Introduction 1

The mathematical practice is full of meta-mathematical considerations, even at the high school level. It is common to find in textbooks statements such as “the intermediate value theorem is equivalent to the least upper bound property” or “give an elementary proof of Euclid’s theorem”. Every mathematician will be convinced that the use of Fermat’s last theorem to prove the irrationality of $2^{1/n}$ is overly sophisticated, and the very distinction between a theorem and a corollary – which are both mathematically true and logically equivalent statements – is purely meta-mathematical. What does it mean for one theorem to imply another? What are the optimal axioms necessary to prove ordinary theorems? These are all questions that reverse mathematics tries to answer. *Reverse mathematics* is originally a meta-mathematical program started in 1972 by Harvey Friedman, seeking for the optimal axioms to prove ordinary theorems, using subsystems of second-order arithmetic. The appellation took over time a broader meaning, encompassing all the sets of tools from proof theory and computability theory to study theorems from a computational perspective.

Intuitively, a theorem A implies a theorem B, or a statement B is a corollary of a theorem A if one can prove B with only elementary methods, using A as a blackbox. The whole difficulty is to find a robust, theory-agnostic notion of “elementary methods”[1] to formalize this intuition. This is where computability theory comes into play: Thanks to the Church-Turing thesis, there is a consensual and robust formalization of the ontological concept of “effective process”. Furthermore, with the popularization of computers and their integration in everyday’s life, the notion of algorithm started to be part of the common knowledge. Last, but not least, by a theorem of Gödel, there is a correspondence between the computably enumerable sets, and the sets definably by a Σ_1-formula in first-order arithmetic, paving the way to a translation of the computability-theoretic concepts to the proof-theoretic realm. All these considerations make the notion of “computable” a good candidate for the definition of “elementary”.

1: Beware, we make here an important distinction between “elementary proof” and “simple proof”. The former concept should be understood as “logically elementary”, that is, involving only logically weak axioms, while the latter is a more human concept which seems harder to formalize. In particular, one can win a Fields medal by proving theorems requiring only weak axioms.

The formal setting of reverse mathematics is therefore subsystems of second-order arithmetic, that is, theories in a two-sorted language with a set of integers and collection of sets of integers.[2] The base theory, RCA_0, captures “computable mathematics”. Thanks to the correspondence between computability and definability, proofs of implications are often witnessed by a computable procedure, and separation proofs mainly consist in constructing models of RCA_0 satisfying some specific computability-theoretic weakness properties.

2: Hilbert and Bernays used second-order arithmetic as a foundational language to reprove ordinary mathematics. They showed through their book *Grundlagen der Mathematik* that a large part of classical mathematics could be casted in this setting and proven using second-order Peano arithmetic (Z_2).

Since the start of reverse mathematics, many theorems have been studied from the core areas of mathematics, including analysis, algebra, topology, and highlighted two main empirical phenomena. First of all, mathematics seem very structured, that is, most theorems from ordinary mathematics are either computationally trivial, or computably equivalent to one of four subsystems of second-order arithmetic, linearly ordered by the implication. Second, a large part of ordinary mathematics requires very weak axiomatic and computability-theoretic power. As mentioned, these phenomena are empirical observations, and there exist two main areas of mathematics escaping these observations: logics and Ramsey theory. Logics, by essence, is meta-mathematical and contains constructions that are designed to outgrow the usual proof-theoretic strengths. Ramsey theory, on the other hand, has no *a priori* reason to be a counter-example to these phenomena, and its study represents one of the most active branches of modern reverse mathematics.

Beyond the comparison of theorems based on a formal notion of elementary proof, reverse mathematics play an important foundational and philosophical role in mathematics thanks to these empirical observations. Indeed, the second observation yields that mathematics is somewhat robust, in the sense that if some inconsistencies were to be discovered in ZFC, one could safely remove many strong axioms while keeping a large part of mathematics. Moreover, all the finitary consequences of RCA_0 are already provable over primitive recursive arithmetic (PRA), a very weak theory arguably capturing finitary mathematics. From this perspective, reverse mathematics can be seen as a partial realization of Hilbert's program as an answer to the foundational crisis of mathematics [1].

1.1 Mathematical problems

Many theorems from ordinary mathematics can be seen as *mathematical problems*, formulated in terms of *instances* and *solutions*. Consider for example the *intermediate value theorem* (IVT), which states, for every continuous function $f : [0,1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$ or $f(1) < 0 < f(0)$, the existence of a real number $x \in [0,1]$ such that $f(x) = 0$. An instance of IVT is a continuous function $f : [0,1] \to \mathbb{R}$ changing its sign over the interval, and a solution to f is a real number $x \in [0,1]$ such that $f(x) = 0$. What is the axiomatic power needed to prove the intermediate value theorem?

First of all, one needs to cast this theorem in the setting of second-order arithmetic, with an appropriate coding. A real number can be represented as a fast-converging Cauchy sequence of rational numbers, hence as a set of integers. At first sight, a continuous function from $\mathbb{R}$ to $[0,1]$ is a third-order object, but since it is fully specified by its values on the rationals, one can also represent a continuous function in second-order arithmetic. Having fixed the representation, both the frameworks of subsystems of second-order arithmetic and computability theory can be applied to the intermediate value theorem.

Thanks to the choice of the base theory, RCA_0, the proof-theoretic analysis of the intermediate value theorem translates to the following computability-theoretic question: *Given a computable instance of the intermediate value theorem, what is the computational content of a solution?* The classical proof of the intermediate value theorem provides an algorithm to find the solution: a dichotomic search. Following the proof, given a computable instance $f : [0,1] \to \mathbb{R}$, one can define a computable fast-converging Cauchy sequence whose limit is a real number x such that $f(x) = 0$, with one subtlety: the natural order between Cauchy sequences is not decidable. Thankfully, one can circumvent this issue using a case analysis, and show the existence of a computable solution. On the other hand, there is provably no single algorithm which takes a code of such a continuous function as an input, and outputs a solution. From a proof-theoretic perspective, the dichotomic search can be formalized with weak induction assumptions, and the intermediate value theorem is provable over RCA_0.

More generally, the reverse mathematical analysis of a theorem, seen as a mathematical problem, answers two families of problematics:

- The *strength* of the theorem as an individual. What axioms are *necessary* and *sufficient* to prove a theorem? Based on the correspondence between definability and computability, these questions are reformulated in the computability-theoretic language as "What is the *computational strength* of a theorem?" One proves lower bounds by constructing instances such that every solution is computationally strong, and upper bounds by proving that every instance admits some computationally weak solution. Consider for example *König's lemma* (KL), which states that every infinite, finitely branching tree admits an infinite path. By a classical result in computability theory, every computable infinite, finitely branching tree admits an infinite $\emptyset''$-computable path, while there exists a computable infinite, finitely branching tree such that every infinite path computes $\emptyset'$. In the reverse mathematical formalism, this translates into an equivalence between KL and ACA_0 over RCA_0, where ACA_0 is a system capturing the arithmetic hierarchy.
- The *comparison* of two theorems. Does theorem A imply theorem B over RCA_0? Let us compare for example König's lemma, and *Ramsey's theorem for pairs and two colors* (RT^2_2). The latter theorem states the existence, for every graph with infinitely many vertices, of an infinite subset of vertices such that the induced sub-graph is either a clique, or an anti-clique. Given an infinite graph (V, E), one can easily compute an infinite, finitely branching tree such that every infinite path codes for a clique or an anti-clique. Intuitively, König's lemma, seen as a mathematical problem, is at least as hard to solve as Ramsey's theorem for pairs. In reverse mathematics, this construction yields a proof that KL implies RT^2_2 over RCA_0. On the other hand, the reverse implication does not hold: a famous theorem from Seetapun states that Ramsey's theorem for pairs and two colors has no coding power, in the sense that for every computable instance of RT^2_2, if every solution computes a fixed set of integers A, then A is computable. From this, one can build a model of $\mathsf{RCA}_0 + \mathsf{RT}^2_2$ which does not contain the halting set, and therefore is not a model of KL, thus RT^2_2 does not imply KL over RCA_0. Note that, while the implication from KL to RT^2_2 is elementary, the proof of Seetapun's theorem involves some very clever techniques from effective forcing.

As it happens, when a problem P implies another problem Q from a proof-theoretic or computability-theoretic viewpoint, the reduction is most of the time rather short, if not straightforward, while the proofs of separations usually involve elaborate forcing arguments to preserve a computability-theoretic weakness property. Separating problems in reverse mathematics and proving upper bounds was at the origin of many developments in effective forcing, with the design of new notions of forcing and preservations properties, tailored to witness subtle combinatorial differences between problems. This resulted into a coherent whole of what could be now called a *separation theory*.

1.2 Separation theory

In classical reverse mathematics, proving that a problem P does not imply another problem Q over RCA_0 requires to construct a model of $\mathsf{RCA}_0 + \mathsf{P}$ which is not a model of Q. Furthermore, one usually wants to build counter-examples which are as close to the intended model a possible. In the case of second-order arithmetic, structures are of the form $\mathcal{M} = (M, S, <, +, \times, 0, 1)$ where M denotes the integers of the model (the first-order part) and $S \subseteq \mathcal{P}(M)$ represents the sets of integers (the second-order part). Almost all the proofs of separations in reverse mathematics involve models $\mathcal{M}$ where the set M is the true set of integers ω, equipped with the standard operations. These models are called *ω-models*, and are fully specified by their second-order part S. It is convenient to identify an ω-model $\mathcal{M}$ with the set S. To summarize, the goal is to obtain an ω-model of $\mathsf{RCA}_0 + \mathsf{P}$ which is not a model of Q.

Models of RCA_0 are well-understood and easy to construct, thank to the clear computability-theoretic interpretation of the axioms of RCA_0. An ω-model $\mathcal{M}$ with second-order part S satisfies RCA_0 if and only if S is a *Turing ideal*, that is, S is a collection of sets satisfying the following two closure properties: First, if $X \in S$ and X computes a set Y, then $Y \in S$. Second, if X and Y belong to S, then their effective union $X \oplus Y = \{2n : n \in X\} \cup \{2n + 1 : n \in Y\}$ also belongs to S. For instance, the collection of all the computable sets forms a Turing ideal, and more generally, given any fixed set X, the collection $\{Y : Y \leq_T X\}$ is a Turing ideal. Last, a union of an increasing sequence of Turing ideals is again a Turing ideal.

The idea to construct an ω-model of $\mathsf{RCA}_0 + \mathsf{P}$ which is not a model of Q goes as follows: First, construct a computable instance X_Q of Q with no computable solution. The solutions of this instance should be as hard to compute as possible, to simplify the construction of the model $\mathcal{M}$. Let $\mathcal{M}_0$ be the ω-model whose second-order part consists of the computable sets. In particular, $\mathcal{M}_0 \models \mathsf{RCA}_0$ but $\mathcal{M}_0$ does not satisfy Q, as the instance X_Q belongs to $\mathcal{M}_0$, but has no solution in $\mathcal{M}_0$. The problem is that $\mathcal{M}_0$ will usually not satisfy P either.

Given an instance $X_0 \in \mathcal{M}_0$ of P with no solution in $\mathcal{M}_0$, we shall construct a solution Y_0, and and extend $\mathcal{M}_0$ into another model $\mathcal{M}_1$ of RCA_0 containing Y_0. In order to obtain a model of RCA_0, the second-order part $\mathcal{M}_1$ must not only contain Y_0, but all the Y_0-computable sets. The initial model $\mathcal{M}_0$ might contain infinitely many P-instances with no solution in $\mathcal{M}_0$, and when extending $\mathcal{M}_0$ into $\mathcal{M}_1$, one might add even more P-instances. We shall therefore carefully list all these instances, and build an increasing sequence $\mathcal{M}_0 \subsetneq \mathcal{M}_1 \subsetneq \mathcal{M}_2 \subsetneq \dots$ of ω-models of RCA_0, such that every P-instance $X \in \mathcal{M}_n$ has a solution in $\mathcal{M}_m$ for some $m \geq n$. Then, letting $\mathcal{M} = \bigcup_n \mathcal{M}_n$, the second-order part is again a Turing ideal, so $\mathcal{M} \models \mathsf{RCA}_0$, and by construction, $\mathcal{M} \models \mathsf{P}$.

There is an important issue in the previous construction: when extending a model $\mathcal{M}_n$ into a larger model $\mathcal{M}_{n+1}$ containing a solution Y_n to a P-instance X_n, one adds many sets, including the Y_n-computable ones, but also the $Y_n \oplus Z$-computable ones for any $Z \in \mathcal{M}_n$. During this extension process, one might inadvertently add a solution to the Q-instance X_Q, loosing our witness of failure of Q. If one is not careful, the final model $\mathcal{M}$ will also satisfy Q. Thankfully, there is some degree of freedom in the choice of a solution Y_n to a P-instance X_n. With an appropriate construction, if $\mathcal{M}_n$ does not contain any Q-solution to X_Q, one might build a P-solution Y_n to X_n such that $\mathcal{M}_{n+1}$ still does not contain any Q-solution to X_Q.

Not containing a solution to X_Q is usually not the good invariant, and part of the difficulty of a proof of separation consists in finding the appropriate computability-theoretic notion of weakness, such that

- There exists a computable instance X_Q of Q with no weak solution.
- For every weak instance X of P, there exists a weak solution.

Thus, a proof of separation of a problem P from a problem Q in reverse mathematics reduces to proving lower bounds to Q and upper bounds to P for an appropriate computability-theoretic notion specific for P and Q.

1.3 Jump control

There are two main families of constructions of solutions to an instance of a problem P: *effective* constructions and *forcing* constructions, the former being often an effectivization of the latter. Forcing therefore plays a central role in reverse mathematics, and in computability theory in general.

Forcing was originally introduced by Paul Cohen to answer open problems in set theory. The main idea is to start with a *ground model* $\mathcal{M}$, and construct a new mathematical object G by approximating it with a set $\mathbb{P}$ of *conditions*. These conditions are partially ordered by a relation $\leq$, intuitively meaning that $q \leq p$ if q is a more precise approximation of G than p. The resulting object G, combined with the model $\mathcal{M}$, defines an *extended model* $\mathcal{M}[G]$, which may not satisfy the same properties. Surprisingly, complex properties of the extended model can already be decided by conditions, in the sense that there exists a *forcing relation* $\Vdash$ between conditions and properties such that, if $p \Vdash \varphi(G)$, then the property $\varphi(G)$ will hold for every appropriate construction containing p. Moreover, the forcing relation is definable with only parameters in the ground model, and because of this, many properties of the extended model $\mathcal{M}[G]$ are *inherited* from the ground model $\mathcal{M}$. Indeed, thanks to the forcing relation, a formula with parameters in the extended model can be translated into another formula in the ground model.

The forcing technique in the computability-theoretic setting shares many features with the set-theoretic setting, with some notable differences: The comprehension scheme in set theory being over all definable formulas, it is sufficient for the forcing relation to be definable in the ground model, to propagate many properties from the ground model to the extended model. In computability theory, on the other hand, the computational content of definable sets is sensitive to the complexity of the defining formula, and one needs to have a forcing relation which is not only definable, but also preserves the complexity of the formulas it forces, in order to propagate computability-theoretic properties. Unfortunately, except for some simple cases such as Cohen forcing, the notions of forcing considered in computability theory do not admit a forcing relation with the desired definitional properties.

The novelty of this book is the emphasis of a related concept, called *forcing question*, which usually admits better definitional features that the associated forcing relation, and is sufficient to propagate computability-theoretic properties from the ground model to the extended model. This notion is not relevant in set theory, as the axioms are coarse enough to define a trivial forcing question from the forcing relation, but are of central interest in computability theory. We call "forcing question" any relation ?$\vdash$ between a condition p and a formula $\varphi(G)$, such that if p ?$\vdash \varphi(G)$ holds, then there is an extension $q \leq p$ forcing $\varphi(G)$, and it not, then there is an extension $q \leq p$ forcing $\neg\varphi(G)$. A forcing question can be thought of as a completion of the forcing relation, dividing the set of conditions into two categories. Contrary to the forcing relation, there is no canonical forcing question, as any condition which forces neither a formula nor its negation can be put in either category. The whole difficulty is to design a forcing question with the appropriate definitional complexity. As we shall see throughout the book, beyond the definitional complexity of the forcing question, its combinatorial properties have a strong impact on the computability-theoretic features of the constructed object. The nth-fold Turing jump of G being $\Sigma^0_n(G)$-complete, the set of techniques for deciding Σ^0_n-formulas is known as *nth jump control*, and essentially consists in designing a forcing question for Σ^0_n-formulas with the appropriate definitional and combinatorial properties.

Although our main motivation is reverse mathematics, the techniques of iterated jump control have applications in many domains of computability theory and weak arithmetic.

1.4 Audience

This book aims at bridging the gap between the general introductory textbooks on computability theory and reverse mathematics on one hand, and the state-of-the-art research articles in reverse mathematics on the other hand. It is therefore not meant to be read as first intention, and assumes a prior knowledge of computability theory. Some familiarities with reverse mathematics would also be beneficial to the reader to give some motivation, although the basic concepts are re-introduced in Chapter 2.

The primary audience is graduate students in computability theory and researcher from other fields wanting to get familiar with the techniques used in reverse mathematics, but I believe it could also be of interest to some other well-established researchers in computability theory, given the recent identification of the forcing question as a central tool to study the computability-theoretic weakness of a forcing notion.

1.5 Book structure

This monograph is not meant to be read linearly, but each chapter forms almost a monolithic block focusing on one aspect of iterated jump control. Because of this, each chapter starts with a list of dependencies.

- *Chapter 2: Prerequisites* presents computability theory, reverse mathematics and forcing in a nutshell. It should not be considered as a proper introduction to these theories, and mostly fixes notation. This chapter can be safely skipped by any researcher familiar with them.
- *Chapter 3: Cone avoidance* introduces the core idea of forcing question through the simplest notion of avoidance, namely, cone avoidance. Although not technically difficult, this is a conceptually important chapter, as it contains many of the important concepts which will be used throughout the book. The highlight application is Seetapun's theorem, stating that Ramsey's theorem for pairs admits cone avoidance.
- *Chapter 4: Lowness* presents an effective version of first-jump control, enabling to construct sets belonging to the arithmetic hierarchy. Besides the intrinsic interest of classifying sets thanks to their definitional complexity, this chapter contains a proof of the low basis theorem for Π^0_1 classes and defines coded Turing ideals, both important notions for

higher jump control. It also contains a proof of a theorem by Cholak, Jocksuch and Slaman, stating that every computable instance of Ramsey's theorem for pairs admits solutions of low_2 degree.

- *Chapter 5: Compactness avoidance* summarizes the interrelationship between the use of compactness argument in theorems and structural properties of the forcing question. It contains, among others, a proof of Liu's theorem, which says that Ramsey's theorem for pairs does not imply weak König's lemma.
- *Chapter 6: Custom properties* gives some examples of separations between combinatorial theorems with custom preservation properties, when the classical computability-theoretic notions fail to separate them. These separations involve the Erdős-Moser theorem, the ascending descending sequence and the chain anti-chain principles.
- *Chapter 7: Conservation theorems* applies a formalized version of the first-jump control techniques to prove conservation theorems over weak theories of second-order arithmetic. It contains a proof of the isomorphism theorem for weak König's lemma by Fiori-Carones, Kołodziejczyk, Wong and Yokoyama. This chapter can be skipped by anyone interested in purely computability-theoretic results.
- *Chapter 8: Forcing design* is the missing link in the thought process leading to a separation between two combinatorial theorems. It rationalizes the steps to design a notion of forcing with a good first-jump control, through the examples of the Erdős-Moser and the free set theorems. This is an independent chapter which, although quite short, I believe is of great importance for the researcher in reverse mathematics. It can be read after Chapter 3.
- *Chapter 9: Jump cone avoidance* studies the relationships between the forcing question and second-jump control through jump cone avoidance. The non-continuous nature of jump functionals raise many new challenges, and the core concepts introduced are of central importance for the remaining chapters. It contains a proof by Monin and Patey that every instance of the pigeonhole principle admits a solution of non-high degree.
- *Chapter 10: Jump compactness avoidance* is probably the most technical chapter of this book, as it combines the complexity of second-jump control with the techniques of compactness avoidance, which happens to raise many issues. The main theorem of this chapter is a theorem by Monin and Patey that every Δ^0_2 set admits an infinite subset in its or its complement whose jump is not of PA degree over $\emptyset'$.
- *Chapter 11: Higher jump cone avoidance* generalizes first and second jump control to higher levels of the arithmetic and the hyperarithmetic hierarchy. The conceptual difficulty mainly comes from the generalization of computability theory to the transfinite realm, known as higher recursion theory.

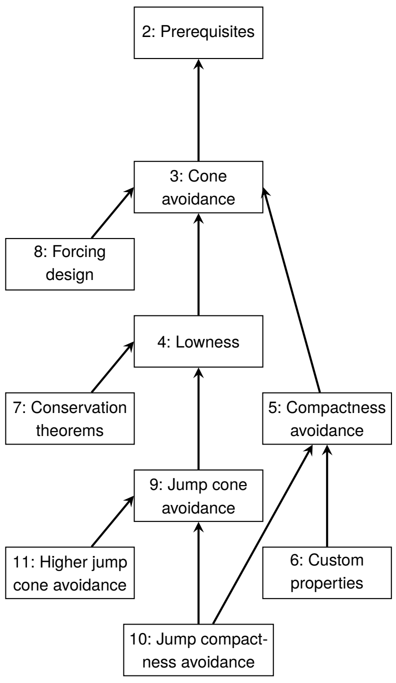

Figure 1.1: Dependencies between the chapters