



# UNIVERSIDAD DE GRANADA

## TRABAJO FIN DE GRADO

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

### Reverse mathematics: axioms and theorems and back again

---

#### Autor

Ignacio Mas Mesa

#### Director

Jesús García Miranda



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN  
FACULTAD DE CIENCIAS

—  
Granada, June 17, 2019



## Matemática inversa: axiomas y teoremas y viceversa

Ignacio Mas Mesa

**Palabras clave:** matemática inversa, axiomas, teoría de la computabilidad, jerarquía aritmética, equivalencia lógica.

### Resumen

La rama de la matemática inversa tal y como la conocemos hoy en día nació en 1975 con el artículo de Harvey Friedman “Some Systems of Second Order Arithmetic and Their Use” [1]. En los años que siguieron a dicho artículo la rama se convirtió en un programa en el que se invirtió una gran cantidad de esfuerzo por parte de diversos investigadores con la intención de desarrollar, mejorar y establecer sus elementos básicos del discurso y principales características definitorias, así como a la profundización en el estudio de sus consecuencias y resultados más interesantes. En 2009, Stephen G. Simpson publicó un tratado de carácter enciclopédico sobre el estado actual de la rama, “Subsystems of Second Order Arithmetic” [2] que hoy en día sigue siendo la referencia de facto sobre la materia y que ha sido frecuentemente consultado y referenciado para la redacción del presente documento. Una introducción más accesible a los principales conceptos presentados en el libro de Simpson fue publicada en 2018 por John Stillwell: “Reverse Mathematics: Proofs from the inside out” [8]. Este libro ha servido como fuente de inspiración para los aspectos más didácticos de este trabajo.

En su forma actual, la matemática inversa se encarga de estudiar la relación entre axiomas y teoremas. Durante la mayor parte de la historia de las matemáticas, los axiomas han sido considerados una herramienta fundamental sobre la que construir el resto de resultados matemáticos mediante un proceso deductivo con el que los matemáticos hoy en día están totalmente familiarizados. Aunque existen algunos ejemplos históricos importantes de estudio de los axiomas en sí mismos, entre los cuales destacan el quinto postulado de Euclides así como el axioma de elección o la hipótesis del continuo en la teoría de conjuntos, la matemática inversa actual estaba sin explorar hasta que Friedman presentó las ideas y herramientas necesarias para la concreción de sus objetivos. En su versión más elemental, la idea clave de la matemática inversa es que la relación entre ciertos axiomas y teoremas no es únicamente de implicación sino de equivalencia. Esto es, en palabras del propio Harvey Friedman: “Cuanto el teorema es demostrado a partir de los axiomas correctos, los axiomas pueden ser demostrados a partir del teorema”.

Por supuesto, esta idea requiere de clarificación y formalización de modo que pueda ser útil para los matemáticos y este será precisamente nuestro principal objetivo en los capítulos venideros.

El capítulo de introducción expone las ideas y conceptos básicos con los que trabajaremos posteriormente de cara a cumplir nuestro objetivo. El principio más destacable de este capítulo es la formalización de la equivalencia a la que hacíamos referencia previamente entre axiomas y teoremas. Para ello, necesitaremos un sistema axiomático base en el que construir la demostración de la equivalencia. La forma clásica de proceder sería asumir el sistema base así como el axioma en el que estamos interesados para demostrar el teorema. La matemática inversa recibe su nombre por la otra parte de la demostración, en la que se asumen el sistema base y el teorema para demostrar el axioma. Es importante tener en cuenta que el sistema base que buscamos debe ser suficientemente expresivo como para poder enunciar el axioma y el teorema (e.g., si se quiere demostrar el teorema de Weierstraß debemos definir en primer lugar qué se entiende por función continua en el sistema base), pero también tan débil que ni el axioma en el que estamos interesados ni el teorema puedan ser demostrados en él.

Después de presentar las ideas y definiciones fundamentales, continuamos con algunos conceptos más avanzados y pasamos a los detalles. Para empezar, realizamos las definiciones necesarias para llegar hasta el concepto de jerarquía aritmética, que nos será de vital importancia para el trabajo posterior y por el que podemos definir una medida de complejidad de ciertas formulas de segundo orden de nuestro lenguaje. También definimos el sistema de la aritmética de segundo orden así como el subsistema en el que estaremos más interesados, a saber,  $RCA_0$ . Por último, demostramos cómo la función emparejadora es capaz de subsumir la estructura de los enteros y los racionales dentro de los naturales y presentamos una definición sintética pero práctica de número real en  $RCA_0$ .

El capítulo siguiente está dedicado a teoría de la computabilidad, que puede parecer una digresión en primera instancia. Empezamos por presentar la aproximación clásica de la teoría de la recursión, en la cual la computabilidad se define inductivamente para una clase reducida de funciones iniciales y para ciertas combinaciones de ellas. También presentamos la alternativa más conocida de las máquinas de Turing y esbozamos una demostración de su equivalencia (en el sentido de que ambas definiciones de computabilidad conducen a la misma clase de funciones). Concluimos esta sección enunciando la tesis de Church-Turing, que nos permitirá proceder de forma más laxa durante el resto del capítulo, pero también confiar en que nuestros resultados pueden ser respaldados por argumentos rigurosos en caso de nece-

sitarlo.

Pasamos entonces a explorar el concepto de enumerabilidad recursiva y presentamos algunos ejemplos y propiedades, descubriendo que las máquinas de Turing (o las funciones parcialmente recursivas, si se quiere) pueden ser enumeradas e incluso demostrando la existencia de una máquina de Turing universal. De este modo encontramos un ejemplo natural de conjunto recursivamente enumerable pero no computable. De hecho, este ejemplo concreto está muy íntimamente relacionado con el problema de la parada, lo cual nos sirve para ligar nuestros desarrollos teóricos con algunos de los aspectos más conocidos de la teoría de la computabilidad.

Más tarde introducimos el concepto de reducción, en particular las reducciones muchos a uno y las reducciones de Turing. Esto nos permite explorar la máquina de Turing con oráculo, la generalización de nuestro trabajo previo a la *computabilidad relativa* y los grados de Turing. Para culminar, todos los aspectos de la teoría de la computabilidad desarrollados hasta el momento nos permiten enunciar el teorema de Post, que relaciona de manera profunda los conceptos estudiados en este capítulo con la jerarquía aritmética. En particular, sin ser muy precisos, descubrimos que la jerarquía aritmética es a los grados de Turing lo que  $RCA_0$  a la computabilidad (o los objetos computables). En otras palabras,  $\Delta_1^0$  es lo mismo que computable y  $\Sigma_1^0$  es equivalente a la enumerabilidad recursiva.

Tras esto, presentamos de nuevo el sistema  $RCA_0$  y algunas de las construcciones en las que estaremos más interesados. Específicamente, definimos en  $RCA_0$  las sucesiones de racionales y de números reales y también las funciones continuas. También en este capítulo se introducen dos subsistemas de la aritmética de segundo orden adicionales:  $WKL_0$  y  $ACA_0$ .

Para  $WKL_0$  definimos en  $RCA_0$ , como de costumbre, los conjuntos finitos, sucesiones finitas, árboles binarios y caminos infinitos en árboles. A continuación presentamos la versión débil del lema de König y definimos el sistema  $WKL_0$  consistente en los axiomas  $RCA_0$  junto con la versión débil del lema de König. Finalmente, las fórmulas aritméticas y el esquema de axioma de inducción de segundo orden habían sido ya introducidos en el primer capítulo, por lo que el sistema  $ACA_0$  con los axiomas de  $RCA_0$ , el esquema de axioma de inducción de segundo orden y la comprensión aritmética es definido sin dificultad. En este capítulo también exploramos otros aspectos de dichos subsistemas como la estructura de sus  $\omega$ -modelos en términos de teoría de la computabilidad así como sus partes de primer orden, i.e., qué fórmulas de primer orden son capaces de demostrar.

Por último, el capítulo final logra el objetivo que nos habíamos fijado al inicio del trabajo y pone en práctica toda la teoría desarrollada para mostrar

algunos ejemplos concretos de matemática inversa. En particular, demostramos que  $WKL_0$  y  $ACA_0$  son cada uno equivalentes a gran cantidad de resultados clásicos del análisis en  $RCA_0$ , como por ejemplo el teorema de Heine-Borel o el de Bolzano-Weierstraß. Así, demostramos cómo la matemática inversa puede ser usada como una herramienta efectiva para determinar la fuerza de distintos teoremas cuando se comparan. Por ejemplo, el hecho de que el teorema de la convergencia monótona sea equivalente a  $ACA_0$  y  $ACA_0$  más fuerte que  $WKL_0$  significa que es *más fuerte* que la compacidad del intervalo  $[0, 1]$ , también conocido como el teorema de Heine-Borel.

Un capítulo adicional para conclusiones y trabajo futuro identifica las principales ideas y resultados desarrollados en los capítulos anteriores y muestra cómo existe una plétora de temas relacionados que no se estudian en este documento.

# Reverse mathematics: axioms and theorems and back again

Ignacio Mas Mesa

**Keywords:** reverse mathematics, axioms, computability theory, arithmetical hierarchy, logical equivalence.

## Overview

The field of reverse mathematics as we know it today began in 1975 with Harvey Friedman’s seminal paper “Some Systems of Second Order Arithmetic and Their Use” [1]. In the following years it became more of a program with a lot of research effort directed towards the development, improvement and settlement of its defining foundations and characteristic, as well as further study of its consequences and more interesting results. In 2009, Stephen G. Simpson published his encyclopaedic treatise on the current state of the field, “Subsystems of Second Order Arithmetic” [2], which has been extensively referenced and consulted for the creation of the present document. A friendlier introduction to some of the most relevant topics covered in Simpson’s book can be found in John Stillwell’s “Reverse Mathematics: Proofs from the Inside Out” [8], which has also served as a source of inspiration for the more didactical parts of this work.

As it stands, reverse mathematics is concerned with the relation between axioms and theorems. For most of the history of mathematics, axioms have been regarded as a foundational tool from which to build the rest of the mathematical results in a deductive process with which mathematicians are nowadays completely familiar. Even though there exist some historically important examples of work in the study of axioms by themselves, most notably Euclid’s fifth postulate and the axiom of choice or the continuum hypothesis for set theory, the current field of reverse mathematics was very much unexplored until Friedman’s introduction of the tools and ideas required to accomplish its goals. In its most basic form, the key idea of reverse mathematics is that for certain axioms and theorems, their relation is not only of implication but also of equivalence. That is, in Harvey Friedman’s own words: “When the theorem is proved from the right axioms, the axioms can be proved from the theorem”. Of course, this idea needs much clarification and formalization in order to be useful for mathematicians, and this is precisely what the bulk of our effort will be in the following chapters.

The introduction chapter lays out the basic ideas and concepts with which we will work in order to reach our ultimate goal. The most important prin-

ciple outlined here is the formalization of what is to be understood when we say some axiom is equivalent to some theorem. For that, we will need a base axiomatic system in which to carry out the proof of equivalence. The classic approach would be to assume the base system **and** the target axiom to show that the theorem holds. Reverse mathematics gets its name from the other part of the proof: assuming the base system and the theorem and proving the axiom. It is important to note here that the base system must be powerful enough to be able to express the axiom and the theorem (e.g., if we were to prove the Weierstraß theorem we first need to define what a continuous function is in our base system), but also weak enough so that neither the target axiom nor the theorem follow from it.

Having presented the fundamental ideas and definitions, we proceed to some more advanced and detailed concepts. First and foremost we make some necessary definitions to ultimately arrive at the concept of the arithmetical hierarchy, which will be capital for our upcoming work, by which we can give a sense of *complexity* to some second order formulas of our language, i.e., how much *complex* is a certain formula. We also define the system of second order arithmetic, and the subsystem we will be most interested in, namely  $\text{RCA}_0$ . Finally, we show how the pairing function is enough to *embed* the integer and rational numbers inside the natural numbers, and present a syntetic but useful definition of real numbers within  $\text{RCA}_0$ .

The following chapter is concerned with computability theory, which may look like a totally unrelated topic at first. We first present the classical approach of recursion theory, in which computability is inductively defined for a reduced class of initial functions and certain allowed *combinations* of them. We also introduce the more well-known Turing machines approach and outline a sketch of a proof for their equivalence (in that they prove the exact same class of functions are *computable*). We end this section by stating the Church-Turing thesis, which allows us to proceed more loosely in the rest of the chapter, but with confidence that our results can be backed up by actually rigorous arguments should we need them.

We then explore the concept of recursive enumerability and present some examples and properties, only to discover that we can actually enumerate Turing machines (or partial recursive functions, for that matter) and even show that there exists a **universal Turing machine**. This way we find a very natural example of a recursively enumerable set that is not computable. In fact, this concrete example is very intimately related to the halting problem, which serves us to tie our theoretical developments with some of the most renowned aspects of computability theory.

Later, we introduce the concept of reduction, particularized to the cases



of many-one reductions and Turing reductions. This allows us to explore oracle Turing machines, the generalization of our previous work in the form of **relativized computability** and Turing degrees. In the end, all of the theoretical aspects of the computability theory developed up until this point allow us to state Post's theorem, which deeply relates the concepts studied in this chapter with the arithmetical hierarchy. In particular, roughly speaking, it turns out that arithmetical hierarchy is to Turing degrees as  $\text{RCA}_0$  is to computability (or computable objects). In other words,  $\Delta_1^0$  is the same as computable and  $\Sigma_1^0$  is equivalent to recursive enumerability.

After that we present again the  $\text{RCA}_0$  system and some of the constructions we will be most interested in. Specifically, we define in  $\text{RCA}_0$  sequences of rational numbers, sequences of real numbers and continuous functions. In this chapter two more subsystems of second order arithmetic are also introduced:  $\text{WKL}_0$  and  $\text{ACA}_0$ .

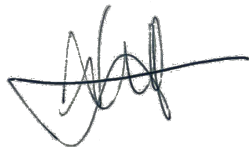
For  $\text{WKL}_0$ , we define in  $\text{RCA}_0$ , as usual, what are finite sets, finite sequences, binary trees and infinite paths in trees. Then we present weak König's lemma, and define the system  $\text{WKL}_0$  comprising the axioms of  $\text{RCA}_0$  plus weak König's lemma. Finally, arithmetical formulas and the full second order axiom scheme of induction were already introduced in the first chapter so the system of  $\text{ACA}_0$  consisting in the axioms of  $\text{RCA}_0$  together with second order induction and arithmetical comprehension is very easily defined. In this chapter we also explore some other aspects of these subsystems such as the structure of their  $\omega$ -models in terms of computability theory and their first order parts, i.e., what first order formulas can they prove.

Lastly, the final chapter accomplishes the goal we set for ourselves at the beginning of the work and successfully uses all the theory developed to showcase some concrete examples of reverse mathematics. In particular, we prove that  $\text{WKL}_0$  and  $\text{ACA}_0$  are each equivalent to a number of classical results in analysis over  $\text{RCA}_0$ , such as the Heine-Borel or the Bolzano-Weierstraß theorems. In doing so, we show how reverse mathematics can be used as an effective tool for determining the relative strength of theorems when compared. For example, the fact that the monotone convergence theorem is equivalent to  $\text{ACA}_0$  and  $\text{ACA}_0$  stronger than  $\text{WKL}_0$  means that it is *stronger* than the compactness of the interval  $[0, 1]$ , otherwise referred to as the Heine-Borel theorem.

An additional chapter for conclusions and further work pinpoints the key ideas and results developed in the preceding chapters and shows how there are plenty of related topics not covered in this document.

---

Yo, **Ignacio Mas Mesa**, alumno del Doble Grado en Ingeniería Informática y Matemáticas de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 71967506T, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



Fdo: Ignacio Mas Mesa

Granada, June 17, 2019

---

D. **Jesús García Miranda**, profesor del Departamento de Álgebra de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado *Reverse mathematics: axioms and theorems and back again*, ha sido realizado bajo su supervisión por **Ignacio Mas Mesa**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a June 17, 2019

**El director:**

**Jesús García Miranda**



# Agradecimientos

---

En primer lugar, quiero agradecerle a toda mi familia el haberme apoyado siempre en todas mis andaduras, incluso cuando la experiencia les decía que mis elecciones podían no ser las óptimas. Gracias en especial a mi madre, mi padre y mis abuelos por haberme proporcionado una educación que me permitirá desarrollarme con mayor facilidad en la etapa de la vida que ahora inicio y por haberme inculcado los ideales de esfuerzo, dedicación y búsqueda del conocimiento. Cualquier desviación por mi parte de estos es sólo atribuible a mis propios errores.

Por otro lado, debo agradecer a la que ha sido mi familia durante estos seis años, tanto a los que han estado desde el principio (Croqueta, S, Fua, Óxido) como a los que llegaron más tarde (Petaca, Linguini, Geyper, Spock, Tallo). A Caracas, por haberme enseñado que por lo general los absolutos no tienen cabida y por ser una fuente constante de inspiración. A Mojo, porque sin él probablemente nunca habría llegado a Granada y mi vida sería completamente distinta; y por ser una persona noble como ninguna. A Gavilán, por haber compartido algunos de los mejores años de nuestras vidas juntos y por seguir siendo un enigma para mí. A Fidel, por las innumerables cachimbos, las veces que ha jugado Blitzcrank en vez de Zed, las horas dedicadas al ajedrez, las matemáticas, la física o la programación; e incluso por las correcciones a este trabajo. Y a Lucía, por enseñarme que el amor y la amistad deben ir de la mano y que “aufabaguera” significa albahaca.

Gracias en general a todas las personas que de un modo u otro han pasado por mi vida y han influido en mi modo de ver el mundo: profesores, conocidos, camareros, panaderos, bibliotecarios, músicos, cineastas, matemáticos o lingüistas. Gracias en particular a toda la gente buena de Granada que ha ayudado a que mi estancia en la ciudad fuera un poco mejor. Este trabajo es también, en parte, suyo.

Agradecer también a la comunidad del software libre, que tanto aporta a la sociedad y que tan frecuentemente es olvidada.

Por último, pero no por ello menos importante, gracias a Andrew L., cuyo anónimo comentario en Facebook me descubrió el mundo de la matemática inversa que, de otro modo, habría permanecido ignoto para mí.

# Contents

---

<b>Contents</b>	<b>12</b>
<b>1 Introduction</b>	<b>15</b>
<b>2 Arithmetic and arithmetization</b>	<b>17</b>
2.1 Arithmetical definability and arithmetical hierarchy . . . . .	20
2.2 The axiom system $\text{RCA}_0$ . . . . .	22
<b>3 Computability</b>	<b>31</b>
3.1 Models of computation and the Church-Turing thesis . . . . .	31
3.1.1 Recursive functions . . . . .	32
3.1.2 Turing machines . . . . .	36
3.1.3 The Church-Turing thesis . . . . .	39
3.2 The limits of computation . . . . .	39
3.2.1 Recursive enumerability . . . . .	39
3.2.2 Enumerating Turing machines . . . . .	42
3.3 Reducibility . . . . .	45
3.3.1 Many-one reducibility . . . . .	46
3.4 Turing reducibility, oracle machines and the Turing universe .	49
3.5 Post's theorem . . . . .	56
<b>4 The formal systems</b>	<b>59</b>
4.1 $\text{RCA}_0$ . . . . .	59
4.1.1 The formal system . . . . .	59
4.1.2 Sequences, functions and continuity . . . . .	59
4.1.3 Characterization of $\omega$ -models . . . . .	61
4.1.4 First order part . . . . .	63
4.1.5 Theorems of $\text{RCA}_0$ . . . . .	63
4.2 $\text{WKL}_0$ . . . . .	64
4.2.1 The formal system . . . . .	64
4.2.2 $\omega$ -models . . . . .	66

4.2.3	First order part . . . . .	68
4.2.4	Theorems of $WKL_0$ . . . . .	69
4.3	$ACA_0$ . . . . .	70
4.3.1	The formal system . . . . .	70
4.3.2	Characterization of $\omega$ -models . . . . .	70
4.3.3	First order part . . . . .	71
4.3.4	Theorems of $ACA_0$ . . . . .	71
<b>5</b>	<b>Some equivalence results over <math>RCA_0</math></b>	<b>73</b>
5.1	$RCA_0$ . . . . .	73
5.1.1	The intermediate value theorem . . . . .	73
5.2	$WKL_0$ . . . . .	76
5.2.1	Sequential Heine-Borel . . . . .	76
5.3	$ACA_0$ . . . . .	79
5.3.1	Consequences of arithmetical comprehension . . . . .	79
5.3.2	Bolzano-Weierstraß proves Cauchy convergence criterion	82
5.3.3	Cauchy convergence criterion proves monotone con- vergence . . . . .	83
5.3.4	Monotone convergence proves arithmetical comprehen- sion . . . . .	83
5.3.5	Sequential least upper bound implies monotone con- vergence . . . . .	85
5.3.6	Theorems equivalent to $ACA_0$ . . . . .	86
<b>6</b>	<b>Conclusions and further work</b>	<b>87</b>
	<b>Bibliography</b>	<b>89</b>





# 1 Introduction

---

*When the theorem is proved from the right axioms, the axioms can be proved from the theorem.*

HARVEY FRIEDMAN

In the beginning, there was  $\omega$ . Or, at least, that is what we will assume for the remainder of this text.

For any attempt at tackling the problem of the foundations of mathematics, a decision must be made as to which undeniable primitives shall the rest of the mathematics be built upon. In our case, we will call  $\omega$  the standard natural numbers with their usual elements, operations and, more generally, properties. Note, however, that this will be an object of our metatheory and, as such, we will try our best to define it inside the logical framework we will work with.

Keeping that in mind, the purpose of this work is to give a gentle introduction to the basic results in the reverse mathematics field/program, whose origins date back to 1975, when Harvey Friedman published his seminal paper “Some Systems of Second Order Arithmetic and Their Use” [1].

As pointed by the quote at the beginning of the section (which in fact appeared in the aforementioned paper), reverse mathematics deals with the relationship between axioms and theorems. It turns out that, more often than not, the theorems not only follow from the axioms but also imply them, so they are, in fact, equivalent (for suitable theorems and axioms, that is).

Alas, we need to tread carefully and be specific about what we mean by *equivalence*. From a logical, proof-theoretic standpoint any two true statements are logically equivalent because they are both true. We will need to develop a base system and then show that, from its point of view, some extension of it is equivalent to some other theorem. Our base system should be powerful enough to be capable of expressing interesting concepts (e.g.,  $f$  is

a continuous function), but also weak enough that the theorems we wish to consider do not already follow from it. Thus, the general procedure for an equivalence claim will be as follows:

1. Show that our base system  $B$ , when augmented to the *target* system  $S$ , can prove the theorem  $T$ . This would be the classical direction, in which theorems follow from axioms.
2. Then, arguing from  $B$ , prove that the theorem  $T$  actually implies the extra axiom(s) in  $S$ . This is the *novelty*, where we show that the proper axiom is indeed needed by  $T$ .

The base system that will be used to show the equivalences is called *recursive comprehension axiom*, or  $\text{RCA}_0$ , for reasons that will become clear later. This will turn out to be, indeed, a very weak system, capable only of proving basic results such as the intermediate value theorem or the Baire category theorem. This is not to say that those theorems are unimportant or trivial, of course, only that they do not require as many or as strong conditions as some others to be proven. However, its expressive power is enough to be able to deal with countable rings and fields, separable metric spaces, functional analysis and even logic itself.

Later on, we will present the systems of  $\text{WKL}_0$ , for weak König's lemma, and  $\text{ACA}_0$ , for *arithmetical comprehension axiom*. These three systems, and others that we will only briefly mention but not actually work with, are essentially equal except for a set (in a sense we will soon explain) existence axiom. In fact, they are actually of strictly increasing strength, which roughly means that if we succeed in showing the equivalence between a theorem and one of these systems, we can get a sense of how strong a theorem is when comparing it to some other, already classified theorems. To be more specific, we will see that monotone convergence is not provable in  $\text{RCA}_0$  but it is in  $\text{ACA}_0$ . So, in that sense, monotone convergence is *stronger* than the intermediate value theorem.

# Arithmetic and arithmetization

## 2

*Mathematics is a game played according to certain simple rules with meaningless marks on paper.*

DAVID HILBERT

Our first step will be to define the basic environment of our forthcoming efforts, namely, the formal system of **second order arithmetic** or  $Z_2$ . A quick word of caution, though: from now on we are going to be discussing *sets* (of natural numbers). Note, however, that, albeit somewhat confusing, this terminology does not bear any relation to set theory, it will be just a commodity. Indeed, sets can be successfully emulated in our logical framework using predicates, i.e., whenever we write  $x \in S$  we actually mean that  $x$  satisfies the predicate  $S$ , or, more succinctly,  $S(x)$  holds. This is to say that there is no need for any underlying set theory in our metatheory (in fact, it would probably be harmful, as we will see later). The reason for this naming convention is to provide a common language that strongly suggests the relation between what can be described using logic and computability theory.

We will need some prior definitions to accomplish the goal of this section. The language of second-order arithmetic,  $L_2$ , consists of:

- an infinite number of symbols for numeric variables, denoted by lower case latin letters  $i, j, k, \dots$  and intended to range over  $\omega = \{0, 1, 2, \dots\}$
- an infinite number of symbols for set variables, denoted by upper case latin letters:  $X, Y, Z, \dots$  and intended to range over subsets of  $\omega$ ,
- the constant symbols 0 and 1,
- the symbols  $+$  and  $\cdot$  for addition and multiplication, respectively,
- the symbols  $=$  and  $<$  for equality and “less than”,
- the symbol  $\in$  for membership, and
- the usual symbols for second order logic, viz.:  $\wedge, \vee, \neg, \implies, \iff, \forall, \exists$

Based on that, we define:

**Numerical terms** the constants 0 and 1, and  $t_1 + t_2$  and  $t_1 \cdot t_2$  whenever  $t_1$  and  $t_2$  are both numerical terms themselves

**Atomic formulas**  $t_1 = t_2$  and  $t_1 < t_2$  whenever  $t_1$  and  $t_2$  are both numerical terms, and  $t \in X$  when  $t$  is a numerical term and  $X$  a set variable

**Formulas** built from atomic formulas using propositional connectives ( $\wedge, \vee, \neg, \implies, \iff$ ) and quantifiers ( $\exists n(\varphi)$  and  $\forall n(\psi)$ )

We will also allow the introduction of parentheses at any given point if they respect the syntactical rules (e.g.,  $((1 + 1) = t)$  is a valid expression, while  $1 + 1)$  or  $($  are not). Finally, a variable not bound by any previous quantifiers will be called a **free variable**, and a formula with no free variables will be called a **sentence**.

We shall also use some common abbreviations in order to alleviate the verbosity of the language. For example, 2 should be understood to represent  $1 + 1$ , 3 is just  $1 + 1 + 1$ , and so on.  $t \notin X$  will mean  $\neg(t \in X)$ . Operators such as  $+$ ,  $\cdot$ ,  $\wedge$ , etc. are left associative, i.e.,  $\varphi \wedge \psi \wedge \theta$  stands for  $(\varphi \wedge \psi) \wedge \theta$ .

Next, we need to discuss the concept of a **model** or **structure**, of our language. A model for  $L_2$  is a septuple  $M = (|M|, S_M, +_M, \cdot_M, 0_M, 1_M, <_M)$ , where  $|M|$  is a *set* (in a metatheoretical sense, so we cannot speak about its nature from within our theory) which serves as the range of the number variables,  $S_M$  is a set of subsets of  $|M|$  serving as the range of the set variables,  $+_M$  and  $\cdot_M$  represent the operations of addition and product on  $|M|$ ,  $0_M$  and  $1_M$  are distinguished elements of  $|M|$ , and  $<_M$  is a binary relation on  $|M|$ . Thus, the interpretation of an  $L_2$  formula in  $M$  is straight-forward.

The intended model for  $L_2$  will be no other than

$$(\omega, \mathcal{P}(\omega), +_\omega, \cdot_\omega, 0_\omega, 1_\omega, <_\omega)$$

This is called the **full** or **standard** semantics. We can, nonetheless, also consider *Henkin* semantics, where we allow  $S_M$  to be an arbitrary, non-empty subset of  $\mathcal{P}(\omega)$ . These semantics give rise to different, simpler (coarser) models that help us in our purpose of determining different axiomatic systems.

The formal system of **second order arithmetic**,  $Z_2$ , consists of (the universal closure of) the following axioms – which are  $L_2$  formulas:

(i) basic axioms

- $n + 1 \neq 0$
- $m + 1 = n + 1 \implies m = n$
- $m + 0 = m$
- $m + (n + 1) = (m + n) + 1$
- $m \cdot 0 = 0$
- $m \cdot (n + 1) = (m \cdot n) + m$
- $\neg(m < 0)$
- $m < n + 1 \iff (m < n \vee m = n)$

(ii) induction axiom

$$(0 \in X \wedge \forall n(n \in X \implies (n + 1) \in X)) \implies \forall n(n \in X)$$

(iii) comprehension scheme

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

where  $\varphi(n)$  is any  $L_2$ -formula in which  $X$  does not occur freely.

Note that in the comprehension scheme  $\varphi(n)$  may contain free variables other than  $n$ , and they will be called **parameters**. For example, if we consider  $\varphi(n) = n \notin Y$ , we can form the axiom

$$\forall Y \exists X \forall n(n \in X \iff n \notin Y)$$

which asserts the existence of the complement of a set.

Finally, we define the formal system of second order arithmetic,  $Z_2$ , to be the formal system in  $L_2$  consisting of the axioms of second order arithmetic and all formulas of  $L_2$  deducible from the arithmetic axioms by the usual logical axioms and rules of inference.

Now that we have established the limits of second order arithmetic, we turn our attention to *subsystems* of it, that is, formal systems in the language of  $L_2$  whose axioms are theorems of  $Z_2$ . Those in which we are interested, namely  $\text{RCA}_0$ ,  $\text{WKL}_0$  and  $\text{ACA}_0$ , consist of the basic axioms, a stronger version of the induction axiom (concretely  $\Sigma_1^0$ -induction), and a weaker comprehension scheme. Note that, even though they have a stronger version of the axiom of induction, they are effectively subsystems because that induction axiom is a consequence of the  $Z_2$  induction axiom and comprehension scheme. We will explore this subject in more detail in later sections.

## 2.1 Arithmetical definability and arithmetical hierarchy

**Definition 2.1.** A formula  $\varphi$  is said to be  $\Sigma_1^0$  if it is equivalent to

$$\exists x_1(\exists x_2(\dots(\exists x_n\psi(x_1, x_2, \dots, x_n))))$$

where  $\psi$  is quantifier-free, for appropriate  $n$  and  $\psi$ .

**Definition 2.2.** A formula  $\varphi$  is said to be  $\Pi_1^0$  if it is equivalent to

$$\forall x_1(\forall x_2(\dots(\forall x_n\psi(x_1, x_2, \dots, x_n))))$$

where  $\psi$  is quantifier-free, for appropriate  $n$  and  $\psi$ .

**Definition 2.3.** A formula  $\varphi$  is said to be  $\Sigma_{n+1}^0$  if it is equivalent to

$$\exists x_1(\exists x_2(\dots(\exists x_m\psi(x_1, x_2, \dots, x_m))))$$

where  $\psi$  is  $\Pi_n^0$  itself, for appropriate  $m, n$  and  $\psi$ .

**Definition 2.4.** A formula  $\varphi$  is said to be  $\Pi_{n+1}^0$  if it is equivalent to

$$\forall x_1(\forall x_2(\dots(\forall x_m\psi(x_1, x_2, \dots, x_m))))$$

where  $\psi$  is  $\Sigma_n^0$ , for appropriate  $m, n$  and  $\psi$ .

In these definitions,  $x_1, x_2, \dots, x_n$  are numerical variables and not set variables. In intuitive terms, a  $\Sigma_n^0$  formula is one equivalent to a formula that begins with some existential quantifiers and alternates between existential and universal quantifiers  $n - 1$  times.  $\Pi_n^0$  formulas are analogous but beginning with a universal quantifier. Now we make a definition that will be useful in later stages:

**Definition 2.5.** A formula  $\varphi$  is said to be  $\Delta_n^0$  if it is both  $\Sigma_n^0$  and  $\Pi_n^0$ .

A special case of quantifiers is that of **bounded quantifiers**, viz.:  $\exists(n < m), \forall(n < m)$ , where  $n$  is a number variable and  $m$  a numerical term that does not contain  $n$ . These quantifiers do not change the language since they can be represented in terms of *normal*, henceforth **unbounded**, quantifiers. For example,  $\exists(n < m)(\varphi(n)) \iff \exists n(n < m \wedge \varphi(n))$  shows the equivalence for the case of the existential quantifier. In a similar vein, we define a formula  $\varphi$  to be a *bounded quantifier formula* when all of its quantifiers are bounded. A bounded quantifier formula is a  $\Sigma_0^0$  formula. Since  $\Sigma_0^0 = \Pi_0^0$ , a bounded quantifier formula is also a  $\Pi_0^0$  and thus a  $\Delta_0^0$ . Intuitively, having to check for

a finite number of cases does not increase the complexity of a formula, so we assign it the least possible classification.

Finally, we define **arithmetical** formulas:

**Definition 2.6.** A formula  $\varphi$  of  $L_2$  is said to be arithmetical if it contains no set quantifiers.

Examples of arithmetical formulas include the basic axioms of  $Z_2$ . An example of a non-arithmetical formula is the comprehension scheme of  $Z_2$ . Note that, even though they cannot contain set quantifiers, they **can** contain set variables. For example,  $\varphi(n) := n \in Y \wedge \exists m(n = 2 \cdot m)$  identifies the even numbers of a (unknown, but otherwise fixed) set  $Y$ .

**Definition 2.7.** A formula  $\varphi$  is said to be in **prenex form** if all of its quantifiers appear at the front

**Theorem 2.1.** Every formula of  $L_2$  is (logically) equivalent to a (possibly syntactically different)  $L_2$  formula in prenex form

*Proof.* First note the following equivalences:

$$\begin{aligned} (\varphi \iff \psi) &\iff (\varphi \implies \psi) \wedge (\psi \implies \varphi) \\ (\varphi \implies \psi) &\iff (\neg\varphi) \vee \psi \\ (\varphi \wedge \psi) &\iff \neg(\neg\varphi \vee \neg\psi) \end{aligned}$$

In other words,  $\{\neg, \vee\}$  is a **functionally complete** set of Boolean connectives. Now notice that we can *push inwards* these connectives when in presence of quantifiers via these equivalences:

$$\begin{aligned} \neg(\exists x(\varphi)) &\iff \forall x(\neg\varphi) \\ \neg(\forall x(\varphi)) &\iff \exists x(\neg\varphi) \\ \varphi \vee \exists x(\psi(x)) &\iff \exists y(\varphi \vee \psi(y)) \\ \varphi \vee \forall x(\psi(x)) &\iff \forall y(\varphi \vee \psi(y)) \end{aligned}$$

Where, in the two last instances, we can replace the bounded variable  $x$  if needed by a variable  $y$  not occurring in  $\varphi$ . By repeatedly applying these rules to a formula  $\varphi$ , we achieve the following equivalence:

$$\varphi \iff Q_1 x_1 (Q_2 x_2 (\dots (Q_n x_n (\psi(x_1, x_2, \dots, x_n))))))$$

Where  $Q_1, Q_2, \dots, Q_n$  are quantifier symbols (either  $\exists$  or  $\forall$ ) and  $\psi$  is quantifier-free.  $\square$

Thanks to the prenex form, we can be sure that every arithmetical formula is assigned a classification in terms of its membership to  $\Sigma_n^0$  or  $\Pi_n^0$ , for suitable  $n$ . Note that, since the addition of *unused* quantifiers doesn't affect the equivalence, once a formula is either  $\Sigma_n^0$  or  $\Pi_n^0$  then it is also  $\Sigma_m^0$  **and**  $\Pi_m^0$  for every  $m > n$ . Hence, we will only be concerned with the minimum  $n$ , since every other classification follows immediately.

Of course, we can extend this notion to sets:

**Definition 2.8.** A set  $X$  is said to be **arithmetically definable**, or arithmetical, if there is an arithmetical formula  $\varphi$  such that  $\forall n(n \in X \iff \varphi(n))$ . In this case, we also say that  $X$  is defined by  $\varphi$ .

**Definition 2.9.** A set  $X$  is said to be  $\Sigma_n^0$  (respectively  $\Pi_n^0$ ) if it is defined by a  $\Sigma_n^0$  (respectively  $\Pi_n^0$ ) formula.

The **arithmetical hierarchy** is the classification of arithmetically definable sets with respect to the *complexity* of the (simplest) formula that describes them. In the next chapter we will explore the astounding relationship between this concept and computability theory.

## 2.2 The axiom system $\text{RCA}_0$

Before proceeding any further, we need to establish the ground upon which we will base all of our work, namely, the axiom system  $\text{RCA}_0$ , the base system to which we alluded in the introduction.

**Definition 2.10.** The formal system  $\text{RCA}_0$  in the language  $L_2$  consists of the following axioms:

1. the basic axioms of  $Z_2$ ,
2.  $\Sigma_1^0$ -induction, i.e.,  $(\varphi(0) \wedge \forall n(\varphi(n) \implies \varphi(n+1))) \implies \forall n\varphi(n)$  where  $\varphi$  is  $\Sigma_1^0$ , and
3. recursive, or  $\Delta_1^0$ , comprehension, i.e.,  $\exists X \forall n(n \in X \iff \varphi(n))$  where  $\varphi$  is  $\Delta_1^0$  and  $X$  does not occur freely in it



Recursive comprehension is also usually defined as

$$\forall n(\varphi(n) \iff \psi(n)) \implies \exists X \forall n(n \in X \iff \varphi(n))$$

where  $\varphi$  is  $\Sigma_1^0$ ,  $\psi$  is  $\Pi_1^0$  and  $X$  does not occur freely in  $\varphi$ .

Based on this, we define the *inner* natural numbers,  $\mathbb{N}$ , as the (unique) set such that  $\forall n(n \in \mathbb{N} \iff n = n)$ , which obviously implies  $\forall n(n \in \mathbb{N})$ . Now we state an important fact about this *newly* found system that will be needed for the upcoming definitions:

**Theorem 2.2.** *We can prove in  $\text{RCA}_0$  that  $(\mathbb{N}, +, \cdot, 0, 1, <)$  is a commutative ordered semiring with cancellation*

*Proof.* The complete proof relies on proving 25 statements that verify the claim made in the statement of the theorem, such as  $m \cdot (n + p) = m \cdot n + m \cdot p$  (distributive property of the product over addition) or  $m + p = n + p \implies m = n$  (cancellation law). The proofs of these statements are a rather mechanical and uninteresting exercise of induction. For more details we refer to [2].  $\square$

The fundamentals about natural numbers have been mostly already established with these definition and the last theorem. Even though the study of the properties of natural numbers can be interesting, we want to show the possibilities of  $\text{RCA}_0$  to work in different areas of mathematics by *embedding* them into the natural numbers. Most notably, we will be concerned with the classical theorems of analysis.

Before getting to the real numbers, however, we must first find how to work with the integers, for example, in  $\text{RCA}_0$ . The most common and simple algebraic way to construct the integers from the natural numbers is to consider ordered pairs of naturals to represent subtraction, and then define an equivalence relationship and take the quotient. Unfortunately, the lack of set theory in our environment prevents us from using  $\mathbb{N} \times \mathbb{N}$  since we have not defined the cartesian product, so we must come up with a different approach to formalize it. In order to do this, we make the following definition:

**Definition 2.11.** The **pairing function**, or map, is defined by  $(i, j) := (i + j)^2 + i$

**Lemma 2.1.**  $\forall n(n^2 \geq n)$

*Proof.* We proceed by induction. Define  $\varphi(n) := n^2 \geq n$ .

- For  $n = 0$ , it is obvious that  $0 \cdot 0 = 0 \geq 0$ , so  $\varphi(0)$  holds.

- For  $n = 1$ , we can see that  $1 \cdot 1 = 1 \geq 1$ , so  $\varphi(1)$  holds and thus  $\varphi(0) \implies \varphi(1)$ .
- Now assume  $n^2 \geq n$  for  $n \geq 1$ . Then, by lemma 2.1,  $(n+1)^2 > n(n+1) > n^2 \geq n$ . Therefore  $\varphi(n) \implies \varphi(n+1)$ .

Finally, applying  $\Sigma_1^0$ -induction we obtain the desired result  $\forall n(n^2 \geq n)$ .  $\square$

**Corollary 2.1.**  $\forall n(n^2 \geq 0)$

There are two important properties of the pairing function that we can prove in  $\text{RCA}_0$ :

**Theorem 2.3.**

$$i \leq (i, j) \wedge j \leq (i, j)$$

$$(i, j) = (i', j') \implies (i = i' \wedge j = j')$$

*Proof.* The first part of the theorem is a direct application of lemma 2.1 and corollary 2.1. For the second part, we reproduce the proof in [2]. If  $k = (i, j)$ , then there exists a unique  $m : m^2 \leq k < (m+1)^2$ . To show that  $m$  exists we consider  $m = (i+j)$ . Indeed,  $((i+j)+1)^2 = (i+j)^2 + 2(i+j) + 1 > (i+j)^2 + i = k$  and obviously  $(i+j)^2 \leq (i+j)^2 + i$ . For uniqueness, assume  $m' \neq m$  verifies the conditions too.

- If  $m < m'$  then  $m' = m+p+1$  for some  $p$ . Therefore  $k \geq m'^2 = (m+p+1)^2 \geq (m+1)^2 > k$ .
- Similarly, if  $m' < m$  then  $m = m'+p'+1$ , so  $k < (m'+1)^2 \leq (m'+p'+1)^2 = m^2 \leq k$ .

By using the monotonicity of  $n^2$  we can derive a contradiction in either case. We can then conclude  $m = m'$  and it is trivial to check that

$$(i, j) = i + m^2 = k = i' + m^2 = (i', j')$$

since we can reuse  $m$  because  $(i, j) = (i', j')$ . It follows that  $i = i' \wedge j = j'$ , thereby completing the proof.  $\square$

The first one is rather technical but nonetheless important and we will need it shortly. The second one simply states, intuitively, that  $(\cdot, \cdot)$  is an injection from  $\mathbb{N} \times \mathbb{N}$  to  $\mathbb{N}$ . Remarkably, this is what allows us to formalize  $\mathbb{N} \times \mathbb{N}$  via recursive comprehension as the (unique) set  $X$  such that

$$n \in X \iff \exists(i \leq n) \exists(j \leq n)((i, j) = n)$$

In fact, this same trick can be used to define  $A \times B$  for any sets  $A, B$  in  $\text{RCA}_0$ .

The fact that the pairing function is injective also means that if we consider the number  $(i, j)$  we can outright use  $i$  and  $j$  for whatever purpose and do not need to worry about well-definedness, i.e., other possible representatives of the number yielding different results. With that in mind, we define the following operations and relations on  $\mathbb{N} \times \mathbb{N}$ :

- $(m, n) +_{\mathbb{Z}} (p, q) := (m + p, n + q)$
- $(m, n) -_{\mathbb{Z}} (p, q) := (m + q, n + p)$
- $(m, n) \cdot_{\mathbb{Z}} (p, q) := (m \cdot p + n \cdot q, m \cdot q + n \cdot p)$
- $(m, n) <_{\mathbb{Z}} (p, q) \iff m + q < n + p$
- $(m, n) =_{\mathbb{Z}} (p, q) \iff m + q = n + p$

It is easily checked that  $<_{\mathbb{Z}}$  is an order relation and  $=_{\mathbb{Z}}$  an equivalence relation on  $\mathbb{N} \times \mathbb{N}$ .

**Definition 2.12.** We say that a natural number  $n$  is an **integer** if it is the least element in its equivalence class induced by  $=_{\mathbb{Z}}$ , that is,  $\forall m (m =_{\mathbb{Z}} n \implies m \geq n)$ .

We define  $\mathbb{Z} \subset \mathbb{N} \times \mathbb{N} \subset \mathbb{N}$  as the set of all integer numbers. We can show in  $\text{RCA}_0$  that  $\mathbb{Z}$  exists, since

$$n \in \mathbb{Z} \iff \exists (i \leq n) \exists (j \leq n) \forall (p \leq n) \forall (q \leq n) (n = (i, j) \wedge (n =_{\mathbb{Z}} (p, q) \implies n < (p, q)))$$

Risking notational ambiguity, we define the previous operations on  $\mathbb{Z}$ , dropping the subscript, in the natural way. For example, for  $a, b \in \mathbb{Z}$  we define  $a + b := c$  where  $c \in \mathbb{Z}$  is the unique integer such that  $a +_{\mathbb{Z}} b =_{\mathbb{Z}} c$ . We are confident that context will be enough to determine the meaning of the symbols in each case.

We can prove the following in  $\text{RCA}_0$ :

**Theorem 2.4.**  $(\mathbb{Z}, +, -, \cdot, 0 = (0, 0), 1 = (1, 0), <)$  is an ordered, commutative, integral domain.

*Proof.* We skip the details since the full proof is quite laborious and does not really add much insight. It will suffice to show, for example, that it is indeed an integral domain, that is,  $(m \neq 0 \wedge n \neq 0) \implies m \cdot n \neq 0$ .

Let  $m = (i, j), n = (p, q) \in \mathbb{Z}$  such that  $m \neq 0, n \neq 0$ . We want to show that  $m \cdot n \neq 0$ . This is equivalent to  $m \cdot_{\mathbb{Z}} n =_{\mathbb{Z}} (ip + jq, iq + pj) \neq_{\mathbb{Z}} 0$ , which, in turn, is equivalent to  $ip + jq \neq iq + pj$ . Assume to the contrary that it is the case that  $ip + jq = iq + pj$ . If either  $i = j$  or  $p = q$  the equality holds trivially, but then either  $m = 0$  or  $n = 0$ , so we reach a contradiction. We may then assume both  $i \neq j \wedge p \neq q$ . By trichotomy, we have  $i \neq j \implies i < j \vee j < i$ , and similarly for  $p$  and  $q$ . We assume  $m < n$  and  $p < q$  and the proof for the other cases is analogous. Then, by theorem 2.2 we have  $\exists k(i + k + 1 = j)$  and  $\exists t(p + t + 1 = q)$ . We rename  $k' := k + 1, t' := t + 1$  and note that  $k', t' > 0$ . Finally, we have:

$$\begin{aligned} jp + jq &= jq + jp \\ (i + k')p + jq &= (i + k')q + jp \\ k'p &= k'q \end{aligned}$$

Where we have used commutativity of addition, distributivity of the product, and the assumption that  $ip + jq = iq + pj$  together with the cancellative law. Again, we have:

$$\begin{aligned} k'q &= k'q \\ k'(p + t') &= k'q \\ k't' &= 0 \end{aligned}$$

But then, noting that  $0 = 0 \cdot t'$  and  $0 < t' \implies t' \neq 0$  we can conclude  $(t' \neq 0 \wedge 0 < k') \implies 0 = 0 \cdot t' < k't'$ . Since, by theorem 2.2,  $\forall n \neg(n < n)$  we have reached a contradiction. Therefore one of our original assumptions must be wrong, and either  $i = j$  or  $p = q$  or, equivalently,  $m = 0$  or  $n = 0$ . This concludes our (sketch of) proof.  $\square$

We can also show  $\mathbb{Z}^+$ , the positive integers, exists as

$$n \in \mathbb{Z}^+ \iff (n \in \mathbb{Z} \wedge 0 <_{\mathbb{Z}} n) \iff \exists(i \leq n) \exists(j \leq n) (n \in \mathbb{Z} \wedge n = (i, j) \wedge (j < i))$$

Hence,  $\mathbb{Z} \times \mathbb{Z}^+$  exists, and we can define the following operations and relations on it:

- $(p, q) +_{\mathbb{Q}} (r, s) := (p \cdot s + q \cdot r, q \cdot s)$
- $(p, q) -_{\mathbb{Q}} (r, s) := (p \cdot s - q \cdot r, q \cdot s)$

- $(p, q) \cdot_{\mathbb{Q}} (r, s) := (p \cdot r, q \cdot s)$
- $(p, q) /_{\mathbb{Q}} (r, s) := (p \cdot s, q \cdot r)$  whenever  $r \in \mathbb{Z}^+$
- $(p, q) <_{\mathbb{Q}} (r, s) \iff p \cdot s < r \cdot q$
- $(p, q) =_{\mathbb{Q}} (r, s) \iff p \cdot s = r \cdot q$

Again,  $<_{\mathbb{Q}}$  and  $=_{\mathbb{Q}}$  can be easily shown to be an order and equivalence relation respectively.

**Definition 2.13.** We say that a natural number  $n \in \mathbb{Z} \times \mathbb{Z}^+$  is a rational number if it is the least element in its equivalence class induced by  $=_{\mathbb{Q}}$ , that is

$$\forall m((m \in \mathbb{Z} \times \mathbb{Z}^+ \wedge m =_{\mathbb{Q}} n) \implies m > n)$$

We define  $\mathbb{Q} \subset \mathbb{Z} \times \mathbb{Z}^+ \subset \mathbb{Z} \times \mathbb{Z} \subset (\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N}) \subset \mathbb{N} \times \mathbb{N} \subset \mathbb{N}$  as the set of all rational numbers. It exists because

$$\begin{aligned} n \in \mathbb{Q} \iff & \exists(i \leq n) \exists(j \leq n) \forall(p \leq n) \forall(q \leq n) ( \\ & i \in \mathbb{Z} \wedge j \in \mathbb{Z}^+ \wedge n = (i, j) \wedge \\ & ((p \in \mathbb{Z} \wedge q \in \mathbb{Z}^+ \wedge n =_{\mathbb{Q}} (p, q)) \implies n < (p, q)) \\ & ) \end{aligned}$$

Similar to what we did with the integers, we drop the subscript and define the operations and relations on  $\mathbb{Q}$ . We can prove the following in  $\text{RCA}_0$ :

**Theorem 2.5.**  $(\mathbb{Q}, +, -, \cdot, /, 0 = (0, 0) = ((0, 0), (0, 0)), 18 = (2, 2) = ((1, 0), (1, 0)), <)$  is an ordered field.

**Definition 2.14.** The **absolute value** of a rational number  $q$ , denoted by  $|q|$ , is defined as follows:

$$|q| := \begin{cases} q & \text{if } 0 \leq q, \\ -q & \text{if } q < 0 \end{cases}$$

**Definition 2.15.** A **sequence** of rational numbers is defined as a function  $f : \mathbb{N} \rightarrow \mathbb{Q}$ . We will usually denote the sequence as  $\{q_i : i \in \mathbb{N}\}$  where  $q_i = f(i)$ .

**Definition 2.16.** A **real number** is defined to be a sequence of rational numbers  $\{q_i : i \in \mathbb{N}\}$  verifying  $\forall i \forall k (|q_i - q_{i+k}| < 2^{-i})$ .

**Definition 2.17.** Two real numbers  $\{q_i : i \in \mathbb{N}\}$  and  $\{r_i : i \in \mathbb{N}\}$  are said to be **equal** if  $\forall i (|q_i - r_i| \leq 2^{1-i})$ .

It is somewhat more difficult but still provable in  $\text{RCA}_0$  that this notion of equality induces an equivalence relation on real numbers. We cannot, however, consider the quotient under this equivalent relation because of the limitations of the language (for more details see [2]). Furthermore, we cannot even construct the set of all such representatives of real numbers in  $\text{RCA}_0$ , since we are restricted to the (monadic) second order arithmetic language,  $L_2$ . This means that we can quantify over **unary relations**, otherwise referred to as **sets**, and not over arbitrary functions or relations. Nonetheless, we may informally use  $\mathbb{R}$  to denote the set of all real numbers, even though it does not formally exist within  $\text{RCA}_0$ . In that case,  $x \in \mathbb{R}$  should be understood as  $x$  is a real number, or  $\forall x \in \mathbb{R}(\dots)$  as  $\forall x(x \text{ is a real number} \implies \dots)$ .

**Definition 2.18.** Let  $x = \{q_i : i \in \mathbb{N}\}, y = \{r_i : i \in \mathbb{N}\}$  be two real numbers. Then  $x + y := \{q_{i+1} + r_{i+1} : i \in \mathbb{N}\}$ .

**Theorem 2.6.** If  $x = \{q_i : i \in \mathbb{N}\}$  and  $y = \{r_i : i \in \mathbb{N}\}$  are real numbers, then so is  $x + y$ .

*Proof.*

$$\begin{aligned} |(q_{i+1} + r_{i+1}) - (q_{i+k+1} + r_{i+k+1})| &= |(q_{i+1} - q_{i+k+1}) + (r_{i+1} - r_{i+k+1})| \\ &\leq |(q_{i+1} - q_{i+k+1})| + |(r_{i+1} - r_{i+k+1})| < 2^{-i-1} + 2^{-i-1} = 2^{-i} \end{aligned}$$

□

**Definition 2.19.** Let  $x = \{q_i : i \in \mathbb{N}\}$  be a real number. Then  $-x := \{-q_i : i \in \mathbb{N}\}$ .

**Theorem 2.7.** If  $x$  is a real number then so is  $-x$ .

*Proof.* It suffices to note that  $|-q_i - (-q_{i+k})| = |q_{i+k} - q_i| = |q_i - q_{i+k}| < 2^{-i}$ . □

**Definition 2.20.** Let  $x, y$  be two real numbers. Then  $x - y := x + (-y)$ .

**Definition 2.21.** Let  $x = \{q_i : i \in \mathbb{N}\}, y = \{r_i : i \in \mathbb{N}\}$  be two real numbers. Then  $x \cdot y := \{q_{k+i} \cdot r_{k+i} : i \in \mathbb{N}\}$ , where  $k$  is the minimum value such that  $2^k \geq |q_0| + |r_0| + 2$ .

**Theorem 2.8.** If  $x = \{q_i : i \in \mathbb{N}\}$  and  $y = \{r_i : i \in \mathbb{N}\}$  are real numbers, then so is  $x \cdot y$ .

*Proof.*

$$\begin{aligned}
& |q_{k+i}r_{k+i} - q_{k+i+j}r_{k+i+j}| \\
&= |(q_{k+i}r_{k+i} - q_{k+i}r_{k+i+j}) + (q_{k+i}r_{k+i+j} - q_{k+i+j}r_{k+i+j})| \\
&\leq |q_{k+i}| \cdot |r_{k+i} - r_{k+i+j}| + |r_{k+i+j}| |q_{k+i} - q_{k+i+j}| \\
&< |q_{k+i}| \cdot 2^{-k-i} + |r_{k+i+j}| \cdot 2^{-k-i} \\
&\leq (|q_0| + 1) \cdot 2^{-k-i} + (|r_0| + 1) \cdot 2^{-k-i} \\
&= (|q_0| + |r_0| + 2) \cdot 2^{-k-i} \\
&< 2^{-i}
\end{aligned}$$

□

**Lemma 2.2.** If  $x = \{q_i : i \in \mathbb{N}\} \neq 0$  is a real number, then  $\exists k \forall i (q_{k+i} \neq 0)$ .

*Proof.* Since  $x \neq 0$  it follows that that  $\exists k |q_k - 0| = |q_k| > 2^{1-k}$ . Assume  $q_{k+i} = 0$  for some  $i$ . Then  $|q_k| = |q_k - q_{k+i}| < 2^{-k} < 2^{1-k} < |q_k|$ , which is a contradiction. □

**Definition 2.22.** Let  $x = \{q_i : i \in \mathbb{N}\}, y = \{r_i : i \in \mathbb{N}\}$  be two real numbers. Then

$$x < y \iff x \leq y \iff \exists k \exists r \forall i (x_{k+i} + 2^{-r} < y_{k+i} + 2^{1-k-i})$$

**Lemma 2.3.** If  $x = \{q_i : i \in \mathbb{N}\} > 0$  is a real number, then  $\exists k \exists t \forall i (q_{k+i} > 2^{-t})$ .

*Proof.* Since  $0 < \{q_i : i \in \mathbb{N}\}$  we have that  $\exists k \exists r \forall i (0 + 2^{-r} < q_{k+i} + 2^{1-k-i})$ . For  $i > r + 2$  we have  $2^{-r} < q_{k+i} + 2^{1-k-i} \leq q_{k+i} + 2^{1-i} < q_{k+i} + 2^{-r-1} \implies 2^{-r-1} < q_{k+i}$ , with  $t = r + 1$ . □

**Definition 2.23.** If  $x = \{q_i : i \in \mathbb{N}\} > 0$  is a real number, then  $\frac{1}{x} := \{\frac{1}{q_{k'+i}} : i \in \mathbb{N}\}$ , where  $k' = k + 2t + 1$  and  $k, t$  are as in lemma 2.3.

**Theorem 2.9.** If  $x > 0$  is a real number, then so is  $\frac{1}{x}$ .

*Proof.* Let  $k'$  be as in definition 2.23. Then:

- $|q_{k'+i} - q_{k'+i+j}| < 2^{-k'-i}$ , because  $x$  is a real number
- $q_{k'+i} \cdot q_{k'+i+j} > 2^{-t} \cdot 2^{-t} = 2^{-2t}$ , because  $x > 0$

$$\text{Then } \left| \frac{1}{q_{k'+i}} - \frac{1}{q_{k'+i+j}} \right| = \frac{|q_{k'+i} - q_{k'+i+j}|}{|q_{k'+i} \cdot q_{k'+i+j}|} < \frac{2^{-k'-i}}{2^{-2t}} < \frac{2^{-k'-i}}{2^{-2t}} = 2^{2t-k'-i} < 2^{-i}. \quad \square$$

**Definition 2.24.** If  $x < 0$  is a real number, then  $\frac{1}{x} := -\frac{1}{-x}$

**Definition 2.25.** Let  $y \neq 0$  and  $x$  be real numbers. Then  $\frac{x}{y} := x \cdot \frac{1}{y}$ .

We state the following theorem without proof because of the sheer amount of work required to prove it.

**Theorem 2.10.** *Let  $x \neq 0$  be a real number. Then  $\frac{x}{x} = 1$ .*

It is possible to prove in  $\text{RCA}_0$ :

**Theorem 2.11.**  *$(\mathbb{R}, +, -, \cdot, /, 0, 1, <, =)$  is an Archimedean ordered field.*

We thus conclude this section having presented the construction and arithmetic of the natural, integer, rational and real numbers, understood as we have defined them. Now it is time to take an unexpected detour towards the study of computability theory.



# 3 Computability

---

*A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine.*

ALAN M. TURING

The theory of computability, or recursion theory, as it was originally named, dates back to the decade of the 1930s and has such prominent figures among its founders as Alan Turing, Emil Post, Kurt Gödel, Alonzo Church or Stephen Kleene. In this chapter we will establish the basic foundations of what later came to be a fully fledged branch of both mathematics and computer science, lying on the edge of both worlds for obvious reasons. More importantly, we will start to highlight the relation between what we just saw in the last chapter and the computable realm.

## 3.1 Models of computation and the Church-Turing thesis

Informally, a model of computation is a collection of definitions and rules that abstractly specify how to perform a *computation*. Since there exist so many, wildly different models of computation it is close to impossible to state the former sentence so that it makes sense for all of them. Some models are more natural, such as finite state automata, Turing machines or unlimited register machines (URM), in that they can be conceptualized as an actual mechanical device that performs certain tasks to produce a result. Others, like Church's  $\lambda$ -calculus, SKI combinators, partial recursive functions or elementary formal systems (due to Raymond Smullyan), *feel* a little alien at first sight since they are more abstract (the aforementioned machine-like nature is not obvious at all). This may explain why almost all modern *computers* (which is also a vague term) resemble Turing machines and not a computer for, say,  $\lambda$ -calculus.

*Remark.* Please note that these models (of computation) do not bear any relation to the **logical** models of a theory that we introduced in the previous chapter and will continue to discuss in later chapters.

However practical these models may be, though, they are still all interesting from a theoretical point of view. Their most interesting theoretical property is their **power**, i.e., the range of functions they can compute. It turns out that not all of them are equivalent, e.g. Turing machines are strictly more powerful than finite state automatas, although in practice, when a model is powerful enough it is actually equivalent to (or just as powerful as) the most powerful models we know of yet.

In the following subsections we will give an overview of the models of partial recursive functions and Turing machines – one because of its historical importance, one because of its dominance –, and present the well-known Church-Turing thesis.

### 3.1.1 Recursive functions

Our first approximation to the computability theory will be that of the recursive functions, since it is arguably the most *mathematical* of them. For the time being, we will call  $\mathbb{N}$  what we previously called  $\omega$ , i.e., the ideal, infinite set of natural numbers that we are used to. We then define:

**Definition 3.1** (Initial functions). The following functions are **primitive recursive**:

- The **zero function**,  $Z : \mathbb{N} \rightarrow \mathbb{N}$ , defined by

$$Z(n) := 0 \forall n$$

- The **successor function**,  $S : \mathbb{N} \rightarrow \mathbb{N}$ , defined by

$$S(n) := n + 1 \forall n$$

- The **projection functions**,  $\pi_i^k : \mathbb{N}^k \rightarrow \mathbb{N}$ , defined by

$$\pi_i^k(x_1, x_2, \dots, x_i, \dots, x_k) = x_i$$

for  $k \geq 1$  and  $1 \leq i \leq k$

**Definition 3.2** (Composition). Let  $f : \mathbb{N}^k \rightarrow \mathbb{N}, g_i : \mathbb{N}^m \rightarrow \mathbb{N}$  for  $1 \leq i \leq k$  be primitive recursive functions. The **composition** of  $f$  with  $g_1, \dots, g_k$  is the function  $h : \mathbb{N}^m \rightarrow \mathbb{N}$ , defined by

$$h(x_1, \dots, x_m) := f(g_1(x_1, \dots, x_m), \dots, g_k(x_1, \dots, x_m))$$

We say that  $h$  is primitive recursive.

**Definition 3.3** (Primitive recursion). Let  $f : \mathbb{N}^k \rightarrow \mathbb{N}, g : \mathbb{N}^{k+2} \rightarrow \mathbb{N}$  be primitive recursive functions. The **primitive recursion** of  $f$  and  $g$  is the function  $h : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ , defined by

$$\begin{aligned} h(0, x_1, \dots, x_k) &:= f(x_1, \dots, x_k) \\ h(n+1, x_1, \dots, x_k) &:= g(n, h(n, x_1, \dots, x_k), x_1, \dots, x_k) \end{aligned}$$

We say that  $h$  is primitive recursive.

*Remark.* Intuitively, the primitive recursion of  $f$  and  $g$  acts like a for loop starting at  $i = 0$  and finishing at  $i = n + 1$ . The arguments  $x_1, \dots, x_k$  are a set of initial conditions immutable by  $f$  and  $g$ . When  $i = 0$ , the initial calculation  $f(x_1, \dots, x_k)$  is performed. From then on, the function  $g$  is fed the current value of  $i$ , the result of the last iteration, and the original, immutable arguments  $x_1, \dots, x_k$ .

An example of a primitive recursive function is the addition function,  $+: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ .

**Theorem 3.1.** *Addition is primitive recursive.*

*Proof.* Let  $f(x) := \pi_1^1(x) = x, g(x, y, z) := S(\pi_2^3(x, y, z)) = S(y)$ . It suffices to note:

$$\begin{aligned} +(0, m) &:= f(m) = m \\ +(n+1, m) &:= g(n, +(n, m), m) = S(+(n, m)) \end{aligned}$$

So  $+(n, m) \equiv n + m$  verifies all the properties of addition and it is clearly primitive recursive.  $\square$

We could proceed in this fashion and show that most of the functions we are used to are, indeed, primitive recursive. In fact, the mathematical community was quite satisfied with this definition of *computability*. It was not until 1928 when a young Willhelm Ackermann, a student of Hilbert's, first published [3] his discovery of a recursive, non-primitive recursive function aptly named Ackermann's function. We present a variation of the original formulation due to Rózsa Péter and Raphael Robinson, which is nowadays the most common way to define it:

**Definition 3.4** (Ackermann's function).

$$A(m, n) := \begin{cases} n + 1 & \text{if } m = 0, \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0, \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

At this point there are two main questions worth discussing:

1. Why is this function not primitive recursive? The proof is technical and arduous, but the main idea is to show that the function  $A(n, n)$  dominates any primitive recursive function  $f$ , i.e., eventually grows (much) faster than  $f$ . That, of course, rules out the possibility of  $A(n, n)$  being primitive recursive, since then it would need to dominate itself. As to why it dominates all primitive recursive functions, a possible proof is to show that the set of functions that are dominated by  $A(n, n)$  includes the initial functions (easy), and that it is closed under composition and primitive recursion (intricate).
2. What do we mean when we say  $A(m, n)$  is *recursive*? The answer to this question is somewhat more discouraging: there is not a formal, completely satisfying definition of recursive (or computable) function. Of course, one can formally define the computable functions to be the  $\mu$ -recursive functions, as we will shortly see. Alternatively, there also exist Turing computable functions, computable functions in the sense of  $\lambda$ -calculus, etc. It turns out all of these definitions yield **exactly** the same class of computable functions so, by lack of a counterexample, we assume that any reasonable model of computation will be equivalent to all the ones we already know, and hence will not alter our notion of computable functions. Furthermore, the custom is to dispense with the formalities of choosing a particular model of computation and showing it can compute a given function, instead invoking the concept of **effectively computable** function, whose definition as “Any function for which there is an intuitively effective process for computing its values” is even vaguer. We will revisit this topic later in this chapter.

It could also be asked why did we not begin by defining ( $\mu$ -)recursive functions. First of all, starting with primitive recursive functions is more historically accurate. On a more practical side, we will lose some properties when we transition to recursive functions. Most notably, primitive recursive functions are **total**, i.e., they produce a resulting value for any possible input. This will not necessarily be the case with recursive functions, which may be **partial**.

**Definition 3.5.** A function  $f$  is defined for  $n$  if it produces a result when (or it allows to be) fed with input  $n$ . We write this as  $f(n) \downarrow$ . Otherwise, we write  $f(n) \uparrow$ .

**Definition 3.6.** A (partial) function  $f$  is **total** if  $\forall n(f(n) \downarrow)$ .

**Definition 3.7.** All primitive recursive functions are defined to be (total) recursive.

**Definition 3.8** ( $\mu$ -operator). Let  $g(x_1, \dots, x_n, m)$  be a partial recursive function. Then so is

$$f(x_1, \dots, x_n) := \mu m[g(x_1, \dots, x_n, m) = 0]$$

where

$$\mu m[g(x_1, \dots, x_n, m) = 0] = k \iff \begin{cases} g(x_1, \dots, x_n, k) = 0, \text{ and} \\ \forall (m < k)(g(x_1, \dots, x_n, m) \downarrow \wedge g(x_1, \dots, x_n, m) \neq 0) \end{cases}$$

*Remark.* Intuitively, the  $\mu$  operator performs a search looking for the least  $m$  such that  $g(x_1, \dots, x_n, m) \downarrow$  and is equal to 0. Notice that such an  $m$  may well not exist (e.g. consider  $g(x, m) = x + m + 1$ , then  $\forall n(f(n) \uparrow)$ ). It is also worth noting that  $= 0$  does not play any important role, we may define the  $\mu$  operator for arbitrary (recursive) relations with a distinguished parameter as Kleene originally did. [4]

**Definition 3.9.** A function  $f$  is **partial recursive** if it is primitive recursive or can be defined from partial recursive functions using (a finite number of applications of) the composition rule, the primitive recursive rule and the  $\mu$  operator.

A partial recursive function that is also total is simply called **recursive**.

In other words, the partial recursive functions are the smallest class of functions that includes the initial functions and is closed under composition, primitive recursion and the  $\mu$  operator. Similarly, we can define recursive sets and relations:

**Definition 3.10** (Recursive sets). Let  $S$  be a set and denote by  $\chi_S$  its indicator or characteristic function, that is:

$$\chi_S(x) := \begin{cases} 1 & \text{if } x \in S, \\ 0 & \text{if } x \notin S \end{cases}$$

$S$  is said to be recursive if  $\chi_S$  is.

**Definition 3.11** (Recursive relations). Let  $R(x_1, \dots, x_k)$  be a relation and denote by  $\chi_R$  its indicator or characteristic function, that is:

$$\chi_R(x_1, \dots, x_k) := \begin{cases} 1 & \text{if } R(x_1, \dots, x_k) \text{ holds,} \\ 0 & \text{if } R(x_1, \dots, x_k) \text{ does not hold} \end{cases}$$

$R$  is said to be recursive if  $\chi_R$  is.

Analogously,  $S$  (respectively  $R$ ) is said to be primitive recursive if  $\chi_S$  (respectively  $\chi_R$ ) is. (Partial) recursive relations are also *logically closed*, in the sense that if  $P$  and  $Q$  are (partial) recursive then so are  $\neg P$ ,  $P \wedge Q$  and  $P \vee Q$ . The analogous for sets is also true, so if  $X, Y$  are (partial) recursive sets then so are  $\mathbb{N} \setminus X$ ,  $X \cap Y$  and  $X \cup Y$ .

### 3.1.2 Turing machines

Turing machines are probably the epitome of models of computation. The intuition is that of a mechanical device with an infinite readable and writable tape subdivided in discrete cells in which the input, the intermediate steps and the final result are all written. The device is equipped with a head, which performs the reading and writing, scans the contents of a cell, outputs a symbol to the same cell and moves either left or right for the next step. Finally, the machine possesses a (finite) set of internal states through which it can jump. These, in the original formulation of Turing, are intended to represent the mental states of an actual human performing the computation. Formally:

**Definition 3.12.** We define a Turing machine  $M$  as a septuple  $(Q, \Gamma, b, \Sigma, \delta, q_0, F)$ , where:

- $Q$ , the set of **states** of  $M$ , is finite and non-empty,
- $\Gamma$ , the **tape alphabet**, is a finite, non-empty set of symbols,
- $b \in \Gamma$  is the **blank symbol**,
- $\Sigma \subseteq \Gamma \setminus \{b\}$  is the set of **input symbols**,
- $\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  is the **transition (partial) function**,
- $q_0 \in Q$  is the **initial state**, and
- $F \subseteq Q$  is the set of **final** or **accepting** states.

$M$  **halts** whenever  $\delta$  is not defined for the current state and tape symbol. If  $M$  is in state  $q \in Q$  when it halts, we say  $M$  **accepts** if  $q \in F$  and  $M$  **rejects** otherwise. This definition is adapted from Hopcroft and Ullman's "Introduction to Automata Theory, Languages and Computation" [5].

We will not deal here with the details, but any introductory course to this topic shows how a great number of modifications can be made to the definition of a Turing machine without changing its *computational power*, that is, the class of functions they can compute. Common examples include multi-track Turing machines, multi-tape Turing machines, two-way infinite tapes, two-dimensional tapes, and, perhaps most notably, non-deterministic Turing machines. This already somewhat suggests that Turing machines are the best we can expect to achieve in terms of computational power, based on a loose definition of *computability* that ultimately depends on our human condition.

**Definition 3.13.** A Turing machine  $M$  is said to compute function  $f$  when, for every  $n \in \mathbb{N}$ , if  $M$  takes input  $n$  it halts and outputs  $f(n)$ . We then say  $f$  is Turing computable and  $f = \varphi_M$ .

This definition can be formalized in many different ways. A rather natural one is to consider  $b = 0, \Sigma = \{0, 1\}$  and specify that  $M$  has input  $n$  if its tape starts with  $n+1$  consecutive 1s from the initial position of the head – this is to ensure we can always start by reading a 1. If  $f$  receives multiple parameters, say  $n_1, \dots, n_k$ , the tape should start with  $n_1 + 1$  consecutive 1s, followed by a 0 and then  $n_2 + 1$  consecutive 1s and so on. Similarly,  $M$  outputs  $n$  if there are exactly  $n$  (possibly non-consecutive) 1s left on the tape when it halts. We do not make distinction in this case between accepting and rejecting states. In these conditions we can prove the following:

**Theorem 3.2.** *The initial functions are Turing computable.*

*Proof.* • The easiest case is that of the successor function,  $S$ . Since we already have  $n + 1$  ones in the tape we can halt immediately and the output is correct.

- For the zero function,  $Z$ , we define the following transition function

$$\delta(q_0, 1) = (q_0, 0, R),$$

which indeed erases all occurring 1s in the tape.

- The projection functions,  $\pi_i^n$ , follow this scheme:

$$\begin{aligned}\delta(q_k, 1) &= (q_{k'}, 0, R), \\ \delta(q_{k'}, 1) &= \begin{cases} (q_{k'}, 1, R) & \text{if } k = i, \\ (q_{k'}, 0, R) & \text{if } k \neq i \end{cases} \\ \delta(q_{k'}, 0) &= (q_{k+1}, 0, R),\end{aligned}$$

renaming the initial state to  $q_1$ .

□

We can also prove, with some additional effort, that substitution, primitive recursion and the  $\mu$  operator can be *emulated* with Turing machines, so the closure properties of the partial recursive functions hold for Turing computable functions. Formally:

**Theorem 3.3.** *All partial recursive functions are Turing computable.*

The converse also holds:

**Theorem 3.4.** *All Turing computable functions are partial recursive.*

*Proof.* The details of the proof are tedious, but the main idea is to encode Turing machines with numbers in a **recursive** fashion. We also need to encode the *instantaneous description* of the machine, i.e., a description of the computation at any given point from which one can resume the process. To describe the whole state of the computation at any step it suffices to give the internal state, the symbol the head is reading, the word (as in a string of symbols) to the left of the current symbol and the word to the right. Assuming the states are numbered and the tape alphabet is binary, we have a natural interpretation of all the items in terms of numbers. Then we just need to show that:

1. The (code for the) initial configuration is a recursive function of the numerical input,
2. the (code for the) configuration at each step is a partial recursive function of the (code for the) configuration of the previous step, and
3. the machine output is a recursive function of the (code for the) machine's halting configuration.

Therefore, the composition of all of them is a partial recursive function that acts exactly as the machine. □



### 3.1.3 The Church-Turing thesis

We conclude this section by stating the well-known Church-Turing thesis, which we will consider a definition:

**Definition 3.14** (Church-Turing thesis). Every function for which there is an intuitively effective process for computing its values can be shown to be computable.

This statement needs some interpretation. First of all, when we say that the function is “computable” we mean Turing computable, or partial recursive, or the analogous for any other equivalent model of computation. Since so many of them are equivalent, it seems the notion of computability transcends any particular model and is agnostic of them. This relates to the first half of the sentence, where we mention “intuitively effective process”. If we regard the Church-Turing thesis as no more than a plain definition, then this would just define “intuitively effective process”, but it actually goes in the other direction: we assume that we, as humans, know how to distinguish intuitively between a process that performs a computation and one that could not possibly succeed at it. Then the thesis reads: if we can think of some process that intuitively can perform certain computation, it surely must be formalizable as, say, a Turing machine. In light of this it is now obvious why so many different models are equivalent, although for some of them the “intuitive” part requires some convincing. Of course this is circular argument, but it gives us confidence that our understanding of computation is good enough.

## 3.2 The limits of computation

In the previous section we have established the fundamental facts about computability. However, perhaps the most interesting topics arise when the requirements are relaxed and *uncomputable* objects are allowed. In this section we will study these objects – that range from numbers to sets, trees or functions – and give precise definitions of what we mean by uncomputable as well as results about their uncomputability.

### 3.2.1 Recursive enumerability

The weakest modification we can think of is that of recursive enumerability, in which, intuitively, we only ask for definitive answers for the positive cases, but allow the other cases to be unanswered. Formally:

**Definition 3.15.** A set  $X$  is **recursively enumerable** (abbreviated **r.e.**) if there exists some Turing machine that, for every input  $n$ , accepts if and only if  $n \in X$ , but may either reject or not halt if  $n \notin X$ . Equivalently,  $X$  is r.e. if there exists some Turing machine that *enumerates* the elements of  $X$ , that is, outputs exactly those  $n$  such that  $n \in X$ , but may never stop doing so (if  $X$  is infinite).

*Remark.* The equivalence of both definitions is natural when one has enough experience with the subject, but may be a little surprising for a newcomer. The argument goes along these lines:

- Assume the first definition holds. We can then proceed by stages, so in the first stage we execute one step of the machine for input 1. In the second stage we perform the second step of the machine for input 1 and the first for input 2. In the third stage, the third step of the machine for input 1, the second for input 2 and the third for input 3 are executed. Whenever we reach a step for some input  $n$  in which the machine accepts, we can stop working on that  $n$  and output it, effectively listing it as a member of  $X$ . Since we assumed the machine always stops for those  $n$  such that  $n \in X$ , we can be sure that eventually all such  $n$  will be output or, in other words, listed. If  $n \notin X$  the machine will either never stop (so it will not appear in the list) or reject, in which case we obviously do not output that number.
- Now assume the second part of the definition holds. For every possible  $n$ , we can start scanning the list and stop whenever we find it. Since the list contains all possible  $n \in X$ , we can be sure that we will eventually find it somewhere in the list. Alternatively, if  $n \notin X$ , we will keep hopelessly looking for  $n$  in the list, never to find it, so the machine will never halt.

Note that for this proof we have relied on the Church-Turing thesis by giving an effectively computable process by which a solution for one of the problems becomes, computably, a solution for the other one, without having actually, explicitly constructed a Turing machine (or a partial recursive function, for that matter) that performs this task. This is a nice commodity since it both eases the process of proving a statement and also provides more insight to the reader as to *how* and *why* the proof works. Unfortunately, the details are lost, but they should be easily reconstructable from the main ideas exposed.

Also, note that due to the equivalence of Turing machines and partial recursive functions the second definition may also be regarded as a machine that performs the computation  $f(0), f(1), f(2), \dots$  for some computable function  $f$  and lists those values  $f(n)$  such that  $f(n) \downarrow$ . This gives us yet another useful characterization for r.e. sets, by which a set  $X$  is r.e. if and only if there exists some partial recursive function  $f$  such that

$$X = \{f(0), f(1), f(2), \dots\} = \text{range}(f).$$

In fact, there is a stronger version of this equivalence by which a **non-empty** set is r.e. if and only if it is the range of a primitive recursive function, but the weak version will be enough for us.

Of course, all computable sets are r.e. since we can effectively decide for each possible candidate whether it is contained in the set or not. A natural question is whether this class of sets includes some non-computable ones, since otherwise the definition would be redundant. The fact that we have a different name for this concept already suggest that they are indeed a proper superclass, as we shall see shortly. First, we will see another classical result that shows that computability and recursive enumerability are actually very much related:

**Theorem 3.5.** *A set  $X$  is computable if and only if both  $X$  and  $\overline{X} := \mathbb{N} \setminus X$  are r.e.*

*Proof.* The forward direction is quite trivial since we already know that  $X$  being computable implies being r.e. For the complement, note that since  $X$  is computable we can also effectively decide which  $ns$  are **not** in  $X$ , so it is r.e. too (in fact computable).

For the backwards implication, we can devise a machine that lists both  $X$  and  $\overline{X}$  separately, so that for any input  $n$  we just need to run it long enough that  $n$  will appear in either the list of  $X$  or  $\overline{X}$ , and then answer accordingly.  $\square$

Another interesting result is a particularization of the equivalence discussed in remark 3.2.1. First we need a definition:

**Definition 3.16.** A total function  $f$  is said to be **increasing** if

$$\forall n(f(n) < f(n+1)).$$

Then we have the following:

**Lemma 3.1.** *An infinite set  $X$  is computable if and only if it is the range of an increasing function.*

*Proof.* • Assume  $X$  is computable, so  $n \in X$  is a computable relation. Then we can define:

$$\begin{aligned} f(0) &= \mu n[n \in X], \\ f(m+1) &= \mu n[n > f(m) \wedge n \in X]. \end{aligned}$$

Thus  $f$  is obviously increasing and its range coincides with  $X$ . Note that in this case we have applied the remark made when we first defined the  $\mu$  operator about how we can replace  $g(x_1, \dots, x_k, n) = 0$  with an arbitrary computable relation. If we wanted to stick to the original definition, we might have considered  $g = \chi_{\overline{X}}$ .

- Assume now  $X$  is indeed the range of some increasing function  $f$ . To determine whether  $n \in X$  or not, we just need to compute the first  $n+1$  values of  $f$ , since they come in increasing order, and check for  $n$  among those values. Formally,

$$n \in X \iff \exists(m \leq n)(f(m) = n).$$

□

### 3.2.2 Enumerating Turing machines

It turns out that if we want to keep studying the nature of Turing machines and some of their most interesting properties we need to enumerate them. This can be accomplished in a number of ways, one of the most common being Gödel numberings. Although we will not provide the details here, the idea is very similar to that of Theorem 3.4.

Making some simplifications over the Turing machine definition (such as the tape alphabet being binary) that **do not** change the class of functions that are Turing computable (in the sense that any general Turing machine can be *translated* to this new subset), we only need to encode the program of the machine (i.e., its transition function) to uniquely represent it. In order to do this we can make use of the fundamental theorem of arithmetic and encode using a finite number of prime powers. If this is done properly, given the code for a Turing machine we just need to check how many times each of those primes divides the code and translate accordingly.

Assume we already have such an encoding called  $gn$  (for Gödel numbering) that takes the description of a Turing machine program and returns a unique natural number. Then we can define

**Definition 3.17.** The  $e^{\text{th}}$  Turing machine program is given by

$$P_e = \begin{cases} P & \text{if } gn^{-1}(e) \downarrow = P, \\ \emptyset & \text{otherwise.} \end{cases}$$

That is, if  $e$  is the code for a program then  $P_e$  is defined as that unique program, otherwise  $P_e$  is the empty program. Note that by our input/output conventions this means it computes the successor function, but we could just as well replace it by any other program.

The  $e^{\text{th}}$  Turing machine,  $T_e$ , is then the Turing machine with program  $P_e$ . Similarly, the  $e^{\text{th}}$  partial recursive function,  $\varphi_e$ , is the function computed by  $T_e$ .

Then we have the following useful result:

**Theorem 3.6** (Enumeration theorem).  *$\varphi_k(n)$  is a partial recursive function, say  $f$ , of  $k$  and  $n$  such that, for each  $k$ ,  $f(k, \cdot) = \varphi_k$ .*

*Proof.* Given  $k$ , we just need to find  $\varphi_k$ , which we can do **computably**, and then compute  $\varphi_k(n)$ .  $\square$

The  $f$  from the last theorem is, in fact, the novelty Turing introduced in his 1936 paper “On computable numbers, with an application to the Entscheidungsproblem” [6]: the **universal Turing machine**.

**Theorem 3.7** (Universal Turing machine). *There exists a Turing machine  $U$  which, if given input  $(e, n)$ , simulates the  $e$ -th Turing machine with input  $n$ . In other words,  $\forall e \forall n (\varphi_U((e, n)) = \varphi_e(n))$ .*

Back to our quest of finding a recursive enumerable but non-computable set, we only need one more previous definition:

**Definition 3.18.** The halting problem asks, given  $e, n \in \mathbb{N}$ , whether  $\varphi_e(n) \downarrow$ . Calling  $T_e$  the  $e$ -th Turing machine, the halting set of  $T_e$ ,  $W_e$ , is defined as the set of inputs that make it halt. Equivalently,  $W_e$  is the domain of  $\varphi_e$ .

$$W_e := \{n : \varphi_e(n) \downarrow\}.$$

The halting problem for  $T_e$  is said to be **solvable** if  $W_e$  is computable.

It turns out that this lets us enumerate recursively enumerable sets (as confusing as that may sound):

**Theorem 3.8** (The normal form theorem for r.e. sets). *A set  $X$  is r.e. if and only if  $X = W_e$  for some  $e$ .*

*Proof.* • If  $X$  is r.e. then define the following partial recursive function:

$$\varphi(n) := \begin{cases} 0 & \text{if } n \in X, \\ \text{undefined} & \text{if } n \notin X \end{cases}$$

Alternatively, it may be easier to visualize the construction of a Turing machine that, given  $n$ , emulates the machine  $M_X$  that checks whether  $n \in X$ . If it ever stops and confirms that  $n \in X$ , the new machine outputs 0. Otherwise, either  $M_X$  never halts or it halts and rejects, in which case we can simply add an infinite loop to our machine.

- Suppose now  $X = W_e$  for some  $e \in \mathbb{N}$ . To show  $X$  is r.e. we can build  $M_X$  by simulating  $T_e$  on input  $n$ . If  $T_e$  ever halts we confirm  $n \in X$ , otherwise we do not answer.

□

*Remark.* This theorem gives us yet another characterization of r.e. sets as exactly the class of sets that are the domain of some partial recursive function.

And now for something completely different:

**Theorem 3.9.** *There exists a r.e. set  $K$  that is not computable.*

*Proof.* Define  $K_0 = \{(e, n) : n \in W_e\}$ . This set is r.e. because, given  $m$ , we can computably decide whether  $m = (e, n)$  for some  $e, n \in \mathbb{N}$ . In that case, retrieve them and computably simulate  $T_e$  with input  $n$ . If  $T_e$  ever halts, then confirm  $(e, n) = m \in K_0$ . Otherwise, we do not answer.

Now define  $K = \{n : n \in W_n\}$ . This set is isomorphic to the diagonal of  $K_0$ , since

$$n \in K \iff n \in W_n \iff (n, n) \in K_0.$$

Therefore,  $K$  is r.e. because we can adapt the algorithm for  $K_0$  to an algorithm for  $K$ . If  $K$  were to be computable, by theorem 3.5  $\bar{K}$  should be r.e. too, and thus  $\bar{K} = W_e$  for some  $e$ . Now, does  $e \in \bar{K}$ ?

$$e \in W_e \iff e \in \bar{K} \iff e \notin K \iff e \notin W_e$$

So we reach a contradiction in both cases. Therefore  $\bar{K}$  **cannot** be r.e., and so  $K$  is not computable. □

Remember the definition of the halting problem? We just found an example of a r.e. set that is not computable. But, being a r.e. set it must be the halting set of some Turing machine. In general:

**Corollary 3.1.** *The halting problem of a given Turing machine is **not** decidable.*

This does not mean that Turing machines cannot have recursive halting sets, as there are plenty of trivial counterexamples to that statement. What it claims is that we cannot possibly devise a general Turing machine (or an algorithm, for that matter) that, given any Turing machine and input, decides whether the machine will halt or not for that input.

Interestingly, we can ask ourselves a sort of converse for this: we know we can enumerate Turing machines but we cannot decide whether they halt, can we instead enumerate only those machines we know for sure have to halt? This is equivalent to enumerating total recursive functions, and it turns out the answer to the question is no by a certain diagonal argument, for assuming  $\{f_i : i \in \mathbb{N}\}$  was an enumeration of said functions, the function  $g(n) := f_n(n) + 1$  would be total and computable by construction but could not possibly appear in our listing. Of course, the collection of all total recursive functions is countable; the assertion is that they cannot be **computably** enumerated.

In fact, the ultimate culprit of this is the universal Turing machine. Generalizing this result, consider a certain class of functions  $C$  including a universal function, i.e., a function  $U$  in  $C$  such that

1. for every  $f$  in  $C$  there exists some natural number  $e$  verifying  $U(e, n) = f(n)$  for all  $n$ , and
2. for every  $e$ , the function  $U_e$  defined by  $U_e(n) := U(e, n)$  is in  $C$ .

If we further assume that all the functions in  $C$  are total and consider the function  $f(n) := U(n, n) + 1$ , it follows that  $f$  cannot be in  $C$ . Otherwise, and under very mild conditions for closure, it means that  $C$  includes some non-total functions. This helps us to improve our intuition about why partial recursive functions are enumerable and have a universal function but (and, in a sense, *because*) they include some non-total functions. On the other hand, primitive recursive functions can be effectively enumerated and are certainly total, but they do not include any *universal primitive recursive* function. Total recursive functions, asking for both totality and universality, lose enumerability.

### 3.3 Reducibility

By this point we already know we can pose problems for which there is no possible *computable* solution. However, a natural question to ask is whether all such unsolvable problems are *equally unsolvable*, or *equally hard*. Similar to what happens with infinite cardinals in set theory, some problems are

harder than others. Intuitively, a problem  $A$  is harder than a problem  $B$  if we can describe a process by which, assuming a solution for problem  $A$  we can get a solution for problem  $B$ , in which case we say  $B$  is reducible to  $A$ . This notion can be formalized in several ways, but we will only present two of the most popular alternatives: **many-one reducibility** and **Turing reducibility**.

### 3.3.1 Many-one reducibility

**Definition 3.19.** Let  $A$  and  $B$  be sets of natural numbers. We say  $B$  is **many-one reducible** or  **$m$ -reducible** to  $A$  (written  $B \leq_m A$ ) if there exists some total recursive function  $f$  such that:

$$\forall n (n \in B \iff f(n) \in A).$$

This definition gives us a rather natural notion of reducibility together with a simple mathematical definition. We can outright proceed to prove some facts about it.

**Lemma 3.2.** For any sets  $A$  and  $B$ , we have  $A \leq_m B \iff \bar{A} \leq_m \bar{B}$ .

*Proof.* Assume  $A \leq_m B$  and let  $f$  be the computable function that witnesses the reducibility. Then

$$n \in \bar{A} \iff \neg(n \in A) \iff \neg(f(n) \in B) \iff f(n) \in \bar{B},$$

so  $f$  itself proves that  $\bar{A} \leq_m \bar{B}$ . It follows from our argument that if  $g$  witnesses  $\bar{A} \leq_m \bar{B}$  then it also proves  $A \leq_m B$ , thus completing our proof.  $\square$

**Lemma 3.3.** 1. If  $A \leq_m B$  and  $B$  is computable, then so is  $A$ .

2. If  $A \leq_m B$  and  $B$  is r.e., then so is  $A$ .

*Proof.* In both cases, let  $f$  be the computable function that proves the reducibility. Then  $\chi_A = \chi_B \circ f$ .

1. If  $B$  is computable then  $\chi_B$  is total recursive, so  $\chi_A$  is total recursive and therefore  $A$  is computable.
2. If  $B$  is r.e. then  $\chi_B$  is partial recursive, so  $\chi_A$  is partial recursive and therefore  $A$  is r.e.

$\square$

This still suits our intuition that reducibility respects *difficulty*. Considering the particular recent example of the set we found to be r.e. but not computable we can prove the following:



**Lemma 3.4.**  $K \leq_m K_0$  and thus  $K_0$  is r.e. but not computable.

*Proof.* Notice that  $n \in K \iff n \in W_n \iff (n, n) \in K_0$ , so  $f(n) = (n, n)$ , being a total recursive function, shows that indeed  $K \leq_m K_0$ . Now if  $K_0$  was computable, by lemma 3.3, we could conclude that  $K$  itself is computable too. But we already proved in theorem 3.9 that  $K$  is not computable, so that means  $K_0$  is also non-computable.  $\square$

In fact, there is a reason we have considered the set  $K_0$  other than being a very natural example: in a sense,  $K_0$  is the *hardest* r.e. set that exists. Formally:

**Lemma 3.5.** A set  $A$  is r.e. if and only if  $A \leq_m K_0$ .

*Proof.* Since  $K_0$  is r.e. we already know  $A \leq_m K_0$  implies that  $A$  is r.e. Assume now  $A$  is r.e., so  $A = W_e$  for some  $e \in \mathbb{N}$ . Using the same argument as before:

$$n \in A \iff n \in W_e \iff (e, n) \in K_0.$$

Therefore,  $f(n) = (e, n)$  shows that  $A \leq_m K_0$ .  $\square$

Thanks to this notion of reducibility we can *almost* actually order sets with respect to their complexity:

**Lemma 3.6.**  $\leq_m$  is a preorder relation.

*Proof.* We need to prove reflexivity and transitivity:

1. For reflexivity, taking  $f(x) = x$  shows  $A \leq_m A$  for every  $A$ .
2. For transitivity, assume  $A \leq_m B$  via  $f$  and  $B \leq_m C$  via  $g$ . Then

$$n \in A \iff f(n) \in B \iff g(f(n)) \in C,$$

so  $h = g \circ f$  is total recursive and shows  $A \leq_m C$ .  $\square$

Unfortunately,  $\leq_m$  is not a *full* order since we cannot distinguish sets only by looking at their upper and lower bounds. Indeed:

**Lemma 3.7.** There exist  $A$  and  $B$  such that  $A \leq_m B$  and  $B \leq_m A$  but  $A \neq B$ .

*Proof.* Any two different computable sets should do. For a concrete example, take  $A = 2\mathbb{N}$ ,  $B = 2\mathbb{N} + 1$  the even and the odd numbers. In both cases  $f(n) = n + 1$  proves the inequality, but they are very much not equal.  $\square$

However, we can use a recurring trick in mathematics in which we treat as equal things that are indistinguishable for some construct – in our case  $\leq_m$  – and pretend they are just a single thing. Formally:

**Definition 3.20.** Two sets  $A$  and  $B$  are said to be **many-one equivalent** (written  $A \equiv_m B$ ) if  $A \leq_m B$  and  $B \leq_m A$ .

**Lemma 3.8.**  $\equiv_m$  is an equivalence relation.

*Proof.* Reflexivity and transitivity follow straight from lemma 3.6. For symmetry we have:

$$A \equiv_m B \iff (A \leq_m B) \wedge (B \leq_m A) \iff (B \leq_m A) \wedge (A \leq_m B) \iff B \equiv_m A,$$

thereby completing our proof.  $\square$

**Definition 3.21.** An equivalence class of  $\equiv_m$  is called an  **$m$ -degree**. We write  $\mathbf{a} = \deg(A)$  where  $A$  is any set and  $\mathbf{a}$  the  $m$ -degree to which it belongs. We call  $\mathcal{D}_m := \mathcal{P}(\mathbb{N}) / \equiv_m$  the collection of all  $m$ -degrees.

*Remark.* To avoid confusion, we shall denote  $m$ -degrees by boldface, lower-case latin letters, whereas sets are denoted by regular, uppercase latin letters.

In light of definition 3.21 we can now naturally adapt many-one reducibility to  $\mathcal{D}_m$ .

**Definition 3.22.** An  $m$ -degree  $\mathbf{a}$  is said to be reducible to  $\mathbf{b}$  (written  $\mathbf{a} \leq_m \mathbf{b}$ ) if there exist sets  $A, B$  such that  $A \in \mathbf{a}$ ,  $B \in \mathbf{b}$  and  $A \leq_m B$ .

With this degree ordering we get a lot of interesting algebraic structure on  $\mathcal{D}_m$  – e.g.  $\mathcal{D}_m$  is a join-semilattice. However, before going too far on this we want to take a step back and reflect on some of the basic properties of  $\leq_m$ . As it stands, our current notion of reducibility is too restrictive: it very accurately distinguishes sets by their *complexity*, but it is a little *too accurate*. In practice it is difficult to show  $m$ -reducibility too, because of how *thin* the  $m$ -degrees are.

Intuitively, we would like a set and its complement to be just as difficult: once we know the answers for one we just need to flip them in order to get the answers for the other.  $K$  gives us a counterexample to that hope: it is r.e. but its complement is certainly not, so if we want to match our intuition we need to come up with a different kind of reduction. In the next section we will present this new reducibility, called **Turing reducibility**, which is actually a weaker version of  $m$ -reducibility, so part of our findings will still be relevant.

Before ending this section, though, it may be worth noting that  $m$ -reducibility is not by any means the best we can do at classifying sets. A natural variation of  $m$ -reducibility is 1-reducibility, where we ask the  $f$  from definition 3.19 to also be one-to-one. This is a strictly stronger notion of reducibility, though not very practical because of how *similar* the sets need to be in order to apply to the definition.

### 3.4 Turing reducibility, oracle machines and the Turing universe

For the purposes of this section we first need to define the concept of **oracle machine**. Intuitively, an oracle is a black box able to compute or solve a problem immediately. Of course, an oracle may just be a Turing machine that computes something (say,  $n \mapsto 3n+5$ ), but that is not very useful because we could just simulate that machine on an outer Turing machine to get the result. The power of oracles comes when we allow them to perform **any** computation, even things that cannot possibly be computed by a Turing machine (such as the Busy Beaver function, for example). Most of the times, though, we will be concerned with decision problems, in which an oracle is posed a question and it answers YES or NO in a single computational step. This kind of oracles is easier to formalize but ultimately equivalent to any other oracle. For example, a busy beaver computing machine may be constructed with an oracle that, when input with  $(n, m)$ , answers YES if and only if  $BB(n) = m$ . Oracle machines are not physically possible, but they constitute a nice theoretical artifact that allows us to further explore the limits of computability.

**Definition 3.23.** An oracle Turing machine is a septuple  $(Q, \Gamma, b, \Sigma, \delta, q_0, F)$  that operates as a two-taped Turing machine with three special states:

- whenever the machine arrives at state  $A$  (for *ask*), the contents of the **oracle tape** are fed to the oracle,
- if the oracle should answer YES, the machine transitions to state  $Y$ ,
- otherwise, if the oracle's answer is NO, the machine transitions to state  $N$ .

We allow these transitions to be *no-op* in the sense that they modify neither the contents of the tape nor move the reading head.

*Remark.* Intuitively, the machine programmer is allowed to implement *hooks* for the answers of the oracle via transitions from states  $Y$  and  $N$ . The transitions from state  $A$  are *hard-coded* by the oracle.

By an abuse of notation, we will usually denote an oracle by the set whose membership problem it solves, i.e., an oracle  $A$  is such that answers all the questions of the form “Is  $n \in A$ ” for some fixed set  $A$ . We are confident that context will be enough to rule out any possible ambiguity.

Thanks to definition 3.23 we arrive at the concept of **relative Turing computability**:

**Definition 3.24.** A partial function  $f$  is  $A$ -Turing computable if it can be computed by a Turing machine with oracle  $A$ .

Note that the fact that we are restricting oracles to solve membership problems of sets does not really affect the definition, it could be any kind of oracle. We could also completely analogously define relative Turing computability for sets and relations in the way we did in previous sections. In general, we will use the symbol  $\leq_T$  to denote Turing reducibility between two sets, functions, problems, etc. In the particular case of sets,  $A \leq_T B$  can be understood as “if I can effectively decide whether  $n \in B$  then I can effectively decide  $n \in A$ ”.

In a similar vein, we can also adapt the Church-Turing thesis to this new realm:

**Definition 3.25** (Relativised Church-Turing thesis). All formalisations of “ $A$  computable from  $B$ ” which are sufficiently reasonable and sufficiently general are equivalent to Turing reducibility, and so can be written  $A \leq_T B$ .

Intuitively, in all of the previous definitions, lemmas, etc. we only allowed objects that can be built from scratch using computational rules (such as composition or minimization). Relative computability is exactly the same but building from a *higher point*, e.g. what could we compute if we assume we can somehow solve the halting problem? This intuition can be formalized, but it is nonetheless important to grasp what this new definition is really introducing.

Just as we did with many-one reducibility, we can define Turing equivalence as follows:

**Definition 3.26.** Two sets  $A$  and  $B$  are said to be **Turing equivalent** (written  $A \equiv_T B$ ) if  $A \leq_T B \wedge B \leq_T A$ .

**Lemma 3.9.**  $\equiv_T$  is an equivalence relation.

*Proof.* It is trivial to check that Turing reducibility is reflexive, and symmetry follows from the definition of  $\equiv_T$ . For transitivity, note that reductions are transitive because:

- If  $A \leq_T B$  every question of the form  $n \in A$  can be computably decided by knowing the answer to finitely many questions of the form  $m_i \in B$ ,
- If  $B \leq_T C$  every question of the form  $m_i \in B$  can be computably decided by knowing the answer to finitely many questions of the form  $k_{ij} \in C$ ,

So we can *compose* these two statements and assert that if both hold then every question of the form  $n \in A$  can be computably decided by knowing the answer to finitely many questions of the form  $k_{ij} \in C$ , and thus  $A \leq_T C$ .  $\square$

It should be easy enough to see how we can adapt all the theory we have developed up until this point about computability to **relative** computability. For this reason we will state several useful lemmas without proof, since the generalizations of the proofs we have already seen are quite natural.

**Definition 3.27.** A set  $B$  is recursively enumerable in  $A$  or  $A$ -r.e. if we can computably enumerate the members of  $B$  using a Turing machine with an oracle for  $A$ .

**Lemma 3.10.** • If  $A \leq_T B$  (that is,  $A$  is  $B$ -computable) then  $A$  is  $B$ -r.e.

- We can enumerate the Turing machines with oracle  $A$ , and hence the  $A$ -partial recursive functions. We will denote them by  $\Phi_e^A$ .
- The halting set of  $\Phi_e^A$  is  $W_e^A$ , which we can  $A$ -recursively enumerate.
- A set  $B$  is  $A$ -r.e. if and only if it is the range of an  $A$ -partial recursive function and the domain of an  $A$ -partial recursive function.

Also:

**Lemma 3.11.** If  $X$  is  $A$ -r.e. and  $A \leq_T B$ , then  $X$  is  $B$ -r.e. too

*Proof.* The sketch of the proof is as follows. We know that, given an oracle for  $A$ , we can recursively enumerate  $X$ ; and given an oracle for  $B$  we can computably decide  $A$ . Therefore, *composing* these two facts as in the proof for lemma 3.9, given an oracle for  $B$  we can answer all the questions of the form  $n \in A$  that we need to recursively enumerate  $X$ , so  $X$  is  $B$ -r.e.  $\square$

**Definition 3.28.** An equivalence class of  $\equiv_T$  is called a **Turing degree**. We write  $\mathbf{a} = \deg(A)$  where  $A$  is any set and  $\mathbf{a}$  the Turing degree to which it belongs. We call  $\mathcal{D} := \mathcal{P}(\mathbb{N})/\equiv_T$  the collection of all Turing degrees, also known as the **Turing universe**.

*Remark.* Note that we have reused part of the notation from  $m$ -degrees since we will not be working with them anymore, so there will be no ambiguity.

**Theorem 3.10.** 1. There is a **least** Turing degree  $\mathbf{0}$ , the collection of all computable sets,

2. Each Turing degree is countably infinite,

3. The collection of degrees bounded by a certain Turing degree  $\mathbf{a}$ , i.e.,  $\{\mathbf{b} \in \mathcal{D} : \mathbf{b} \leq_T \mathbf{a}\}$ , is countable,

4.  $\mathcal{D}$  is uncountable

*Proof.* 1. A set reducible to a computable set must itself be computable, since oracles for computable sets do not augment the computational capabilities of a Turing machine. The answers the oracle gives might just have been computed by the machine, therefore effectively eliminating the need for the oracle.

2. Let  $A \in \mathbf{a}$  for some set  $A$  and some Turing degree  $\mathbf{a}$ . Then  $\mathbf{a} = \{X \in \mathcal{P}(\mathbb{N}) : X \equiv_T A\} \subseteq \{X \in \mathcal{P}(\mathbb{N}) : X \leq_T A\}$ .  $X \leq_T A$  is equivalent to saying that  $X$  is  $A$ -computable, which implies  $X$  is  $A$ -r.e. But to any  $A$ -r.e. set  $X$  corresponds some (possibly many)  $A$ -partial recursive function  $\Phi_e^A$ . Therefore  $|\mathbf{a}| \leq |\{\Phi_e^A : e \in \mathbb{N}\}| = \aleph_0$ , so  $\mathbf{a}$  is countable. To see  $\mathbf{a}$  is infinite, let  $A \in \mathbf{a}$  again. For every  $i \in \mathbb{N}$ , define:

$$A_i := \begin{cases} A \setminus \{i\} & \text{if } i \in A, \\ A \cup \{i\} & \text{if } i \notin A \end{cases}$$

Then

$$i \in A_i \iff i \notin A \iff i \notin A_j$$

for  $i \neq j$ . In particular,  $A_i \neq A_j$  for any  $i \neq j$ . It is clear that  $A_i$  can be computed from  $A$ . For the reverse implication, assume we have an oracle for  $A_i$ . Then

$$n \in A \iff \begin{cases} n \in A_i & \text{if } n \neq i, \\ i \notin A_i & \text{if } n = i \end{cases}$$

So  $A_i \equiv_T A$  for every  $i \in \mathbb{N}$ . Finally,  $\{A_i : i \in \mathbb{N}\} \subseteq \mathbf{a}$ , so  $\mathbf{a}$  is infinite.

3. Let  $L = \{X \in \mathcal{P}(\mathbb{N}) : X \leq_T A\}$  for some  $A \in \mathbf{a}$ . As in the previous proof,  $L$  is a subset of the collection of  $A$ -r.e. sets, so  $L$  is countable by the same argument. Now

$$L = \bigcup_{\mathbf{b} \in \{\mathbf{c} \in \mathcal{D} : \mathbf{c} \leq_T \mathbf{a}\}} \mathbf{b}$$

and we already know  $\mathbf{b}$  is infinite. If there were uncountably many  $\mathbf{b} : \mathbf{b} \leq_T \mathbf{a}$  then  $L$  itself would be uncountable, so  $\{\mathbf{b} \in \mathcal{D} : \mathbf{b} \leq_T \mathbf{a}\}$  must be countable.

4.  $\mathcal{P}(\mathbb{N}) = \bigcup_{\mathbf{a} \in \mathcal{D}} \mathbf{a}$ . Since Turing degrees are countable but  $\mathcal{P}(\mathbb{N})$  is not,  $\mathcal{D}$  itself must be uncountably infinite. □

As we briefly mentioned for  $m$ -degrees,  $\mathcal{D}$  can be endowed with a (join-)semilattice structure as follows:

**Definition 3.29** (Turing join). Given two sets  $A$  and  $B$ , their **recursive** or **Turing join** is defined as

$$A \oplus B := \{2n : n \in A\} \cup \{2m+1 : m \in B\}.$$

**Lemma 3.12.** *Let  $A, B$  be sets. Then  $(A \leq_T A \oplus B) \wedge (B \leq_T A \oplus B)$ .*

*Proof.* Assume we are given an oracle for  $A \oplus B$ . Given  $n \in \mathbb{N}$

- $n \in A \iff 2n \in A \oplus B,$
- $n \in B \iff 2n+1 \in A \oplus B.$

□

**Lemma 3.13.** *Let  $A, B, C$  be sets such that  $A \leq_T C, B \leq_T C$ . Then  $A \oplus B \leq_T C$ .*

*Proof.* Assume we can compute  $A$  and  $B$  given an oracle for  $C$ . Given  $m \in \mathbb{N}$ , we can computably decide whether  $m$  is even or odd and get the corresponding  $n$ .

- If  $m$  is even,  $m \in A \oplus B \iff n \in A$ , which we can computably decide with the oracle  $C$ ,
- If  $m$  is odd,  $m \in A \oplus B \iff n \in B$ , which we can computably decide with the oracle  $C$ ,

So  $A \oplus B \leq_T C$ . □

**Lemma 3.14.** *Let  $A, B$  be sets. Then  $A \oplus B \equiv_T B \oplus A$ .*

*Proof.* From lemma 3.12,  $(A \leq_T A \oplus B) \wedge (B \leq_T A \oplus B)$ . Exchanging their places, we get  $(A \leq_T B \oplus A) \wedge (B \leq_T B \oplus A)$ . Using lemma 3.13 twice we get both  $A \oplus B \leq_T B \oplus A$  and  $A \oplus B \geq_T B \oplus A$ , so we can conclude  $A \oplus B \equiv_T B \oplus A$ .  $\square$

**Definition 3.30.** Let  $\mathbf{a}, \mathbf{b}$  be Turing degrees and  $A \in \mathbf{a}, B \in \mathbf{b}$ . The **join** of  $\mathbf{a}$  and  $\mathbf{b}$  is defined as follows:

$$\mathbf{a} \vee \mathbf{b} := \deg(A \oplus B) = \{X \in \mathcal{P}(\mathbb{N}) : X \equiv_T A \oplus B\}.$$

**Theorem 3.11.** *The Turing join induces a join operation on  $\mathcal{D}$ .*

*Proof.*  $\mathbf{a} \vee \mathbf{b}$  is clearly a degree by definition.

- $\mathbf{a} \leq_T \mathbf{a} \vee \mathbf{b}, \mathbf{b} \leq_T \mathbf{a} \vee \mathbf{b}$  follows from lemma 3.12.
- $(\mathbf{a} \leq_T \mathbf{c}) \wedge (\mathbf{b} \leq_T \mathbf{c}) \implies \mathbf{a} \vee \mathbf{b} \leq_T \mathbf{c}$  follows from lemma 3.13.

Therefore  $\mathbf{a} \vee \mathbf{b}$  is the **least upper bound** of  $\mathbf{a}$  and  $\mathbf{b}$  in  $\mathcal{D}$ .  $\square$

We are almost ready to define the most important operation on Turing degrees, at least for our purposes. If we go back to lemma 3.5, we found  $K_0$  to be, in terms of  $m$ -reducibility, the *hardest* possible r.e. set, and it was certainly not computable. We can reproduce the same construction for an arbitrary set  $A$ , giving:

**Definition 3.31.**  $K_0^A := \{(e, n) : n \in W_e^A\}$ .

This is actually the definition of the Turing jump:

**Definition 3.32.** Given a set  $A$ , the **Turing jump** of  $A$  is defined as:

$$A' := K_0^A = \{(e, n) : n \in W_e^A\}.$$

This definition can be iterated, so the  $(n+1)$ -th Turing jump of  $A$  is defined as:

$$A^{(n+1)} := (A^{(n)})'.$$

**Theorem 3.12.** *Let  $A, B \subseteq \mathbb{N}$ . Then:*

- (i)  $A'$  is  $A$ -r.e.
- (ii) A set  $B$  is  $A$ -r.e. if and only if  $B \leq_m A'$ .
- (iii)  $A \leq_T A'$ , but  $A' \not\leq_T A$ .
- (iv)  $A \equiv_T B \implies A' \equiv_T B'$ .



*Proof.* (i) Assume we have an oracle for  $A$ . Then to answer  $k \in A'$  we just need to:

- a) Compute  $e, n$  such that  $(e, n) = k$  (if they exist),
- b) compute, using the oracle,  $\Phi_e^A(n)$ .

If the computation of  $\Phi_e^A(n)$  ever halts it means  $(e, n) \in W_e^A$ , so  $k \in A'$ .

(ii) Note that  $A' = K_0^A$ . Then:

- If  $B$  is  $A$ -r.e. by lemma 3.10 we have  $B = W_e^A$  for some  $e \in \mathbb{N}$ . Consider the computable function  $f(n) := (e, n)$ . Then

$$n \in B \iff n \in W_e^A \iff (e, n) \in K_0^A \iff f(n) \in K_0^A,$$

so  $B \leq_m A'$ .

- If  $B \leq_m A'$  there must exist a computable function  $f$  such that  $n \in X \iff f(n) \in A'$ . But  $A'$  is  $A$ -r.e., so to answer a question of the form  $n \in B$  we just need to answer  $f(n) \in A'$ , which is semidecidable given an oracle for  $A$ . Hence  $B$  is  $A$ -r.e.

(iii) The first part is a corollary of the previous item, taking  $B = A$  and using that  $m$ -reducibility implies Turing reducibility. For the second part assume  $A' \leq_T A$ . Using a similar argument to that of theorem 3.9, it can be shown that  $K^A := \{n : n \in W_n^A\}$  is  $A$ -r.e. but not  $A$ -computable. But, again, adapting the argument from lemma 3.4 we get  $K^A \leq_m K_0^A = A'$  which we know is  $A$ -r.e. Finally, using that  $m$ -reducibility implies Turing reducibility

$$K^A \leq_T A' \leq_T A,$$

but that would mean  $K^A$  is  $A$ -computable, which is false. Therefore  $A' \not\leq_T A$ .

(iv) Since  $A \equiv_T B$  it follows that  $A \leq_T B$ . By part (i) we know  $A'$  is  $A$ -r.e., and by lemma 3.11 that means it is also  $B$ -r.e. But then by part (ii)  $A' \leq_m B'$ , so  $A' \leq_T B'$ . The reverse inequality follows completely analogously.  $\square$

**Corollary 3.2.** *By part (iv) the Turing jump is well-defined on Turing degrees as  $\mathbf{a}' := \deg(A')$  for some  $A \in \mathbf{a}$ .*

The structure of the Turing universe is really interesting and has been extensively studied, but unfortunately is beyond our scope. For the interested reader we refer to Cooper's "Computability theory" [7] for more details and further in-depth references.

### 3.5 Post's theorem

We have finally arrived at what represents the pinnacle of the connection between computability theory and the arithmetical hierarchy. In order to fully state Post's theorem we just need one additional definition:

**Definition 3.33** (*m-completeness*). Given a collection  $\mathcal{X} \subseteq \mathcal{P}(\mathbb{N})$ , a set  $A \in \mathcal{X}$  is said to be *m-complete* for  $\mathcal{X}$  (or just  *$\mathcal{X}$ -complete*) if  $X \leq_m A$  for all  $X \in \mathcal{X}$ .

We will not prove the following since a rigorous proof would require the formalization of Turing machines in arithmetic, which we have only hinted.

**Theorem 3.13** (Post's theorem). 1. A set  $A$  is  $\Sigma_{n+1}^0$  if and only if  $A$  is r.e. in  $\emptyset^{(n)}$  or, in other words,  $A \leq_m \emptyset^{(n+1)}$ .

2.  $\emptyset^{(n)}$  is  $\Sigma_n^0$  complete for every  $n > 0$ .

However, we can prove this practical corollary:

**Corollary 3.3.** (i) A set  $A$  is r.e. if, and only if, it is  $\Sigma_1^0$  with no set parameters.

(ii) A set  $B$  is co-r.e. if, and only if, it is  $\Pi_1^0$  with no set parameters.

(iii) A set  $C$  is computable if, and only if, it is  $\Delta_1^0$  with no set parameters.

*Proof.* • Assume  $A$  is r.e. Let  $f$  be a function such that  $\text{range}(f) = A$ . Then

$$\forall n (n \in A \iff \exists m (f(m) = n)).$$

Therefore,  $A$  is  $\Sigma_1^0$  with no set parameters.

- Assume now  $A$  is  $\Sigma_1^0$  with no set parameters, so it is given by some  $\Sigma_1^0$  formula  $\varphi(n)$  which is in turn equivalent to  $\exists j \psi(n, j)$  for some  $\Sigma_0^0$  formula  $\psi$ , where both  $\varphi$  and  $\psi$  have no set parameters. Using the Church-Turing thesis, define the following partial computable function

$$f(n) = \begin{cases} 0 & \text{if } \varphi(n), \\ \uparrow & \text{otherwise.} \end{cases}$$

For every  $n$ , we can compute  $f(n)$  by checking in order  $\psi(n, 0), \psi(n, 1), \dots$ . Since  $\psi$  is  $\Sigma_0^0$  it can only contain equations and bounded quantifiers, which clearly are computable. Then it follows

$$\forall n (n \in A \iff f(n) = 0 \iff f(n) \downarrow).$$

But that means  $A$  is the domain of  $f$ , which is a partial computable function. By theorem Theorem 3.8, we have that  $A$  is r.e.

Now (ii) follows because if  $B$  is defined by  $\varphi$  then

$$B \text{ is co-r.e.} \iff \mathbb{N} \setminus B \text{ is r.e.} \iff \neg\varphi \text{ is } \Sigma_1^0 \iff \varphi \text{ is } \Pi_1^0.$$

Similarly for (iii),  $C$  is computable if, and only if, it is both r.e. and co-r.e., if, and only if, it is both  $\Sigma_1^0$  and  $\Pi_1^0$ .  $\square$



# The formal systems

*Some people are always critical of vague statements. I tend rather to be critical of precise statements; they are the only ones which can correctly be labeled wrong.*

RAYMOND S. SMULLYAN

In this chapter we will continue delving into the nuances of  $\text{RCA}_0$  as well as present the formal systems of  $\text{WKL}_0$  and  $\text{ACA}_0$ . We shall give an overlook of the main possible *constructions*, their models or structures, their first order parts, and theorems and *non-theorems* of each of them.

## 4.1 $\text{RCA}_0$

### 4.1.1 The formal system

We reproduce here the Definition 2.10 we made back in Section 2.2 for the sake of completeness:

**Definition 4.1.** The formal system  $\text{RCA}_0$  in the language  $L_2$  consists of the following axioms:

1. the basic axioms of  $Z_2$ ,
2.  $\Sigma_1^0$ -induction, i.e.,  $(\varphi(0) \wedge \forall n(\varphi(n) \implies \varphi(n+1))) \implies \forall n\varphi(n)$  where  $\varphi$  is  $\Sigma_1^0$ , and
3. recursive, or  $\Delta_1^0$ , comprehension, i.e.,  $\exists X \forall n(n \in X \iff \varphi(n))$  where  $\varphi$  is  $\Delta_1^0$  and  $X$  does not occur freely in it

### 4.1.2 Sequences, functions and continuity

Real numbers were defined in the first chapter as sequences of rationals with certain properties, which in turn were defined as functions from  $\mathbb{N}$  to  $\mathbb{Q}$ , but we did not actually define what we meant by function. Recalling the construction of the cartesian product of two sets, we define:

**Definition 4.2.** A function  $f : A \rightarrow B$  is a subset of  $A \times B$  such that, for every  $a \in A$ , there exists exactly one  $b \in B$  verifying  $(a, b) \in f$ . We denote such a  $b$  by  $f(a)$ .

Of course, not every *imaginable* function from  $A$  to  $B$  is actually realizable in  $\text{RCA}_0$ , unless it is already part of the model. For the case of  $\mathbb{N}$  and  $\mathbb{Q}$ , this means that we cannot expect every sequence of rationals to be constructible in  $\text{RCA}_0$  or, in other words, there are **uncomputable reals**. It turns out the only functions that  $\text{RCA}_0$  can prove exist, for every model, are precisely computable functions (albeit not all of them, since the graph of a computable function need not be computable).

Similar to real numbers (as sequences of rational numbers), we can define sequences of real numbers.

**Definition 4.3.** A double sequence of rationals is a function  $q : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Q}$ , denoted  $\{q_{mn} : m \in \mathbb{N}, n \in \mathbb{N}\}$ , where  $q_{mn} := q(m, n)$ .

A sequence of real numbers is a double sequence of rationals such that, for each  $i \in \mathbb{N}$ ,  $\{q_{in} : n \in \mathbb{N}\}$  is a real number. We will often denote this by  $\{x_i : i \in \mathbb{N}\}$ , where each  $x_i$  is to be interpreted as a real number.

Finally, we will need to define continuity.

**Definition 4.4.** A **continuous function** is a set  $\Phi \subseteq \mathbb{N} \times \mathbb{Q} \times \mathbb{Q}^+ \times \mathbb{Q} \times \mathbb{Q}^+$  verifying:

**Compatibility**  $(a, r)\Phi(b, s) \wedge (a, r)\Phi(b', s') \implies |b - b'| \leq s + s'$ ,

**Monotonicity**  $(a, r)\Phi(b, s) \wedge (a', r') < (a, r) \implies (a', r')\Phi(b, s)$ ,

**Inclusion**  $(a, r)\Phi(b, s) \wedge (b, s) < (b', s') \implies (a, r)\Phi(b', s')$ .

where

- $(a, r)\Phi(b, s) \iff \exists n((n, a, r, b, s) \in \Phi)$ ,
- $(a, r) < (a', r') \iff |a - a'| + r < r'$ .

Roughly speaking, we are describing a continuous function by a sequence of pairs of balls centered around rational numbers and with positive rational radii. If we are lucky, this will carry enough information so as to completely determine the value of the function at a given point. Formally:

**Definition 4.5.** Given a continuous function  $\Phi$ , a real number  $x$  is said to belong to the **domain** of  $\Phi$  if

$$\forall \varepsilon \exists (a, r, b, s) (\varepsilon > 0 \implies ((a, r)\Phi(b, s) \wedge (|x - a| < r) \wedge s < \varepsilon)), \quad (4.1)$$

where  $\varepsilon \in \mathbb{Q}$  and we have abbreviated the existence quantifiers for  $a, r, b, s$  in a single one. In case  $x$  is in the domain of  $\Phi$ ,  $\Phi(x)$  is defined as the unique (up to real numbers equality) real number  $y$  such that

$$\forall (a, r, b, s) (((a, r)\Phi(b, s) \wedge |x - a| < r) \implies |y - b| < s). \quad (4.2)$$

*Remark.* Existence of  $\Phi(x)$  when  $x$  belongs to the domain is provable in  $\text{RCA}_0$  using the  $\mu$  operator [2], but we will not provide the details here. Uniqueness follows from the fact that  $\mathbb{Q}$  is densely ordered, since if we assume  $x$  satisfies Equation (4.1) and there exist  $y, y' \in \mathbb{Q} : y \neq y'$  verifying Equation (4.2) we can take  $\varepsilon = \frac{|y - y'|}{2} > 0$  in Equation (4.1). Then we get  $a, r, b, s$  satisfying  $(a, r)\Phi(b, s) \wedge (|x - a| < r) \wedge s < \varepsilon$ , but by Equation (4.2) it follows

$$(|y - b| < s) \wedge (|y' - b| < s) \wedge (s < \varepsilon = \frac{|y - y'|}{2}),$$

quickly reaching a contradiction via the triangular inequality. It is also worth noting that all this construction can be adapted to arbitrary complete separable metric spaces (instead of just  $\mathbb{R}$ ) thanks to the fact that these spaces are the completion of a countable set.

Continuous functions as presented here share some of the properties of the common continuous functions of analysis, but they also lack some of them (e.g. the image of  $[0, 1]$  by a continuous function need not attain a supremum). We will discuss later in this section which results about continuous functions are provable in  $\text{RCA}_0$ .

### 4.1.3 Characterization of $\omega$ -models

In this subsection we turn our attention to the models of  $\text{RCA}_0$ . In particular, we will be concerned with  $\omega$ -**models**, i.e., models of the form

$$(\omega, \mathcal{S}, +_\omega, \cdot_\omega, 0_\omega, 1_\omega, <_\omega)$$

where the operations and elements are fixed, but we allow different collections of subsets of  $\omega$  over which set variables may range. Just as in the intended model for  $L_2$  we had  $\mathcal{S} = \mathcal{P}(\omega)$ , this is certainly not the case for  $\text{RCA}_0$ . The intended model for  $\text{RCA}_0$  should have exactly those sets that are definable via the recursive comprehension axiom. Formally:

**Theorem 4.1** ( $\omega$ -models of  $\text{RCA}_0$ ). *A collection  $\mathcal{S} \subseteq \mathcal{P}(\omega)$  of sets of natural numbers is an  $\omega$ -model of  $\text{RCA}_0$  if, and only if,*

- (i)  $\mathcal{S} \neq \emptyset$ ,
- (ii)  $A \in \mathcal{S}, B \in \mathcal{S} \implies A \oplus B \in \mathcal{S}$ , and
- (iii)  $A \in \mathcal{S} \wedge B \leq_T A \implies B \in \mathcal{S}$ .

*Proof.* We will start by showing these conditions are necessary for all  $\omega$ -models of  $\text{RCA}_0$ .

1. Taking  $\varphi(n) \iff n = n$ , which is obviously  $\Delta_1^0$ , gives us  $\mathbb{N}$ . So  $\mathbb{N} \in \mathcal{S} \implies \mathcal{S} \neq \emptyset$ .
2. Assuming  $A$  and  $B$  are in the model we can use them as parameters of a formula. Namely, let

$$\varphi(n) \iff \exists(m \leq n)((n = 2m \wedge m \in A) \vee (n = 2m + 1 \wedge m \in B))$$

which is a  $\Sigma_1^0 \subset \Sigma_1^0 \cap \Pi_1^0$  formula. It is clear that  $\forall n(n \in A \oplus B \iff \varphi(n))$ , so  $A \oplus B$  is definable in  $\text{RCA}_0$  and so it must be in the model.

3. If  $B \leq_T A$  we know  $B$  is computable in  $A$  or, in other words,  $\Delta_1^0$  with parameter  $A$  (sometimes noted  $\Delta_1^{0,A}$ ), so  $\text{RCA}_0$  can prove  $B$  exists. Therefore, the model must contain it and hence  $B \in \mathcal{S}$ .

For sufficiency, assume a collection  $\mathcal{S}$  with the aforementioned properties. Let  $\varphi$  be a  $\Delta_1^0$  formula. If it uses no parameters it defines a computable set, which is Turing reducible to any other set, so it must be in  $\mathcal{S}$ . If it uses a parameter, the resulting set is computable in the parameter, so for any possible  $X \in \mathcal{S}$  we have that  $\varphi_X := \{n \in \mathbb{N} : \varphi^X(n)\}$  is computable in  $X$ , where  $\varphi^X$  is  $\varphi$  with its parameter interpreted as  $X$ . Therefore,  $\varphi_X \leq_T X$ , so  $\varphi_X \in \mathcal{S}$ . Finally, if  $\varphi$  uses parameters  $A_1, A_2, \dots, A_k$  then  $\varphi^{[A_i:1 \leq i \leq k]}$  is computable in  $A_1 \oplus A_2 \oplus \dots \oplus A_k$ , which is also in  $\mathcal{S}$  by (ii), so  $\varphi^{[A_i:1 \leq i \leq k]} \leq_T A_1 \oplus A_2 \oplus \dots \oplus A_k$  and thus  $\varphi^{[A_i:1 \leq i \leq k]} \in \mathcal{S}$ .  $\square$

**Corollary 4.1.** *There exists a **minimal** model of  $\text{RCA}_0$ ,  $\text{REC}$ , consisting of all the computable subsets of  $\omega$ .*

$$\text{REC} := \{X \subseteq \omega : X \text{ is recursive}\}$$



#### 4.1.4 First order part

The first order part of a given model  $(|M|, \mathcal{S}_M, +_M, \cdot_M, 0_M, 1_M, <_M)$  is just the structure  $(|M|, +_M, \cdot_M, 0_M, 1_M, <_M)$ , that is, removing the collection of subsets of  $|M|$  over which second order formulas are allowed to range, since first order formulas only quantify elements of the model (natural numbers in the case of  $\omega$ -models). Similarly, first order formulas can only have elements of the model  $|M|$  as parameters. We call  $\Sigma_1^0$ -PA the system resulting from taking first order arithmetic (with the usual axioms for arithmetic) plus  $\Sigma_1^0$ -induction, i.e.,

$$(\varphi(0) \wedge \forall n(\varphi(n) \implies \varphi(n+1))) \implies \forall n(\varphi(n)),$$

where  $\varphi$  is any  $\Sigma_1^0$  formula in the language of  $L_1$ . It is therefore clear that all the axioms of  $\Sigma_1^0$ -PA are axioms of  $\text{RCA}_0$ . Conversely, given a model of  $\Sigma_1^0$ -PA,  $(|M|, +_M, \cdot_M, 0_M, 1_M, <_M)$  there exists  $\mathcal{S}_M \subseteq \mathcal{P}(|M|)$  such that

$$(|M|, \mathcal{S}_M, +_M, \cdot_M, 0_M, 1_M, <_M)$$

is a model for  $\text{RCA}_0$ . In particular, we can consider  $\mathcal{S}_M$  to be the collection of subsets of  $|M|$  that are definable by a  $\Delta_1^0$  formula allowing parameters from  $|M|$ . This shows that, given any sentence  $\psi$  in the language of  $L_1$ ,  $\psi$  is provable in  $\Sigma_1^0$ -PA if, and only if, it is provable in  $\text{RCA}_0$ , since by Gödel's completeness theorem

$$\Sigma_1^0\text{-PA} \vdash \psi \iff \Sigma_1^0\text{-PA} \models \psi \iff \text{RCA}_0 \models \psi \iff \text{RCA}_0 \vdash \psi.$$

Therefore, we can conclude the first order part of  $\text{RCA}_0$  is  $\Sigma_1^0$ -PA.

#### 4.1.5 Theorems of $\text{RCA}_0$

In this subsection we will just state without proof some of the theorems of  $\text{RCA}_0$ .

**Theorem 4.2** (Theorems of  $\text{RCA}_0$ ). *The following are provable in  $\text{RCA}_0$ :*

1. *The Baire category theorem,*
2. *the intermediate value theorem,*
3. *Urysohn's lemma for complete separable metric spaces,*
4. *the soundness theorem,*
5. *a weak version of Gödel's completeness theorem,*

6. existence of an algebraic closure of a countable field, and
7. the Banach-Steinhaus uniform boundedness principle.

## 4.2 $\text{WKL}_0$

### 4.2.1 The formal system

#### Previous definitions

For the introduction of the system  $\text{WKL}_0$  we will need some additional definitions that ultimately allow us to present the weak König's lemma. Please note that these definitions are made in  $\text{RCA}_0$ , so that we can later use it to prove the equivalence of  $\text{WKL}_0$  to other theorems within  $\text{RCA}_0$ .

**Definition 4.6.** A set  $X$  is said to be **finite** if  $\exists n \forall i (i \in X \implies i < n)$

**Theorem 4.3.** A finite set can be encoded by a natural number.

*Proof.* We will not give the details of the proof here since it is rather technical, but they can be consulted in [2]. The basic idea is to use Gödel's  $\beta$  function to find suitable  $k, m, n$  such that  $\forall i (i \in X \iff (i < k \wedge (m(i+1)+1) \mid n))$  and thus, in a sense, coding the elements of  $X$  thanks to relative primality. The least number of the form  $(k, (m, n))$  is called the **code** of  $X$ . This, fortunately, is unique in the sense that different sets cannot possibly have the same code, so it is enough to uniquely determine the set it encodes.  $\square$

**Theorem 4.4.** The set of all codes for finite sets,  $\text{Fin}$ , exists in  $\text{RCA}_0$ .

*Proof.*  $\text{Fin}$  is just the set of all triples of the form  $(k, (m, n))$  for  $k, m, n \in \omega$  since, given  $k, m, n$ , they necessarily define a finite set by  $\Delta_1^0$  comprehension

$$\exists X \forall i (i \in X \iff (i < k \wedge (m(i+1)+1) \mid n)).$$

Therefore,

$$\forall x (x \in \text{Fin} \iff \exists (k \leq x) \exists (m \leq x) \exists (n \leq x) (x = (k, (m, n)))).$$

$\square$

**Definition 4.7.** A **finite sequence** of natural numbers is a finite set  $X$  verifying:

1.  $\forall n \exists i \exists j (n \in X \iff n = (i, j))$ ,

2.  $\forall i \forall j \forall k ((i, j) \in X \wedge (i, k) \in X) \implies j = k$ ,
3.  $\exists l \forall i \exists j ((i, j) \in X \iff i < l)$ .

Such an  $l$  is unique and is called the **length** of the sequence. The **code** for a finite sequence is just its code as a finite set.

**Theorem 4.5.** *The set of all codes for finite sequences,  $\text{Seq}$ , exists in  $\text{RCA}_0$ .*

*Proof.* Of course, a code for a finite sequence is also a code for a finite set, so by Theorem 4.4 we can use  $\text{Fin}$  as a parameter to define  $\text{Seq}$ . Now note that, after appropriately renaming the variables in Definition 4.7, we can replace all the quantifiers in the definition by their bounded counterparts, using the  $k$  from the finite set code to bound them. For example, if  $x \in \text{Seq}$  then

$$\exists(k \leq x) \exists(r \leq x) (\forall(n \leq k) \exists(i \leq k) \exists(j \leq k) (x = (k, r) \wedge (n \in X \iff n = (i, j)))).$$

Hence, by bounding all the quantifiers in the three conditions we can conclude that  $\text{Seq}$  exists by  $\Delta_1^0$  comprehension.  $\square$

**Definition 4.8.** Let  $s \in \text{Seq}$  be the code for some finite sequence  $X$ . We define:

- $\text{lh}(s)$  is the **length** of  $s$  or  $X$ , defined as the  $l$  in Definition 4.7.
- For all  $i < \text{lh}(s)$ ,  $s(i)$  is the  $i$ -th element of  $X$ , i.e., the unique  $j$  such that  $(i, j) \in X$ .
- Given  $t$  also in  $\text{Seq}$ , the **concatenation** of  $s$  and  $t$ ,  $s \frown t$ , is the code for the finite sequence

$$\{(0, s(0)), \dots, (\text{lh}(s)-1, s(\text{lh}(s)-1)), (\text{lh}(s), t(0)), \dots, (\text{lh}(s)+\text{lh}(t)-1, t(\text{lh}(t)-1))\}.$$

Therefore,  $\text{lh}(s \frown t) = \text{lh}(s) + \text{lh}(t)$ .

- $t$  is an **initial segment** of  $s$ , written  $t \subset s$ , whenever

$$\text{lh}(t) \leq \text{lh}(s) \wedge \forall(i < \text{lh}(t))(t(i) = s(i)).$$

**Definition 4.9** (Trees). The **full binary tree**, denoted  $2^{<\omega}$ , is the set of all finite sequences whose elements are only 0 or 1:

$$\forall n (n \in 2^{<\omega} \iff \forall(i \leq n)(n \in \text{Seq} \wedge (i < \text{lh}(n) \implies n(i) < 2))).$$

Hence,  $2^{<\omega}$  exists in  $\text{RCA}_0$  by  $\Sigma_0^0$  comprehension.

A **binary tree** is a subset  $T \subseteq 2^{<\omega}$  satisfying:

$$\forall \sigma \forall \tau ((\sigma \in T \wedge \tau \subset \sigma) \implies \tau \in T).$$

A binary tree is said to be **infinite** if it has infinitely many elements. For example, the full binary tree is infinite.

Finally, we define paths:

**Definition 4.10.** A **path** in a given binary tree  $T$  is just one of the (codes of the) finite sequences it contains, i.e., some  $\sigma \in 2^{<\omega} \cap T = T$ .

An **infinite path** of  $T$  is a function  $f : \mathbb{N} \rightarrow \{0, 1\}$  such that all the initial segments of the (infinite) sequence it defines are in  $T$ , i.e.

$$\forall n (\{(0, f(0)), (1, f(1)), \dots, (n, f(n))\} \in T)$$

We are at last in position to state the axioms of WKL<sub>0</sub>

### Axioms

**Definition 4.11.** The formal system of WKL<sub>0</sub> in the language  $L_2$  consists of the axioms of RCA<sub>0</sub> together with the **weak König's lemma**: every infinite binary tree has an infinite path.

*Remark.* Weak König's lemma can actually be expressed by a formula of  $L_2$  (as a scheme with parameter  $T$  an infinite binary tree). As a possible formulation, let

$$\text{Func}(f) \iff \forall i \exists n \forall m ((i, n) \in f \wedge ((i, m) \in f \implies n = m))$$

be a predicate of sets asserting a set is the code of a function. Then the weak König's lemma reads

$$\exists \pi \forall n \exists \sigma (\text{Func}(\pi) \wedge ((n, 0) \in \pi \vee (n, 1) \in \pi) \wedge \sigma \in T \wedge \text{lh}(\sigma) = n+1 \wedge \forall (k \leq n) (\sigma(k) = \pi(k))).$$

That is, there exists a function  $\pi : \mathbb{N} \rightarrow \{0, 1\}$  such that all of its initial segments are in  $T$ . By definition,  $\pi$  is an infinite path of  $T$ .

#### 4.2.2 $\omega$ -models

A natural question is whether WKL<sub>0</sub> is actually more powerful than RCA<sub>0</sub> or, in other words, does weak König's lemma follow from RCA<sub>0</sub>? To answer this question we will actually show that not all models of RCA<sub>0</sub> are models

of  $WKL_0$ . Assuming  $RCA_0$  is **sound**, i.e., it cannot prove a formula that is not true on every model of  $RCA_0$ , it will follow that  $WKL_0$  is actually stronger than  $RCA_0$ .

In order to do this, we first need to define recursive separability:

**Definition 4.12.** Given two disjoint sets  $A, B$ , a **separating set** is some  $C \subseteq \omega$  such that

$$\forall n((n \in A \implies n \in C) \wedge (n \in B \implies n \notin C)).$$

If no such computable  $C$  exists, we say  $A$  and  $B$  are **recursively inseparable**.

**Lemma 4.1** (Recursively inseparable sets). *The sets*

$$A = \{k : \varphi_k(k) = 0\},$$

$$B = \{k : \varphi_k(k) = 1\},$$

where  $\varphi_k$  is the  $k^{\text{th}}$  partial computable function, are recursively inseparable.

*Proof.* Assume they are recursively separable, i.e., there is some recursive set  $C$  that separates both sets. Since  $C$  is recursive  $\chi_C$  is total computable, so  $\chi_C = \varphi_i$  for some  $i \in \omega$ , so

$$i \in A \implies i \in C \iff \chi_C(i) = 1 \iff \varphi_i(i) = 1 \iff i \in B$$

and, similarly,

$$i \in B \implies i \notin C \iff \chi_C(i) = 0 \iff \varphi_i(i) = 0 \iff i \in A.$$

But both of them lead to contradiction since  $A$  and  $B$  are obviously disjoint, so  $C$  cannot be recursive and thus  $A$  and  $B$  are recursively inseparable.  $\square$

Now for the counterexample:

**Theorem 4.6** (Computable tree with no computable infinite path). *There exists an infinite binary tree  $T$  whose vertices constitute a computable set but whose infinite paths are all non-computable.*

*Proof.* We construct  $T$  by stages. Letting  $A$  and  $B$  be as in Lemma 4.1, it should be clear that both of them are r.e., so we can enumerate them also by stages. We may safely assume that the enumerations of  $A$  and  $B$  at any given stage  $n$  list no numbers larger than  $n$ . The idea then is to use the result of

the enumerations by stage  $n$  to computably decide which vertices of the full binary tree at level  $n$  we should include in  $T$ . For example, say by stage six we have found  $1, 2, 3 \in A$  and  $5 \in B$ . Then every sequence of the form  $111 * 0^*$  should be put in  $T$ . Note that when we include new sequences in  $T$  that determine level  $n$ , all of their initial segments have already been included in  $T$  in previous stages, since they separated  $A$  and  $B$  up until that point in time. Therefore  $T$  is a binary tree. Also,  $T$  is computable for we can decide whether a given finite sequence  $\sigma \in T$  by running the computation of  $T$  for  $\text{lh}(\sigma)$  stages.

Now let  $C$  be any separating set of  $A$  and  $B$ , which we already know cannot be recursive. We can identify  $C$  with  $\chi_C$  and  $\chi_C$  with an infinite path of the full binary tree. Note that  $C$  being non-computable means that  $\chi_C$  will not be computable either, so by  $\chi_C$  we mean here the non-computable, total version of the indicator function of  $C$ . Finally, the initial segments of  $\chi_C$  separate  $A$  and  $B$  up until stage  $n$ , so they are all in  $T$ . Therefore,  $\chi_C$  is an infinite path of  $T$ .

Conversely, assume a given infinite path  $\pi$  of  $T$  as in Section 4.2.1. If  $\pi$  does not separate  $A$  and  $B$ , there exists some  $n$  such that

$$(\pi(n) = 0 \wedge n \in A) \vee (\pi(n) = 1 \wedge n \in B).$$

In either case, since  $n$  is in  $A$  or in  $B$  it will appear on the enumeration of the corresponding set. When that happens, say at stage  $m \geq n$ , no finite sequence of length greater than  $m$  that is an initial segment of  $\pi$  will be in  $T$ , since by that point it will be known that  $\pi$  fails to separate  $A$  and  $B$ . Thus,  $\pi$  cannot be an infinite path of  $T$  unless it separates  $A$  and  $B$ .

In sum, the infinite paths of  $T$  are precisely those that separate  $A$  and  $B$ , which are known to be recursively inseparable, so the infinite paths of  $T$  themselves must be non-computable.  $\square$

If we recall Corollary 4.1,  $\text{RCA}_0$  has a minimal model  $\text{REC}$  consisting of only the computable subsets of  $\omega$ . By Theorem 4.6, the tree  $T$  is computable so  $T \in \text{REC}$ . However, the infinite paths of  $T$  are non-computable so they do not exist in  $\text{REC}$ . Therefore, if we reason within  $\text{RCA}_0$ ,  $T$  has no infinite path even though  $T$  itself is an infinite binary tree, so the weak König's lemma cannot possibly be proven in  $\text{RCA}_0$  since it fails in the model  $\text{REC}$ .

### 4.2.3 First order part

We have shown how weak König's lemma can be expressed as a second order formula, i.e., one that quantifies over sets. First order theories, however, are

not capable of doing so. Intuitively, weak König's lemma is about sequences of elements of a given binary tree, stating that there exists an infinite such sequence with certain properties. However, first order arithmetic cannot even define trees since it has no sets. Since the formal system of  $\text{WKL}_0$  is just  $\text{RCA}_0$  plus weak König's lemma, and the latter is not expressible as a first order formula, it follows that the first order part of  $\text{WKL}_0$  is the same as that of  $\text{RCA}_0$ , viz.  $\Sigma_1^0\text{-PA}$ .

#### 4.2.4 Theorems of $\text{WKL}_0$

In this subsection we will just state without proof some of the theorems of  $\text{WKL}_0$ .

**Theorem 4.7** (Theorems of  $\text{WKL}_0$ ). *The following are provable in  $\text{WKL}_0$ :*

1. *The sequential Heine-Borel covering lemma: every covering of the closed interval  $[0, 1]$  by a sequence of open intervals has a finite subcovering,*
2. *every continuous real-valued function on a compact metric space is bounded,*
3. *every continuous real-valued function on a compact metric space is uniformly continuous,*
4. *every continuous real-valued function on  $[0, 1]$  is Riemann integrable,*
5. *the Weierstraß theorem: every continuous real-value function on a compact metric space attains a maximum,*
6. *the Peano existence theorem for solutions of differential equations,*
7. *Gödel's completeness theorem,*
8. *existence of prime ideals in countable commutative rings,*
9. *every countable field of characteristic 0 has a unique algebraic closure,*
10. *Brouwer's fixed point theorem, and*
11. *the Hahn-Banach theorem for complete separable metric spaces.*

### 4.3 $\text{ACA}_0$

#### 4.3.1 The formal system

**Definition 4.13.** The formal system of  $\text{ACA}_0$  in the language  $L_2$  consists of the axioms of  $\text{RCA}_0$  together with the second order axiom scheme of induction

$$(0 \in X \wedge \forall n(n \in X \implies n+1 \in X)) \implies \forall n(n \in X),$$

and the **arithmetical comprehension axiom**:

$$\exists X \forall n(n \in X \iff \varphi(n)),$$

where  $\varphi$  is any arithmetical (i.e., it has no set quantifiers) formula of  $L_2$  in which  $X$  does not occur freely.

*Remark.* It turns out that arithmetical comprehension – or  $\Sigma_0^1$  comprehension, as it is sometimes called – is actually equivalent to just  $\Sigma_1^0$  comprehension. Note the swap of the subindex and the superindex:  $\Sigma_0^1$  is, roughly speaking,  $\bigcup_n \Sigma_n^0$ , i.e., all of the arithmetical formulas/sets. A proof of this equivalence can be found in [2].

#### 4.3.2 Characterization of $\omega$ -models

Just as we did for the case of  $\text{RCA}_0$ , we will give here a characterization of the  $\omega$ -models of  $\text{ACA}_0$  in terms of computability theory.

**Theorem 4.8** ( $\omega$ -models of  $\text{ACA}_0$ ). *A collection  $\mathcal{S} \subseteq \mathcal{P}(\omega)$  of sets of natural numbers is an  $\omega$ -model of  $\text{ACA}_0$  if, and only if,*

- (i)  $\mathcal{S} \neq \emptyset$ ,
- (ii)  $A \in \mathcal{S}, B \in \mathcal{S} \implies A \oplus B \in \mathcal{S}$ ,
- (iii)  $A \in \mathcal{S} \wedge B \leq_T A \implies B \in \mathcal{S}$ , and
- (iv)  $A \in \mathcal{S} \implies A' \in \mathcal{S}$ .

*Proof.* The proof is similar to that of  $\text{RCA}_0$  for the first three items. Turing jump closure is necessary because  $A'$  is  $A$ -r.e., so it is definable by a  $\Sigma_1^0$  formula with parameter  $A$ .

Conversely, assume  $\varphi$  is  $\Sigma_1^0$ , so it defines a set  $X$  from parameters  $A_1, A_2, \dots, A_n$ . Let  $A = A_1 \oplus A_2 \oplus \dots \oplus A_n$ , then  $X$  is  $A$ -r.e. By Theorem 3.12, this means  $X \leq_m A'$ , so  $X \leq_T A'$  and therefore  $X \in \mathcal{S}$ . Since  $\Sigma_1^0$  comprehension implies



arithmetical comprehension by Section 4.3.1, it is enough to prove this for  $\Sigma_1^0$  formulas.  $\square$

**Corollary 4.2.** *There exists a minimal model of  $\text{ACA}_0$ ,  $\text{ARITH}$ , consisting of all the arithmetical subsets of  $\omega$ .*

$$\text{ARITH} := \{X \subseteq \omega : \exists n(X \leq_T \emptyset^{(n)})\}$$

In fact, by closer inspection of the models of  $\text{ACA}_0$  and  $\text{WKL}_0$ , we can actually show that  $\text{ACA}_0$  proves the consistency of  $\text{WKL}_0$ . By Gödel's incompleteness theorem, this means that  $\text{ACA}_0$  must be strictly stronger than  $\text{WKL}_0$ . The interested reader can check the details of the proof in [2].

### 4.3.3 First order part

In this case, the first order part of  $\text{ACA}_0$  is the Peano arithmetic, or PA, which in turn is a friendly name for  $Z_1$ . To see why, first note that all of the axioms of  $Z_1$  are either axioms or theorems of  $\text{ACA}_0$ , since the first order axiom scheme of induction of  $Z_1$  follows from the arithmetical comprehension axiom and the second order axiom scheme of induction. Therefore if any sentence in the language of  $L_1, \sigma$ , is a theorem of  $Z_1$  then it is also a theorem of  $\text{ACA}_0$ .

Conversely, consider a model  $(|M|, +_M, \cdot_M, 0_M, 1_M, <_M)$  of  $Z_1$ . Then there exists  $\mathcal{S}_M \subseteq \mathcal{P}(|M|)$  such that

$$(|M|, \mathcal{S}_M, +_M, \cdot_M, 0_M, 1_M, <_M)$$

is a model of  $\text{ACA}_0$ . In particular, we can consider  $\mathcal{S}_M$  to be the collection of sets definable in the  $Z_1$  model using parameters from  $|M|$ . This shows that, given any sentence  $\psi$  in the language of  $L_1$ ,  $\psi$  is provable in PA if, and only if, it is provable in  $\text{ACA}_0$ , since by Gödel's completeness theorem

$$PA \vdash \psi \iff PA \models \psi \iff \text{ACA}_0 \models \psi \iff \text{ACA}_0 \vdash \psi.$$

Therefore, we can conclude the first order part of  $\text{ACA}_0$  is PA.

### 4.3.4 Theorems of $\text{ACA}_0$

In this subsection we will just state without proof some of the theorems of  $\text{ACA}_0$ .

**Theorem 4.9** (Theorems of  $\text{ACA}_0$ ). *The following are provable in  $\text{ACA}_0$ :*

1. *the full König lemma, which applies to infinite but finitely branching trees,*

2. *the Bolzano-Weierstraß theorem,*
3. *the sequential least upper bound property,*
4. *the monotone convergence theorem, and*
5. *the Cauchy convergence criterion.*

# 5

## Some equivalence results over $\text{RCA}_0$

---

*[...] quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus*

GOTTFRIED W. VON LEIBNIZ

In the present chapter we will cover some equivalences that can be proven in  $\text{RCA}_0$  between some of the theorems mentioned in the previous chapter and the three axiomatic systems we have introduced. In particular, we will show:

- $\text{RCA}_0$  can prove the intermediate value theorem,
- $\text{WKL}_0$  is equivalent to the (sequential) Heine-Borel theorem, and
- $\text{ACA}_0$  is equivalent to the Bolzano-Weierstraß theorem, sequential least upper bound property, the monotone convergence theorem and the Cauchy convergence criterion over  $\text{RCA}_0$ .

### 5.1 $\text{RCA}_0$

#### 5.1.1 The intermediate value theorem

Using Definition 4.4 for continuous functions, we state:

**Theorem 5.1.** *Let  $\phi : [0, 1] \rightarrow \mathbb{R}$  be a continuous function. If  $\phi(0) < 0$  and  $\phi(1) > 0$ , then there exists some  $x$  such that  $0 < x < 1$  and  $\phi(x) = 0$ .*

*Proof.* If  $\phi(q) = 0$  for some  $q \in \mathbb{Q} \cap [0, 1]$  we are done, since  $q$  obviously exists and satisfies the given conditions. Otherwise, we can assume  $\phi$  does not vanish on any rational point in the unit interval. Let

$$\varphi(q) \stackrel{\Delta}{\iff} \exists a \exists r \exists b \exists s ((a, r) \Phi(b, s) \wedge |q - a| < r \wedge b + s < 0),$$

and

$$\psi(q) \stackrel{\Delta}{\iff} \forall a \forall r \forall b \forall s ((a, r) \Phi(b, s) \wedge |q - a| < r) \implies b - s < 0).$$

Assuming  $\phi$  is defined at  $q$ , it is clear that  $\varphi(q) \implies \phi(q) < 0 \implies \psi(q)$ . For the reverse implications, we study first the necessity of  $\varphi$ , that is,  $\phi(q) < 0 \implies \varphi(q)$ . Assume  $\neg\varphi(q)$ , that is

$$\forall a \forall r \forall b \forall s (\neg((a, r) \Phi(b, s)) \vee |q - a| \geq r \vee b + s \geq 0),$$

which is logically equivalent to

$$\forall a \forall r \forall b \forall s (((a, r) \Phi(b, s)) \wedge |q - a| < r) \implies b + s \geq 0).$$

Since  $q$  belongs to the domain of  $\phi$  the antecedent must be true for some  $a, r, b, s$ . In that case,  $b + s \geq 0$ . In fact, we can choose  $a, r, b, s$  such that  $|\phi(q) - b| < s$  and  $s < \varepsilon$  for each  $\varepsilon \in \mathbb{Q}^+$ . Assume also  $\phi(q) < 0$ . Then we can take some positive rational number  $\varepsilon < \frac{|\phi(q)|}{2}$  to obtain

$$b + s = b - s + 2s < \phi(q) + 2s < \phi(q) + 2\varepsilon < \phi(q) + |\phi(q)| = 0.$$

So we have reached a contradiction, and thus our assumptions must be false, i.e.,

$$\neg(\neg\varphi(q) \wedge \phi(q) < 0) \iff (\varphi(q) \vee \neg(\phi(q) < 0)) \iff (\phi(q) < 0 \implies \varphi(q)).$$

To show  $\psi$  is sufficient we proceed in a very similar manner. Assume  $\psi(q)$  and  $\phi(q) > 0$  (since we have ruled out the possibility  $\phi(q) = 0$ ). Take some positive rational  $\varepsilon < \frac{\phi(q)}{2}$ . It follows that

$$b - s = b + s - 2s > \phi(q) - 2\varepsilon > \phi(q) - \phi(q) = 0.$$

Again, we have reached a contradiction, so  $\psi(q) \implies \phi(q) < 0$ . Combining these facts, it is clear that

$$\varphi(q) \iff \phi(q) < 0 \iff \psi(q),$$

where  $\varphi(q)$  is  $\Sigma_1^0$  and  $\psi(q)$  is  $\Pi_1^0$ . By  $\Delta_1^0$  comprehension, there exists  $\phi^-$ , the set of all rationals in  $[0, 1]$  where  $\phi$  is negative.

By primitive recursion using parameter  $\phi^-$ , we define the following sequences:

$$a_0 = 0, b_0 = 1,$$

$$a_n = \begin{cases} c_n & \text{if } c_n \in \phi^-, \\ a_n & \text{if } c_n \notin \phi^- \end{cases}$$

$$b_n = \begin{cases} b_n & \text{if } c_n \in \phi^-, \\ c_n & \text{if } c_n \notin \phi^- \end{cases}$$

where  $c_n = \frac{a_n + b_n}{2}$ . So, the idea is to check at each step whether the image of the midpoint of the current  $a_n, b_n$  is negative. We can do this since  $c_n \in \mathbb{Q}$  and we have shown  $\phi^-$  exists in  $\text{RCA}_0$ . It is straightforward to check that

- $\forall n(\phi(a_n) < 0 < \phi(b_n))$ , since we are choosing  $a_n$  and  $b_n$  to satisfy that condition,
- $\forall n(|b_n - a_n| = 2^{-n})$  since we are halving the length of the interval at each step.

By  $\Sigma_1^0$ -induction, we have:

- $|a_0 - a_1| \leq 2^{-1}$  because  $a_1$  is either 0 or  $\frac{1}{2}$ ,
- Assume  $|a_n - a_{n+1}| \leq 2^{-n-1}$ . If  $a_{n+1} = a_{n+2}$  the inequality holds. Otherwise,

$$|a_{n+1} - a_{n+2}| = \left| a_{n+1} - \frac{a_{n+1} + b_{n+1}}{2} \right| = \left| \frac{b_{n+1} - a_{n+1}}{2} \right| = 2^{-n-2},$$

so  $\forall n(|a_n - a_{n+1}| \leq 2^{-n-1})$ . Finally,

$$|a_n - a_{n+k}| = \left| \sum_{i=0}^{k-1} a_{n+i+1} - a_{n+i} \right| \leq \sum_{i=0}^{k-1} |a_{n+i+1} - a_{n+i}| \leq \sum_{i=0}^{k-1} 2^{-n-i-1} < 2^{-n},$$

and similarly for  $b_n$ . Thus we can conclude that both of them represent a real number and in fact they represent the same real number, since  $|a_n - b_n| = 2^{-n} < 2^{-n+1}$ . Also, it is obvious that  $0 \leq a_n < b_n \leq 1$ . Call  $x$  the real number they represent, so  $0 \leq x \leq 1$ , and suppose  $\phi(x) \neq 0$ , say  $\phi(x) < 0$ . Let  $(u, r)\Phi(v, s)$  be such that  $|x - u| < r$  and  $s < \frac{|\phi(x)|}{2}$ . Following the same argument as in our proof of necessity of  $\varphi(q)$ , we get  $v + s < 0$ . Now since  $b_n$  converges to  $x$ , we can choose a sufficiently large  $n$  such that  $|b_n - u| < r$ . But then  $\phi(b_n) < v + s < 0$ , which contradicts our construction of  $b_n$ . Analogously, we can show that  $\neg(\phi(x) > 0)$ , so the only remaining possibility is  $\phi(x) = 0$ . Since  $0 \leq x \leq 1$  but  $\phi(0) \neq 0 \neq \phi(1)$ , the last requirement of  $0 < x < 1$  is also fulfilled.  $\square$

## 5.2 WKL<sub>0</sub>

### 5.2.1 Sequential Heine-Borel

We will subdivide the prove of the equivalence in two different theorems, one showing that weak König's lemma is sufficient and one showing it is necessary.

**Theorem 5.2** (Weak König's lemma proves Heine-Borel theorem). *Given two sequences of real numbers,  $c_i, d_i$  such that*

$$\forall x(0 \leq x \leq 1 \implies \exists i(c_i < x < d_i)),$$

*then*

$$\exists n \forall x(0 \leq x \leq 1 \implies \exists(i < n)(c_i < x < d_i)).$$

*Proof.* We will assume that  $c_i$  and  $d_i$  are sequences of rational numbers for the moment. For each  $s \in 2^{<\omega}$ , let

$$a_s = \sum_{i=0}^{\text{lh}(s)-1} \frac{s(i)}{2^{i+1}},$$

$$b_s = a_s + 2^{-\text{lh}(s)}.$$

Intuitively, we can interpret any finite binary sequence  $s \in 2^{<\omega}$  as a binary number  $a_s$ . Hence,  $[a_s, b_s]$  represents a subinterval of the form  $[\frac{m}{2^l}, \frac{m+1}{2^l}]$ , where  $l$  is the length of  $s$ . Keeping this in mind, we build a binary tree  $T$  where

$$s \in T \iff \neg \exists(i \leq \text{lh}(s))(c_i < a_s < b_s < d_i).$$

In other words, we include a vertex of the full binary tree at level  $n$  in  $T$  whenever the interval it represents is not fully covered by some open interval  $(c_i, d_i)$  with  $i \leq n$ . Note that  $T$  exists by  $\Sigma_0^0$  comprehension.

Now imagine there is an infinite path in  $T$  that leads to some  $0 \leq x \leq 1$ . That would mean that, no matter how far we look in the sequences, the unique subinterval of length  $2^{-n}$  of the form  $[\frac{m}{2^n}, \frac{m+1}{2^n}]$  to which  $x$  belongs is not completely covered by any  $[c_i, d_i]$  for  $i \leq n$ . That means  $x$  will not be covered by any  $[c_i, d_i]$  with arbitrary  $i$ . Indeed, let  $\pi : \mathbb{N} \rightarrow \{0, 1\}$  be a path in  $T$ , and let

$$x := \sum_{k=0}^{\infty} \frac{\pi(k)}{2^{k+1}}.$$

If we denote by  $\pi[n]$  the initial segment  $\{(0, \pi(0)), \dots, (n, \pi(n))\}$ , it is clear that  $a_{\pi[n]} \leq x \leq b_{\pi[n]}$ . Now since we assumed the open intervals  $(c_i, d_i)$  cover  $[0, 1]$ , let  $i$  be such that  $c_i < x < d_i$ . We can find  $n$  as large as necessary so that  $n > i$  and  $2^{-n} < d_i - c_i$ . Then  $[a_{\pi[n]}, b_{\pi[n]}]$  is an interval of length  $2^{-n-1} < 2^{-n} < d_i - c_i$  that contains  $x$ , so  $c_i < a_{\pi[n]} < b_{\pi[n]} < d_i$ . Then  $\pi[n] \notin T$  because the associated interval is covered, so  $\pi$  is not an infinite path. By weak König's lemma, that means  $T$  is finite.

Now let  $n$  be such that  $\forall s (s \in T \implies \text{lh}(s) < n)$ , i.e., some level after which  $T$  has no more vertices, which must exist because  $T$  is finite. Then

$$\forall s (\text{lh}(s) = n \implies s \notin T \implies \exists (i \leq n) (c_i < a_s < b_s < d_i)).$$

Since all the  $[a_s, b_s]$  for  $\text{lh}(s) = n$  cover  $[0, 1]$  and  $(c_i, d_i)$  cover each  $[a_s, b_s]$ , we conclude that the open intervals  $(c_i, d_i)$  for  $i \leq n$  cover  $[0, 1]$ .

Finally, assume  $c_i$  and  $d_i$  are sequences of real numbers. Consider the  $\Sigma_1^0$  formula

$$\varphi(q, r) \stackrel{\Delta}{\iff} q \in \mathbb{Q} \wedge r \in \mathbb{Q} \wedge \exists i (c_i < q < r < d_i).$$

It should be clear that it holds for an infinite number of pairs  $q, r$ , so it defines an infinite r.e. set (because it is  $\Sigma_1^0$ ) and so there exists a computable function  $f : \mathbb{N} \rightarrow \mathbb{Q} \times \mathbb{Q}$  such that

$$\forall q \forall r (\varphi(q, r) \iff \exists n (f(n) = (q, r)),$$

i.e., the range of  $f$  is the collection of all pairs of rational numbers  $q, r$  such that  $\varphi(q, r)$ . It stands to reason that we can replace  $c_i$  and  $d_i$  by  $q_j$  and  $r_j$ , where  $(q_j, r_j) = f(j)$  and apply the previous proof for rational open intervals.  $\square$

**Theorem 5.3** (Sequential Heine-Borel proves the weak König's lemma). *The sequential Heine-Borel theorem implies the weak König's lemma in  $\text{RCA}_0$ .*

*Proof.* Before proceeding with the actual proof, we give some context and point out the main ideas that will be used. Assuming the sequential Heine-Borel theorem, we will show that a binary tree with no infinite paths must be finite. In order to do this we will rely heavily on the Cantor set,  $C$ . Indeed, we can interpret an infinite path in the full binary tree  $2^\omega$  as a unique point in  $C$ . The idea then is to consider all the intervals that are not in  $C$ , and a few intervals that cover  $C$  that we will choose to suit the given tree. Applying the Heine-Borel theorem, we get a finite collection of intervals that cover  $[0, 1]$ . This will ultimately imply, because of the construction, that the tree is finite.

For starters, consider a finite sequence  $s \in 2^{<\omega}$ . We define

$$a_s = \sum_{i=0}^{\text{lh}(s)-1} \frac{2s(i)}{3^{i+1}},$$

$$b_s = a_s + \frac{1}{3^{\text{lh}(s)}}.$$

These represent the intervals that are not eliminated from  $C$  by stage  $n$ , where  $n = \text{lh}(s)$ . In other words, for any  $n \in \mathbb{N}$ , if we consider all the finite binary sequences  $s$  such that  $\text{lh}(s) = n$  (equivalently, all the paths in the full binary tree to vertices at level  $n$ ), their associated intervals  $[a_s, b_s]$  cover  $C$ . In fact, given some infinite path of the full binary tree  $\pi$ , the path represents the unique  $x \in [0, 1]$  such that  $\forall n (a_{\pi[n]} \leq x \leq b_{\pi[n]})$ . However, we need open intervals in order to apply Heine-Borel, so we define

$$a'_s = a_s - \frac{1}{3^{\text{lh}(s)+1}},$$

$$b'_s = b_s + \frac{1}{3^{\text{lh}(s)+1}}.$$

These intervals still cover  $C$  in the sense described before, but they are open. More importantly, they are disjoint for different paths of the same length. Intuitively, we are extending each  $[a_s, b_s]$  by a third of its length to the left and to the right. So, for example,  $a_{\{(0,0)\}} = 0$ ,  $b_{\{(0,0)\}} = \frac{1}{3}$  is the left interval in the Cantor set after the first step, where  $(\frac{1}{3}, \frac{2}{3})$  is eliminated; whereas  $a_{\{(0,0)\}} = -\frac{1}{9}$ ,  $b_{\{(0,0)\}} = \frac{4}{9}$ . Note that we are using the same notation for open intervals and for the pairing function. We will try to be explicit about what we mean each time, but the reader is free to replace it by the less used notation  $]a, b[$  for open intervals.

Similarly, we define

$$\alpha_s := b_{s \smallfrown \{(0,0)\}} = a_s + \frac{1}{3^{\text{lh}(s)+1}},$$

$$\beta_s := a_{s \smallfrown \{(0,1)\}} = a_s + \frac{2}{3^{\text{lh}(s)+1}}.$$

These represent the endpoints of the subinterval removed from  $[a_s, b_s]$  in the next step. For example,  $\emptyset$  represents the path to the root of the full binary tree, which is the only node at level 0. Then  $(\alpha_\emptyset, \beta_\emptyset) = (\frac{1}{3}, \frac{2}{3})$  is the subinterval removed at step 1.

Assume now we are given some binary tree  $T \subseteq 2^{<\omega}$  with no infinite paths. We call a vertex  $u$  of  $2^{<\omega}$  a **fallen leaf** of  $T$  if

$$u \notin T \wedge \forall t (t \subsetneq u \implies t \in T).$$



Intuitively, the fallen leaves of  $T$  are just the immediate children of the leaves in  $T$ . By the previous formalization, the set  $\widetilde{T}$  of fallen leaves of  $T$  exists by  $\Sigma_0^0$  comprehension (note that we can replace the  $\forall t$  quantifier by  $\forall(t < u)$  since the code for a proper initial segment of a finite sequence is always smaller than the code for the sequence in question. This fact should be self-evident and thus we will not provide a detailed proof of it). Now consider  $x \in C$ , which is given by an infinite path of the full binary tree  $\pi$ . Since  $T$  contains no infinite path, there exists some  $n$  such that  $\pi[n] \notin T$ . However,  $\pi[0] = \emptyset$  certainly is in  $T$ . Consider the first  $m$  such that  $\pi[m] \notin T$ , which exists by minimization ( $\mu$  operator) and is greater than 0. Then  $\pi[m]$  is a fallen leaf of  $T$ , i.e.,  $\pi[m] \in \widetilde{T}$ . Furthermore,  $a_{\pi[m]} \leq x \leq b_{\pi[m]}$ , so  $x$  is indeed covered by the interval associated to some path  $u \in \widetilde{T}$ . In fact, we can replace  $[a_u, b_u]$  by  $(a'_u, b'_u)$  to get an open interval that also contains  $x$ , since  $[a_u, b_u] \subsetneq (a'_u, b'_u)$  for all  $u \in 2^{<\omega}$ . By the arbitrariness of  $x$ , we conclude that the sequence of intervals  $(a'_u, b'_u) : u \in \widetilde{T}$  covers  $C$ . On the other hand, the intervals  $(\alpha_s, \beta_s) : s \in 2^{<\omega}$  cover  $[0, 1] \setminus C$ . We can then merge both sequences to get a single sequence of open intervals that cover  $[0, 1]$ . Applying now the Heine-Borel theorem, it follows that a finite number of them also cover the unit interval.

Since the intervals  $(a'_u, b'_u)$  have been stretched sideways, they intersect non-trivially some of the  $(\alpha_s, \beta_s)$ , so we can get rid of some of those intervals and still cover  $[0, 1]$ . However, none of those intervals  $(\alpha_s, \beta_s)$  contain any point in  $C$ , so we must keep all of the  $(a'_u, b'_u)$  in order to cover  $C$ , and therefore  $[0, 1]$ . It follows then that  $\widetilde{T}$  must be finite. Since  $\widetilde{T}$  gives us the same information as  $T$ , because  $s \in T \iff \exists u(u \in \widetilde{T} \wedge s \subsetneq u)$  it follows that  $T$  itself must be finite, thereby completing the proof.  $\square$

### 5.3 $\text{ACA}_0$

In this section we will prove the equivalence, over  $\text{RCA}_0$ , of a number of theorems of classical analysis and the arithmetical comprehension. We will divide the full proof in subproofs and defer the statement of the full theorem to Theorem 5.10. Note that in order to ease the arguments in the proof, but without loss of generality, we will assume that we can always work with the unit interval  $[0, 1]$ . The generalizations of these results to arbitrary intervals should be straightforward.

#### 5.3.1 Consequences of arithmetical comprehension

**Theorem 5.4.**  *$\text{ACA}_0$  proves the Bolzano-Weierstraß theorem: given a bounded sequence  $\{x_i : i \in \mathbb{N}\}$  of real numbers, there exists a real number  $x$  such that*

1.  $\forall \varepsilon \exists n_0 \forall n ((\varepsilon > 0 \wedge n > n_0) \implies x_n < x + \varepsilon)$ , i.e.,  $x$  is an eventual upper bound of  $x_n$ ,
2.  $\forall \varepsilon \forall n_0 \exists n (\varepsilon > 0 \implies (n > n_0 \wedge |x_n - x| < \varepsilon))$ , i.e.,  $x$  is the least such upper bound.

Such an  $x$  is also usually called the **limit superior** of  $\{x_i : i \in \mathbb{N}\}$ , denoted by

$$\limsup_{i \rightarrow \infty} x_i.$$

Furthermore,  $\{x_i : i \in \mathbb{N}\}$  has a convergent subsequence, i.e., there exists an strictly increasing function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\{x_{g(i)} : i \in \mathbb{N}\}$  is a convergent sequence of real numbers.

*Proof.* Define  $\varphi(i, m) \stackrel{\Delta}{\iff} m < 2^i \wedge \forall n \exists k (k > n \wedge m \cdot 2^{-i} \leq x_k \leq (m+1) \cdot 2^{-i})$ . Intuitively,  $\varphi$  determines whether the interval  $[\frac{m}{2^i}, \frac{m+1}{2^i}]$  contains infinitely many  $x_k$ . Then:

$$\forall n (n \in f \iff \exists i \exists m \forall p ((i, m) = n \wedge \varphi(i, m) \wedge (p > m \implies \neg \varphi(i, m)))).$$

$f$  exists by arithmetical comprehension and is the code for a function that, given  $i$ , computes the largest  $m$  such that  $\varphi(i, m)$ . Consider sequences  $a_i = f(i) \cdot 2^{-i}$ ,  $b_i = a_i + 2^{-i}$ . It is straightforward to check that both  $\{a_i : i \in \mathbb{N}\}$  and  $\{b_i : i \in \mathbb{N}\}$  define real numbers and that they in fact represent the same real number since  $\forall i (|a_i - b_i| = 2^{-i})$ . We claim that  $x = \{a_i : i \in \mathbb{N}\} = \limsup_{i \rightarrow \infty} x_i$ .

1. Fix  $\varepsilon > 0$  and assume the contrary, that is,  $\forall n_0 \exists n (n_0 < n \wedge x_n \geq x + \varepsilon)$ . Consider  $k$  so large that  $\frac{1}{2^k} < \varepsilon$ . Then

$$x + \varepsilon \geq a_k + \varepsilon > a_k + \frac{1}{2^k} = b_k,$$

so

$$\forall n_0 \exists n (n_0 < n \wedge x_n > b_k).$$

In other words, for every  $n_0 > k$  we can find some  $n > n_0 > k$  such that  $x_n > b_k$ . This contradicts the fact that  $[a_k, b_k]$  is the rightmost subinterval of length  $2^{-k}$  that contains infinitely many  $x_i$ , by construction. Therefore, our assumption is false and the assertion is proved.

2. Similarly  $\varepsilon > 0, n_0 > 0$  and assume  $\forall n (n > n_0 \implies |x_n - x| \geq \varepsilon)$ . Consider  $k$  so large that  $k > n_0, \frac{1}{2^{k-1}} < \varepsilon$  and, again, take  $n$  as large as necessary so that  $x_n \in [a_k, b_k]$ . Then

$$|x_n - x| = |x_n - a_k + a_k - x| \leq |x_n - a_k| + |a_k - x| \leq 2^{-k} + |a_k - x| \leq 2^{-k} + 2^{-k} = 2^{-k+1} < \varepsilon,$$

so our assumption is false and thus we conclude

$$\forall \varepsilon \forall n_0 \exists n (\varepsilon > 0 \implies (n > n_0 \wedge |x_n - x| < \varepsilon)).$$

We have successfully proved that  $x = \limsup_{i \rightarrow \infty} x_i$ , but it remains to find a convergent subsequence. Define the following function by minimization:

$$g(0) = 0,$$

$$g(n+1) = \mu m [m > g(n) \wedge |x - x_m| \leq 2^{-n}],$$

which is total because for every  $n$  there are infinitely many  $x_m$  satisfying the condition by the construction of  $x$ . It should now be clear that the sequence  $\{x_{g(i)} : i \in \mathbb{N}\}$  converges to  $x$ .  $\square$

**Theorem 5.5.**  *$\text{ACA}_0$  proves the sequential least upper bound property: given a bounded sequence  $\{x_i : i \in \mathbb{N}\}$  of real numbers, there exists a real number  $x$  such that*

1.  $\forall i (x_i \leq x)$ ,
2.  $\forall \varepsilon \exists i (\varepsilon > 0 \implies x_i + \varepsilon > x)$ .

Such an  $x$  is also usually called the **supremum** of  $\{x_i : i \in \mathbb{N}\}$ , denoted by

$$\sup_{i \rightarrow \infty} x_i.$$

*Proof.* Similar to our previous proof, the idea is to halve the current interval at each step and choose the rightmost one that contains some element of the sequence. The intersection of those intervals (the limit of its endpoints) is the supremum of the sequence.

Define  $\varphi(i, m) \stackrel{\Delta}{\iff} m < 2^i \wedge \exists n (m \cdot 2^{-i} \leq x_n \leq (m+1) \cdot 2^{-i})$ , and the function  $f$  analogous to that in Theorem 5.4, gives us the largest  $f(i) = m$  such that  $\varphi(i, m)$ , i.e., the rightmost subinterval of length  $2^{-i}$  that contains some element of the sequence. Let  $a_i$  and  $b_i$  be the endpoints of those intervals. Then both of them represent the same real number  $x$ . Then

1. Assume that  $x_i > x$  for some  $i \in \mathbb{N}$  and take  $\varepsilon = x_i - x > 0$ . Let  $k$  be so large that  $\frac{1}{2^k} < \varepsilon$ . Then  $a_k \leq x \leq b_k < x_i$ . That contradicts the choice of  $f(k)$ , so our assumption was false. Therefore,  $\forall i (x_i \leq x)$ .
2. Fix  $\varepsilon > 0$  and consider  $k$  so large that  $\frac{1}{2^k} < \varepsilon$ ,  $m > k : a_k \leq x_m \leq b_k$ . Then

$$x_m \geq a_k = b_k - \frac{1}{2^k} > b_k - \varepsilon \geq x - \varepsilon.$$

This completes our proof that  $x$  is indeed the supremum of the sequence.  $\square$

### 5.3.2 Bolzano-Weierstraß proves Cauchy convergence criterion

**Theorem 5.6.**  $\text{RCA}_0$  proves that the Bolzano-Weierstraß theorem implies the Cauchy convergence criterion, i.e., given a sequence  $\{x_i : i \in \mathbb{N}\}$ , if it is Cauchy

$$\forall \varepsilon \exists n_0 \forall n ((\varepsilon > 0 \wedge n > n_0) \implies (|x_{n_0} - x_n| < \varepsilon)),$$

then it is convergent.

*Proof.* Assume the sequence is Cauchy. Take  $\varepsilon = 1$  and let  $n_0$  be such that the condition holds for said  $\varepsilon$ . Then  $\{x_i : i \in \mathbb{N}\}$  is bounded for  $i \leq n_0$  because it is finite, and it is also bounded for  $i > n_0$  because it is Cauchy. Therefore it is bounded, so we can apply the Bolzano-Weierstraß theorem. Let  $g$  be the increasing function such that  $\{x_{g(i)} : i \in \mathbb{N}\}$  is convergent to  $x$  and fix some  $\varepsilon > 0$ . Then:

$$\exists i \forall n (n > i \implies |x_{g(n)} - x| < \frac{\varepsilon}{2},$$

$$\exists n_0 \forall n (n > n_0 \implies |x_n - x_{n_0}| < \frac{\varepsilon}{4}.$$

Take  $m = i + n_0 + 1$  for given  $\varepsilon$ . It follows that, for  $n > m$

$$\begin{aligned} |x - x_n| &= |x - x_{g(n)} + x_{g(n)} - x_n| \leq |x - x_{g(n)}| + |x_{g(n)} - x_n| \\ &< \frac{\varepsilon}{2} + |x_{g(n)} - x_{n_0} + x_{n_0} - x_n| \\ &\leq \frac{\varepsilon}{2} + |x_{g(n)} - x_{n_0}| + |x_{n_0} - x_n| < \frac{\varepsilon}{2} + \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \varepsilon. \end{aligned}$$

Therefore, the original sequence also converges to  $x$ .  $\square$

### 5.3.3 Cauchy convergence criterion proves monotone convergence

**Theorem 5.7.**  *$\text{RCA}_0$  proves that the Cauchy convergence criterion implies the monotone convergence theorem, i.e., any bounded, increasing sequence  $\{x_i : i \in \mathbb{N}\}$  is convergent.*

*Proof.* Assume  $\{x_i : i \in \mathbb{N}\}$  is given with the aforementioned properties, viz.: bounded and increasing. If it is Cauchy then we can use the Cauchy convergence criterion to show that it must converge, so suppose it is not Cauchy. That means

$$\exists \varepsilon \forall n_0 \exists n (\varepsilon > 0 \wedge n > n_0 \wedge |x_n - x_{n_0}| \geq \varepsilon).$$

In fact, since the sequence is increasing we can just write  $x_n - x_{n_0} \geq \varepsilon$ . Now take some such  $\varepsilon > 0$ , set  $n_0 = 0$  and find the corresponding  $n$ , say  $n_1$ , such that  $n_1 > n_0 \wedge x_{n_1} - x_{n_0} \geq \varepsilon$ . Having defined  $n_k$ , we find  $n_{k+1} > n_k$  such that  $x_{n_{k+1}} - x_{n_k} \geq \varepsilon$ . Note that we can in fact choose them so that  $x_{n_{k+1}} - x_{n_k} > \varepsilon$ , because if they did not exist the sequence would already be convergent.

We have then found a sequence  $\{y_i = x_{n_i} : i \in \mathbb{N}\}$ , which is a subsequence of  $\{x_i : i \in \mathbb{N}\}$ , such that  $y_{n+k} - y_n > k\varepsilon$ , which obviously cannot be bounded, so  $\{x_i : i \in \mathbb{N}\}$  itself cannot be bounded and we have reached a contradiction. Therefore,  $\{x_i : i \in \mathbb{N}\}$  must be Cauchy, so it is convergent by the Cauchy convergence criterion and thus the monotone convergence theorem is implied.  $\square$

### 5.3.4 Monotone convergence proves arithmetical comprehension

We will need a previous lemma for this theorem. This was mentioned in Section 4.3.1 but not proved:

**Lemma 5.1.** *The following are equivalent over  $\text{RCA}_0$ :*

- (i) *arithmetical comprehension axiom,*
- (ii)  $\Sigma_1^0$  *comprehension, and*
- (iii) *For any one-to-one function  $f : \mathbb{N} \rightarrow \mathbb{N}$  there exists a set  $R$  such that*

$$\forall n (n \in R \iff \exists m (f(m) = n)),$$

*i.e.,  $R$  is the range of  $f$  and it exists.*

*Proof.* (i)  $\implies$  (ii) *A fortiori.*

(ii)  $\implies$  (iii) *A fortiori.*

(iii)  $\implies$  (ii) Let  $\varphi(n)$  be a  $\Sigma_1^0$  formula. If there exists some finite set  $X$  such that

$$\forall n(n \in X \iff \varphi(n)).$$

Then  $X$  is computable so it exists in  $\text{RCA}_0$ . Otherwise, let  $\varphi(n)$  be equivalent to  $\exists j\psi(n, j)$  for some  $\Sigma_0^0$  formula  $\psi$ . By  $\Sigma_0^0$  comprehension, let  $Y$  be the set of all pairs  $(n, j)$  such that  $\psi(n, j)$  holds, which is infinite. Let  $g : \mathbb{N} \rightarrow Y$  be a function that enumerates  $Y$  in strictly increasing order. Then the range of  $\pi_1 \circ g$ , where  $\pi_1$  denotes the first projection function (i.e.,  $\forall n\forall j(\pi_1((n, j)) = n)$ ), is exactly  $X$ , the set of all  $n$  such that  $\varphi(n)$  holds.

(ii)  $\implies$  (i) Since any arithmetical formula is  $\Sigma_n^0$  for sufficiently large  $n$ , we just need to prove  $\Sigma_1^0$  comprehension implies  $\Sigma_n^0$  comprehension for every  $n$ . We proceed by induction:

1. Obviously,  $\Sigma_1^0$  comprehension implies  $\Sigma_0^0$  comprehension,
2. Assume now  $\Sigma_1^0$  comprehension implies  $\Sigma_k^0$  comprehension for some  $k \in \omega$ . Let  $\varphi(n)$  be some  $\Sigma_{k+1}^0$  formula. Then  $\varphi(n)$  is equivalent to  $\exists j\psi(n, j)$  for some  $\Pi_k^0$  formula  $\psi$ , so  $\neg\psi$  is  $\Sigma_k^0$ . By  $\Sigma_k^0$  comprehension, let  $Y$  be the set of all the pairs  $(n, j)$  such that  $\neg\psi(n, j)$  holds. Then, by  $\Sigma_1^0$  comprehension, let  $X$  be the set of all  $n$  such that  $\exists j((n, j) \notin Y)$ . Then  $n \in X \iff \varphi(n)$  and we have shown  $X$  exists, so  $\Sigma_1^0$  comprehension also implies  $\Sigma_{k+1}^0$  comprehension.

By induction, this mean  $\Sigma_1^0$  comprehension proves  $\Sigma_n^0$  comprehension for every  $n \in \omega$ , so  $\Sigma_1^0$  comprehension proves arithmetical comprehension.

□

We can now prove the theorem:

**Theorem 5.8.**  *$\text{RCA}_0$  proves the monotone convergence theorem implies arithmetical comprehension.*

*Proof.* Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be an injective function. Define the following sequence:

$$c_n = \sum_{i=0}^n 2^{-f(i)}.$$

Note that  $\forall n(0 < c_n < 2 \wedge c_{n+1} > c_n)$ , so  $c_n$  is bounded and strictly increasing. Therefore, by the monotone convergence theorem, there exists a real number  $c$  that is the limit of the sequence. Note that, for all  $k$ ,

$$\exists i(f(i) = k) \iff \forall n(|c_n - c| < 2^{-k} \implies \exists(i \leq n)(f(i) = k),$$

where the left hand side is  $\Sigma_1^0$  and the right hand side is  $\Pi_1^0$ . Therefore, by  $\Delta_1^0$  comprehension,

$$\exists R \forall n(n \in R \iff \exists i(f(i) = n).$$

In other words, the range of any injective function exists, which by Lemma 5.1 implies arithmetical comprehension.  $\square$

### 5.3.5 Sequential least upper bound implies monotone convergence

**Theorem 5.9.**  *$\text{RCA}_0$  proves the sequential least upper bound property implies the monotone convergence theorem.*

*Proof.* Let  $\{x_i : i \in \mathbb{N}\}$  be a bounded, increasing sequence of real numbers. By the sequential least upper bound property, let  $x$  be the supremum of the given sequence. Then:

$$\forall i(x_i \leq x),$$

which implies  $x - x_i = |x - x_i|$  for every  $i$ . Furthermore,

$$\forall \varepsilon \exists i(\varepsilon > 0 \implies x_i > x - \varepsilon),$$

which is equivalent to  $x - x_i < \varepsilon$ . But  $x - x_i = |x - x_i|$ , so

$$\forall \varepsilon \exists i(\varepsilon > 0 \implies |x - x_i| < \varepsilon).$$

Finally, note that since the sequence is increasing,  $|x - x_{i+k}| = x - x_{i+k} \leq x - x_i < \varepsilon$ , so we conclude:

$$\forall \varepsilon \exists n_0 \forall n((\varepsilon > 0 \wedge n > n_0) \implies |x - x_n| < \varepsilon).$$

$\square$

### 5.3.6 Theorems equivalent to $\text{ACA}_0$

We are finally in position to state the complete theorem:

**Theorem 5.10.** *The following are pairwise equivalent over  $\text{RCA}_0$ :*

- (i) *arithmetical comprehension axiom,*
- (ii) *the Bolzano-Weierstraß theorem,*
- (iii) *the Cauchy convergence criterion,*
- (iv) *the monotone convergence theorem, and*
- (v) *the sequential least upper bound property.*

*Proof.* • (i)  $\implies$  (ii) is Theorem 5.4,

- (ii)  $\implies$  (iii) is Theorem 5.6,
- (iii)  $\implies$  (iv) is Theorem 5.7, and
- (iv)  $\implies$  (i) is Theorem 5.8.

On the other hand,

- (i)  $\implies$  (v) is Theorem 5.5,
- (v)  $\implies$  (iv) is Theorem 5.9, and
- (iv)  $\implies$  (i) is Theorem 5.8.

□



# Conclusions and further work

*We must not believe those, who today, with philosophical bearing and deliberative tone, prophesy the fall of culture and accept the ignorabimus. For us there is no ignorabimus, and in my opinion none whatever in natural science. In opposition to the foolish ignorabimus our slogan shall be: Wir müssen wissen — wir werden wissen.*

DAVID HILBERT

In this document we have presented the fundamental ideas about reverse mathematics. We started by defining the language and formal system of second order arithmetic, only to take a step back and present the quintessential subsystem  $\text{RCA}_0$ , the basis for all of our later work. In doing so we provide a powerful framework in which many of the concepts of *ordinary mathematics* can be expressed, such as functions, sequences, real numbers or continuous functions; yet not powerful enough to prove the theorems we are interested in studying. Thus, by considering increasingly stronger axioms, we trusted that not only would those axioms prove some of the most *common* theorems in mathematics, but that, by carefully choosing said axioms, we would be able to show that the theorems under consideration imply them.

Then we turned our attention to computability theory, showing that it bears a strong relation to the subsystem  $\text{RCA}_0$  and, in fact, can successfully encode the main ideas of said system in a seemingly independent environment. We developed enough theory to be able to state the cornerstone theorem that relates both realms: the Post theorem.

Finally, we defined the stronger subsystems  $\text{WKL}_0$  and  $\text{ACA}_0$ , studied some of their properties and went on to proof the claimed equivalences. In particular, we have mainly focused our attention in some of the classical results of analysis, viz.: the intermediate value theorem, the Heine-Borel theorem, the Bolzano-Weierstraß theorem, the least upper bound property for sequences of real numbers, the monotone convergence theorem and the Cauchy convergence criterion. In this fashion, we have showed that each of this theorems *correspond* to some of the three subsystems, in the sense that

we can prove in  $\text{RCA}_0$  that the axiom implies the theorem and the theorem implies the axiom.

It is in this sense that we conclude that reverse mathematics provides us with a powerful set of tools with which to discern the actual strength of a given theorem, by examining its consequences so as to find the *right axiom* from which it can be proved. Also, it is a useful lens through which to look at the problems of the foundations of mathematics.

Of course, this is by no means a complete work and many interesting topics have not been covered. To begin with, and albeit not inherently useful for the matter at hand, the computability theory chapter might have delved deeper in the rich structure of Turing degrees, showcasing some of its most well-known properties. For example, the fact that no meet operation (compatible with the natural join it has) can be defined is fascinating in and of itself. Also, the study of r.e. degrees and the priority method presents itself as a wonderful opportunity to improve our understanding of the more down-to-earth parts of the computability realm. Finally, even though the necessity of classical logic (and thus of the law of excluded middle) for our work renders all attempts at translating these proofs into programs via the Curry-Howard isomorphism inane, we could have used some proof assistant such as Coq to verify the correctness of our results.

On a more mathematical side, we have extensively used the arithmetical hierarchy to fulfill our needs. However, there is no reason to stop there since the hyperarithmetical hierarchy provides a clear path to follow in an effort to generalize the ideas presented here. Furthermore, we have limited ourselves to three of the main systems of reverse mathematics but there is a lot of theory developed for some stronger systems such as  $\text{ATR}_0$  (for *arithmetical transfinite recursion* axiom) or  $\Pi_1^1\text{-CA}_0$ . These have been intentionally left out in order to provide a more condensed and clearer overview of the reverse mathematics field, and because they are related to lesser known mathematical theorems that do not possess the same sort of appeal to general audiences as the ones presented here involving classical analysis. However, they are related with the Ramsey theorems (certain results in combinatorics) and  $\omega$ -incompleteness. More explicitly, it turns out that every Ramsey theorem  $\text{RT}(k)$  for  $k \in \omega$  is provable in  $\text{ACA}_0$ , but the full theorem  $\forall k(\text{RT}(k))$  is not [8], although it is strong enough to prove arithmetical comprehension. Finally, a deeper development of the model theory of these systems as presented in Simpson's treatise [2] would have constituted a nice addition to our study of said systems.

# Bibliography

---

- [1] Harvey Friedman. “Some systems of second order arithmetic and their use”. In: (1975), pp. 235–242.
- [2] Stephen G. Simpson. *Subsystems of Second Order Arithmetic*. 2nd ed. Perspectives in Logic. Cambridge University Press, 2009. DOI: [10 . 1017 / CB09780511581007](https://doi.org/10.1017/CB09780511581007).
- [3] Wilhelm Ackermann. “Zum hilbertschen aufbau der reellen zahlen”. In: *Mathematische Annalen* 99.1 (1928), pp. 118–133.
- [4] Stephen Cole Kleene. “Introduction to metamathematics”. In: (1968).
- [5] John E Hopcroft and Jeffrey D Ullman. “Introduction to Automata Theory, Languages and Computation. Adison-Wesley”. In: *Reading, Mass* (1979).
- [6] Alan M. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: *Proceedings of the London Mathematical Society* 2.42 (1936), pp. 230–265. URL: [http: / /www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf](http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf).
- [7] S Barry Cooper. *Computability theory*. Chapman and Hall/CRC, 2017.
- [8] John Stillwell. *Reverse Mathematics: Proofs from the Inside Out*. Princeton University Press, 2018. ISBN: 9780691177175.