Tutorial Series on Reverse Mathematics

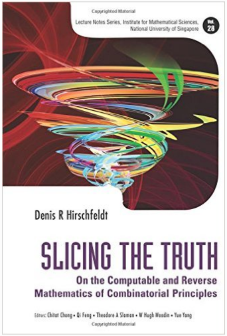
Denis R. Hirschfeldt — University of Chicago

2017 NZMRI Summer School, Napier, New Zealand

Tutorial Series on Reverse Mathematics

Denis R. Hirschfeldt — University of Chicago

2017 NZMRI Summer School, Napier, New Zealand



Part I: Background

A Bit of Historical Context

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

Russell's Paradox

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

Russell's Paradox: Let $S = \{A : A \notin A\}$. Is $S \in S$?

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

Russell's Paradox: Let $S = \{A : A \notin A\}$. Is $S \in S$?

Crisis in foundations

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

Russell's Paradox: Let $S = \{A : A \notin A\}$. Is $S \in S$?

Crisis in foundations

Hilbert's Program: prove the consistency of mathematics via finitistic methods

A Bit of Historical Context: Abstraction and the Loss of Certainty

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Increase in power, but also a loss of intuition

Increased demand for rigor

Cantor's Paradise

Russell's Paradox: Let $S = \{A : A \notin A\}$. Is $S \in S$?

Crisis in foundations

Hilbert's Program: prove the consistency of mathematics via finitistic methods

Gödel's Second Incompleteness Theorem

A Bit of Historical Context: Reverse Mathematics

Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.

— *Jorge Luis Borges*

> *Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.*
>
> — *Jorge Luis Borges*

We can still try to understand how much axiomatic power given theorems need.

A Bit of Historical Context: Reverse Mathematics

Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.

— Jorge Luis Borges

We can still try to understand how much axiomatic power given theorems need.

Fix a weak base axiomatic system B.

A Bit of Historical Context: Reverse Mathematics

> *Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.*
>
> — *Jorge Luis Borges*

We can still try to understand how much axiomatic power given theorems need.

Fix a weak base axiomatic system B.

Given a theorem T, we can find an axiomatic system $S \supseteq B$ sufficient to prove T.

A Bit of Historical Context: Reverse Mathematics

Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.

— *Jorge Luis Borges*

We can still try to understand how much axiomatic power given theorems need.

Fix a weak base axiomatic system B.

Given a theorem T, we can find an axiomatic system $S \supseteq B$ sufficient to prove T.

If we can then also show that the axioms of S are provable from $B + T$, then we know S is *exactly* what we need to prove T.

A Bit of Historical Context: Reverse Mathematics

> *Nada se edifica sobre la piedra, todo sobre la arena, pero nuestro deber es edificar como si fuera piedra la arena.*
>
> — *Jorge Luis Borges*

We can still try to understand how much axiomatic power given theorems need.

Fix a weak base axiomatic system B.

Given a theorem T, we can find an axiomatic system $S \supseteq B$ sufficient to prove T.

If we can then also show that the axioms of S are provable from $B + T$, then we know S is *exactly* what we need to prove T.

We can also compare theorems in terms of implication over B.

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{\text{19}^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

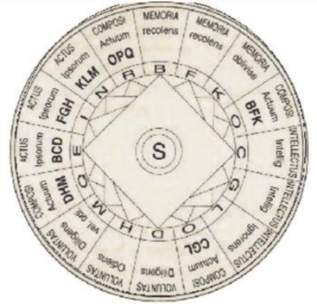
Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

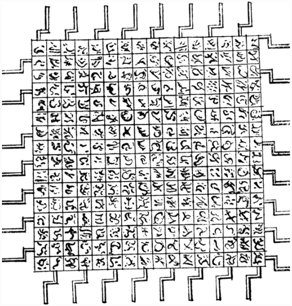
Ramon Llull (c. 1232–1315), Gottfried Leibniz (1646–1716)

A Bit of Historical Context: Algorithms in Mathematics

A Llullian circle

The writing machine at the Grand Academy of Lagado (*Gulliver's Travels*, 1726)





A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{\text{19}^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Ramon Llull (c. 1232–1315), Gottfried Leibniz (1646–1716)

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Ramon Llull (c. 1232–1315), Gottfried Leibniz (1646–1716)

Emil du Bois-Reymond's *ignoramus et ignorabimus* (1880) and Hilbert's "*Wir müssen wissen—wir werden wissen.*" (1930)


WIR MÜSSEN WISSEN
WIR WERDEN WISSEN

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Ramon Llull (c. 1232–1315), Gottfried Leibniz (1646–1716)

Emil du Bois-Reymond's *ignoramus et ignorabimus* (1880) and Hilbert's *"Wir müssen wissen—wir werden wissen."* (1930)

A Bit of Historical Context: Algorithms in Mathematics

Concrete, algorithmic mathematics $\xrightarrow{19^{\text{th}}\text{ c.}}$ Abstract mathematics

Loss of algorithmic content

Increased interest in the notion of computability

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Ramon Llull (c. 1232–1315), Gottfried Leibniz (1646–1716)

Emil du Bois-Reymond's *ignoramus et ignorabimus* (1880) and Hilbert's "*Wir müssen wissen—wir werden wissen.*" (1930)

Hilbert's 10^{th} Problem: algorithm to decide whether a given Diophantine equation has a solution

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

Examples require a formal notion of computability.

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

Examples require a formal notion of computability.

Various proposed definitions by Church, Gödel, Herbrand, Kleene in the 1930's

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

Examples require a formal notion of computability.

Various proposed definitions by Church, Gödel, Herbrand, Kleene in the 1930's

Turing's machine-based definition (1936)

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

Examples require a formal notion of computability.

Various proposed definitions by Church, Gödel, Herbrand, Kleene in the 1930's

Turing's machine-based definition (1936)

All of these definitions are equivalent.

A Bit of Historical Context: Computability Theory

Despite Hilbertian optimism, not all problems have algorithms.

Examples require a formal notion of computability.

Various proposed definitions by Church, Gödel, Herbrand, Kleene in the 1930's

Turing's machine-based definition (1936)

All of these definitions are equivalent.

Church-Turing Thesis: This definition captures the intuitive notion of "computable".

A Bit of Historical Context: Computability Theory

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

A Bit of Historical Context: Computability Theory

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Thm (Church; Turing). There is no such algorithm.

A Bit of Historical Context: Computability Theory

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Thm (Church; Turing). There is no such algorithm.

Hilbert's 10^{th} Problem: algorithm to decide whether a given Diophantine equation has a solution

Thm (Davis; Putnam; Robinson; Matiyasevich). There is no such algorithm.

A Bit of Historical Context: Computability Theory

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Thm (Church; Turing). There is no such algorithm.

Hilbert's 10^{th} Problem: algorithm to decide whether a given Diophantine equation has a solution

Thm (Davis; Putnam; Robinson; Matiyasevich). There is no such algorithm.

Many other objects have been shown to be noncomputable.

A Bit of Historical Context: Computability Theory

Hilbert's *Entscheidungsproblem*: algorithm to decide, given a set of axioms A and a statement σ, whether σ follows from A

Thm (Church; Turing). There is no such algorithm.

Hilbert's 10^{th} Problem: algorithm to decide whether a given Diophantine equation has a solution

Thm (Davis; Putnam; Robinson; Matiyasevich). There is no such algorithm.

Many other objects have been shown to be noncomputable.

Computability theory has tools to compare such objects.

A Bit of Computability Theory

A Bit of Computability Theory

We look at countably infinite objects built out of finite ones, e.g. sets of natural numbers, sets of finite strings, functions $\mathbb{N} \to \mathbb{N}$, etc.

A Bit of Computability Theory

We look at countably infinite objects built out of finite ones, e.g. sets of natural numbers, sets of finite strings, functions $\mathbb{N} \to \mathbb{N}$, etc.

Computability for such objects can be

thought of via an informal idea of algorithm;

defined formally using a model such as Turing machines.

We look at countably infinite objects built out of finite ones, e.g. sets of natural numbers, sets of finite strings, functions $\mathbb{N} \to \mathbb{N}$, etc.

Computability for such objects can be

thought of via an informal idea of algorithm;

defined formally using a model such as Turing machines.

We can list all Turing machines (with inputs and outputs in $\mathbb{N}$), in such a way that we can simulate the computation of the e^{th} machine on input n using a universal Turing machine.

A Bit of Computability Theory

We look at countably infinite objects built out of finite ones, e.g. sets of natural numbers, sets of finite strings, functions $\mathbb{N} \to \mathbb{N}$, etc.

Computability for such objects can be

thought of via an informal idea of algorithm;

defined formally using a model such as Turing machines.

We can list all Turing machines (with inputs and outputs in $\mathbb{N}$), in such a way that we can simulate the computation of the e^{th} machine on input n using a universal Turing machine.

A Turing machine may fail to halt on a given input, so this list yields a list $\Phi_0, \Phi_1, \ldots$ of all *partial* computable functions.

A Bit of Computability Theory

We look at countably infinite objects built out of finite ones, e.g. sets of natural numbers, sets of finite strings, functions $\mathbb{N} \to \mathbb{N}$, etc.

Computability for such objects can be

thought of via an informal idea of algorithm;

defined formally using a model such as Turing machines.

We can list all Turing machines (with inputs and outputs in $\mathbb{N}$), in such a way that we can simulate the computation of the e^{th} machine on input n using a universal Turing machine.

A Turing machine may fail to halt on a given input, so this list yields a list $\Phi_0, \Phi_1, \ldots$ of all *partial* computable functions.

We write $\Phi_e(n)\downarrow$ to mean that Φ_e is defined on n.

The Halting Problem is $\emptyset' = \{\langle e, n\rangle : \Phi_e(n)\downarrow\}$.

The Halting Problem is $\emptyset' = \{\langle e, n\rangle : \Phi_e(n)\downarrow\}$.

Thm (Turing). $\emptyset'$ is not computable.

The Halting Problem is $\emptyset' = \{\langle e, n \rangle : \Phi_e(n)\downarrow\}$.

Thm (Turing). $\emptyset'$ is not computable.

Pf. By diagonalization: Suppose that $\emptyset'$ is computable.

Then so is $f(e) = \begin{cases} \Phi_e(e) + 1 & \text{if } \langle e, e \rangle \in \emptyset' \\ 0 & \text{otherwise.} \end{cases}$

Thus $\Phi_e = f$ for some e.

Then $\Phi_e(e)\downarrow = f(e) = \Phi_e(e) + 1. \quad \square$

The Halting Problem is $\emptyset' = \{\langle e, n\rangle : \Phi_e(n)\downarrow\}$.

Thm (Turing). $\emptyset'$ is not computable.

Pf. By diagonalization: Suppose that $\emptyset'$ is computable.

Then so is $f(e) = \begin{cases} \Phi_e(e) + 1 & \text{if } \langle e, e\rangle \in \emptyset' \\ 0 & \text{otherwise.} \end{cases}$

Thus $\Phi_e = f$ for some e.

Then $\Phi_e(e)\downarrow = f(e) = \Phi_e(e) + 1.$ $\square$

A similar proof shows that there is no effective list of all total computable functions.

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

A Bit of Computability Theory

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

So are the sets in the Entscheidungsproblem and in Hilbert's 10^{th} problem.

A Bit of Computability Theory

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

So are the sets in the Entscheidungsproblem and in Hilbert's 10th problem.

A is computable relative to B if there is an algorithm for computing A if given access to B.

Can be formalized using Turing machines with oracle tapes.

A Bit of Computability Theory

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

So are the sets in the Entscheidungsproblem and in Hilbert's 10^{th} problem.

A is computable relative to B if there is an algorithm for computing A if given access to B.

Can be formalized using Turing machines with oracle tapes.

We write $A \leqslant_{\mathrm{T}} B$ and say that A is Turing reducible to B.

A Bit of Computability Theory

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

So are the sets in the Entscheidungsproblem and in Hilbert's 10^{th} problem.

A is computable relative to B if there is an algorithm for computing A if given access to B.

Can be formalized using Turing machines with oracle tapes.

We write $A \leqslant_{\mathrm{T}} B$ and say that A is Turing reducible to B.

If $A \leqslant_{\mathrm{T}} B$ and $B \leqslant_{\mathrm{T}} A$ then A and B are Turing equivalent.

The resulting equivalence classes are the Turing degrees.

$\emptyset'$ is not computable, but it is computably enumerable (c.e.).

So are the sets in the Entscheidungsproblem and in Hilbert's 10^{th} problem.

A is computable relative to B if there is an algorithm for computing A if given access to B.

Can be formalized using Turing machines with oracle tapes.

We write $A \leqslant_{\mathrm{T}} B$ and say that A is Turing reducible to B.

If $A \leqslant_{\mathrm{T}} B$ and $B \leqslant_{\mathrm{T}} A$ then A and B are Turing equivalent.

The resulting equivalence classes are the Turing degrees.

The degree of the join $A \oplus B = \{2n : n \in A\} \cup \{2n + 1 : n \in B\}$ is the least upper bound of the degrees of A and B.

For a c.e. A, define a partial computable f s.t. $f(n)\downarrow$ iff $n \in A$.

$\emptyset'$ can tell whether $f(n)\downarrow$, so A is computable relative to $\emptyset'$.

A Bit of Computability Theory

For a c.e. A, define a partial computable f s.t. $f(n)\downarrow$ iff $n \in A$.

$\emptyset'$ can tell whether $f(n)\downarrow$, so A is computable relative to $\emptyset'$.

We say that $\emptyset'$ is a complete c.e. set.

For a c.e. A, define a partial computable f s.t. $f(n)\downarrow$ iff $n \in A$.

$\emptyset'$ can tell whether $f(n)\downarrow$, so A is computable relative to $\emptyset'$.

We say that $\emptyset'$ is a complete c.e. set.

The undecidability of the Entscheidungsproblem and of Hilbert's 10^{th} problem are proved by encoding $\emptyset'$.

So the corresponding c.e. sets are also complete, i.e., they are in the same Turing degree as $\emptyset'$.

For a c.e. A, define a partial computable f s.t. $f(n)\downarrow$ iff $n \in A$.

$\emptyset'$ can tell whether $f(n)\downarrow$, so A is computable relative to $\emptyset'$.

We say that $\emptyset'$ is a complete c.e. set.

The undecidability of the Entscheidungsproblem and of Hilbert's 10^{th} problem are proved by encoding $\emptyset'$.

So the corresponding c.e. sets are also complete, i.e., they are in the same Turing degree as $\emptyset'$.

Thm (Friedberg; Muchnik). There are noncomputable, incomplete c.e. sets.

There are also non-c.e. sets that are computable relative to $\emptyset'$, including co-c.e. sets but also many others.

We can relativize other computability-theoretic concepts.

We can relativize other computability-theoretic concepts.

We can define the concept of being c.e. relative to X.

We can relativize other computability-theoretic concepts.

We can define the concept of being c.e. relative to X.

Let $\Phi_0^X, \Phi_1^X, \ldots$ be the functions that are partial computable relative to X.

We can define the Halting Problem relative to X as $X' = \{\langle e, n\rangle : \Phi_e^X(n)\downarrow\}$.

We can relativize other computability-theoretic concepts.

We can define the concept of being c.e. relative to X.

Let $\Phi_0^X, \Phi_1^X, \ldots$ be the functions that are partial computable relative to X.

We can define the Halting Problem relative to X as $X' = \{\langle e, n \rangle : \Phi_e^X(n)\downarrow\}$.

We call this the (Turing) jump of X.

If $X \leqslant_{\mathrm{T}} Y$ then $X' \leqslant_{\mathrm{T}} Y'$, but not necessarily vice-versa.

We can relativize other computability-theoretic concepts.

We can define the concept of being c.e. relative to X.

Let $\Phi_0^X, \Phi_1^X, \dots$ be the functions that are partial computable relative to X.

We can define the Halting Problem relative to X as $X' = \{\langle e, n\rangle : \Phi_e^X(n)\downarrow\}$.

We call this the (Turing) jump of X.

If $X \leqslant_T Y$ then $X' \leqslant_T Y'$, but not necessarily vice-versa.

Computability-theoretic results tend to relativize.

We can relativize other computability-theoretic concepts.

We can define the concept of being c.e. relative to X.

Let $\Phi_0^X, \Phi_1^X, \dots$ be the functions that are partial computable relative to X.

We can define the Halting Problem relative to X as $X' = \{\langle e, n\rangle : \Phi_e^X(n)\downarrow\}$.

We call this the (Turing) jump of X.

If $X \leqslant_{\mathrm{T}} Y$ then $X' \leqslant_{\mathrm{T}} Y'$, but not necessarily vice-versa.

Computability-theoretic results tend to relativize.

E.g., X' is not computable relative to X, and is complete for sets c.e. relative to X.

Part II: Computability-Theoretic Comparison

An Example: Versions of König's Lemma

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

T is finitely branching if for each $\sigma \in T$, $|\{n : \sigma n \in T\}| < \infty$.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

T is finitely branching if for each $\sigma \in T$, $|\{n : \sigma n \in T\}| < \infty$.

T is binary if it is a subset of $2^{<\omega}$.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

T is finitely branching if for each $\sigma \in T$, $|\{n : \sigma n \in T\}| < \infty$.

T is binary if it is a subset of $2^{<\omega}$.

A path on T is a $P \in \mathbb{N}^{\omega}$ s.t. every initial segment of P is in T.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

T is finitely branching if for each $\sigma \in T$, $|\{n : \sigma n \in T\}| < \infty$.

T is binary if it is a subset of $2^{<\omega}$.

A path on T is a $P \in \mathbb{N}^{\omega}$ s.t. every initial segment of P is in T.

Put a topology on $\mathbb{N}^{\omega}$ by taking $\{X : \sigma \prec X\}$ as basic open sets.

Then $\mathcal{C}$ is closed iff it is the set of paths on a tree.

Trees and Paths

A tree is a subset T of $\mathbb{N}^{<\omega}$ closed under initial segments.

T is computable if there is an algorithm for determining whether a given σ is in T.

T is finitely branching if for each $\sigma \in T$, $|\{n : \sigma n \in T\}| < \infty$.

T is binary if it is a subset of $2^{<\omega}$.

A path on T is a $P \in \mathbb{N}^{\omega}$ s.t. every initial segment of P is in T.

Put a topology on $\mathbb{N}^{\omega}$ by taking $\{X : \sigma \prec X\}$ as basic open sets.

Then $\mathcal{C}$ is closed iff it is the set of paths on a tree.

Put a measure on 2^{ω} by letting $\mu(\{X : \sigma \prec X\}) = 2^{-|\sigma|}$.

Versions of König's Lemma

König's Lemma: Every infinite, finitely branching tree has a path.

Versions of König's Lemma

König's Lemma: Every infinite, finitely branching tree has a path.

Weak König's Lemma: Every infinite binary tree has a path.

Versions of König's Lemma

König's Lemma: Every infinite, finitely branching tree has a path.

Weak König's Lemma: Every infinite binary tree has a path.

Weak Weak König's Lemma: Every binary tree T s.t.

$$\liminf_n \frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n} > 0$$

has a path.

Versions of König's Lemma

König's Lemma: Every infinite, finitely branching tree has a path.

Weak König's Lemma: Every infinite binary tree has a path.

Weak Weak König's Lemma: Every binary tree T s.t.

$$\liminf_n \frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n} > 0$$

has a path.

Bounded König's Lemma: Every infinite binary tree T s.t.

$$|\{\sigma \in T : |\sigma| = n\}| < c$$

for some c has a path.

Versions of König's Lemma

KL: Infinite, finitely branching trees have paths.

WKL: Infinite binary trees have paths.

WWKL: Fat binary trees have paths.

BKL: Skinny infinite binary trees have paths.

Versions of König's Lemma

KL: Infinite, finitely branching trees have paths.

WKL: Infinite binary trees have paths.

WWKL: Fat binary trees have paths.

BKL: Skinny infinite binary trees have paths.

WKL says that 2^ω is compact.

Versions of König's Lemma

KL: Infinite, finitely branching trees have paths.

WKL: Infinite binary trees have paths.

WWKL: Fat binary trees have paths.

BKL: Skinny infinite binary trees have paths.

WKL says that 2^ω is compact.

KL says that certain subspaces of $\mathbb{N}^\omega$ are compact, but these subspaces are not as effectively presented.

Versions of König's Lemma

KL: Infinite, finitely branching trees have paths.

WKL: Infinite binary trees have paths.

WWKL: Fat binary trees have paths.

BKL: Skinny infinite binary trees have paths.

WKL says that 2^ω is compact.

KL says that certain subspaces of $\mathbb{N}^\omega$ are compact, but these subspaces are not as effectively presented.

WKL: Find an element of a closed set.

WWKL: Find an element of a closed of positive measure.

BKL: Find an element of a finite set.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

We can find τ_n computably.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

We can find τ_n computably.

$P = \lim_n \tau_n$, so T has a computable path.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

We can find τ_n computably.

$P = \lim_n \tau_n$, so T has a computable path.

In fact, every path on T is computable.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

We can find τ_n computably.

$P = \lim_n \tau_n$, so T has a computable path.

In fact, every path on T is computable.

More generally, even if T is not computable, the above procedure is computable *relative to T*.

Bounded König's Lemma and Computability

Let T be a computable infinite binary tree s.t. $|\{\sigma \in T : |\sigma| = n\}| < c$ for all n.

There is a $\sigma \in T$ extended by a unique path P on T.

For each $n > |\sigma|$, there is a unique $\tau_n \succ \sigma$ of length n s.t. T is infinite above τ_n.

We can find τ_n computably.

$P = \lim_n \tau_n$, so T has a computable path.

In fact, every path on T is computable.

More generally, even if T is not computable, the above procedure is computable *relative to* T.

Thus BKL is computably true.

Thm (Kreisel). There is a computable infinite binary tree with no computable path.

Thus WKL is not computably true, and hence neither is KL.

Kreisel's tree can be fat, so WWKL is also not computably true.

Thm (Kreisel). There is a computable infinite binary tree with no computable path.

Thus WKL is not computably true, and hence neither is KL.

Kreisel's tree can be fat, so WWKL is also not computably true.

To build such a tree, we diagonalize against all potential computable paths.

Thm (Kreisel). There is a computable infinite binary tree with no computable path.

Thus WKL is not computably true, and hence neither is KL.

Kreisel's tree can be fat, so WWKL is also not computably true.

To build such a tree, we diagonalize against all potential computable paths.

There is a computable infinite, finitely branching tree T s.t. every path of T computes $\emptyset'$.

Thm (Kreisel). There is a computable infinite binary tree with no computable path.

Thus WKL is not computably true, and hence neither is KL.

Kreisel's tree can be fat, so WWKL is also not computably true.

To build such a tree, we diagonalize against all potential computable paths.

There is a computable infinite, finitely branching tree T s.t. every path of T computes $\emptyset'$.

There is a computable infinite, finitely branching tree T with no path computable relative to $\emptyset'$.

Let T be a computable infinite binary tree.

Thm (Kreisel). T has a path $P \leqslant_{\mathbf{T}} \emptyset'$.

An example is the leftmost path of T.

Let T be a computable infinite binary tree.

Thm (Kreisel). T has a path $P \leqslant_{\mathbf{T}} \emptyset'$.

An example is the leftmost path of T.

Thm (Shoenfield). T has a path $P <_{\mathbf{T}} \emptyset'$.

Thus WKL is strictly weaker than KL in at least two senses.

Let T be a computable infinite binary tree.

Thm (Kreisel). T has a path $P \leqslant_{\mathbf{T}} \emptyset'$.

An example is the leftmost path of T.

Thm (Shoenfield). T has a path $P <_{\mathbf{T}} \emptyset'$.

Thus WKL is strictly weaker than KL in at least two senses.

But just how much weaker?

Weak König's Lemma and Computability

Let T be a computable infinite binary tree.

Thm (Kreisel). T has a path $P \leqslant_{\mathrm{T}} \emptyset'$.

An example is the leftmost path of T.

Thm (Shoenfield). T has a path $P <_{\mathrm{T}} \emptyset'$.

Thus WKL is strictly weaker than KL in at least two senses.

But just how much weaker?

Low Basis Thm (Jockusch and Soare). T has a path P s.t. $P' \leqslant_{\mathrm{T}} \emptyset'$.

Such a P is called low.

Weak König's Lemma and Computability

Let T be a computable infinite binary tree.

Thm (Kreisel). T has a path $P \leqslant_{\mathrm{T}} \emptyset'$.

An example is the leftmost path of T.

Thm (Shoenfield). T has a path $P <_{\mathrm{T}} \emptyset'$.

Thus WKL is strictly weaker than KL in at least two senses.

But just how much weaker?

Low Basis Thm (Jockusch and Soare). T has a path P s.t. $P' \leqslant_{\mathrm{T}} \emptyset'$.

Such a P is called low.

This theorem relativizes: If the binary tree T is computable relative to X then T has a path P s.t. $(P \oplus X)' \leqslant_{\mathrm{T}} X'$.

Computable Entailment

Second-Order Statements

Versions of KL are second-order statements, involving quantification over first-order (finite) objects and second-order (countably infinite) objects.

Second-Order Statements

Versions of KL are second-order statements, involving quantification over first-order (finite) objects and second-order (countably infinite) objects.

We can encode finite objects as natural numbers: e.g., strings, rationals, finite sets, . . .

We can encode countably infinite objects as sets of natural numbers: e.g., infinite sequences, trees, groups, reals, . . .

Versions of KL are second-order statements, involving quantification over first-order (finite) objects and second-order (countably infinite) objects.

We can encode finite objects as natural numbers: e.g., strings, rationals, finite sets, ...

We can encode countably infinite objects as sets of natural numbers: e.g., infinite sequences, trees, groups, reals, ...

So we might encode a $\sigma \in 2^{<\omega}$ of length n as $2\sigma(0) + 4\sigma(1) + \cdots + 2^n\sigma(n-1)$.

Then a tree is just a particular kind of subset of $\mathbb{N}$.

Thus we can work in second-order arithmetic.

Π^1_2 Statements

Statements involving only first-order quantification are called arithmetic.

Π^1_2 Statements

Statements involving only first-order quantification are called arithmetic.

Version of KL are of the form

$$\forall X\,[\Theta(X) \;\rightarrow\; \exists Y\,\Psi(X,Y)],$$

where Θ and Ψ are arithmetic.

Π^1_2 Statements

Statements involving only first-order quantification are called arithmetic.

Version of KL are of the form

$$\forall X\,[\Theta(X) \;\rightarrow\; \exists Y\,\Psi(X, Y)],$$

where Θ and Ψ are arithmetic.

We can think of such a statement as a problem:

An instance is an X s.t. $\Theta(X)$ holds.

A solution to X is a Y s.t. $\Psi(X, Y)$ holds.

Π^1_2 Statements

Statements involving only first-order quantification are called arithmetic.

Version of KL are of the form

$$\forall X\,[\Theta(X) \rightarrow \exists Y\,\Psi(X,Y)],$$

where Θ and Ψ are arithmetic.

We can think of such a statement as a problem:

An instance is an X s.t. $\Theta(X)$ holds.

A solution to X is a Y s.t. $\Psi(X,Y)$ holds.

Solving an instance of WKL takes less power than solving an instance of KL.

Π^1_2 Statements

Statements involving only first-order quantification are called arithmetic.

Version of KL are of the form

$$\forall X\,[\Theta(X) \rightarrow \exists Y\,\Psi(X,Y)],$$

where Θ and Ψ are arithmetic.

We can think of such a statement as a problem:

An instance is an X s.t. $\Theta(X)$ holds.

A solution to X is a Y s.t. $\Psi(X,Y)$ holds.

Solving an instance of WKL takes less power than solving an instance of KL.

But what about multiple instances?

Turing Ideals

A Turing ideal is an $\mathcal{I} \subseteq 2^{\mathbb{N}}$ s.t. if $B_1, \ldots, B_n \in \mathcal{I}$ and A is computable relative to $B_1, \ldots, B_n$ then $A \in \mathcal{I}$.

Turing Ideals

A Turing ideal is an $\mathcal{I} \subseteq 2^{\mathbb{N}}$ s.t. if $B_1, \ldots, B_n \in \mathcal{I}$ and A is computable relative to $B_1, \ldots, B_n$ then $A \in \mathcal{I}$.

A problem P holds in $\mathcal{I}$ if for every instance X of P in $\mathcal{I}$, there is a solution Y to X in $\mathcal{I}$.

A Turing ideal is an $\mathcal{I} \subseteq 2^{\mathbb{N}}$ s.t. if $B_1, \ldots, B_n \in \mathcal{I}$ and A is computable relative to $B_1, \ldots, B_n$ then $A \in \mathcal{I}$.

A problem P holds in $\mathcal{I}$ if for every instance X of P in $\mathcal{I}$, there is a solution Y to X in $\mathcal{I}$.

P computably entails Q, written as $P \vDash_{\mathbf{c}} Q$, if Q holds in every Turing ideal in which P holds.

P and Q are computably equivalent if they hold in the same Turing ideals.

Turing Ideals

A Turing ideal is an $\mathcal{I} \subseteq 2^{\mathbb{N}}$ s.t. if $B_1, \ldots, B_n \in \mathcal{I}$ and A is computable relative to $B_1, \ldots, B_n$ then $A \in \mathcal{I}$.

A problem P holds in $\mathcal{I}$ if for every instance X of P in $\mathcal{I}$, there is a solution Y to X in $\mathcal{I}$.

P computably entails Q, written as $P \vDash_{\mathbf{c}} Q$, if Q holds in every Turing ideal in which P holds.

P and Q are computably equivalent if they hold in the same Turing ideals.

A statement Φ of second-order arithmetic holds in $\mathcal{I}$ if Φ is true when $\exists X$ and $\forall X$ are replaced by $\exists X \in \mathcal{I}$ and $\forall X \in \mathcal{I}$.

Building Turing Ideals

Clearly KL $\vDash_{\mathbf{c}}$ WKL and WKL $\vDash_{\mathbf{c}}$ WWKL.

BKL holds in every Turing ideal, so WWKL $\vDash_{\mathbf{c}}$ BKL.

Building Turing Ideals

Clearly KL $\models_{\mathbf{c}}$ WKL and WKL $\models_{\mathbf{c}}$ WWKL.

BKL holds in every Turing ideal, so WWKL $\models_{\mathbf{c}}$ BKL.

The computable sets form a Turing ideal $\mathcal{I}$, and WWKL does not hold in $\mathcal{I}$, so BKL $\not\models_{\mathbf{c}}$ WWKL.

Clearly KL $\models_{\mathbf{c}}$ WKL and WKL $\models_{\mathbf{c}}$ WWKL.

BKL holds in every Turing ideal, so WWKL $\models_{\mathbf{c}}$ BKL.

The computable sets form a Turing ideal $\mathcal{I}$, and WWKL does not hold in $\mathcal{I}$, so BKL $\not\models_{\mathbf{c}}$ WWKL.

Thm (Scott/Jockusch and Soare/Friedman). WKL $\not\models_{\mathbf{c}}$ KL.

The proof uses the relativized Low Basis Theorem: If the binary tree T is computable relative to X then T has a path P s.t. $(P \oplus X)' \leqslant_{\mathbf{T}} X'$.

Clearly KL $\vDash_{\mathbf{c}}$ WKL and WKL $\vDash_{\mathbf{c}}$ WWKL.

BKL holds in every Turing ideal, so WWKL $\vDash_{\mathbf{c}}$ BKL.

The computable sets form a Turing ideal $\mathcal{I}$, and WWKL does not hold in $\mathcal{I}$, so BKL $\nvDash_{\mathbf{c}}$ WWKL.

Thm (Scott/Jockusch and Soare/Friedman). WKL $\nvDash_{\mathbf{c}}$ KL.

The proof uses the relativized Low Basis Theorem: If the binary tree T is computable relative to X then T has a path P s.t. $(P \oplus X)' \leqslant_{\mathbf{T}} X'$.

Thm (Yu and Simpson). WWKL $\nvDash_{\mathbf{c}}$ WKL.

The proof uses the theory of algorithmic randomness.

Computable equivalence

BKL, WWKL, WKL, and KL represent important complexity levels.

Computable equivalence

BKL, WWKL, WKL, and KL represent important complexity levels.

Computably equivalent to BKL (i.e., computably true):

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

  ⋮

Computable equivalence

BKL, WWKL, WKL, and KL represent important complexity levels.

Computably equivalent to BKL (i.e., computably true):

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

  ⋮

Computably equivalent to WWKL:

- the Vitali Covering Theorem
- the monotone convergence theorem for Lebesgue measure on [0, 1]
- the existence of (relatively) Martin-Löf random sequences

  ⋮

Computable equivalence

Computably equivalent to WKL:

- the uniqueness of algebraic closures for fields
- the existence of prime ideals for commutative rings
- the Compactness Theorem for first-order logic
- the Extreme Value Theorem
- Brouwer's Fixed Point Theorem

  ⋮

Computable equivalence

Computably equivalent to WKL:

- the uniqueness of algebraic closures for fields
- the existence of prime ideals for commutative rings
- the Compactness Theorem for first-order logic
- the Extreme Value Theorem
- Brouwer's Fixed Point Theorem

⋮

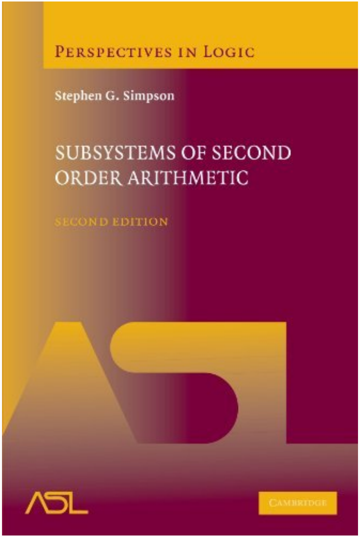
Computably equivalent to KL:

- the existence of maximal ideals for commutative rings
- the existence of bases for vector spaces
- the Bolzano-Weierstraß Theorem
- the existence of the Turing jump

⋮

Part III: Reverse Mathematics

Stephen G. Simpson, *Subsystems of Second Order Arithmetic*



Second-Order Arithmetic and RCA_0

Second-Order Arithmetic

We work in a language with number variables, set variables, and symbols $0, 1, S, <, +, \cdot, \in$.

We work in a language with number variables, set variables, and symbols $0, 1, S, <, +, \cdot, \in$.

Again we encode finite objects as natural numbers and infinite objects as sets of natural numbers.

Second-Order Arithmetic

We work in a language with number variables, set variables, and symbols $0, 1, S, <, +, \cdot, \in$.

Again we encode finite objects as natural numbers and infinite objects as sets of natural numbers.

Reverse Mathematics: fix a weak base system and calibrate the strength of principles by considering implications over this system.

Often in terms of a few subsystems of second-order arithmetic.

Second-Order Arithmetic

Full second-order arithmetic consists of

- axioms for a discrete ordered commutative semiring
- comprehension:

$$\exists X \, \forall n \, [n \in X \;\leftrightarrow\; \varphi(n)]$$

  for all formulas φ s.t. X is not free in φ

- induction:

$$(\varphi(0) \;\wedge\; \forall n \, [\varphi(n) \;\rightarrow\; \varphi(n+1)] \;\rightarrow\; \forall n \, \varphi(n)$$

  for all formulas φ

Second-Order Arithmetic

Full second-order arithmetic consists of

- axioms for a discrete ordered commutative semiring
- comprehension:

$$\exists X\, \forall n\, [n \in X \;\leftrightarrow\; \varphi(n)]$$

  for all formulas φ s.t. X is not free in φ

- induction:

$$(\varphi(0) \;\wedge\; \forall n\, [\varphi(n) \;\rightarrow\; \varphi(n+1)] \;\rightarrow\; \forall n\, \varphi(n)$$

  for all formulas φ

We obtain subsystems by limiting comprehension and induction.

A Hierarchy of Arithmetic Formulas

A bounded quantifier is one of the form $\forall x < t$ or $\exists x < t$.

A bounded-quantifier formula is an arithmetic formula in which all quantifiers are bounded.

A Hierarchy of Arithmetic Formulas

A bounded quantifier is one of the form $\forall x < t$ or $\exists x < t$.

A bounded-quantifier formula is an arithmetic formula in which all quantifiers are bounded.

A Σ_n^0 formula is one of the form

$$\exists x_1 \, \forall x_2 \, \exists x_3 \, \forall x_4 \cdots Q x_n \, \varphi,$$

where φ is a bounded-quantifier formula and Q is $\exists$ if n is odd and $\forall$ if n is even.

A Hierarchy of Arithmetic Formulas

A bounded quantifier is one of the form $\forall x < t$ or $\exists x < t$.

A bounded-quantifier formula is an arithmetic formula in which all quantifiers are bounded.

A Σ_n^0 formula is one of the form

$$\exists x_1 \, \forall x_2 \, \exists x_3 \, \forall x_4 \cdots Q x_n \, \varphi,$$

where φ is a bounded-quantifier formula and Q is $\exists$ if n is odd and $\forall$ if n is even.

A Π_n^0 formula is one of the form

$$\forall x_1 \, \exists x_2 \, \forall x_3 \, \exists x_4 \cdots Q x_n \, \varphi,$$

where φ is a bounded-quantifier formula and Q is $\forall$ if n is odd and $\exists$ if n is even.

A Hierarchy of Arithmetic Formulas

A bounded quantifier is one of the form $\forall x < t$ or $\exists x < t$.

A bounded-quantifier formula is an arithmetic formula in which all quantifiers are bounded.

A Σ_n^0 formula is one of the form

$$\exists x_1\ \forall x_2\ \exists x_3\ \forall x_4 \cdots Qx_n\ \varphi,$$

where φ is a bounded-quantifier formula and Q is $\exists$ if n is odd and $\forall$ if n is even.

A Π_n^0 formula is one of the form

$$\forall x_1\ \exists x_2\ \forall x_3\ \exists x_4 \cdots Qx_n\ \varphi,$$

where φ is a bounded-quantifier formula and Q is $\forall$ if n is odd and $\exists$ if n is even.

These formulas can have free variables.

RCA_0 is obtained by restricting:

- comprehension to Δ^0_1-comprehension:

$$\forall n\,[\varphi(n) \;\leftrightarrow\; \psi(n)] \;\rightarrow\; \exists X\,\forall n\,[n \in X \;\leftrightarrow\; \varphi(n)]$$

for all φ, ψ s.t. φ is Σ^0_1 and ψ is Π^0_1, and X is not free in φ

- induction to Σ^0_1-induction:

$$(\varphi(0) \;\wedge\; \forall n\,[\varphi(n) \;\rightarrow\; \varphi(n+1)]) \;\rightarrow\; \forall n\,\varphi(n)$$

for all Σ^0_1 formulas φ

The Weak Base System RCA_0

RCA_0 is obtained by restricting:

- comprehension to Δ^0_1-comprehension:

$$\forall n\,[\varphi(n) \leftrightarrow \psi(n)] \rightarrow \exists X\, \forall n\, [n \in X \leftrightarrow \varphi(n)]$$

  for all φ, ψ s.t. φ is Σ^0_1 and ψ is Π^0_1, and X is not free in φ

- induction to Σ^0_1-induction:

$$(\varphi(0) \wedge \forall n\,[\varphi(n) \rightarrow \varphi(n+1)]) \rightarrow \forall n\, \varphi(n)$$

  for all Σ^0_1 formulas φ

This choice of base system creates a tight connection between this approach and computable entailment.

Provable in RCA_0

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

$\vdots$

Some Equivalences over RCA_0

Provable in RCA_0

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

⋮

Provably equivalent to WWKL over RCA_0:

- the Vitali Covering Theorem
- the monotone convergence theorem for Lebesgue measure on [0, 1]
- the existence of (relatively) Martin-Löf random sequences

⋮

Some Equivalences over RCA_0

Provably equivalent to WKL over RCA_0:

- the uniqueness of algebraic closures for fields
- the existence of prime ideals for commutative rings
- the Compactness Theorem for first-order logic
- the Extreme Value Theorem
- Brouwer's Fixed Point Theorem

  ⋮

Provably equivalent to KL over RCA_0:

- the existence of maximal ideals for commutative rings
- the existence of bases for vector spaces
- the Bolzano-Weierstraß Theorem
- the existence of the Turing jump

  ⋮

Computability and Definability

The Arithmetic Hierarchy

A first-order formula is one with no set variables.

$A \subseteq \mathbb{N}$ is defined in $\mathbb{N}$ by a first-order formula $\varphi(y)$ if: $k \in A$ iff $\varphi(k)$ holds in $\mathbb{N}$.

The Arithmetic Hierarchy

A first-order formula is one with no set variables.

$A \subseteq \mathbb{N}$ is defined in $\mathbb{N}$ by a first-order formula $\varphi(y)$ if: $k \in A$ iff $\varphi(k)$ holds in $\mathbb{N}$.

A set is Σ^0_n if it is defined in $\mathbb{N}$ by some Σ^0_n first-order formula.

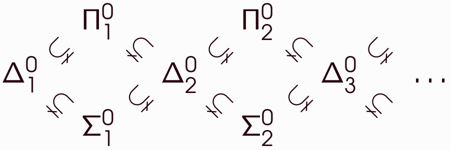
A set is Π^0_n if it is defined in $\mathbb{N}$ by some Π^0_n first-order formula.

A set is Δ^0_n if it is both Σ^0_n and Π^0_n.

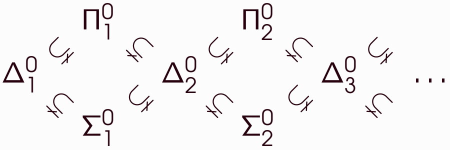
A set is arithmetic if it is in one of these classes.

The Arithmetic Hierarchy

$$
\begin{array}{ccccccccccc}
 & & \Pi^0_1 & & & & \Pi^0_2 & & & & \\
 & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & \\
\Delta^0_1 & & & & \Delta^0_2 & & & & \Delta^0_3 & & \cdots \\
 & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & \\
 & & \Sigma^0_1 & & & & \Sigma^0_2 & & & &
\end{array}
$$



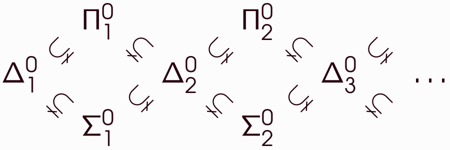
Thm (Kleene). A is Σ^0_1 iff A is c.e. Thus A is Δ^0_1 iff A is computable.



Thm (Kleene). A is Σ^0_1 iff A is c.e. Thus A is Δ^0_1 iff A is computable.

Recall that Z' is the Halting Problem relative to Z.

Define $X^{(n)}$ as follows: $X^{(0)} = X$ and $X^{(n+1)} = (X^{(n)})'$.



Thm (Kleene). A is Σ^0_1 iff A is c.e. Thus A is Δ^0_1 iff A is computable.

Recall that Z' is the Halting Problem relative to Z.

Define $X^{(n)}$ as follows: $X^{(0)} = X$ and $X^{(n+1)} = (X^{(n)})'$.

Thm (Post). A set is Σ^0_{n+1} iff it is c.e. relative to $\emptyset^{(n)}$, and is Δ^0_{n+1} iff it is computable relative to $\emptyset^{(n)}$.

Relativizing the Arithmetic Hierarchy

All of this can be relativized to any $S \subseteq \mathbb{N}$:

Relativizing the Arithmetic Hierarchy

All of this can be relativized to any $S \subseteq \mathbb{N}$:

Consider formulas with one free set variable X.

$A \subseteq N$ is defined in $(\mathbb{N}, S)$ by $\varphi(y, X)$ if: $k \in A$ iff $\varphi(k, S)$ holds in $\mathbb{N}$.

Relativizing the Arithmetic Hierarchy

All of this can be relativized to any $S \subseteq \mathbb{N}$:

Consider formulas with one free set variable X.

$A \subseteq N$ is defined in $(\mathbb{N}, S)$ by $\varphi(y, X)$ if: $k \in A$ iff $\varphi(k, S)$ holds in $\mathbb{N}$.

A set is Σ^0_n relative to S if it is defined in $(\mathbb{N}, S)$ by some Σ^0_n formula.

A set is Π^0_n relative to S if it is defined in $(\mathbb{N}, S)$ by some Π^0_n formula.

A set is Δ^0_n relative to S if it is both Σ^0_n and Π^0_n relative to S.

Relativizing the Arithmetic Hierarchy

All of this can be relativized to any $S \subseteq \mathbb{N}$:

Consider formulas with one free set variable X.

$A \subseteq N$ is defined in $(\mathbb{N}, S)$ by $\varphi(y, X)$ if: $k \in A$ iff $\varphi(k, S)$ holds in $\mathbb{N}$.

A set is Σ^0_n relative to S if it is defined in $(\mathbb{N}, S)$ by some Σ^0_n formula.

A set is Π^0_n relative to S if it is defined in $(\mathbb{N}, S)$ by some Π^0_n formula.

A set is Δ^0_n relative to S if it is both Σ^0_n and Π^0_n relative to S.

Post's Theorem holds in relativized form.

In particular, A is Δ^0_1 relative to S iff A is computable relative to S.

RCA_0 and Computability

Recall that RCA_0 is obtained by restricting:

- comprehension to Δ^0_1-comprehension:

$$\forall n\,[\varphi(n) \leftrightarrow \psi(n)] \rightarrow \exists X\,\forall n\,[n \in X \leftrightarrow \varphi(n)]$$

  for all φ, ψ s.t. φ is Σ^0_1 and ψ is Π^0_1, and X is not free in φ

- induction to Σ^0_1-induction:

$$(\varphi(0) \wedge \forall n\,[\varphi(n) \rightarrow \varphi(n+1)]) \rightarrow \forall n\,\varphi(n)$$

  for all Σ^0_1 formulas φ

RCA_0 and Computability

Recall that RCA_0 is obtained by restricting:

- comprehension to Δ^0_1-comprehension:

$$\forall n\,[\varphi(n) \leftrightarrow \psi(n)] \rightarrow \exists X\,\forall n\,[n \in X \leftrightarrow \varphi(n)]$$

  for all φ, ψ s.t. φ is Σ^0_1 and ψ is Π^0_1, and X is not free in φ

- induction to Σ^0_1-induction:

$$(\varphi(0) \land \forall n\,[\varphi(n) \rightarrow \varphi(n+1)]) \rightarrow \forall n\,\varphi(n)$$

  for all Σ^0_1 formulas φ

Δ^0_1-comprehension is (relative) computable comprehension.

Indeed, RCA stands for Recursive Comprehension Axiom.

Models of RCA_0

A model in the language of second-order arithmetic consists of a first-order part $\mathcal{N} = (N; 0_N, 1_N, S_N, <_N, +_N, \cdot_N)$ and a second-order part $\mathcal{S} \subseteq 2^N$.

Models of RCA_0

A model in the language of second-order arithmetic consists of a first-order part $\mathcal{N} = (N; 0_N, 1_N, S_N, <_N, +_N, \cdot_N)$ and a second-order part $\mathcal{S} \subseteq 2^N$.

If $\mathcal{N}$ is the standard natural numbers, we call this an ω-model and identify it with $\mathcal{S}$.

Models of RCA_0

A model in the language of second-order arithmetic consists of a first-order part $\mathcal{N} = (N; 0_N, 1_N, S_N, <_N, +_N, \cdot_N)$ and a second-order part $\mathcal{S} \subseteq 2^N$.

If $\mathcal{N}$ is the standard natural numbers, we call this an ω-model and identify it with $\mathcal{S}$.

Thm (Friedman). $\mathcal{S}$ is an ω-model of RCA_0 iff $\mathcal{S}$ is a Turing ideal.

Cor. If $\mathsf{RCA}_0 + P \vdash Q$ then $P \vDash_{\mathbf{c}} Q$.

Models of RCA_0

A model in the language of second-order arithmetic consists of a first-order part $\mathcal{N} = (N; 0_N, 1_N, S_N, <_N, +_N, \cdot_N)$ and a second-order part $\mathcal{S} \subseteq 2^N$.

If $\mathcal{N}$ is the standard natural numbers, we call this an ω-model and identify it with $\mathcal{S}$.

Thm (Friedman). $\mathcal{S}$ is an ω-model of RCA_0 iff $\mathcal{S}$ is a Turing ideal.

Cor. If $RCA_0 + P \vdash Q$ then $P \vDash_{\mathbf{c}} Q$.

The converse does not always hold because non-ω-models of RCA_0 exist, but it often does.

The Reverse-Mathematical Universe

Mathematics in RCA_0

Several theorems can be proved in RCA_0, e.g. many basic properties of the natural numbers and the reals, as well as

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

$\vdots$

Mathematics in RCA_0

Several theorems can be proved in RCA_0, e.g. many basic properties of the natural numbers and the reals, as well as

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

  ⋮

But the computable sets form an ω-model of RCA_0, so theorems that are not computably true cannot be proved in RCA_0.

Mathematics in RCA_0

Several theorems can be proved in RCA_0, e.g. many basic properties of the natural numbers and the reals, as well as

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

  ⋮

But the computable sets form an ω-model of RCA_0, so theorems that are not computably true cannot be proved in RCA_0.

Limited induction also plays a role.

Mathematics in RCA_0

Several theorems can be proved in RCA_0, e.g. many basic properties of the natural numbers and the reals, as well as

- the existence of algebraic closures of fields
- Gödel's Completeness Theorem for theories
- the Intermediate Value Theorem
- the Tietze Extension Theorem for complete separable metric spaces

  ⋮

But the computable sets form an ω-model of RCA_0, so theorems that are not computably true cannot be proved in RCA_0.

Limited induction also plays a role.

Thm (Yokoyama). BKL is not provable in RCA_0.

Other Subsystems of Second-Order Arithmetic

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0 implies arithmetic induction.

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0 implies arithmetic induction.

A Turing ideal is an ω-model of ACA_0 iff it is closed under jumps.

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0 implies arithmetic induction.

A Turing ideal is an ω-model of ACA_0 iff it is closed under jumps.

RCA_0 + Σ^0_1-comprehension implies ACA_0.

Other Subsystems of Second-Order Arithmetic

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0 implies arithmetic induction.

A Turing ideal is an ω-model of ACA_0 iff it is closed under jumps.

RCA_0 + Σ^0_1-comprehension implies ACA_0.

KL is equivalent to ACA_0 over RCA_0.

Other Subsystems of Second-Order Arithmetic

ACA_0: RCA_0 + arithmetic comprehension:

$$\exists X \forall n [n \in X \leftrightarrow \varphi(n)]$$

for all arithmetic φ s.t. X is not free in φ

ACA_0 implies arithmetic induction.

A Turing ideal is an ω-model of ACA_0 iff it is closed under jumps.

RCA_0 + Σ^0_1-comprehension implies ACA_0.

KL is equivalent to ACA_0 over RCA_0. So are

- the existence of maximal ideals for commutative rings
- the existence of bases for vector spaces
- the Bolzano-Weierstraß Theorem
- the existence of the Turing jump

  ⋮

Other Subsystems of Second-Order Arithmetic

WKL_0: RCA_0 + Weak König's Lemma

Other Subsystems of Second-Order Arithmetic

WKL_0: RCA_0 + Weak König's Lemma

WKL_0 is arithmetically conservative over RCA_0, i.e., if WKL_0 proves an arithmetic statement, then so does RCA_0.

So WKL_0 has the same amount of induction as RCA_0.

Other Subsystems of Second-Order Arithmetic

WKL_0: RCA_0 + Weak König's Lemma

WKL_0 is arithmetically conservative over RCA_0, i.e., if WKL_0 proves an arithmetic statement, then so does RCA_0.

So WKL_0 has the same amount of induction as RCA_0.

ω-models of WKL_0 are also known as Scott sets.

Other Subsystems of Second-Order Arithmetic

WKL_0: RCA_0 + Weak König's Lemma

WKL_0 is arithmetically conservative over RCA_0, i.e., if WKL_0 proves an arithmetic statement, then so does RCA_0.

So WKL_0 has the same amount of induction as RCA_0.

ω-models of WKL_0 are also known as Scott sets.

Equivalents of WKL_0

- the uniqueness of algebraic closures for fields
- the existence of prime ideals for commutative rings
- the Compactness Theorem for first-order logic
- the Extreme Value Theorem
- Brouwer's Fixed Point Theorem

  ⋮

Other Subsystems of Second-Order Arithmetic

WWKL_0: RCA_0 + Weak Weak König's Lemma

Equivalents of WWKL_0:

- the Vitali Covering Theorem
- the monotone convergence theorem for Lebesgue measure on $[0, 1]$
- the existence of (relatively) Martin-Löf random sequences

⋮

ATR_0: RCA_0 + arithmetic transfinite recursion

Other Subsystems of Second-Order Arithmetic

ATR_0: RCA_0 + arithmetic transfinite recursion

Equivalents of ATR_0

- comparability of well-orderings
- Ulm's Theorem on Abelian p-groups
- the Perfect Set Theorem

  ⋮

Other Subsystems of Second-Order Arithmetic

ATR_0: RCA_0 + arithmetic transfinite recursion

Equivalents of ATR_0

- comparability of well-orderings
- Ulm's Theorem on Abelian p-groups
- the Perfect Set Theorem

$\vdots$

Π^1_1-CA_0: RCA_0 + Π^1_1-comprehension

A Π^1_1 formula is one of the form $\forall X\ \varphi$, where φ is arithmetic.

Other Subsystems of Second-Order Arithmetic

ATR_0: RCA_0 + arithmetic transfinite recursion

Equivalents of ATR_0

- comparability of well-orderings
- Ulm's Theorem on Abelian p-groups
- the Perfect Set Theorem

⋮

Π^1_1-CA_0: RCA_0 + Π^1_1-comprehension

A Π^1_1 formula is one of the form $\forall X\ \varphi$, where φ is arithmetic.

Equivalents of Π^1_1-CA_0

- every countable Abelian group is the direct sum of a divisible group and a reduced group
- the Cantor-Bendixson Theorem

⋮

Relationships between Subsystems of Second-Order Arithmetic

$$\Pi^1_1\text{-}CA_0$$

$$\downarrow$$

$$ATR_0$$

$$\downarrow$$

$$ACA_0$$

$$\downarrow$$

$$WKL_0$$

$$\downarrow$$

$$WWKL_0$$

$$\downarrow$$

$$RCA_0$$

Ramsey's Theorem

$[X]^n$ is the set of n-element subsets of X.

A k-coloring of $[X]^n$ is a map $c : [X]^n \to k$.

A set $H \subseteq X$ is homogeneous for c if $|c([H]^n)| = 1$.

Ramsey's Theorem for n-tuples and k colors (RT^n_k): Every k-coloring of $[\mathbb{N}]^n$ has an infinite homogeneous set.

Ramsey's Theorem

$[X]^n$ is the set of n-element subsets of X.

A k-coloring of $[X]^n$ is a map $c : [X]^n \to k$.

A set $H \subseteq X$ is homogeneous for c if $|c([H]^n)| = 1$.

Ramsey's Theorem for n-tuples and k colors (RT^n_k): Every k-coloring of $[\mathbb{N}]^n$ has an infinite homogeneous set.

For $j, k \geqslant 2$, RT^n_j and RT^n_k are equivalent over RCA_0.

$[X]^n$ is the set of n-element subsets of X.

A k-coloring of $[X]^n$ is a map $c : [X]^n \to k$.

A set $H \subseteq X$ is homogeneous for c if $|c([H]^n)| = 1$.

Ramsey's Theorem for n-tuples and k colors (RT^n_k): Every k-coloring of $[\mathbb{N}]^n$ has an infinite homogeneous set.

For $j, k \geqslant 2$, RT^n_j and RT^n_k are equivalent over RCA_0.

$\mathrm{RT}^n_{<\infty}$ is $\forall k\ \mathrm{RT}^n_k$ and RT is $\forall n\, \forall k\ \mathrm{RT}^n_k$.

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

The Reverse Mathematics of Ramsey's Theorem

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

Thm (Seetapun). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{ACA}_0$.

The Reverse Mathematics of Ramsey's Theorem

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

Thm (Seetapun). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{ACA}_0$.

Thm (Hirst). $\mathsf{WKL}_0 \nvdash \mathsf{RT}^2_2$.

The Reverse Mathematics of Ramsey's Theorem

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

Thm (Seetapun). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{ACA}_0$.

Thm (Hirst). $\mathsf{WKL}_0 \nvdash \mathsf{RT}^2_2$.

Thm (Liu). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{WWKL}_0$.

The Reverse Mathematics of Ramsey's Theorem

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

Thm (Seetapun). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{ACA}_0$.

Thm (Hirst). $\mathsf{WKL}_0 \nvdash \mathsf{RT}^2_2$.

Thm (Liu). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{WWKL}_0$.

$\mathsf{RCA}_0 \vdash \mathsf{RT}^1_k$ for any $k \in \mathbb{N}$, but:

Thm (Hirst). $\mathsf{RCA}_0 \nvdash \mathsf{RT}^1_{<\infty}$.

The Reverse Mathematics of Ramsey's Theorem

Thm (Jockusch/Simpson). For $n \geqslant 3$, $\mathsf{RCA}_0 \vdash \mathsf{RT}^n_2 \leftrightarrow \mathsf{ACA}_0$.

Thm (Seetapun). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{ACA}_0$.

Thm (Hirst). $\mathsf{WKL}_0 \nvdash \mathsf{RT}^2_2$.

Thm (Liu). $\mathsf{RCA}_0 + \mathsf{RT}^2_2 \nvdash \mathsf{WWKL}_0$.

$\mathsf{RCA}_0 \vdash \mathsf{RT}^1_k$ for any $k \in \mathbb{N}$, but:

Thm (Hirst). $\mathsf{RCA}_0 \nvdash \mathsf{RT}^1_{<\infty}$.

Thm (Jockusch). $\mathsf{ACA}_0 \nvdash \mathsf{RT}$.

Some consequences of RT^2_2

Ascending / Descending Sequence Principle (ADS): Every infinite linear order has an infinite ascending or descending sequence.

Some consequences of RT^2_2

Ascending / Descending Sequence Principle (ADS): Every infinite linear order has an infinite ascending or descending sequence.

Chain / Antichain Principle (CAC): Every infinite partial order has an infinite chain or antichain.

Some consequences of RT^2_2

Ascending / Descending Sequence Principle (ADS): Every infinite linear order has an infinite ascending or descending sequence.

Chain / Antichain Principle (CAC): Every infinite partial order has an infinite chain or antichain.

Atomic Model Theorem (AMT): Every complete atomic theory has an atomic model.

Some consequences of RT^2_2

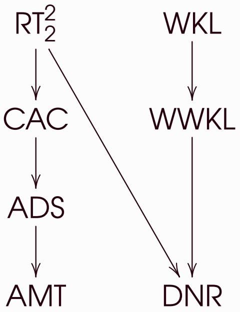
Ascending / Descending Sequence Principle (ADS): Every infinite linear order has an infinite ascending or descending sequence.

Chain / Antichain Principle (CAC): Every infinite partial order has an infinite chain or antichain.

Atomic Model Theorem (AMT): Every complete atomic theory has an atomic model.

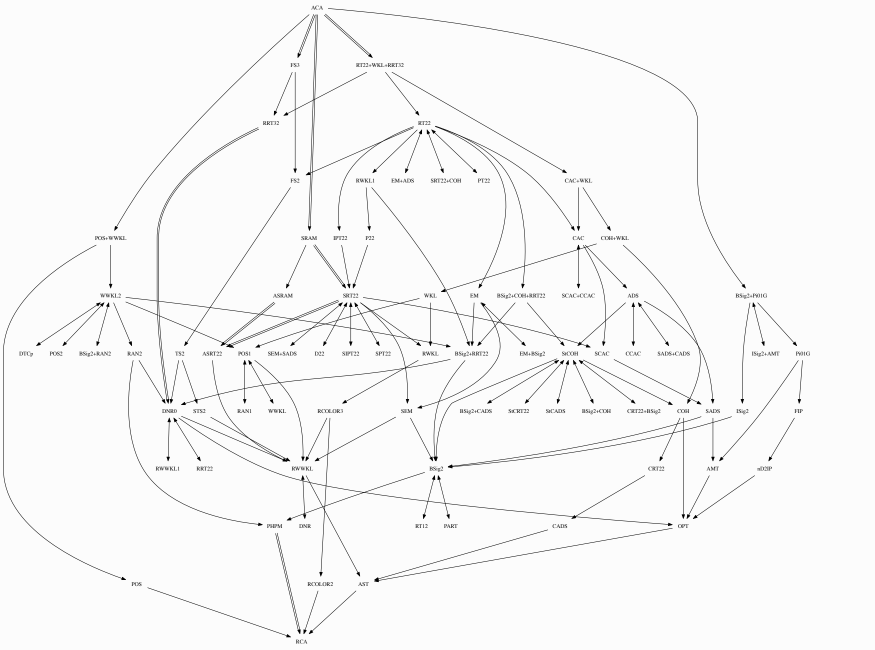
Existence of Diagonally Nonrecursive Functions (DNR): For every X, there is a function f s.t. $f(e) \neq \Phi_e^X(e)$ for all e.

A Small Part of the Universe Between RCA_0 and ACA_0



Combined results of Yu and Simpson; Giusto and Simpson; Ambos-Spies, Kjos-Hanssen, Lempp, and Slaman; Hirschfeldt and Shore; Hirschfeldt, Jockusch, Kjos-Hanssen, Lempp, and Slaman; Hirschfeldt, Shore, and Slaman; Liu; and Lerman, Solomon, and Towsner.

A Larger Part of the Universe Between RCA_0 and ACA_0



Tutorial Series on Reverse Mathematics

Denis R. Hirschfeldt — University of Chicago

2017 NZMRI Summer School, Napier, New Zealand