_1 or Π^0_1, then it automatically implies comprehension for all arithmetic formulas:

Lemma 2.2 [**over** RCA_0]: Σ^0_1-comprehension is equivalent to full arithmetic comprehension.

Proof. We will induct on formula complexity. Assuming Σ_n and Π_n comprehension, we'll show that Σ_{n+1} and Π_{n+1} comprehension also hold.

Let $\varphi(x)$ be a formula of arithmetic complexity Σ_{n+1}, so that

$$\varphi(x) = \exists y : \psi(x, y)$$

where $\psi \in \Pi_n$. By the inductive hypothesis, there is a set

$$A := \{(x, y) : \psi(x, y)\}$$

and thus we can write $\varphi(x)$ as the Σ_1 formula

$$\varphi(x) = \exists y : (x, y) \in A.$$

Now φ-comprehension follows from Σ_1-comprehension.

Likewise, if $\varphi(x)$ is Π_{n+1}, then its complement is Σ_{n+1} and has a corresponding set, hence $\varphi(x)$ does as well, since the existence of complements follows from Δ_0-comprehension. □

Thus, the natural choice for extending the arithmetic domain of comprehension gives us ACA_0, the *Arithmetical Comprehension Axiom*. Naturally, ACA_0 has arithmetical induction as well. It is still weaker than *full* second-order arithmetic, however, because it does not allow comprehension and induction over Σ^1_n or Π^1_n formulas.

By Post's Theorem, ACA_0 is also equivalent (over RCA_0) to the statement that for every set X, there exists a set X' having the properties of the Turing jump of X. This implies, in particular, that in ω-models of ACA_0, $\mathcal{S}$ itself is closed under the Turing jump operation, in addition to Δ_1-comprehension. This characterization is convenient to work with, and we will use it often.

There are more powerful theories between ACA_0 and full second-order arithmetic, but for our purposes here, ACA_0 will be the ceiling. It is strong enough to prove essentially all of the familiar results about $\mathbb{R}$. For example, we can now go beyond what was possible in RCA_0 and prove the completeness of $\mathbb{R}$:

Lemma 2.3 [over ACA_0]: $\mathbb{R}$ is complete, i.e. every Cauchy sequence has a limit.

Proof. Let $x^0, x^1, x^2, \dots$ be a Cauchy sequence of in $\mathbb{R}$, with each x^j denoting the sequence of rational approximations $(x^j_0, x^j_1, \dots)$. For each k, define j_k such that

$$j_k = (\mu j)(\forall i > j) : |x^{j_k} - x^i| \leq 2^{-k}.$$

Now define the real number L by

$$L_k = x^{j_{k+1}}_{k+2}.$$

First, to show that this is actually a real number, for $m > k$,

$$\begin{aligned} |L_k - L_m| &\leq |x^{j_{k+1}}_{k+2} - x^{j_{k+1}}| + |x^{j_{k+1}} - x^{j_{m+1}}| + |x^{j_{m+1}} - x^{j_{m+1}}_{m+2}| \\ &\leq 2^{-k-2} + 2^{-k-1} + 2^{-m-2} \\ &\leq 2^{-k}. \end{aligned}$$

And moreover, L is the limit of x^j, as

$$(\forall j \geq j_k) : |L_k - x^j| \leq 2^{-k}.$$

□

If this proof were attempted in RCA_0, it would fail because RCA_0 cannot prove the sequence j_k exists; it is definable, but not computable. The core issue is that if a sequence has *unknown* convergence rate, then there is no way to compute its limit. In contrast, RCA_0 can prove that nested intervals with length approaching 0 will converge around a limit point, since the closeness of the approximation is known at every stage.

More surprising is the fact that the converse is also true: the completeness of $\mathbb{R}$ *implies* ACA_0!

Theorem 2.4 [over RCA_0]: Completeness of $\mathbb{R} \implies \mathsf{ACA}_0$.

Proof. It suffices to show that if $\mathbb{R}$ is complete then for any set A, the Turing jump A' exists. Let

$$c_j := \sum_{k=0}^{j} \Phi^A(k)[j]\downarrow \cdot 2^{-k}.$$

Each c_j is A-computable, therefore the sequence (c_j) exists. It's also clearly Cauchy. By the completeness of $\mathbb{R}$, this c_j has a limit $c \in \mathbb{R}$. From c, one can compute all of the binary digits of c, which correspond to values of A'. Thus, A' exists. □

This is our first concrete example of a reverse mathematical result—we have established an equivalence between an axiomatic theory and a result of that theory.

2.3. **König's Lemma and Weak König's Lemma.** Several of the set-existence principles discussed so far have been *problems*, by which we mean statements of the form

$$\forall X : \Theta(X) \implies \exists Y : \Gamma(X, Y).$$

$\Theta(X)$ means that X is an *instance* of the problem, and $\Gamma(X, Y)$ means that Y is a *solution* to the instance posed by X. For example,

"for every set X there exists a set Y so that Y is the Turing jump of X,"

and

"for every sequence (x_j) there is a real number y such that y is the limit of (x_j)."

These two problems each have the property that there can be at most one solution. But one can also consider problems with potentially many different solutions. We study problems from a reverse-mathematical perspective by treating them as axiomatic additions to RCA_0.

A *tree* is a set of finite sequences which is downward closed (i.e. if $\sigma \in T$ then all of its initial segments are in T). A *path* in T is an infinite sequence whose initial segments are all in T. We use the notation $[\,T\,]$ to mean the set of paths in T.

Definition (König's Lemma or KL): König's Lemma is the statement that for every infinite tree $T \subseteq \mathbb{N}^{\mathbb{N}}$ in which each node has *finite degree*, there exists an infinite path in T.

Imagine standing at the root node of T and staring out at an infinite expanse of pathways in front of you. Each step forward advances you down one sequence, and you cannot go back once a step is taken. Can you manage to avoid hitting any dead ends?

The set of nodes in T which extend to paths is Π_2 relative to T. Thus, with knowledge of ACA_0 sets, it is possible to find such a path. With only computable sets, there seems no way forward; indeed, ACA_0 is sufficient *and necessary* to prove König's Lemma:

Theorem 2.5 [over RCA_0]: $\mathsf{ACA}_0 \iff \mathsf{KL}$.

Proof. ($\mathsf{ACA}_0 \implies \mathsf{KL}$): Given $T \subseteq \mathbb{N}^{\mathbb{N}}$, we can prove the existence of a path $P \in [\,T\,]$ as follows: first, ACA_0 shows the existence of a set $\mathrm{Ext}(T) \subseteq T$ defined by

$$\sigma \in \mathrm{Ext}(T) \iff (\forall n)(\exists \tau \in \mathbb{N}^n) : \tau \succ \sigma \wedge \tau \in T.$$

Now using $\mathrm{Ext}(T)$, we define a path $\ell := (\sigma_0, \sigma_1, \dots)$ recursively via

$$\sigma_{j+1} = \sigma_j^\frown x_j \text{ where } x_j := (\mu x)\big(\sigma_j^\frown x_j \in \mathrm{Ext}(T)\big)$$

Every initial segment of ℓ is in $\mathrm{Ext}(T)$ and hence in T, so it suffices to show that ℓ is infinite.

We can inductively prove that ℓ contains a sequence of length $\geq n$ for each n: for $n = 0$ it is vacuous. For the inductive step, suppose $\sigma_n \in \ell$ with $|\sigma_n| = n$. $\sigma_n \in \mathrm{Ext}(T)$, so it has children $\tau_1, \tau_2, \dots, \tau_k \in T$. If $\tau_j \notin \mathrm{Ext}(T)$ for all these τ_j, then each one has a largest depth d_j of all its descendants, but then σ_n's descendants would be bounded in depth by $\max\{\tau_1, \tau_2, \dots, \tau_k\}$, a contradiction of $\sigma_n \in \mathrm{Ext}(T)$, thus some $\tau_j \in \mathrm{Ext}(T)$, hence σ_{n+1} is well-defined.

($\mathsf{KL} \implies \mathsf{ACA}_0$): We will show via KL that for every set A, the Turing jump A' exists. Define a tree T as follows: a sequence σ belongs to T if for all $e < |\sigma|$, either $\sigma(e)$ is the halting time of $\Phi_e^A(e)$, or $\sigma(e) = 0$ and $\Phi_e^A(e)$ has not halted at time $|\sigma|$. This is indeed a tree because every requirement of membership for σ is also required of its extensions.

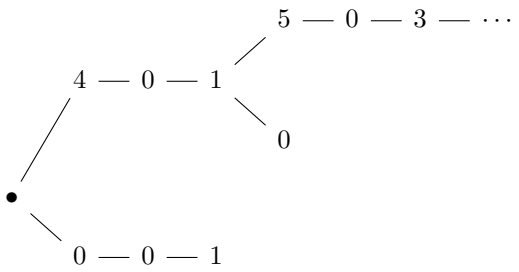


FIGURE 2. The tree T, in the case where the path begins $4, 0, 1, 5, 0, 3, \dots$

Membership in T is computable (since checking a *particular* halting time can be done in finite time) thus RCA_0 proves that this tree T exists. For each e, there are exactly two possible values that $\sigma(e)$ could ever have for any $\sigma \in T$: either 0, or the halting time (if it exists). If $\Phi_e^A(e)$ does halt, then for sufficiently short sequences σ, it is possible that both values could appear in T.

Every $\Phi_e^A(e)$ has either a correct halting time or never halts, thus there is an infinite path in T consisting of these correct answers. No other infinite path can be contained in T because whether or not $\Phi_e^A(e)$ halts, there is a depth beyond which no node $\sigma \in T$ can be wrong about e: if it halts, this depth is the halting time, and if it diverges, then no σ can be wrong at *any* depth. Thus, the set produced by KL is this path, from which A' can easily be computed. Thus A' exists. □

It might be somewhat surprising that a problem like KL can prove the comprehension axiom for every arithmetic formula. In the T constructed for the previous proof, though $\sigma \in T$ can be computed, the questions of whether σ has any children and how many children σ has are both non-computable. This is why non-computable information—the halting time of $\Phi_e^A(e)$—can be coded in the paths of T.

We might also ask about the strength of a weakened version of KL; one which bears only on trees where the nodes of a fixed depth can be explored computably:

Definition (Weak König's Lemma or WKL): WKL states that for any infinite *binary* tree T (considered as a subset of $2^{\mathbb{N}}$), T contains an infinite path.

Note the distinction between *binary* and *degree-at-most-2*. The tree utilized in the proof that $\mathsf{KL} \Rightarrow \mathsf{ACA}_0$ has degree at most 2 at each node, but from the perspective of computability it is not a binary tree, since there are more than 2 (indeed, infinitely many) *potential* children of each node, all of which would need to be checked in order to compute the tree up to a fixed depth.

Weak König's Lemma is equivalent to the strengthened statement for trees $T \in \mathbb{N}^{\mathbb{N}}$ whose nodes are bounded by some computable function b, in the sense that

$$\sigma \in T \implies \sigma(n) < b(n)\ \forall n.$$

Like binary trees and unlike general trees in $\mathbb{N}^{\mathbb{N}}$, these bounded trees can be computably explored up to any finite depth. For any computably bounded tree T, define

$$f : T \to 2^{\mathbb{N}}$$

by mapping each σ to the concatenation of the binary representations of $\sigma(j)$ for each j, with $\lceil \log_2(b(j)) \rceil$ bits in each place.

$$\boxed{5}, \boxed{3}, \boxed{19}, \boxed{4}, \boxed{11}, \ldots$$
$$\Downarrow$$
$$\boxed{1,0,1}, \boxed{0,0,0,0,0,0,1,1}, \boxed{1,0,0,1,1}, \boxed{0,0,1,0,0}, \boxed{1,0,1,1}, \ldots$$

Let $S \subseteq 2^{\mathbb{N}}$ be defined by $\sigma \in S$ iff σ is an initial segment in some $f(\tau)$. Since $b(n)$ is computable, this S is T-computable, and any path in S corresponds to a path in T. Thus, the computably-bounded case of KL reduces to WKL.

We denote the theory $\mathsf{RCA}_0 + \mathsf{WKL}$ as WKL_0. A priori, it's not clear whether WKL_0 is actually weaker than ACA_0, or even whether it's stronger than RCA_0. This question will be resolved in the next section. For now, let's get an idea of what WKL_0 can do on its own.

From the perspective of real analysis, WKL_0 is notable for its ability to prove compactness results:

Theorem 2.6 (Heine-Borel) [over WKL_0]: The closed interval $[0,1] \subset \mathbb{R}$ is compact; that is, every countable collection of open intervals (c_i, d_i) covering $[0,1]$ has a finite subcover.

Proof. To each finite binary string $\sigma \in 2^{\mathbb{N}}$, we associate a closed interval $[a_\sigma, b_\sigma]$ defined by

$$a_\sigma := \sum_{k<|\sigma|} \sigma(k) \cdot 2^{-k+1}, \quad b_\sigma := a_\sigma + 2^{-|\sigma|}$$

So that each $[a_\sigma, b_\sigma]$ is partitioned in half by the intervals $[a_{\sigma,0}, b_{\sigma,0}]$ and $[a_{\sigma,1}, b_{\sigma,1}]$.

The idea behind this proof is this: we would like to find a tree T such that

(1) $\sigma \notin T \implies (\exists i \in \mathbb{N}) : [a_\sigma, b_\sigma] \subset (c_i, d_i)$,
(2) $\ell \in [T] \implies (\forall \sigma \prec \ell)(\forall i \in \mathbb{N}) : [a_\sigma, b_\sigma] \not\subset (c_i, d_i)$.

Suppose such a T exists. Every path $\ell \in [T]$ corresponds to a real number $x = (a_\sigma)_{\sigma \prec \ell} \in [0,1]$. Because (c_i, d_i) are an open cover, there is some i for which $x \in (c_i, d_i)$, and thus some σ for which

$$x \in [a_\sigma, b_\sigma] \subset (c_i, d_i).$$

Thus, by (2), $[T] = \emptyset$. By WKL_0, this implies that T is *finite*. Then there is a depth d at which $|\sigma| = d \Rightarrow \sigma \notin T$, which implies by (1) that each of the 2^d intervals $[a_\sigma, b_\sigma]$ is covered by some (c_i, d_i). These 2^d covering intervals cover all of $[0, 1]$.

Now we show that RCA_0 proves the existence of such a tree. Let

$$\varphi(\sigma) := (\forall i \in \mathbb{N}) : [a_\sigma, b_\sigma] \not\subset (c_i, d_i).$$

Note that

$$[a_\sigma, b_\sigma] \not\subset (c_i, d_i) \iff (a_\sigma \leq c_i) \vee (b_\sigma \geq d_i)$$

which is co-c.e., thus $\varphi(\sigma)$ is also co-c.e. (Π_1 is co-c.e. by Post's theorem). Thus, let Φ be an algorithm such that $\Phi(\sigma) \uparrow \iff \varphi(\sigma)$. We define T by

$$\sigma \in T \iff (\forall \tau \prec \sigma) : \Phi(\tau)[|\sigma|] \uparrow .$$

This T is computable, and one can check that it satisfies (1) and (2). □

And conversely, WKL_0 is *exactly* the necessary axiom to prove this:

Theorem 2.7 [over RCA_0]: Heine-Borel $\implies \mathsf{WKL}_0$.

Proof. Let $T \subseteq 2^{\mathbb{N}}$ be a binary tree with no infinite paths. We can map $2^{\mathbb{N}}$ bijectively to the Cantor set via

$$f : \sigma \mapsto \sum_k \sigma(k) \cdot \frac{2}{3^{k+1}}.$$

For $\sigma \in 2^{\mathbb{N}}$, define $f(\sigma)$ similarly. Let a_σ, b_σ be the points

$$a_\sigma := f(\sigma) - 3^{-|\sigma|-1}, \quad b_\sigma := f(\sigma) + 4 \cdot 3^{-|\sigma|-1}$$

so that

$$f(\tau) \in (a_\sigma, b_\sigma) \iff \tau \succ \sigma.$$

Let $U \subseteq 2^{\mathbb{N}}$ be the set of $\sigma \notin T$ for which the parent of σ is in T. We consider the collection of open intervals

$$\mathcal{U} := \{(a_\sigma, b_\sigma)\}_{\sigma \in U}.$$

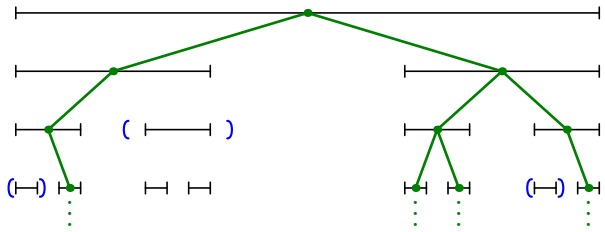


FIGURE 3. T embedded into the Cantor set. The blue intervals are $\mathcal{U}$.

Because there are no infinite paths in T, every infinite binary sequence ℓ has a smallest initial segment σ where it first exits T, for which the real number $f(\ell)$ is in (a_σ, b_σ). This shows that $\mathcal{U}$ covers all of $f(2^{\mathbb{N}})$, i.e. the Cantor set.

Now, Heine-Borel implies that $[0, 1]$ is compact, and the Cantor set is closed in $[0, 1]$ hence it is compact as well. Thus there must be a finite subcover of the Cantor set among $\mathcal{U}$, represented by some $\sigma_1, \sigma_2, \ldots, \sigma_j \in U$. Every $\tau \in T$ must be an initial segment of one of these σ_i, so T is bounded in depth by $\max\{|\sigma_1|, |\sigma_2|, \ldots, |\sigma|_j\}$, and hence finite. □

These results suggest the idea that WKL_0 is the "combinatorial core" of compactness, one of the many insights to come out of reverse mathematics. But the question remains: is WKL actually weaker than KL over RCA_0?

3. Separation of Problems

The goal for this section is to prove that WKL_0 does not prove KL, but we will approach the question more generally: given two problems A and B, how can we prove that $\mathsf{A} \not\Longrightarrow \mathsf{B}$? We do it by building a model of A in which B does not hold. If there is such a model, then that immediately precludes any proof of $\mathsf{A} \Longrightarrow \mathsf{B}$.

We've already implicitly used this idea in a simple form when showing that RCA_0 does not prove the existence of any non-computable set. Similarly, by again working with the $\mathcal{S} = \Delta_1$ model of RCA_0, we can show that WKL_0 is stronger than RCA_0:

Theorem 3.1: $\mathsf{RCA}_0 \not\Longrightarrow \mathsf{WKL}_0$.

Proof. In this model, we can construct a binary tree T in S for which there is no path in S. In particular, we will ensure that the path is not computable by making the eth element of all paths differ from $\Phi_e(e)$: define T by

$$\sigma \in T \iff (\forall e < |\sigma|) : \neg\big(\Phi_e(e)[|\sigma|]\downarrow \wedge\ \sigma(e) = \Phi_e(e)\big).$$

That is, T severs a path at the depth when it first detects that $\sigma(e) = \Phi_e(e)$ for any e where $\Phi_e(e)$ halts, and it eventually detects all such discrepancies. Thus, any infinite path (there are many) must disagree with all computable functions, and hence is non-computable. □

Note the difference in strength between KL and WKL here: from the perspective of ACA_0, as shown in the previous section, there exists a computable tree $T \subseteq \mathbb{N}^{\mathbb{N}}$ whose only path computes $\emptyset'$. In contrast, among *binary* trees $T \subseteq 2^{\mathbb{N}}$, we can force all paths to be non-computable but cannot hit any particular complexity. We could say that KL *avoids* solutions below $\emptyset'$, whereas WKL only avoids (as far as we know) solutions equivalent to $\emptyset$. These are examples of *strength* properties. To separate WKL from KL, we will show that WKL holds some *weakness* property.

Showing that a problem is not provable in RCA_0 is a convenient case because RCA_0 has a standard model that we already understand well. But what does a model of WKL_0 look like? Here, there is not a simple choice for $\mathcal{S}$ we can just use—we have to make one ourselves. This section will be about the process of model construction and the many tools one can use during that process.

3.1. Methods of Model-building.

Suppose that we want to construct a model of $\mathsf{B} \wedge \neg \mathsf{A}$, where A, B are two principles between RCA_0 and ACA_0.

To construct a model of B is fairly simple: you just keep adding solutions to B instances until (after $\mathbb{N}$-many steps) everything has a solution. Define sets $Z_0 \leq_T Z_1 \leq_T \dots$ inductively as follows: first, let $Z_0 = \emptyset$. To define Z_j for $j \geq 1$, let

$$X := \text{ the } e_j\text{th set computed by } Z_{n_j}$$

where $j \mapsto (e_j, n_j)$ is an ordering of $\mathbb{N} \times \mathbb{N}$ such that $n_j < j$ (so that Z_j is not defined referencing itself). If this X is a B-instance without a Z_{j-1}-computable solution, choose a solution Y and let

$$Z_j := Z_{j-1} \oplus Y.$$

Otherwise we let $Z_j = Z_{j-1}$. Finally, define $\mathcal{S}$ to be the Turing ideal generated by the sets Z_j.

For every B-instance $X \in \mathcal{S}$, there is a finite stage n in which X is first computable from Z_n, and X is the eth set computable from Z_n for some e, so there must be a stage $j > n$ for which $(e_j, n_j) = (e, n)$ and so a solution to X is added (or already exists).

The difficulty is not in modeling B, but in *not* modeling A. It suffices to pick a particular A-instance and deliberately avoid adding any of its solutions to $\mathcal{S}$ at any stage. In the case of $\mathsf{A} = \mathsf{KL}$, for example, we might avoid adding $\emptyset'$. But moreover, one also has to avoid adding sets which compute $\emptyset'$, and instances whose only solutions compute $\emptyset'$, and so on. Thus, we look for a sharp dividing line between sets we must add (e.g. $\emptyset$) and sets we don't want to add (e.g. $\emptyset'$) for which we can keep the construction on the right side of the line.

Definition (Preservation and Avoidance): Let $\mathcal{Q}$ be some collection of sets that is downward-closed under $\leq_T$ (but not necessarily a Turing ideal). We say that B *preserves* $\mathcal{Q}$ if for every $Z \in \mathcal{Q}$ and every B-instance $X \leq_T Z$, there is a solution Y such that $Y \oplus Z \in \mathcal{Q}$.

If $\mathcal{Q}$ is upward-closed under $\leq_T$ and B preserves $\overline{\mathcal{Q}}$, then we say B *avoids* $\mathcal{Q}$.

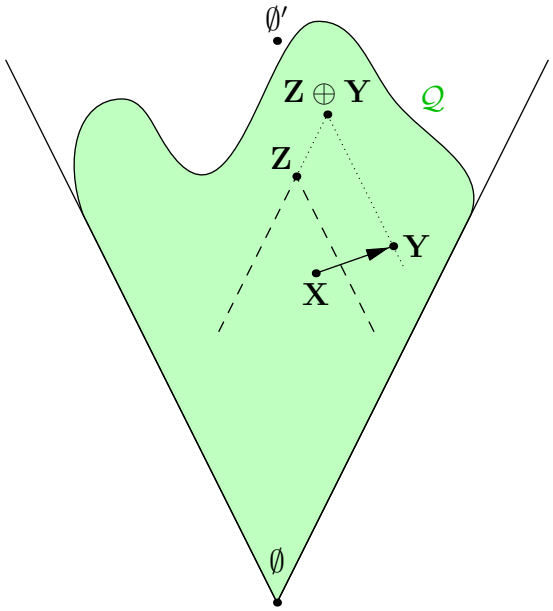


FIGURE 4. $\mathcal{Q}$-preservation inside the order $\leq_T$. Here, $\mathcal{Q}$ separates $\emptyset$ from $\emptyset'$.

Clearly, if we find a suitable $\mathcal{Q}$ and prove that B preserves $\mathcal{Q}$, then a model can be constructed as above while keeping $Z_0, Z_1, Z_2, \cdots \in \mathcal{Q}$, and hence $\mathcal{S} \subset \mathcal{Q}$, since by construction every set in $\mathcal{S}$ is computable from some Z_j.

Now the question becomes choosing a suitable class as $\mathcal{Q}$. Below are some key examples of $\mathcal{Q}$ commonly used in preservation and avoidance arguments:

- **Cone Avoidance**: Let $\mathcal{Q}$ be the "cone above" some non-computable set C, i.e.

  $$\mathcal{Q} := \{A : C \leq_T A\}.$$

  In this case, if B avoids $\mathcal{Q}$ for all C, we say that B *admits cone avoidance*.

- **Low Degree**: Let $\mathcal{Q}$ be the collection of *low* sets, i.e.

  $$\mathcal{Q} := \{A : A' \equiv_T \emptyset'\}.$$

  If B preserves $\mathcal{Q}$, we say that B *admits low solutions*.
  There is a similar notion of low_n for each $n \in \mathbb{N}$, where $\mathcal{Q}$ is defined

  $$\mathcal{Q} := \{A : A^{(n)} \equiv_T \emptyset^{(n)}\}.$$

- **Hyperimmune-free**: A function f is said to be *hyperimmune* if it is not dominated (i.e. bounded above except at finitely-many values) by any computable function.

  A set is *hyperimmune-free* if it does not compute any hyperimmune function. Being hyperimmune-free is clearly downward-closed under $\leq_T$. If B preserves

  $$\mathcal{Q} := \{A : A \text{ is hyperimmune-free}\},$$

  we say that B *admits hyperimmune-free solutions*.

- **PA Degree**: A is of *PA degree* if every computable tree $T \subseteq 2^{\mathbb{N}}$ has an A-computable path[9]. From our discussions of KL and WKL so far, it follows that $\emptyset$ does not have PA degree, but $\emptyset'$ does. If B avoids
$$\mathcal{Q} := \{A : A \text{ has PA degree}\},$$
we say that B *admits PA avoidance*.

Note that all four of these classes $\mathcal{Q}$ separate $\emptyset$ from $\emptyset'$ (choosing $C = \emptyset'$ in the case of cone avoidance). In general, one can relativize each one to separate other pairs of sets.

To make a model of WKL_0 and not KL, it will suffice to show that WKL has any of the above four properties. In fact, we will show all of them except PA avoidance, which WKL does not admit.

3.2. **Forcing and its Uses.** To show any $\mathcal{Q}$-preservation for WKL entails constructing a path Y through each Z-computable binary tree such that $Y \oplus Z \in \mathcal{Q}$. To make this and similar constructions easier to follow, we'll use a system of bookkeeping called *forcing*.

Forcing is a method of specifying an object (typically a subset of $\mathbb{N}$), which we call $\mathtt{G}$, through a series of approximations. The construction follows a descending path through a partially-ordered collection of *conditions*, denoted P. In each stage, the condition p is *extended* to some $p^* \leq p$ (the extension is considered *lower* in the order because it allows a smaller space for possible sets $\mathtt{G}$).

To instantiate a notion of forcing, then, requires a partially-ordered set $(P, \leq)$, and an order-preserving interpretation of each $p \in P$ as a formula. We also require that P enjoy the "saturation" property that every descending chain of conditions is satisfied by some $\mathtt{G}$.

Before explaining how to use forcing to construct sets with specific desirable properties, let's see a couple of examples of different notions of forcing, each with its own interpretation of P:

Cohen Forcing:
- $P = 2^{\mathbb{N}}$, ordered by $p^* \leq p := p^* \succeq p$.
- The condition p is interpreted as $p \prec \mathtt{G}$.

Jockusch-Soare Forcing:
- P is the set of infinite *computable* sub-trees of $2^{\mathbb{N}}$, ordered by $p^* \leq p := p^* \subseteq p$.
- The condition p is interpreted as $\mathtt{G} \in [p]$, i.e. $\mathtt{G}$ is a path in p.
- Unlike the other two forcing notions here, it is not immediately clear that a descending sequence of Jockusch-Soare conditions is satisfied (i.e. a descending sequence of trees has a common path). But this actually follows from the compactness of $[2^{\mathbb{N}}]$: each $[p]$ is a closed (and thus compact) subset of $[2^{\mathbb{N}}]$, and a descending sequence of compact sets always has a nonempty intersection.

Mathias Forcing:
- P is the set of pairs (E, R), where E is a finite set and R is an infinite set, with $E < R$. The order is $(E^*, R^*) \leq (E, R)$ if $E^* \supseteq E \cup R$ and $R^* \subseteq R$. We think of R as the "reservoir" of elements that are available to be added to E.
- (E, R) is interpreted as $E \subseteq \mathtt{G} \subseteq E \cup R$.

Notice that each of these three forcing notions has a different range of formulas that its conditions can articulate. For example, consider the two statements
$$\text{"}\mathtt{G} \text{ is infinite" and "}\overline{\mathtt{G}} \text{ is infinite."}$$

In Cohen forcing, there is no condition which could decide either of these for $\mathtt{G}$, since at any point the undecided elements could all be put in $\mathtt{G}$ or $\overline{\mathtt{G}}$.

In Jockusch-Soare forcing, there are conditions deciding both—in fact, a condition can specify any computable $\mathtt{G}$ entirely by being a tree with only one path.

In Mathias forcing, there is an asymmetry: no condition can decide whether $\mathtt{G}$ is infinite because at condition (E, R), $\mathtt{G}$ could be all of $E \cup R$ (infinite) or only E (finite) or anything in between. However, (E, R) *can* decide that $\overline{\mathtt{G}}$ is infinite, when $\overline{R}$ is infinite.

[9]The name "PA degree" comes from the fact that a set A has PA degree if and only if it can compute a complete, consistent extension of Peano Arithmetic. Another equivalent condition is that A can compute a function which can accurately tell *which* of two algorithms halts, given that at least one does.

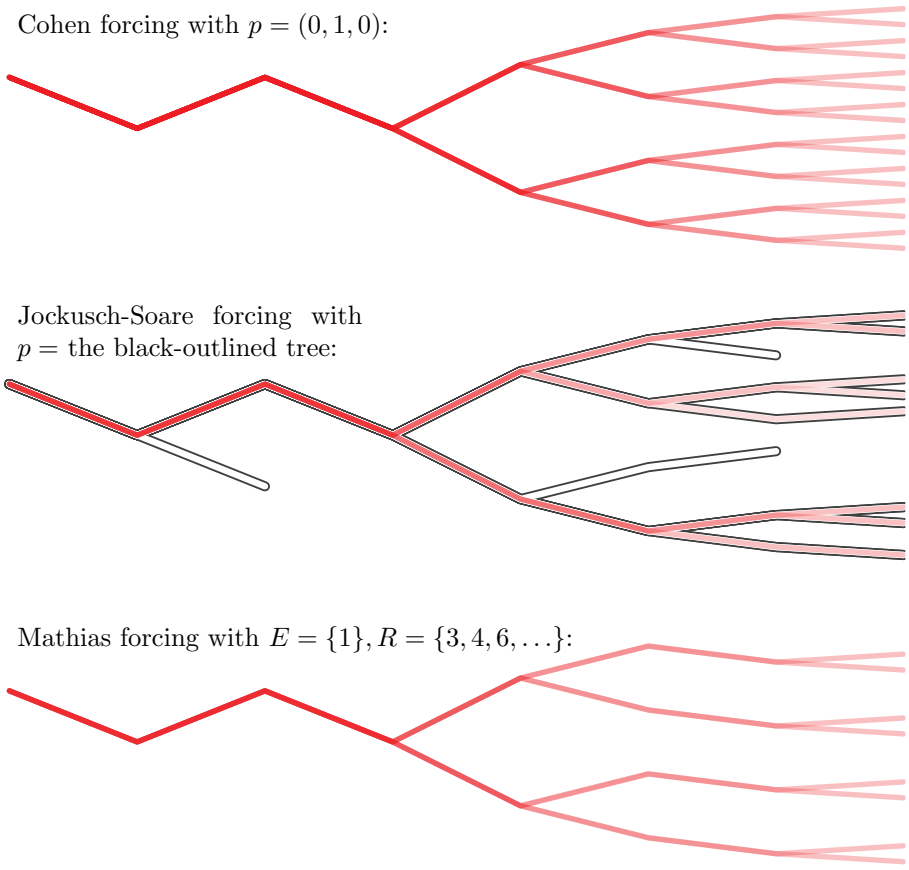


FIGURE 5. The space of sets (visualized as paths in $2^{\mathbb{N}}$) obeying each of three conditions.

Though they differ in precision, all three types of conditions are considerably less precise than second-order arithmetic in general. This is what P gives up in exchange for its saturation property. Conditions in P cannot usually enforce the properties that we want in $\mathtt{G}$, but we can try to get at these properties through infinite descending sequences in P, and get a corresponding $\mathtt{G}$ using saturation.

For example, suppose we are working in the setting of Cohen forcing and we want to produce a $\mathtt{G}$ which is infinite. As we know, there is no individual condition in P that can guarantee this. However, for each $n \in \mathbb{N}$ there are $p \in P$ which imply $|\mathtt{G}| > n$. In fact, the property $|\mathtt{G}| > n$ is equivalent to obeying at least one condition $p \in P$ which has more than n 1's. If the set of such p is denoted $E_n \subset P$, then we say

$$|\mathtt{G}| > n \iff \mathtt{G} \text{ “meets” } E_n.$$

To meet a condition-set means to obey at least one of its conditions. We can express $|\mathtt{G}| = \mathbb{N}$ by

$$|\mathtt{G}| = \mathbb{N} \iff (\forall n) : \mathtt{G} \text{ meets } E_n.$$

So if we have a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq \cdots$$

for which $p_n \in E_n$ for each n, then the $\mathtt{G}$ approximated by these conditions will meet each E_n and hence be infinite. And it is easy to construct such a sequence, since any condition at all can be extended to one in E_n, which is to say that E_n is *dense* in P.

Of course, this is a very roundabout way of proving that an infinite set exists when one can much more easily name a particular infinite set. But when the desired properties are more complicated, forcing becomes increasingly helpful as a way of organizing the construction of $\mathtt{G}$.

In general, when we want a set $\mathtt{G}$ with some particular property $F(\mathtt{G})$, we express F as a countable list of *dense* condition-sets $D_e \subset P$,

$$(D_0, D_1, D_2, \dots)$$

such that if $\mathtt{G}$ meets each D_e then $F(\mathtt{G})$ holds. In all of the cases in this paper, meeting D_e will be equivalent to some definable property of $\mathtt{G}$, but this is not always the case in general. Using density, we produce a descending sequence of conditions

$$p_0 \geq p_1 \geq p_2 \geq \cdots$$

where $p_e \in D_e$ for each e.

Expressing F as a countable conjunction of dense condition-sets—and proving that they are dense—is the essential piece of any forcing argument.

A descending sequence $p_0 \geq p_1 \geq \dots$ which meets every dense condition-set in some collection is called *generic*, and a set $\mathtt{G}$ obeying it is called a generic set. The name "generic" is quite fitting, since it seems much more unusual to *not* meet a dense set at $\mathbb{N}$-many opportunities than to meet it once. If there exists any sequence of dense condition-sets implying F, as described above, then we say that $F(\mathtt{G})$ holds for any *sufficiently generic* $\mathtt{G}$. In such a situation, we say that F is *forced*. It may also be that given a certain condition p, any sufficiently generic $\mathtt{G}$ obeying p is forced to hold $F(\mathtt{G})$, in which case we say p forces F.

Note that if F_1, F_2 are both forced, with the two corresponding sequences of dense condition-sets (D^1_e) and (D^2_e), then $F_1 \wedge F_2$ is also forced, with the sequence

$$(D^1_0, D^2_0, D^1_1, D^2_1, \dots).$$

And naturally the same is true for any countable collection of forced formulas. Thus, conveniently, we may treat every desired property separately.

When aiming to construct a set by forcing, the choice of which forcing notion to use is critical. As we've seen, the strengths of each system differ. Cohen forcing is strictly coarser than both Jockusch-Soare and Mathias forcing, and the latter two each have their own areas of high and low precision. If we plan to achieve some property $\varphi(\mathtt{G})$ at the eth stage of the construction, i.e. to correspond to a dense subset $D_e \subset P$, then P must be coarse enough that no $p \in P$ can ever imply $\neg(\varphi(\mathtt{G}))$, yet fine enough that there are $p \in P$ (and densely-many!) implying $\varphi(\mathtt{G})$. This is analogous to the choice of a suitable inductive hypothesis: it must be strong enough to prove the inductive step, but weak enough that it can be maintained in each step.

In order to tweak the coarseness of P, we will often start with one of the three main forcing notions as a base and then add further restrictions onto P. We may even add on new objects to be tracked. For example, we will later see an instance of Mathias forcing which builds two sets at once, each with its own reservoir.

3.3. **The Weakness of WKL.** With the structure of forcing as a guide, we can approach the question of whether WKL_0 is genuinely weaker than ACA_0 or not. To do this, it will suffice to show that WKL_0 admits preservation of some $\mathcal{Q}$ which separates it from $\emptyset'$, which will allow for the construction of a model of WKL_0 without $\emptyset'$ in it. We can actually show that WKL preserves several of the classes $\mathcal{Q}$ that we introduced.

In all of these preservation results, we will use a version of Jockusch-Soare forcing, as it is a natural way to constrain objects that we are thinking of as paths in the first place. In each one, note how the fact that T is computably bounded plays an essential role. This is to be expected, given that KL does not admit any of these weaknesses.

Theorem 3.2 (WKL Admits Hyperimmune-Free Solutions): If Z is hyperimmune-free and T is a Z-computable binary tree, there is an infinite path $\ell \in [T]$ such that $\ell \oplus Z$ is hyperimmune-free; that is, every function computed by $\ell \oplus Z$ is dominated by some computable function.

Proof. We'll use Jockusch-Soare forcing with subtrees of T.

We want to construct $\mathtt{G} \in [T]$ such that

$$(\forall e) : \Phi_e^{\mathtt{G} \oplus Z} \text{ is computably bounded}$$

for which it suffices to meet the condition sets

$$D_e := \{p : \ell \in [\, p\,] \Rightarrow \Phi_e^{\ell \oplus Z} \text{ is computably-bounded}\}$$

for each e. We will show that D_e is dense.

Assume otherwise, i.e. that for some tree p there is no extension p^* meeting D_e. This implies that there is some depth d_x such that $\Phi_e^{\sigma \oplus Z}(x)[d_x] \downarrow$ for $|\sigma| \geq d_x$; otherwise,

$$p^* := \{\sigma \in p : \Phi_e^{\sigma \oplus Z}(x)[|\sigma|] \uparrow\}$$

would be an infinite Z-computable tree whose paths ℓ all have $\Phi_e^{\ell \oplus Z}(x) \uparrow$, and thus meet D_e.

Moreover, d_x is Z-computable as a function of x, since p is binary and thus one can explore all depths d until finding one for which $\Phi_e^{\sigma \oplus Z}(x)[d] \downarrow$ for all $|\sigma| = d$. Using this fact, for any given $\ell \in [\, p\,]$ we can construct a Z-computable function $g(x)$ that dominates $\Phi_e^{\ell \oplus Z}$:

$$g(x) := \max\{\Phi_e^{\sigma \oplus Z}(x) : |\sigma| = d_x\}.$$

And because Z itself is hyperimmune-free, there is also a computable function dominating $g(x)$. Thus, in this case, p already meets D_e. □

Theorem 3.3 (WKL Admits Cone Avoidance): Given a Z-computable binary tree $T \subset 2^{\mathbb{N}}$ and a set $C \not\leq_T Z$, there is a path $\ell \in [\, T\,]$ for which $\ell \oplus Z$ does not compute C.

Proof. We use Jockusch-Soare forcing, with the modification that all trees in P are subsets of T and are Z-computable. The desired condition, not computing C, can be expressed as

$$(\forall e) : \Phi_e^{\mathtt{G} \oplus Z} \neq C,$$

and $\Phi_e^{\mathtt{G} \oplus Z} \neq C$ is equivalent to meeting the condition set

$$D_e := \{p : \ell \in [\, p\,] \Rightarrow \Phi_e^{\ell \oplus Z} \neq C\}.$$

Thus, it suffices to show that D_e is dense in P, i.e. that for any tree $p \in P$, there is an extension $p^* \leq p$ (i.e. a subtree) for which $\ell \in [\, p^*\,] \Rightarrow \Phi_e^{\ell \oplus Z} \neq C$.

Assume for the sake of contradiction that all extensions $p^* \leq p$ in P contain a path ℓ with $\Phi_e^{\ell \oplus Z} = C$. We will use this fact to compute C from Z, showing a contradiction. The key is that this assumption greatly limits the extent to which the computable functions $\Phi_e^{\sigma \oplus Z}$ can differ from C, in two ways:

(a) No $\sigma \in p$ that extends to a path in p has $\Phi^{\sigma \oplus Z} \perp C$; otherwise,

$$p^* = \{\tau \in p : \tau \succeq \sigma\}$$

would be infinite and meet D_e.

(b) For every x, there is some depth d_x such that $\Phi^{\sigma \oplus Z}(x) \downarrow$ for all $\sigma \in p$ with $|\sigma| \geq d_x$; otherwise,

$$p^* = \{\tau \in p : \Phi^{\tau \oplus Z}(x)[|\tau|] \uparrow\}$$

would be infinite and also all $\ell \in [\, p^*\,]$ have $\Phi_e^{\ell \oplus Z}(x) \uparrow$, thus p^* would meet D_e.

Given d_x, we can calculate $C(x)$ as follows: let

$$D = \{\sigma \in p : |\sigma| = d_x\}.$$

Because p is a *binary* tree, D is Z-computable. We know by (b) that every $\sigma \in D$ has $\Phi^{\sigma \oplus Z}(x) \downarrow$, so we can Z-computably split D into

$$D = A \cup B = \{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$$

where $\Phi^{a_j \oplus Z}(x) = 0$ and $\Phi^{b_j \oplus Z}(x) = 1$. By (a), either A or B (whichever disagrees with C) is entirely non-extendable, and hence the depth of its descendants is bounded. Then for each $d' > d$, we can compute from Z the set of nodes of depth d' (again relying on the fact that p is binary) and check which are descended from A versus B. For sufficiently large d', only one of the two groups will remain, whence we will have computed $C(x)$ from Z. □

From cone avoidance, it also follows as a corollary that WKL_0 has no minimal model; any non-computable C can be avoided. In other words, like RCA_0, WKL_0 does not imply the existence of

any particular non-computable set, though it does imply the existence of *some* non-computable set.

Next, we'll show that WKL also admits low solutions. This argument will be a little bit different from the previous two. It is an example of *effective forcing*, in which a constructed object is made to be A-computable (for some A) by deciding each of its values at a particular finite stage, and choosing each extension p^* in an A-computable way.

Theorem 3.4 (WKL Admits Low Solutions): If $Z \subseteq \mathbb{N}$ is low, then for any Z-computable infinite binary tree $T \subset 2^{\mathbb{N}}$, there is an infinite path $\ell \in [\, T \,]$ such that $\ell \oplus Z$ is low, i.e. $(\ell \oplus Z)' \leq_T \emptyset'$.

Proof. We'll construct a path $\mathtt{G} \in [\, T \,]$ *and* a $\emptyset'$-computable function f which computes $(\mathtt{G} \oplus Z)'$.
We employ a variant of Jockusch-Soare forcing as follows:

- $p \in P$ are of the form $p = (U, \sigma)$, where
  - U is an infinite, Z-computable subtree of T,
  - σ is a finite binary sequence,
  - $g' \succ \sigma$ for all $g \in [\, U \,]$.
- $(U^*, \sigma^*) \leq (U, \sigma)$ if $U^* \subseteq U$ and $\sigma^* \succeq \sigma$.
- (U, σ) is interpreted as $\mathtt{G} \in [\, U \,]$ and $\sigma \prec f$.

The properties of P already guarantee that $f = \ell'$, so the only thing to be forced is that f is total:

$$(\forall e) : f(e) \downarrow,$$

and $f(e) \downarrow$ is equivalent to meeting the condition set

$$D_e := \{(U, \sigma) : |\sigma| \geq e\}.$$

To ensure that f is $\emptyset'$-computable, we will explicitly construct an extension for (U, σ) computably relative to $\emptyset'$, and thus every place of f will be $\emptyset'$-computable.

We want an extension (U^*, σ^*) such that all $\mathtt{G} \in [\, U^* \,]$ have the same halting behavior on e, i.e.

$$\Phi_e^{\mathtt{G} \oplus Z}(e) \downarrow \iff \sigma^*(e),$$

and we need to decide $\sigma^*(e)$ in a $\emptyset'$-computable way. It might be that U already has this property, so we first check if this is the case. We can do this Z'-computably (and hence $\emptyset'$-computably, since Z is low): it is equivalent to checking

$$\Psi(U) := (\exists\, d)(\forall\, \tau \in U) : (|\tau| = d) \Rightarrow \Phi_e^{\tau \oplus Z}(e)[|\tau|] \downarrow$$

Note that the $\forall$ is bounded because U is a *binary* tree, so $\Psi(U)$ is Σ_1^Z and hence Z'-computable. If $\Psi(U)$ is true, then $\Phi_e^{\mathtt{G} \oplus Z}(e) \downarrow$ for all $\mathtt{G} \in [\, U \,]$, so we can extend (U, σ) to $U^* := U$ and $\sigma^*(e) = 1$. On the other hand, if $\Psi(U)$ is false, the subtree

$$E := \{\tau \in U : \Phi_e^{\tau \oplus Z}(e)[|\tau|] \uparrow\}$$

is infinite and Z-computable, so we can take $U^* := U \cap E$ with $\sigma^*(e) = 0$.

In summary, we can extend (U, σ) in a way depending only on the $\emptyset'$-computable function $\Psi(U)$:

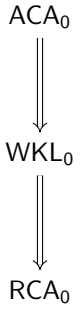
$$(U^*, \sigma^*) := \begin{cases} (U, \sigma^\frown 1) & \text{if } \Psi(U), \\ (U \cap E, \sigma^\frown 0) & \text{if } \neg\Psi(U). \end{cases}$$

In either case, $\Phi_e^{\mathtt{G} \oplus Z}(e)$ has the same halting behavior for all $\mathtt{G} \in [\, U^* \,]$, and it is matched by $\sigma^*(e)$. Since every extension is found $\emptyset'$-computably, $f = (\mathtt{G} \oplus Z)'$ is $\emptyset'$-computable, thus $\mathtt{G} \oplus Z$ is low, as desired. □

With WKL shown to admit solutions in several complexity classes not containing $\emptyset'$, we can handily reach this section's goal:

Theorem 3.5: $\mathsf{WKL}_0 \not\Rightarrow \mathsf{ACA}_0$.

Proof. This follows from any of the preservation results shown for WKL, which allow us to build a model of WKL_0 consisting entirely of $\emptyset'$-cone-avoiding or low or hyperimmune-free sets. The resulting model does not contain $\emptyset'$ as a set, therefore it is not a model of KL (or ACA_0). □



Historically, the hierarchy of these three theories has been a focus of reverse mathematics. They are the bottom three of the "big five" hierarchy, which also includes the stronger theories ATR_0 and $\Pi^1_1\mathsf{CA}_0$.[10] The big five are notable because of how many theorems from other areas have been shown equivalent in strength to one of them.

We have seen already that the completeness of $\mathbb{R}$ is equivalent to ACA_0 and the compactness of $[0,1]$ is equivalent to WKL_0. One can actually show that a wide range of results in algebra, analysis, logic, and combinatorics are equivalent to one of these levels. A few key examples are listed below.[11]

Note that the language of second-order arithmetic is limited in what it can express; all results below referring to algebraic structures and metric spaces apply only to *countable* structures and *separable* metric spaces, as general uncountable sets are third-order objects and thus cannot be discussed directly in second-order arithmetic. Continuous functions on separable metric spaces are determined by their values on a countable dense set, and thus they can be described as second-order objects.

- RCA_0
  - Intermediate Value Theorem
  - Nested Interval Completeness
  - Baire Category Theorem
  - Existence of algebraic closures
- WKL_0
  - Heine-Borel Theorem (compactness of $[0,1]$)
  - The Hahn-Banach Theorem
  - Riemann integrability of continuous functions
  - Uniqueness of algebraic closures
  - Existence of prime ideals in commutative rings
  - Gödel's Completeness and Compactness Theorems
- ACA_0
  - Completeness of $\mathbb{R}$
  - Bolzano-Weierstrass Theorem (sequential compactness of $[0,1]$)
  - Existence of maximal ideals in commutative rings
  - Existence of bases for $\mathbb{Q}$-vector spaces
  - König's Lemma

In the next section, we will set our reverse-mathematical techniques onto Ramsey's Theorem, a problem from infinitary combinatorics, and investigate its relationship to this hierarchy.

[10] ATR_0 stands for Arithmetic Transfinite Recursion. Roughly speaking, it extends ACA_0 to allow induction and comprehension along any well-order, rather than just $\mathbb{N}$. $\Pi^1_1\mathsf{CA}_0$ extends comprehension to Π^1_1 formulas. It is strictly weaker than full second-order arithmetic but stronger than ATR_0.

[11] Proofs of all of these equivalences can be found in [8].

4. Ramsey Theory

So far, we've developed a nice hierarchy of three naturally-arising theories. We've seen that many statements in second-order arithmetic can be located in one of these three levels. To complicate this picture, we now consider Ramsey's Theorem:

Definition (Ramsey's Theorem in degree n for k colors, or RT^n_k): Given a k-coloring of $[\mathbb{N}]^n$ (the size-n subsets of $\mathbb{N}$), i.e. a function $c : [\mathbb{N}]^n \to \{1, 2, \dots, k\}$, there exists an infinite set X that is *c-homogeneous*, i.e. c is constant on $[X]^n$.

In this section, we'll use the methods developed in previous sections—forcing in particular—to determine the precise strength of RT^n_k relative to the Big Five. This was not known for general $n, k \in \mathbb{N}$ until Liu closed the last remaining case in 2011.

4.1. RT over RCA_0.

First, we'll see what can be proven about RT^n_k over RCA_0. RCA_0 does prove some relationships between RT^n_k for different n and k, but we will see that it does not actually prove any RT^n_k for $n > 1$.

Lemma 4.1 (Ramsey Theory over RCA_0):

(a) For $k \geq 2$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^1_k$.
(b) For $k \geq 2$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_k \iff \mathsf{RT}^n_{k+1}$.
(c) For $n \geq 1$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^{n+1}_k \implies \mathsf{RT}^n_k$.

Proof. (a): This is essentially the infinitary pigeonhole principle for a fixed number of colors. Given a coloring $c : \mathbb{N} \to \{1, 2, \dots, k\}$, the homogeneous sets

$$c^{-1}(j) := \{x : c(x) = j\}$$

are all computable from c, thus exist in RCA_0. One of these sets must be unbounded; if there exist upper bounds a_j on each of these sets so that $c^{-1}(j) < a_j$, then the union of all of them is bounded above by $\max\{a_1, a_2, \dots, a_k\}$, but their union is $\mathbb{N}$, which is unbounded.

(b): Given a $(k+1)$-coloring $c : [\mathbb{N}]^n \to \{1, 2, \dots, k+1\}$, we can produce a slightly reduced k-coloring by identifying 1 and 2:

$$c^* : A \mapsto \begin{cases} 1 & c(A) = 1 \\ c(A) - 1 & c(A) > 1 \end{cases}$$

This c^* clearly c-computable, thus it is a function in RCA_0. Assuming RT^n_k, there exists an infinite c^*-homogeneous set X. If X is any color other than 1, then X is also c-homogeneous. Otherwise, $c^*(A) \in \{1, 2\}$ for $A \in [X]^n$, so c^* is a 2-coloring of X. Thus, we can apply RT^n_2 (which clearly follows from RT^n_k) to c on X to get an infinite c-homogeneous set. Either case implies RT^n_{k+1}.

(c): Given a k-coloring $c : [\mathbb{N}]^n \to \{1, 2, \dots, k\}$, we have a k-coloring

$$c^* : [\mathbb{N}]^{n+1} \to \{1, 2, \dots, k\} \text{ given by } c^*(A) = c(A \setminus \min(A)).$$

c^* is clearly c-computable, thus it is also a function in RCA_0. Assuming RT^{n+1}_k, there exists a c^*-homogeneous set X (say with color j). For any size-n subset $B \subset (X \setminus \min(X))$, we have

$$c(B) = c^*(B \cup \{\min(X)\}) = j,$$

thus $X \setminus \min(X)$ is an infinite c-homogeneous set of color j, implying that RT^n_k also holds. □

Thus, RCA_0 proves that the strength of RT^n_k (for $k \geq 2$) depends only on n, and that it is non-decreasing in n. As a corollary, RCA_0 proves that the weakest case of RT is RT^2_2. The natural question is whether RCA_0 can actually prove this simplest case, and the answer is: no. To show this, we will give a computable 2-coloring of $[\mathbb{N}]^2$ with no computable infinite homogeneous set.

Theorem 4.2: $\mathsf{RCA}_0 \not\Longrightarrow \mathsf{RT}^2_2$.

Proof. We'll describe a coloring algorithm that systematically makes every infinite computable set non-homogeneous. In fact, we can do even better and avoid all infinite c.e. sets. Let

$$A_e = \{n : \Phi_e(n)\downarrow\}$$

and denote by $A_e[s]$ the approximation of A_e up to input s and for s steps:

$$A_e[s] := \{n < s : \Phi_e(n)[s]\downarrow\}.$$

Note that $A_e[s]$ is computable and that $\lim_s A_e[s] = A_e$. We will ensure that if $A_e[s]$ gets sufficiently large, then we eventually color two of its edges oppositely.

The construction will proceed in $\mathbb{N}$-many stages: in stage s, we decide the colors of edges

$$E_s := \{(0,s),(1,s),\ldots,(s-1,s)\}.$$

During stage s, for each $e < s$ we compute the set of $x \in A_e[s]$ such that (x,s) is still uncolored. If this set has at least two elements a, b, we color their edges oppositely:

$$c((a,s)) = 0, \quad c((b,s)) = 1.$$

Otherwise we do nothing and move on. Note that at most 2 edges of E_s can be colored for each $e < s$. At the end of stage s, some of E_s may remain uncolored—their colors don't matter.

Now to show that the coloring works: if A_e is infinite, then for sufficiently large s,

$$|A_e[s]| \geq 2e.$$

For such s, our coloring is guaranteed to find $a, b \in A_e[s]$ with (a,s) and (b,s) uncolored, since at most $2(e-1)$ edges of E_s can be colored up to this point in stage s. And because A_e is infinite, there will be some sufficiently large s which is also in A_e, so that $(a,s),(b,s)$ are both edges in A_e, and thus A_e is not homogeneous. Thus, we've colored all infinite c.e. sets non-homogeneously. □

This argument focuses on the c.e. sets, but in fact it can be extended to all *limit-computable* sets, i.e. sets A such that

$$A(n) = \lim_m g(n,m)$$

for some computable $g(n,m)$. This requires only a minor change: define $A_e[s]$ as

$$A_e[s] := \{n < s : g_e(n,s)[s]\downarrow = 1\}$$

where (g_e) enumerates the algorithms on two inputs. Define c in the same way relative to $A_e[s]$, and c will color every limit-computable A non-homogeneously by the same argument.

So RT^2_2 implies the existence of non-limit-computable sets, but WKL_0 *does not*. Because WKL admits low solutions, one can build an entirely low model of WKL_0. Low sets are $\emptyset'$-computable, and hence limit-computable as well,[12] so we have a model of WKL_0 whose sets are all limit-computable. Thus, WKL_0 is not strong enough to prove RT^2_2 (or any other RT^n_k) either!

4.2. RT **over** ACA_0**.** First, ACA_0 is strong enough to prove RT^n_k for all n and k:

Theorem 4.3: $\mathsf{ACA}_0 \implies \mathsf{RT}^n_k$.

Proof. Given a coloring $c : [\mathbb{N}]^n \to \{1,2,\ldots,k\}$, we say that a set X is *pre-homogeneous* if the coloring of n-edges in X does not depend on their largest element, in the sense that

$$\forall s \in [X]^{n-1} \text{ and } a,b \in X \text{ with } a,b > \max(s), \quad c(s \cup \{a\}) = c(s \cup \{b\}).$$

Such c and X give rise to a derived coloring

$$c^* : [X]^{n-1} \to \{1,2,\ldots,k\} \quad c^*(s) = c(s \cup \{a\}) \text{ where } a \in X, a > \max(s).$$

Any c^*-homogeneous set is also c-homogeneous, so if such an X exists for any coloring c, then $\mathsf{RT}^{n-1}_k \Rightarrow \mathsf{RT}^n_k$. We've shown that RT^1_k is provable in RCA_0, so to prove RT^n_k for every n, it suffices to show that an infinite, pre-homogeneous set exists for every coloring.

[12] In fact, $\emptyset'$-computable and limit-computable are equivalent properties. This is Shoenfield's Limit Lemma.

Now given c we'll construct an infinite pre-homogeneous set $\mathtt{G}$ via a variation of Mathias forcing:

- $p \in P$ are of the form (E, R) where E is finite, R is infinite, and $E < R$.
  - Additionally, E must be pre-homogeneous with respect to c.
  - All elements of R must be able to be added to E while maintaining pre-homogeneity.
- $(E^*, R^*) \leq (E, R)$ if $E \subseteq E^* \subseteq E \cup R$ and $R^* \subseteq R$.
- (E, R) is interpreted as $E \subseteq \mathtt{G} \subseteq E \cup R$.

We'd like to construct a $\mathtt{G}$ that is infinite, for which it suffices to meet

$$D_e := \{(E, R) : \max E > e\}$$

for all e. Any such $\mathtt{G}$ will automatically be pre-homogeneous because of the additional constraints on P. We'll show that D_e is dense.

Let $(E, R) \in P$ and fix e. For each $\sigma : [E]^{n-1} \to \{0, 1\}$ (of which there are finitely many) let

$$R_\sigma := \{x \in R : (\forall A \in [E]^{n-1}) : c(A \cup \{x\}) = \sigma(A)\},$$

so that every element of R_σ has the same coloring behavior with respect to E, and

$$R = \bigcup_\sigma R_\sigma.$$

R is infinite, so by the pigeonhole principle, at least one of these R_σ is infinite. Selecting such an R_σ, let $x \in R_\sigma$ be its least element above e, and choose the extension

$$(E^*, R^*) := (E \cup \{x\}, R_\sigma \cap (x, \infty))$$

which indeed meets D_e, and one can check that $(E^*, R^*) \in P$. Thus, D_e is dense, as desired. □

A note on the complexity of this construction: R_σ is c-computable, and testing whether R_σ is infinite is c''-computable, as being infinite is a Π_2 property. This shows that one can find an infinite pre-homogeneous set that is c''-computable. Thus, one can reduce an instance c of RT^n_k to a c''-computable instance of RT^{n-1}_k. To prove RT^n_k, this reduction must be done $n-1$ times, after which RT^1_k can be computably solved, giving a $c^{(2n-2)}$-computable solution to c.

We can also show a partial reversal of this result: RT^3_2 (and thus all RT^n_k for $n \geq 3$) implies $\mathsf{ACA_0}$. To prove this requires an A-computable coloring of triples whose infinite homogeneous sets all compute A':

Theorem 4.4 (Jockusch): $\mathsf{RT}^3_2 \implies \mathsf{ACA_0}$.

Proof. Let A be a set. We will describe a Δ^A_1 coloring $c : [\mathbb{N}]^3 \to \{0, 1\}$ whose only infinite homogeneous sets compute A', and thus show that A' exists for all A.

Specifically, consider the following coloring: given a triple (a, s, t) with $a < s < t$, let

$$c : (a, s, t) \mapsto \begin{cases} 0 & \exists (e < a) : \Phi^A_e(e)[s] \uparrow \wedge \Phi^A_e(e)[t] \downarrow \\ 1 & \text{otherwise} \end{cases}$$

There are no infinite c-homogeneous sets of color 0; if such a set $\{a_1, a_2, \dots\}$ existed, then there would be Turing machines *of index at most* a_1 which halt in time-windows $(a_2, a_3], (a_3, a_4], \dots$ but this is impossible because there are only finitely many of these.

Thus, the infinite set X guaranteed by RT^3_2 must have color 1. Let $X = \{a_1, a_2, \dots\}$. Given X, we can compute A': to test whether $e \in A'$, take some $a_p > e$ (one exists because X is infinite) and check $\Phi^A_e(e)[a_{p+1}]$. If this doesn't halt then it never will, since $c((a_p, a_q, a_{q+1})) = 1$ for $q > p$, hence $\Phi^A_e(e)$ does not halt in $(a_q, a_{q+1}]$ for any $q > p$, and these intervals cover the rest of $\mathbb{N}$. □

This proves that RT^3_2 is equivalent to $\mathsf{ACA_0}$, and likewise for RT^n_k with $n \geq 3$, so now RT^2_2 is the only outstanding case. It is unclear how one could encode the halting problem with only a coloring of pairs. Indeed, there turns out to be a genuine difference in strength between RT^2_2 and RT^3_2. To prove this, we will use a forcing argument to show that RT^2_2 admits cone avoidance.

Theorem 4.5 (Seetapun): $\mathsf{RT}^2_2 \not\Longrightarrow \mathsf{ACA}_0$.

Proof. We will prove this by establishing cone avoidance for RT^2_2. That is, we'll show that for any Z, if $c : [\mathbb{N}]^2 \to \{0,1\}$ is Z-computable and $C \not\leq_T Z$, then there is an infinite c-homogeneous set $\mathtt{G}$ for which $Z \oplus C \not\leq_T \mathtt{G}$. We'll assume $Z = \emptyset$ wlog, as it does not affect the argument.

We can't simply choose a color and begin the construction, since it's possible that there is only an infinite homogeneous set of one color, and it's not apparent which color will work. Instead, we can start constructing *two* homogeneous sets at once, one in each color, and show that *at least one* will have the desired properties.

Proceeding by contradiction, we'll assume that no such $\mathtt{G}$ exists. This assumption is necessary to force both sets to be infinite.

We construct $\mathtt{G}_0$ and $\mathtt{G}_1$ with a variation of 2-fold Mathias forcing defined as follows:

- $p \in P$ are of the form (E_0, E_1, R), where E_0, E_1 are finite, R is infinite, and $E_0, E_1 < R$.
  - Additionally, E_0 and E_1 must be c-homogeneous of colors 0 and 1 respectively.
  - All elements of R can be added to E_0 *or* E_1 while maintaining homogeneity.
  - R must not compute C.
- $(E_0^*, E_1^*, R^*) \leq (E_0, E_1, R)$ if $E_i \subseteq E_i^* \subseteq E_i \cup R$ for $i \in \{0,1\}$ and $R^* \subseteq R$.
- (E_0, E_1, R) is interpreted as $E_i \subseteq \mathtt{G}_i \subseteq E_i \cup R$ for $i \in \{0,1\}$.

Now we seek to force three properties:

(1) $\mathtt{G}_0$ is infinite.
(2) $\mathtt{G}_1$ is infinite.
(3) At least one of $\mathtt{G}_0$ and $\mathtt{G}_1$ does not compute C.

We'll show separately that each one is forced.

First, some notation: for any set A and color $i \in \{0,1\}$, define the i-neighborhood of A as

$$N_i(A) := \{x : (\forall a \in A) : c((a,x)) = i\}.$$

Note that the conditions on P imply that $R \subseteq N_0(E_0) \cap N_1(E_1)$ for $(E_0, E_1, R) \in P$.

(1): It suffices to meet the condition-sets

$$D_e := \{(E_0, E_1, R) : (\max\{E_0\} \geq e)\}$$

for $e \in \mathbb{N}$. If a given (E_0, E_1, R) has no extension meeting D_e, then for every $x \geq e$ in R,

$$(E_0^*, E_1^*, R^*) := (E_0 \cup \{x\}, E_1, R \cap N_0(x))$$

must not be a valid extension, otherwise it would meet D_e. This can only be because $R \cap N_0(x)$ is finite for all $x \geq e$ in R. But if this is true, then $R \cap N_1(x)$ is infinite for all $x \geq e$ in R, which makes it R-computable to find an infinite c-homogeneous set of color 1: the extension

$$(E_0^*, E_1^*, R^*) := (E_0, E_1 \cup \{x\}, R \cap N_1(x))$$

(where $x = \min\{R_{\geq e}\}$) is guaranteed to be valid, so one can repeatedly choose such extensions, resulting in a $\mathtt{G}_1$ that is R-computable and infinite, and hence witnesses cone avoidance. We had assumed that such $\mathtt{G}_1$ did not exist, so we must now assume that D_e is dense.

(2): This follows similarly.

(3): This property is the most substantive. The only case we must avoid is both $\mathtt{G}_0$ and $\mathtt{G}_1$ compute C, i.e. for some $e_0, e_1 \in \mathbb{N}$, $\Phi_{e_0}^{\mathtt{G}_0} = \Phi_{e_1}^{\mathtt{G}_1} = C$. Thus, it suffices to meet the condition-sets

$$D_{(e_0,e_1)} := \{(E_0, E_1, R) : (\Phi_{e_0}^{\mathtt{G}_0} \neq C) \vee (\Phi_{e_1}^{\mathtt{G}_1} \neq C)\}$$

for $(e_0, e_1) \in \mathbb{N}^2$. We aim to show that $D_{(e_0,e_1)}$ is dense. Assume otherwise, and that (E_0, E_1, R) is a condition in P with no extensions meeting $D_{(e_0,e_1)}$.

We define an *i-fork* to be a pair of finite i-homogeneous sets (X, Y) for which

$$\exists w < \max(X \cup Y) : \quad \Phi_{e_i}^{E_i \cup X}(w)[\max(X \cup Y)] \downarrow \neq \Phi_{e_i}^{E_i \cup Y}(w)[\max(X \cup Y)].$$

Whereas the property of differing from C is non-computable, the two extensions in an i-fork differ from *one another* in a bounded way, which can be verified R-computably. Say an i-fork is *valid* if

$$(E_i \cup X, E_{1-i}, R^*) \quad \text{and} \quad (E_i \cup Y, E_{1-i}, R^*)$$

are both in P, for some R^*. If (X, Y) is valid, then one of these two extensions (the one that differs from C) must meet $D_{(e_0,e_1)}$, thus it suffices to show that a valid i-fork exists. We'll first show the existence of many i-forks, and then show there is a valid one among them.

Lemma ($*$): Every infinite set $R^* \subseteq R$ which does not compute C must contain an i-fork.

Proof. Suppose otherwise. We have assumed that no extension meets $D_{(e_0,e_1)}$. This implies that for every w there is some finite $X \subset R^*$ with

$$\Phi_{e_i}^{E_i \cup X}(w)[\max(X)] \downarrow = C(w).$$

Otherwise, the extension (E_0, E_1, R^*) would meet $D_{(e_0,e_1)}$, as it implies $\Phi_{e_i}^{\mathtt{G_i}}(w)$ is either undefined or disagrees with $C(w)$. If, in addition, there are no i-forks in R^*, then

$$\Phi_{e_i}^{E_i \cup X}(w)[\max(X)] \in \{\uparrow, C(w)\}$$

for all finite $X \subset R^*$, since no two X can contradict one another. Now we can check all such X until halting, thus computing $C(w)$ from R^*, which contradicts the assumption that $C \not\leq_T R^*$. □

As an immediate corollary, there are infinitely many i-forks (one can take $R^* = R \cap \{x > M\}$ to get an i-fork of arbitrarily high minimum). So let (X_j, Y_j) be a *computable* sequence of 0-forks (one can compute it by repeatedly taking the next largest 0-fork in some computable ordering):

$$X_0 < Y_0 < X_1 < Y_1 < X_2 < Y_2 < \dots$$

By assumption, all of these 0-forks are invalid. This is actually a rather strong condition, because it tells us that for any potential reservoir R^*, each $X_j \cup Y_j$ contains some z_j for which $N_1(z_j) \cap R^*$ is infinite; otherwise, $R^* \cap_{z \in X_j \cup Y_j} N_0(z)$ would be a valid reservoir for the fork (X_j, Y_j).

The goal now is to leverage the invalidity of the 0-forks to produce a valid 1-fork within (z_j). We'll first show that such a sequence z_j of sufficient depth d will eventually include a 1-fork. Then, we'll find a reservoir R^* with respect to which all $z \in \{X_j \cup Y_j\}_{j \leq d}$ have edges of only one color. This R^* will serve as reservoir to the 1-fork included in (z_j), hence the 1-fork will be valid.

Construct a computable tree $T \subseteq \mathbb{N}^{\mathbb{N}}$ from all sequences σ with

$$\sigma(e) \in X_e \cup Y_e$$

such that the immediate parent of σ does not contain any 1-forks. T is computably bounded (since $X_e \cup Y_e$ is finite), so if T is infinite then it contains a C-cone avoiding path by WKL cone avoidance. But by $(*)$, any infinite non-C-computing set necessarily contains 1-forks, so *there can be no such path*, and thus T must be finite. Let its depth be d.

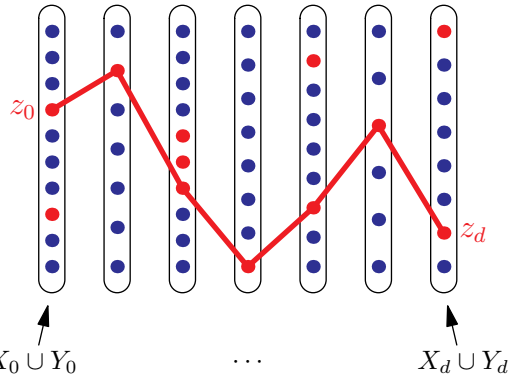


FIGURE 6. The elements of $X_j \cup Y_j$ for $j \leq d$ colored by τ (blue = 0, red = 1). The sequence (z_j) must contain a valid 1-fork with reservoir R^*.

Finally, for each function

$$\tau : \bigcup_{j=0}^{d} X_j \cup Y_j \to \{0,1\}$$

let

$$R_\tau := \{r \in R : (\forall z \in \mathrm{Dom}(\tau)) : c((z,r)) = \tau(z)\}.$$

There are finitely many such τ since T is finite, and R_τ finitely partitions R into R-computable parts. For some τ, R_τ will be infinite: call this R^*. Since the 0-forks (X_j, Y_j) are all invalid, there is a sequence $z_j \in X_j \cup Y_j$ which all have $\tau(z_j) = c(z_j, R^*) = 1$. By the definition of T, there is a 1-fork among this sequence. This 1-fork is valid, with R^* as its corresponding reservoir. □

This shows that RT^2_2 is strictly weaker than RT^3_2, casting RT^2_2 out of the $\mathsf{RCA}_0 - \mathsf{WKL}_0 - \mathsf{ACA}_0$ hierarchy, though we haven't yet ruled out the possibility that RT^2_2 lies between WKL_0 and ACA_0. Seetapun's proof was published in 1995, but the relationship between RT^2_2 and WKL_0 was left open until 2011, when Jiayi Liu proved that $\mathsf{RT}^2_2 \not\Longrightarrow \mathsf{WKL}$. Liu showed that RT^2_2 admits PA avoidance by using a Mathias forcing argument.

Theorem 4.6 (Liu): $\mathsf{RT}^2_2 \not\Longrightarrow \mathsf{WKL}$.

Proof. Omitted. See [2] or [6] for full proofs of Liu's Theorem. □

With Liu's proof, RT^2_2 could be placed firmly outside of the Big Five hierarchy:

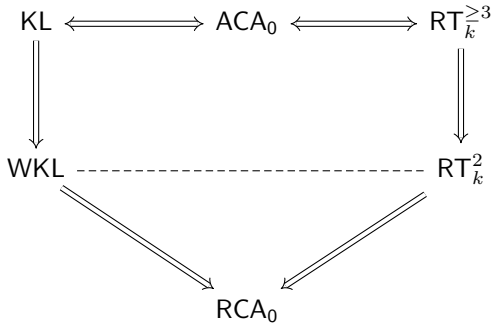


FIGURE 7. The full dependency chart between RT and KL. WKL_0 and RT^2_k are independent.

But unlike WKL, which we find embedded within proofs of all sorts of major theorems throughout mathematics, RT^2_2 is mysteriously not equivalent to any other major theorems. RT^2_2 is its own principle, with its own set of consequences. Much of the work in reverse mathematics in recent decades has been dedicated to studying it.

4.3. **Study of RT^2_2.** In this final section, we'll take a look at the world of RT^2_2 and several related coloring principles, and see some longstanding open problems. First, we introduce two new properties a coloring $c : [\mathbb{N}]^2 \to \{0,1\}$ can have on a domain X:

- *Stable*: For $x \in X$, c takes a single value on all but finitely-many edges containing x.
- *Transitive* For $x < y < z \in X$, $c((x,y)) = c((y,z)) = q \implies c((x,z)) = q$.

Stability and Transitivity turn out to be quite significant classes of colorings. In a sense, they are the only two classes that must be considered; as we will see, if RT^2_2 holds on all stable colorings and all transitive colorings, then it holds in general.

RT^n_k says *any* coloring has an infinite subset on which it is *homogeneous*. This principle can be weakened in several ways. If $\mathcal{C}$ and $\mathcal{D}$ are two classes of colorings, let

$$\mathcal{C} \rightarrowtail \mathcal{D} \text{ denote } \text{“}\forall c \in \mathcal{C},\ \exists \text{ an infinite subset } X \subseteq \mathbb{N} \text{ such that } c \in \mathcal{D} \text{ on } X.\text{”}$$

Note that $\rightarrowtail$ is transitive. Using this notation,

$$\mathsf{RT}^2_2 := \mathit{Any} \rightarrowtail \mathit{Homogeneous}.$$

RT^2_2 can be split into several interesting coloring principles based on the coloring classes of stable and transitive, many of which are collected in Figure 8.

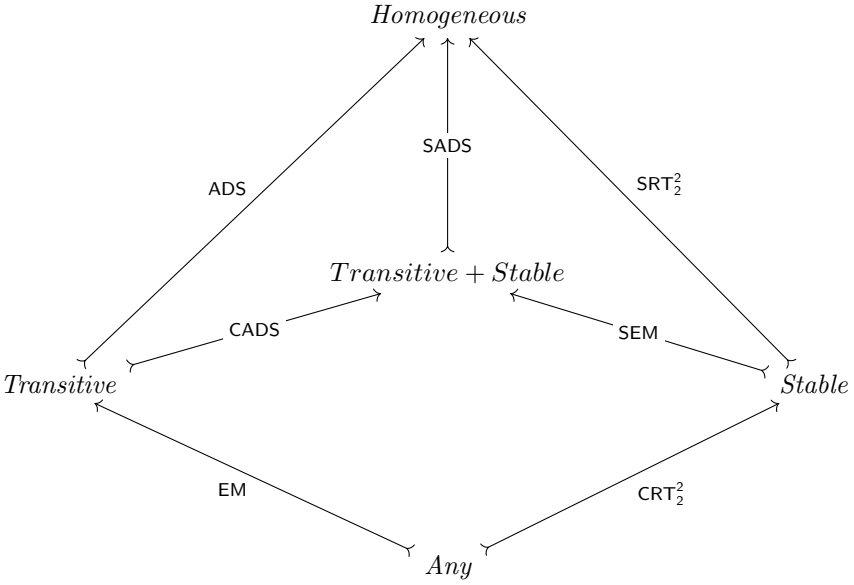


FIGURE 8. A decomposition of the coloring principles below RT^2_2.

The abbreviations are:

- RT: Ramsey's Theorem.
  - SRT = Stable Ramsey's Theorem.
  - CRT = Cohesive Ramsey's Theorem.
- ADS: The Ascending/Descending Sequence Principle.
  - For every linear order on $\mathbb{N}$, there is an infinite increasing or decreasing sequence. A linear order can be expressed as a transitive coloring of $[\mathbb{N}]^2$.
  - SADS = Stable Ascending/Descending Sequence.
  - CADS = Cohesive Ascending/Descending Sequence.
- EM: The Erdős-Moser Principle.
  - This is sometimes called the "Tournament Principle," because it can be interpreted as the statement "for every infinite set of teams, there is an infinite subset on which teams can be linearly ordered by their strength."
  - SEM = Stable Erdős-Moser.

It is known that all of these principles lay in the expanse strictly between RCA_0 and RT^2_2, though in many cases this is very nontrivial. In terms of their relative strengths, some equivalences can be found by identifying paths in the diagram: for instance,

$$\mathsf{ADS} + \mathsf{EM} \iff \mathsf{SRT}^2_2 + \mathsf{CRT}^2_2 \iff \mathsf{RT}^2_2.$$

But there are also some unexpected implications—for example, we will show that $\mathsf{ADS} \Rightarrow \mathsf{CRT}^2_2$. In fact, ADS proves the *Cohesive Set Principle* (COH), which is a stronger result.

COH states that for any sequence of functions $f_i : \mathbb{N} \to \{0, 1\}$, there is a set A for which all of the f_i are stable (i.e. converge to 0 or 1) on A—we say that A is a *cohesive set* for (f_i). CRT^2_2 follows from a special case of COH where

$$f_i(x) := c((i, x))$$

and, say, $f_i(i) := 0$ (the diagonal values do not matter to the cohesiveness of A). The essential difference is that $f_i(x)$ must be stable on A for all i, whereas $c((i, x))$ only needs to be stable for $i \in A$.

Lemma 4.7: $\mathsf{ADS} \implies \mathsf{COH}$.

Proof. Given a sequence of functions $f_i : \mathbb{N} \to \{0,1\}$, let $f(x)$ denote the binary sequence

$$f(x) := (f_1(x), f_2(x), f_3(x), \dots).$$

We may assume that f is injective[13]. Between two binary sequences there is a lexicographic order; e.g. $(1,0,0,1,1,\dots) > (1,0,0,1,0,\dots)$. Let $c : [\mathbb{N}]^2 \to \{0,1\}$ color pairs based on this ordering:

$$\text{for } x < y, \quad c : (x,y) \mapsto \begin{cases} 0 & f(x) < f(y) \\ 1 & f(x) > f(y) \end{cases}$$

Given that $f(x) \neq f(y)$, the sequences differ at some finite index, so c is computable relative to the functions f_i. Then, by ADS there must exist an infinite ascending (wlog) subsequence

$$A = \{a_1, a_2, \dots\},$$

so that

$$f(a_1) < f(a_2) < f(a_3) < \dots$$

Now we can show (in RCA_0) that A is a cohesive set for the functions f_i: for any n, the n-tuple

$$(f_1(a_j), f_2(a_j), \dots, f_n(a_j))$$

is non-decreasing in j, and takes at most 2^n values, so it changes at most 2^n times. In particular, this implies $\{f_n(a_j)\}_j$ changes between 0 and 1 at most 2^n times, and thus is eventually constant. As this is true of all n, A is cohesive for the functions f_i. □

Thus, as promised, RT^2_2 can be reduced to the two cases of transitive and stable colorings, as

$$\mathsf{SRT}^2_2 + \mathsf{ADS} \implies \mathsf{SRT}^2_2 + \mathsf{CRT}^2_2 \iff \mathsf{RT}^2_2 .$$

This decomposition is unique to $n = 2$, as the property of transitivity does not make sense for colorings with $n \geq 3$. The decomposition can be used to more easily prove weakness principles for RT^2_2 by proving them for SRT^2_2 and ADS (or COH), which are often simpler. Both Seetapun's Theorem and Liu's Theorem have "modernized" proofs along these lines.

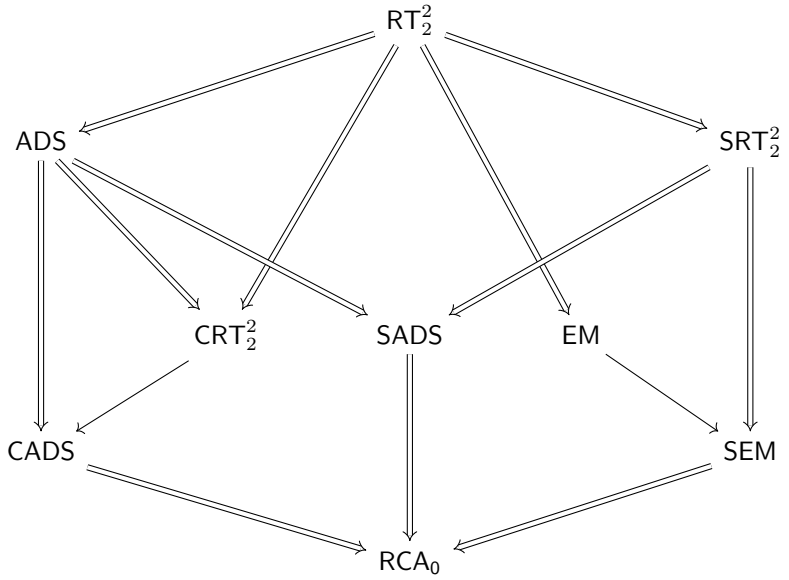


FIGURE 9. Known dependencies between principles below RT^2_2.

[13]In general, we can interleave the sequence f_i with a characteristic functions χ_i for each i, which makes f injective. Proceeding with the proof will yield a set A that is cohesive for both f_i and χ_i, which also serves as a cohesive set for just the f_i.

The presently-known implications among the coloring principles discussed in this section are listed above in Figure 9.

The single arrows denote implications that are not proven to be strict (i.e. may be equivalences). Two open questions are whether SEM is weaker than EM and whether CADS is weaker than CRT^2_2. Between CRT^2_2, SADS, and EM, five of six non-implications have been proven; the only unproven one is whether $\mathsf{EM} \Rightarrow \mathsf{CRT}^2_2$, or more generally whether $\mathsf{EM} \Rightarrow \mathsf{COH}$. The precise strength of EM remains uncertain.

The full state of progress on these and many other questions in reverse mathematics is cataloged and displayed in the Reverse Mathematics Zoo, though the diagrams there are not up to date.

Acknowledgments

I would like to thank Professors Maryanthe Malliaris and Denis Hirschfeldt for giving helpful advice and direction during this project. I am especially grateful to my REU mentor Miles Kretschmer for teaching me all that I know about reverse mathematics and providing excellent feedback on dozens of drafts of this paper.

References

[1] Jeremy Avigad. *Mathematical Logic and Computation*. Cambridge University Press, 2023. ISBN: 978-1-108-47875-5.

[2] Damir D. Dzhafarov and Carl Mummert. *Reverse Mathematics*. Theory and Applications of Computability. Springer, 2022. ISBN: 978-3-031-11366-6.

[3] Harvey M. Friedman. "The Emergence of (Strict) Reverse Mathematics". In: (2021). URL: https://u.osu.edu/friedman.8/foundational-adventures/downloadable-manuscripts/.

[4] John P. Burgess George S. Boolos and Richard C. Jeffrey. *Computability and Logic*. Cambridge University Press, 2007. ISBN: 978-0-521-87752-7.

[5] Ivor Grattan-Guinness. *The Search For Mathematical Roots, 1870-1940*. Princeton University Press, 2001. ISBN: 978-0-691-05858-0.

[6] Denis R. Hirschfeldt. *Slicing the Truth*. Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore. World Scientific, 2014. ISBN: 978-981-4612-62-3.

[7] Ludovic Patey. *Lowness and Avoidance*. Unpublished, 2024-ongoing. URL: https://ludovicpatey.com/lowness-avoidance/.

[8] Stephen G. Simpson. *Subsystems of Second Order Arithmetic*. Perspectives in Logic. Cambridge University Press, 2009. ISBN: 978-0-521-88439-6.

[9] Robert I. Soare. *Turing Computability*. Theory and Applications of Computability. Springer, 2016. ISBN: 978-3-642-31932-7.