

COMPUTABILITY

S. Barry Cooper

University of Leeds, Leeds LS2 9JT, U.K.

pmt6sbc@leeds.ac.uk

What is computability? What does it tell us, and why should we be interested in it? Unlike with more obviously practical topics, we do need to ask such questions before talking about technical aspects.

Computability as a field is broadly defined by the work and research interests of Alan Turing, who started out as a mathematician. Its distinctive features are:

- *A multidisciplinary approach to fundamental computational problems facing scientists from many different areas, such as computer science, mathematics, physics, and philosophy.* Turing did seminal work on artificial intelligence, software design, the emergence of natural structures, and connectionist models of computation.
- *A concern with saying something **about** computation as an important everyday activity.* The invention of computational models, such as the Turing machine, inevitably revealed that many very natural problems were not practically solvable, and even unsolvable in principle.
- *A close historical and continuing involvement with the development of computing, going back over sixty years.* Although computers came out of experimental and ad hoc work, theory and the ‘big picture’ have played a hugely important role in taking computing techniques to higher levels — just as consciousness marks human activity apart from that of the rest of the animal kingdom.

- *The development and investigation of deep and mathematically interesting computational structures, which tell us more about what can and cannot be computed.* Turing's notion of an oracle machine provides a natural model of computationally complex environments, structuring information, and hence the real world, in remarkably informative ways. But more of this later.

So it is only in the last century that computability became both a driving force in our daily lives, and a concept one could talk about with any sort of precision. Computability as a *theory* is a specifically twentieth century development, and so of course is the computer, and this is no coincidence. But this contemporary awareness and understanding of the computational content of everyday life has its roots in a rich history.

INTRODUCTION

We can see now that the world changed in 1936, in a way quite unrelated to the newspaper headlines of that year, concerned with such things as the civil war in Spain, economic recession, and the Berlin Olympics. The end of that year saw the publication of a thirty-six page paper by a young mathematician, Alan Turing, claiming to solve a longstanding problem of the distinguished German mathematician David Hilbert. A byproduct of that solution was the first machine-based model of what it means for a number-theoretic function to be computable, and the description of what we now call a *Universal Turing Machine*. At a practical level, as Martin Davis describes in his 2001 book *Engines of Logic: Mathematicians and the Origin of the Computer*, the logic underlying such work became closely connected with the later development of real-life computers. The stored program computer on ones desk is a descendant of that first universal machine. What is less often remembered is Turing's theoretical contribution to the understanding of the *limitations* on what computers can do. There are quite easily described arithmetical functions which are not computable by *any* computer, however powerful. And even the advent of quantum computers will not change this.

Before computers, computer programs used to be called *algorithms*. Algorithms were just a finite set of rules, expressed in everyday language, for performing some general task. What is special about an algorithm is that its rules can be applied in a potentially unlimited number instances of a particular situation. We talk about the *algorithmic content* of Nature when we recognise patterns in natural phenomena which appear to follow general rules. One of the main tasks of science, at least since the time of Isaac Newton, is to make mathematically explicit the algorithmic content of the world about us. A more recent task is to come to terms with, and analyse, the theoretical obstacles to the scientific approach. This is where the discovery of *incomputability*, and the theory which flows from that discovery, play such an important role.

Of course, algorithms have played an explicit role in human affairs since very early times. In mathematics, the first recorded algorithm is that of Euclid, for finding the greatest common factor of two integers, from around 300 BC. The word ‘algorithm’ is derived from the name of the mathematician al-Khwarizmi, who worked at the court of Mamun in Baghdad around the early part of the 9th century AD.

One should mention two modern age contributors to the pre-history of computability before Hilbert’s decisive intervention. The famous German philosopher and mathematician Gottfried Leibniz (1646–1716) believed that the principles of reasoning could be reduced to a formal symbolic system, an algebra or calculus of thought, in which controversy would be settled by calculations. His main achievement here was to produce a machine that did not just do additions and subtractions (as did Pascal’s calculating machine), but one which was also able to perform multiplications and divisions.

But ambitious as was Leibniz’s dream of reducing all reasoning to calculation, it was Charles Babbage (born December 26, 1791 in Teignmouth, Devon) who, along with Turing, has become described as the “Father of Computing”. His Difference Engine No.1, completed in 1832, was invented to compile mathematical tables and remains one of the finest examples of precision engineering of the time. What was so new was that the Analytical Engine was intended to perform not just one mathematical task but

any kind of calculation. An example of this versatility was provided by his devoted follower Ada Byron (daughter of the poet), Lady Lovelace who in describing how the engine might calculate Bernoulli numbers, is credited with the first “computer program”.

But it was David Hilbert’s famous address to the 1900 International Congress of Mathematicians in Paris, which set out a mathematical agenda which turned out to be of fundamental importance to the history of computability. Many of the twenty-three problems he posed have been solved. But a main theme running through them still preoccupies us, although in ways very different to what Hilbert would have expected. Essentially, Hilbert hoped to reduce a wide spectrum of familiar problems to computation. For instance, Hilbert’s tenth problem, asked for an algorithm for locating solutions to Diophantine equations. Another, leading to Turing’s seminal 1936 paper, was the question (Hilbert’s ‘Entscheidungsproblem’) of whether there is an algorithm for deciding of a given sentence whether it is logically valid or not. More generally, he raised the question of whether there exist *unsolvable* problems in mathematics. Or whether there exist computational tasks for which there is no valid program. Hilbert believed there were no unsolvable problems, famously declaring in Königsberg in September 1930:

For the mathematician there is no Ignorabimus, and, in my opinion, not at all for natural science either. . . . The true reason why [no one] has succeeded in finding an unsolvable problem is, in my opinion, that there *is no* unsolvable problem.

The capturing of the *notion* of computability via abstract mathematical models led to a very different view.

MODELS OF COMPUTABILITY

Coincidentally, just the day before Hilbert’s declaration quoted above, what became known as *Gödel’s Incompleteness Theorem* was being quietly announced by the young Kurt Gödel at another meeting in the same city. An

important technical feature of the proof of the theorem was the first formalisation of the notion of a computable function, enabling us to *talk about* computability from the outside. It was not long before there were a number of formalisations, or models, of computability. Here are some of the more important ones:

I. The Recursive Functions. This was Gödel's formalisation, and, based in logic, is the closest one gets to the way one would describe an algorithm in everyday language. Stephen Kleene gave this model of computability its final form, and developed the theory of computability under the name of *recursive function theory* (or just *recursion theory*). It was not until the late 1990s that the nineteenth century term 'recursion' lost its dominance over the terminology of computability theory and resumed something like its original technical meaning.

II. The λ -Computable Functions. The lambda calculus, first developed by Alonzo Church (founder member of the Association for Symbolic Logic, the main organisation of mathematical logicians) and his student Kleene, has become a very important way of presenting computations so as to minimise distinctions between notations, and is basic to certain programming languages.

III. The Turing Computable Functions. Turing machines provided a first machine model of computability. What makes them still basic to theoretical computer science is that every computational act is carried out in atomic detail. If one is interested in measuring *complexity* of computations, then Turing machines give a true measure of the work done.

IV. Markov Algorithms. Another quite influential model was that of the Markov Normal Algorithm, which acts on particular words in some chosen language, with operations implementing simple substitutions of sub-words. This another model relevant to programming.

V. URM Computable Functions. Unlimited Register Machines were invented somewhat later than Turing machines. The concept is usually associated with Shepherdson and Sturgis (1963), although related models

were studied by others such as Ershov (1958) and Minsky (1961). The memory of a URM functions more obviously like that of a modern day computer.

All of these frameworks enable one to effectively list *all possible* algorithms of that kind, and to use the list to devise a problem which cannot be solved by such an algorithm. This is essentially what Turing did in constructing a *universal* Turing machine, and hence finding a problem unsolvable by such a machine. By arguing convincingly that *any* algorithm could be performed by a suitable Turing machine, he was able to conclude that there existed problems that were *unsolvable* by any algorithm. Church, also in 1936, did something similar using λ -computability instead.

For further details of these, one should go to any good introduction to computability, such as [1].

An important fact is that however different these notions appear to be, they all lead to the same class of functions. Even more remarkable is that computability appears to exist independently of any language used to describe it. Any sufficiently general model gives the *same* class of functions — this assertion is captured in the *Church-Turing Thesis*, and has stood the test of time.

Of course, all these notions of computability deal with discrete data. This does reflect everyday practice, in that scientific measurements can only be made to a given level of exactness, and this is reflected in the sort of data can computers work with. However, the *mathematics* of the real world does sometimes require us to say something about computations over continuous data, that is data described by real numbers, and this requires extended models, as we shall see below. Turing himself introduced the concept of a *computable real number*.

Finally, notice that it does not change our thesis to allow in our computations *non-deterministic* steps, wherein we are allowed free choice between a list of computational actions. For instance, any function computable by a non-deterministic Turing machine can also be computed by a deterministic one. But the picture changes radically, as we see in the next section,

if we compute *relative* to incomplete information. And the jury is still out on what happens when we work with time or space bounds on our computations. For instance, in the context of polynomial time bounds, the fundamental open question $P =? NP$ asks whether there are functions computable in polynomial time using non-deterministic Turing machines which are not so computable with deterministic ones.

INCOMPUTABILITY AND ORACLE MACHINES

Turing machines can become data for other Turing machines to compute with. This observation enabled the design of a *Universal Turing Machine* which could be used to simulate computations of *any* other machine. The key technical idea here is to code information, such as machines or their computations, as machine-friendly data, such as numbers or binary sequences. This is the theoretical basis for modern computers, which treat programs as data, to be stored and used as needed.

Having theoretically captured computers, it is a short step to finding quite natural problems which are beyond the grasp of any computer, however large or efficient. For instance, there are all sorts of general questions about *how* a given Turing machine computes which cannot be answered by any computer program. The best known of these is the so-called *Halt-ing Problem*, which is that of determining of an arbitrary Turing machine whether it will successfully compute – that is, halt and give an output – for arbitrary input data.

So incomputability emerges at the edge of computability. Its origins are mathematically uncomplicated enough to produce this complex and intimate relationship. And just as the non-computable universe is woven from the algorithmic fabric of everyday life, so the structures on which we base its analysis are derived from appropriate computable relationships on information content — itself abstracted from the way science describes the material universe. The standard model of computationally complex environments first appeared in a difficult and still slightly mysterious paper of Turing's from 1939. The *Turing universe* is a structuring of the binary real numbers

based on the notion of an *oracle Turing machine*, which models what we can mathematically map of algorithmic interactions between data — coded as reals — which may or may not be algorithmic in origin. Oracle Turing machines also allow us to compare the solvability of different mathematical problems, which is why Emil Post called the mathematical structure *degrees of unsolvability* — today more usually called the *Turing degrees*, denoted by \mathcal{D} . The Turing degrees consist of equivalence classes of binary reals under the equivalence relation of being Turing computable from each other. We often write $A \leq_T B$ for “ A is Turing computable from B ”.

This model can be refined in various ways to take account of constraints of time or space imposed on computations. Just as Turing machines provide a useful basis on which to base the study of *complexity* of computations, so oracle Turing machines provide degree structures relevant to complexity-theoretic questions.

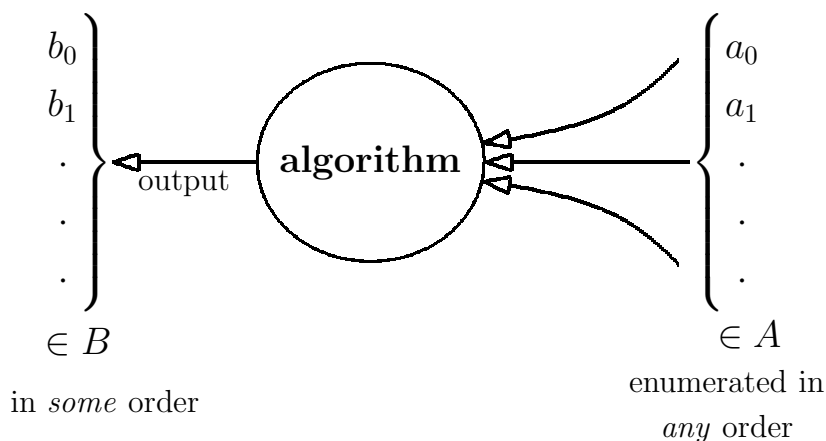
Actually, there is a more general model based on *non-deterministic* Turing machines, which is applicable to computations relative to *partial* information, or equivalently, relative to data which is *enumerated* in real-time rather than being available as required.

What makes computation relative to partial information *different* is that we cannot computably tell whether our oracle is going to answer or not. In the new model, there is not just one computation dependent on guaranteed oracle responses to our queries. It is more like real life where our search for useful knowledge is only partially rewarded and new information emanates from our environment in a fairly surprising and unpredictable way.

This gives rise to one slightly different way of looking at things, based on looking what we *do* — how we *guess* and pursue different alternative computations. This leads to *nondeterministic Turing machines*, with corresponding reducibility $A \leq_{NT} B$.

Another viewpoint emphasises how knowledge is *delivered* to us by our source of outside information — in a sense *enumerated*, and according to a timetable not under our own control. This leads to a notion of *enumeration reducibility*. Formally, we define $B \leq_e A$ to mean we can computably enumerate the members of B from an enumeration of the members of A —

where this enumeration of B does not depend on the *order* in which A is enumerated. Here is a picture:



We have in mind here the real world where we make scientific calculations according to the available data, but where the eventual answers we get do not depend on the order of discovery of these data.

These two viewpoints appear to give us *two* models, though it can be easily proved that both are equivalent. Another important fact is that for computations relative to total functions, the computational models based on deterministic or non-deterministic machines are the same, so there is a natural embedding of the structure \mathcal{D} of the Turing degrees within the extended *enumeration degree* structure \mathcal{D}_e . There are a number of deep and intractable open problems concerning the relationship between these two structures.

Of course, there are underlying fundamental questions here concerning incomputability in mathematics and real life:

How rich a variety of unsolvable problems is there? Does incomputability impinge on everyday life? And if so, can we find an informative theory of incomputability?

These questions can be seen to underlie many bitter debates in science and mathematics, and prominent figures can be found ranged on both sides

of this controversy. It has to be said that there are as yet no generally agreed answers to these questions, but quite a lot of pointers to positive ones. But there exist fascinating discussions concerning extensions of the Church–Turing Thesis to the material Universe (see Section I.8 of Volume I of Odifreddi’s book on *Classical Recursion Theory*) and of incomputability in Nature (see, for example, Roger Penrose’s *The Emperor’s New Mind*).

Even more divisive is the debate as to how the human mind relates to practical incomputability. The unavoidable limitations on computers suggest that mathematics — and life in general — may be an essentially *creative* activity which transcends what computers do.

To what extent can the human mind be likened to a Turing machine or a large computer?

The basic inspiration for Alan Turing’s computing machines was, of course, the human mind, with things like “states of mind” feeding into the way he described the way his machines worked. Turing made clear in a number of places which side of the argument he was on. On the other, we feel subjectively that our mental processes are not entirely mechanical, in the sense that a Turing machine is. And various people have explicated these feelings to a point where it can be argued convincingly that these feelings have more than purely subjective content. For instance, there is the famous and influential book of Jacques Hadamard on *The Psychology of Invention in the Mathematical Field*, or the philosophically remarkable *Proofs and Refutations: The Logic of Mathematical Discovery* by Imre Lakatos. In science, Karl Popper effectively demolished the inductive model of scientific discovery — as was accomplished, more debatably, by Thomas Kuhn at the social level. This raises the question of how to model the way theories are hypothesised, via a process which seems neither random nor simply mechanical.

A purely mathematical answer to the question is very difficult. Roger Penrose (in his *Shadows of the Mind*) has argued (unsuccessfully it seems) that the overview we have of Gödel’s Incompleteness Theorem for axiomatic arithmetic shows that the human mind is not constrained by that theorem.

But it is hard to be clear what it is that the human mind may be doing that Turing machines are incapable of. Obviously it will help to know more about both the physical and the logical structures involved. What is really needed is an *alternative* mathematical model to that of the Turing machine, and providing this must be one of the main aims of computability theory. Some speculations in this direction are provided in the 2003 paper by myself and George Odifreddi on “Incomputability in Nature”.

COMPUTABILITY AND INFORMATION

A large part of the scientific enterprise is bringing plausible descriptions of reality within practical computational frameworks. As we have seen, it is easy to describe classically incomputable objects from algorithmic ingredients, raising the question of to what extent this is mirrored in real-world. It is also the case that in mathematics, the links between computability and descriptions in natural languages is an intimate one which has been extensively mapped out.

Descriptions in natural languages give rise to various hierarchies. At a very basic level, one can build the *arithmetical hierarchy* by starting with computable relations on numbers — essentially, general statements about numbers which do not involve quantifiers — and adding existential and universal quantifiers. There are other hierarchies which mix language and computational elements. Degree structures and hierarchies are two complementary ways of looking more closely at the universe of incomputable — or computable — objects.

We are all familiar with the hierarchical structure of science itself. Within the life sciences, say, we have the fragmented focus on the quantum level, on atoms, on molecules, on cells, on multicellular organisms, on social structures. Within this descriptive framework the dynamic relationships at each level have to be investigated within the local constraints operating at each level. From a basic mathematical perspective, we find different levels of the arithmetical hierarchy reveal an analogous dynamic infrastructure — its analysis based on a detailed examination of algorithmic relationships.

Within the Turing degrees, one has the degree $\mathbf{0}$ of the computable reals, and then the degree $\mathbf{0}'$ of familiar unsolvable problems such as the Halting Problem. We call $\mathbf{0}'$ the *Turing jump* of $\mathbf{0}$. In fact, given a Turing degree \mathbf{a} and a set $A \in \mathbf{a}$, one can *relativise* the halting problem by considering it for a universal Turing machine U with oracle A . This will give us a *halting set* A' of inputs for which U computes, its Turing degree \mathbf{a}' being the *Turing jump* of \mathbf{a} . Of course, one can apply this jump operation to $\mathbf{0}'$ and get $\mathbf{0}''$, $\mathbf{0}'''$, ..., $\mathbf{0}^{(n)}$, ... etc.

The nice thing is it turns out that the Turing jump is closely related to how one uses the language of simple high-school arithmetic, codified within the arithmetical hierarchy. This is the basis of *Post's Theorem*, which relates statements in first-order arithmetic to iterations of the Turing jump. For instance, if one wanted to test arbitrary Turing machines to see if they were defined on every input, one would have to decide a statement which involved a universal quantifier followed by an existential quantifier, and this would require an oracle at the $\mathbf{0}''$ level.

There are mathematically interesting reducibilities which can be got by relativising and extending the arithmetical hierarchy. One might think of these as giving “degrees of describability”. One gets notions of *arithmetical in* and *hyperarithmetical in*, with their corresponding degree structures. You can take this approach further, getting reducibilities derived from hierarchies based on quantification over sets or functions, or even more general quantification.

COMPUTABLE ENUMERABILITY

Particular interest attaches to sets of numbers which can be *computably enumerated*. This is because many naturally occurring mathematical problems seem to involve such sets, including the Halting Problem. And in the context of the arithmetical hierarchy, they are the sets definable from a computable relation using just one existential quantifier, and are called Σ_1^0 sets — where the subscript tells us there is just one quantifier, and the superscript tells us it is just number variables we are quantifying over. There are many unsolved

problems relating to its degree structure \mathcal{E} .

Many naturally occurring incomputable sets turn out to be computably enumerable. A basic question motivating research since the earliest days is: *Just how rich is the Turing structure of the computably enumerable sets?*

There are very different ways of looking at this question, each with its own strengths and technical beauties. An intuitively satisfying approach — first tried by Emil Post — is to look for links between natural information content and relations on \mathcal{D} . Another is to delve into the intricacies of \mathcal{D} by directly constructing interesting features of the Turing universe. It is the relationship between these approaches which seems to have a special potential for modelling aspects of the material Universe. This is an area in which the techniques are quite hard to handle even at the classical level — and it is not surprising that their wider potential is largely unrealised.

One approach involves the search for richness of information corresponding to local degree theoretic structure. Ideally we would like something corresponding to the arithmetical hierarchy below \emptyset' . One can use *jump inversion* to bring aspects of that very natural hierarchy down to the local level. The resulting *high/low hierarchy* provides an invaluable frame of reference at the local level. But it is hard to characterise in terms of *natural* information content, or to describe in the local structure of \mathcal{D} .

(1) The *high/low hierarchy* is defined by

$$\mathbf{High}_n = \{\mathbf{a} \leq \mathbf{0}' \mid \mathbf{a}^{(n)} = \mathbf{0}^{(n+1)}\}, \quad \mathbf{Low}_n = \{\mathbf{a} \leq \mathbf{0}' \mid \mathbf{a}^{(n)} = \mathbf{0}^{(n)}\},$$

for each $n \geq 1$.

(2) If $\deg(A) \in \mathbf{High}_n$ we say A and $\deg(A)$ are *high_n*. We similarly define the *low_n* sets and degrees. For $n = 1$ we often drop the subscript — A and $\deg(A)$ are *low* if $A' \in \mathbf{0}'$, and *high* if $A' \in \mathbf{0}''$.

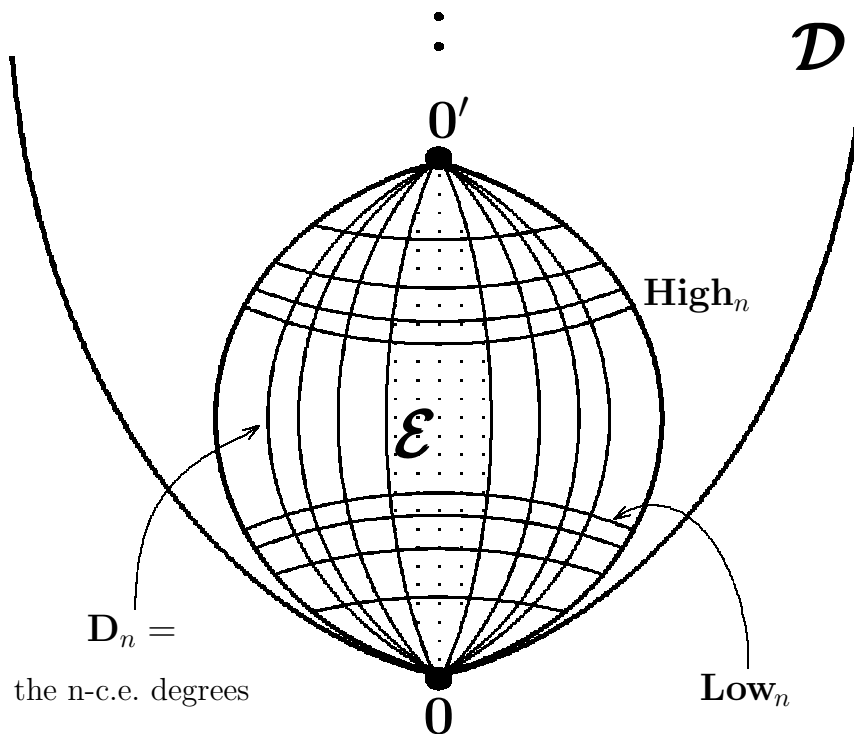
Intuitively, \mathbf{a} is *high_n* or *low_n* according as $\mathbf{a}^{(n)}$ takes its greatest or least possible value.

If the high/low hierarchy is thought of as defining a *horizontal* stratifi-

cation of \mathcal{D} below $\mathbf{0}'$, there is another very important hierarchy whose effect is *vertical*. The *n-c.e. hierarchy* — independently devised by Putnam and Gold around 1965 — inductively builds on the way we can form new sets using boolean combinations of c.e. sets. We get d.c.e. sets — or equivalently 2-c.e. sets — from the c.e. sets by forming *differences* $A - B$ of c.e. sets. More generally, an *n-c.e.* set is got via boolean operations on c.e. sets allowing up to n differences. Another way of looking at this generalisation of the notion of computably enumerable is that the n provides a bound on the number of mistakes one is allowed to make in computing the status of potential members of the set. So for a c.e. set A we can make just one mistake, deciding on at most one occasion to change the status of a number from being not in A to being in A .

A degree \mathbf{a} is *d.c.e.* if it contains a d.c.e. set. And for each $n \geq 2$ we write \mathbf{D}_n for the n^{th} level of the corresponding *n-c.e. hierarchy* of *n-c.e.* degrees.

This is the local framework we get from these two basic hierarchies:



One should not be misled by the diagram into thinking that either of the hierarchies eventually includes all of the Turing universe below $\mathbf{0}'$. Far from it. This can only be achieved by extending the levels of the hierarchies into the transfinite, as was done by Yuri Ershov, using the constructive ordinals to notate the different levels, in a sequence of three papers in 168–70.

There are other hierarchies based on notions of randomness, forcing, etc., though the local significance of these is limited or unclear. Randomness-related notions have led to some surprising refinements of the low and high degrees. Further information on these can be found in two comprehensive books on computability and randomness, one by Rodney Downey and Denis Hirschfeld, and another by André Nies. See below for some basic definitions, and some comments on the relationship between mathematical randomness and its quantum counterpart.

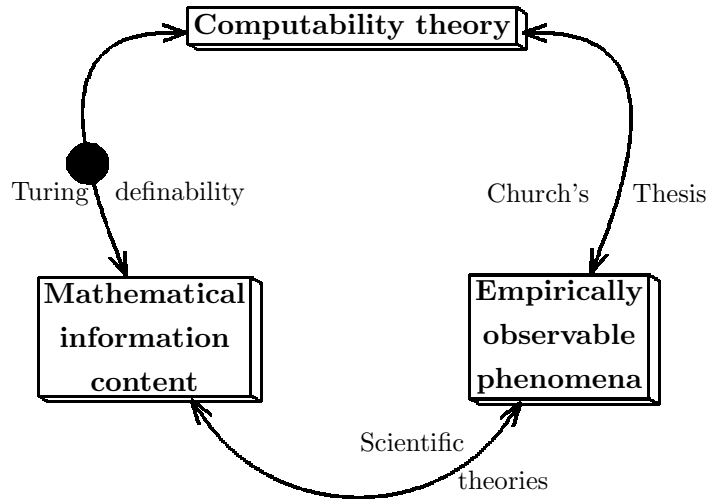
POST'S PROGRAMME

As we have seen, basic to the relevance of computability theory is the investigation of the extent to which it explains and structures naturally arising information, initially within the mathematical context. We now have some fine hierarchies which provide invaluable landmarks in this exploration.

The programme of mapping out the various links between information and computability-theoretic structure can be traced back to the seminal 1944 Bulletin of the American Mathematical Society paper of Emil Post. It was Post — before anyone had discovered the local hierarchies — who pointed the way. Let me try and say something about why Post's work is still so important to us, and how his approach relates to basic science. And why certain difficult technical problems in computability theory promise to have far-reaching implications.

The diagram below is an attempt to show how computability relates to the primary ingredients of science. These are firstly *observation* — that is, our experience of interacting with the Universe. And then mathematical *descriptions*, or information content, pinning down plausible relationships on the Universe in a widely communicable form. Computability is intrinsic

to both, and at the same time stands outside, its theory a sort of meta-science.



Process, causality, algorithmic content — all basic aspects — perhaps *the most* basic aspect of the real world of observation. And it is computability theory — suitably fleshing out and qualifying the Church–Turing Thesis — which mathematically models this. But this is not the only such modelling process. Science routinely builds much more specific mathematical models of natural phenomena, codifying all sorts of observed data into general laws. What is different about computability is that it also has something to say about this extraction of information content as an aspect of the real world. It has the potential to explain how this *information* content — natural laws — relates to the basic *algorithmic* content of the Universe.

The technical expression of this relationship is the notion of *Turing definability*. It is basic to understanding how the beautiful descriptions science gives us of the real world actually derive their material substance.

Definability in a structure is a key mathematical concept, and not widely understood. It is easy to give an intuitive idea of what definability is and how it relates to another useful notion, that of *invariance*. This is not necessarily because the notions are very simple ones (they are not), but because they do correspond to phenomena in the real world which we already, at some

level, are very familiar with.

As one observes a rushing stream, one is aware that the dynamics of the individual units of flow are well understood. But the relationship between this and the continually evolving forms manifest in the streams surface is not just too *complex* to analyse — it seems to depend on globally emerging relationships not derivable from the local analysis. The form of the changing surface of the stream appears to constrain the movements of the molecules of water, while at the same time being traceable back to those same movements. The mathematical counterpart is the relationship between familiar operations and relations on structures, and globally arising new properties based on those locally encountered ones. The emergence of form from chaos, of global relations within turbulent environments, is a particularly vivid metaphor for the assertion of definability, or invariance. Let us take a simple mathematical example from arithmetic.

Given the usual operation $+$ of addition on the set \mathbb{Z} of integers, it is easy to see that the set Ev of even integers is describable from $+$ within \mathbb{Z} via the formula

$$x \in Ev \iff (\exists y)(y + y = x).$$

So all we mean by a relation being *definable* from some other relations and/or functions on a given domain is that it can be *described* in terms of those relations and/or functions in some agreed standard language. Of course, there are languages of varying power we can decide on. In the above example, we have used very basic first order language, with finitary quantification over individual elements — we say that Ev is *first order definable* from $+$ over \mathbb{Z} . What has happened is that we started off with just an arithmetical operation on \mathbb{Z} but have found it distinguishes certain subsets of \mathbb{Z} from all its other subsets. Intuitively, we first focused on a dynamic flow within the structure given locally by applications of the form $n + m$ to arbitrary integers m, n . But then, standing back from the structure, we observed something global — \mathbb{Z} seemed to fall into two distinct parts, with flow relative to even integers constrained entirely within Ev , and flow from outside Ev being directed into Ev — with Ev being a maximal such subset of \mathbb{Z} . From within the

structure, $+$ is observable and can be algorithmically captured. Further than that, we are dealing with “laws” which cannot be related to the local without some higher analysis. This feature of the integers is not, of course, a deep one, but it does act as a basic metaphor for other ways in which more or less unexpected global characteristics of structures emerge quite deterministically from local infrastructure.

For the following definition, the *first order language* for \mathcal{D} is one with just the basic variables, brackets, quantifiers and logical connectives, and one 2-place symbol for the ordering \leq .

(1) Let $R(\mathbf{x}_1, \dots, \mathbf{x}_k)$ be a relation on \mathcal{D} .

We say that R is *Turing definable* — or *definable in \mathcal{D}* — if there is some first order formula $\varphi(x_1, \dots, x_k)$ in the language for \mathcal{D} such that for all $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathcal{D}$ $R(\mathbf{a}_1, \dots, \mathbf{a}_k)$ holds if and only if $\varphi(\mathbf{a}_1, \dots, \mathbf{a}_k)$ holds in \mathcal{D} .

(2) If \mathcal{A} is some family of sets, we say \mathcal{A} is *Turing definable* if the set of all Turing degrees of members of \mathcal{A} is Turing definable.

As a simple example, notice that $\mathbf{0}$ is Turing in \mathcal{D} via the formula

$$\varphi(\mathbf{x}) \iff_{\text{defn}} (\forall \mathbf{y}) [\mathbf{x} \leq \mathbf{y}].$$

One can, of course, talk about definability in other structures. For instance, $\mathbf{0}_e$ is definable in \mathcal{D}_e , and $\mathbf{0}'$ is definable in \mathcal{E} .

The notion of *invariance* gives a useful, if slightly more abstract, way of looking at definability. Being able to uniquely *describe* a feature of a structure is a measure of its uniqueness. But some feature of a structure may be quite unique, without one being able to describe that uniqueness in everyday language. Mathematically, we use the notion of *automorphism* to capture the idea of a *reorganisation* of a structure which does not change any of its properties. A feature of that structure is *invariant* if it is left fixed by any automorphism of the structure. Obviously if one can uniquely describe such a feature, it must be invariant, but not necessarily conversely.

(1) Let $R(\mathbf{x}_1, \dots, \mathbf{x}_k)$ be a relation on \mathcal{D} .

Then R is *Turing invariant* if for every automorphism $\psi : \mathcal{D} \rightarrow \mathcal{D}$ and every $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathcal{D}$ we have $R(\mathbf{a}_1, \dots, \mathbf{a}_k)$ holds if and only if we have $R(\psi(\mathbf{a}_1), \dots, \psi(\mathbf{a}_k))$ holds in \mathcal{D} .

(2) If \mathcal{A} is some family of sets, we say \mathcal{A} is *Turing invariant* if the set of all Turing degrees of members of \mathcal{A} is Turing invariant.

Only in recent years have we become aware that much of the past fifty years' research into computability has actually been about Turing definability and invariance. Current research focuses on getting optimal Turing definitions of the various levels of the n-c.e. hierarchies we have been looking at. There are also deep questions concerning the nature of the automorphism groups pertaining to different degree structures.

For more background information the joint article by myself and George Odifreddi is an approachable source. For more technical material try volume II of Odifreddi's mammoth *Classical Recursion Theory*.

TURING INVARIANCE IN THE REAL WORLD

What is far from clear is the extent to which the hierarchies and fine-structure theories built from simple computational ingredients are mirrored in the material world. On the other hand, the recognition of the real-world computational content of definability does seem to offer a widely applicable explanatory potential.

It is not surprising that attention has turned to Turing's universe of computably related reals as providing a model for scientific descriptions of a computationally complex real universe.

Let us now return to what the Turing model can do. Let us try to be more clear about how, from very simple beginnings, we can get from the basic fact of existence to what is for us an even greater puzzle — because we have to take what is happening under the umbrella of sufficient reason — the quite amazing emergence of individual entities. From this point of view, it is not quantum ambiguity which is surprising, but the existence of the well-defined world of our everyday experience.

More generally, we have the problem that even though we have natural laws to help us understand much of what happens in the universe, we have no idea where those laws themselves come from. Their apparent arbitrariness lies at the root of the present day confusion of speculative science, verging on the metaphysical.

For Alan Guth in 1997, the problem is:

“ If the creation of the universe can be described as a quantum process, we would be left with one deep mystery of existence: What is it that determined the laws of physics? ”

While in 1987 Roger Penrose asks for a *strong determinism*, according to which:

“ ... all the complication, variety and apparent randomness that we see all about us, as well as the precise physical laws, are all exact and unambiguous consequences of one single coherent mathematical structure. ”

The match between mathematics and experience has become more all-embracing, with string theory perhaps the most ambitious of the attempts to unify the two. The Turing model may be as yet very far from clarifying the specific details of relativity or quantum theory, but it does promise a release from the arbitrariness to which all less basic theories — superstring theory, M-theory, inflation, decoherence, the pilot wave, gauge theory, etc. — are subject, and is based almost entirely upon experience.

What the Turing model primarily tells us about is not an emergence of particular events from events, but of natural laws from the structure of information content.

What does the Turing model suggest regarding the basic structure of matter and the laws governing it?

What we know of the Turing universe is consistent with the possibility that the information content or level of interactivity of a given entity may be insufficient to guarantee it a unique relationship to the global structure. This is what one might expect to apply at an early stage in the development of the universe, or at levels where there is not a sufficient density

of interactions to give information a global role. A number of classic experiments on subatomic particles confirm such a prediction. On the other hand, mathematically entangling such low level information content, perhaps with content at levels of the Turing universe at which rigidity sets in, will inevitably produce new content corresponding to a Turing invariant real. The prediction is that there is a level of material existence which does not display such ambiguity as seen at the quantum level, and whose interactions with the quantum level have the effect of removing such ambiguity — confirmed by our everyday experience of a classical level of reality, and by the familiar ‘collapse of the wave function’ associated with observation of quantum phenomena. Since there is no obvious mathematical reason why quantum ambiguity should remain locally constrained, there may be an apparent non-locality attached to the collapse. Such a non-locality was first suggested by the well-known Einstein-Podolsky-Rosen thought experiment, and, again, has been confirmed by observation. The way in which definability asserts itself in the Turing universe is not known to be computable, which would explain the difficulties in predicting exactly how such a collapse might materialise in practice, and the apparent randomness involved.

As we have already mentioned, the Turing model may have implications for how the laws of nature immanently arise. And also how they collapse near the big bang ‘singularity’, and the occurrence or otherwise of such a singularity. What we have in the Turing universe are not just invariant individuals, but a rich infrastructure of more general Turing definable relations. These relations grow out of the structure, and constrain it, in much the same sort of organic way observable in familiar emergent contexts. These relations operate at a universal level. The prediction is that a Universe *with sufficiently developed information content* to replicate the defining content of the Turing universe will manifest corresponding material relations. The existence of such relations one would expect to be susceptible to observation, these observations in turn suggesting regularities capable of mathematical description. And this is what the history of science confirms. The conjecture is that there is a corresponding parallel between natural laws and relations which are definable in an appropriate fragment of the Turing universe.

The early Universe one would not expect to replicate such a fragment. The homogenisation and randomisation of information content consequent on the extreme interconnectivity of matter would militate against higher order structure. The manifest fragment of the Turing universe, based on random reals, might still contain high information content, but content dispersed and made largely inaccessible to the sort of Turing definitions predicted by the theory. Projected singularities, such as within black holes or associated with boundary states of the Universe, depend on a constancy of the known laws of physics. But immanently originating laws must be of global extraction. This means that their detailed manifestations may vary with global change, and disappear even.

Notice the difference here between what we are saying, and what the upholders of the various versions of Everett's many worlds scenario are. On the one hand, we have an application of the principle of sufficient reason to the world as we know it, which gives a plausible explanation of quantum ambiguity, the dichotomy between quantum and classical reality, and promises some sort of reconciliation between science, the humanities, and our post-modern everyday world. On the other we have something more like metaphysics.

The Turing model, and its connections with emergence, also lead us to expect the familiar fragmentation of science, and human knowledge in general. As we know from computability theory, a Turing definition of a given relation does not necessarily yield a computable relationship with the defining information content. But working within the relations at a given level, there may well be computable relationships emerging, which may become the basis for a new area of scientific investigation. For instance research concerning the cells of a living organism may not be usefully reduced to atomic physics, but deals with a higher level of directly observed regularities. Sociologically, one studies the interactions governing groups of people with only an indirect reference to psychological or biological factors. Entire relations upon cells (humans) defined in some imperfectly understood way by the evolutionary process provide the raw material underlying the new discipline, which seeks to identify a further level of algorithmic content.

There are questions about the range of possibilities embodied in such things as quantum ambiguity: Going from the uniqueness of a defined phenomenon to — what? Are there any overall constraints apart from those imposed by the mathematics specific to the emergent structures? There seems to be one unavoidable rule — obvious when it is pointed out — which is that each superimposed alternative must be viable by itself. Which, in addition to the specifics, demands that the information content develops within the rules experience and the computability theory lead us to expect. In particular, there can be at most countably many such alternatives. It is known that there exist at most countably many Turing automorphisms.

For a more detailed review of the deep and fundamental computability-theoretic problems reviewed here see [2].

RANDOMNESS VERSUS INCOMPUTABILITY

The concept of randomness is a very significant one for computer scientists, and there are important questions concerning the relationship between quantum randomness, computability and mathematical notions of randomness. Its mathematical theory took off with Martin-Löf's notion of algorithmic randomness around 1966. Randomness has been pursued in many different ways, most notably by Andrei Kolmogorov, Gregory Chaitin, Ray Solomonoff and Robert Solovay.

Unlike for incomputability, there is no clear-cut definition of randomness. A common misconception, particularly among non-mathematicians, is that there is an *absolute* notion of randomness. In fact, randomness is always relative to what it is trying to *avoid*. Another is that randomness goes with a lack of information content. The mathematical theory tells us that an object can be both very random, and contain a lot of information — the randomness consists in the *concealment* of information. Randomness of a real does imply incomputability, but not conversely.

Here is the natural statistical definition of a random real Martin-Löf came up with, one of a number of related definitions. Let μ be a Lebesgue measure.

(1) An $f \in 2^\omega$ is *Martin-Löf random* — or *1-random* — if for every c.e. sequence $\{A_i\}_{i \geq 0}$ of Σ_1^0 sets $A_i \subseteq 2^\omega$ such that $\mu(A_i) \leq 2^{-i}$, we do not have $f \in$ each A_i .

(2) More generally, we say $\mathcal{A} \subseteq 2^\omega$ has Σ_1^0 -measure zero if $\mathcal{A} \subseteq \bigcap_{i \geq 0} A_i$ for some such $\{A_i\}_{i \geq 0}$.

Another way of phrasing this definition is in terms of a *Martin-Löf test* T which is a c.e. set of pairs of the form (i, σ) of numbers i and binary strings $\sigma \in 2^{<\omega}$. Then the sets A_i arise via

$$A_i = \bigcup_{(i, \sigma) \in T} N_\sigma.$$

We say f *fails* the test T if $f \in \bigcap_{i \geq 0} A_i$, and otherwise *passes* the test T . The 1-random reals are the ones which pass every Martin-Löf test.

Intuitively, random reals are very hard to approximate. f is random if it is not in any constructively defined null subset of the measure space. The practical effects of randomness are very much like those of incomputability — random objects are indeed very hard to computably predict. But you should not confuse these two overlapping but distinct ideas. What makes for randomness is not just incomputability, but *concealed* information content — sometimes very high information content, but inaccessible to the point of worthlessness in any practical sense. For this reason notions of randomness have important consequences for coding theory.

The coding connection can be made clearer via Chaitin’s definition of a random real, which is also very persuasive and is equivalent to Martin-Löf’s definition. Chaitin says f should be random if it is *algorithmically incompressible* — that is, if the “size” of any program for describing $f \upharpoonright x$ is not significantly smaller than x . I would recommend Chaitin’s book on *Algorithmic Information Theory* as an entertaining introduction to his ideas on randomness and its wide-ranging consequences.

Since $\mu(2^\omega) = 1$, μ is called a *probability measure*. It turns out that if \mathcal{A} is a sufficiently constructive property of reals which is true with probability 1, then every random real *must* have property \mathcal{A} .

A basic result of Martin-Löf from 1970 is that there is a universal Martin-Löf test T with $\mathcal{U}_i = \bigcup_{(\sigma) \in T} N_\sigma$ each i , where

- (1) $\mathcal{U}_0 \supseteq \mathcal{U}_1 \supseteq \dots$
- (2) $\forall i [\mu(\mathcal{U}_i) < 2^{-i}]$.
- (3) $\mathcal{A} \subseteq \bigcap_{i \geq 0} \mathcal{U}_i$ for every $\mathcal{A} \subset 2^\omega$ of Σ_1^0 -measure zero.

It easily follows that the class of all random reals has measure one. That is, *most* reals are random. There are many questions relating to the degrees of 1-random reals.

Computability theory gives us a rather simpler definition of randomness which avoids the use of measure. Here is an equivalent way of describing the 1-random reals. It is essentially Bob Solovay's characterisation of randomness, as simplified by Rod Downey.

- (1) A *Solovay test* T is a c.e. set of binary strings with $\sum_{\sigma \in T} 2^{-|\sigma|} < \infty$.
- (2) A set A *fails* a Solovay test T if for infinitely many $\sigma \in T$ we have $\sigma \subset A$ — and otherwise A *passes* the test T .
- (3) The set A is 1-random if it passes every Solovay test.

A Solovay test is a c.e. sequence of guesses about beginnings of A . The condition $\sum_{\sigma \in T} 2^{-|\sigma|} < \infty$ makes sure the guesses are bold enough — the lengths of the strings in T must increase fast enough to ensure the infinite sum converges. T cannot play safe. Random sets are chaotic enough to eventually escape the predictions made by T . *Eventually* is the key word here. We cannot ask f to completely avoid any sufficiently thin Σ_1^0 class of functions. The above definition is the next best thing.

Of course, there is no such thing as complete randomness. If f avoids one kind of class, it necessarily gets captured by the members of another — a 1-random f is captured by all sufficiently fat Σ_2^0 classes of reals.

Kučera has shown that although every degree above $\mathbf{0}'$ is 1-random, but that the 1-random degrees are not closed upwards. Given that there are 1-random members of $\mathbf{0}'$, we can ask if there are any *natural* 1-random members. If we look at the halting problem for the universal Turing machine U in a rather different way, we get one. This was Chaitin's idea:

Think of any string $p \in 2^\omega$ as a program for U , which you run by inputting it to U in the usual way. A program p may or may not halt — but you can compute the probability that for any set of instructions, a universal Turing machine will halt in the form of the binary real:

$$\Omega = \sum_{p \text{ halts}} 2^{-|p|}$$

Then Ω turns out to be 1-random, and it is in $\mathbf{0}'$. There has been a lot of interest in Chaitin's Ω , for obvious reasons — it is not every day we come across a new real number.

COMPUTABILITY OF STRUCTURES

Sometimes we look at real situations and struggle to come up with usable descriptions of them. In an abstract sense, what we are doing is trying to approximate reality in a computationally manageable way. It can also happen that we make plans and theorise, and then have to manage reality to fit the design. These are related problems, and both lead us to the study of *computable model theory*. In this section we will be concerned with the computability of the structures on which theories are based, and which in turn arise from those theories.

To talk about computability of a model \mathcal{M} in the everyday sense, it is appropriate to assume it to be a structure with a *countable* domain M and a countable set of relations and functions over it. Again in keeping with everyday situations, we need to assume a *presentation* of \mathcal{M} , whereby the members of M have labels we can compute over. Usually we will implicitly assume $M = \mathbb{N}$, with the relations and functions and constants of \mathcal{M} number-theoretic. To talk about \mathcal{M} in a natural way, we need the first order language $\mathcal{L}_{\mathcal{M}}$ which has function and predicate symbols corresponding to the functions and relations of \mathcal{M} , together with individual constants \mathbf{a} corresponding to the members $a \in M$. We write $\text{Th}(\mathcal{M}, a \in M)$ for the set of all sentences of $\mathcal{L}_{\mathcal{M}}$ which are true in \mathcal{M} . Then:

(1) A model \mathcal{M} is *computable* if its domain M is computable and its relations and functions are uniformly computable.

(2) A model is *decidable* if M is computable, and $\text{Th}(\mathcal{M}, a \in M)$ is computable.

To take account of the underlying presentation we occasionally say \mathcal{M} is *computably presented* instead of just computable. And we sometimes say \mathcal{M} is *computably presentable* to mean it is *isomorphic* to a computable — that is computably presented — model.

It turns out that quite familiar theories lead to incomputable models. Take the usual first-order theory PA of arithmetic. It has the familiar *standard* model \mathfrak{N} with domain $M = \mathbb{N}$, which is certainly computable. Notice that \mathfrak{N} is a what we call a *prime* model of PA , in the sense that every model of PA has a submodel isomorphic to \mathfrak{N} . But (essentially from Gödel's incompleteness theorem), we know that nonstandard models of PA exist. And, by a theorem of Tennenbaum, \mathfrak{N} is the *only* computable model of PA , up to isomorphism. Actually, with a little sleight-of-hand, you can get incomputable *standard* models of PA .

It is straightforward to say what we mean by the *degree* of a model \mathcal{M} , where we assume the domain $M = \mathbb{N}$. We take it to be the l.u.b. of the degree of M and the degree of its relations and functions. To do this we use the *atomic diagram* of \mathcal{M} , which puts together all the basic information about \mathcal{M} in one set of formulas.

Then incomputable models with simple origins are mathematically very common:

(1) For every c.e. degree \mathbf{a} there is an axiomatisable first order theory \mathcal{T} with a model \mathcal{M} of degree \mathbf{a} , and

(2) Given $A \subseteq \mathbb{N}$, there is a model \mathcal{A} of PA such that $\mathcal{A} \cong \mathfrak{N}$ and $\text{deg}(\mathcal{A}) = \text{deg}(A)$.

Let us finish this brief introduction to computable models with a closer look at the general question:

- *What is the relationship between the degree of a theory \mathcal{T} and the degrees*

of its models?

One can get some strikingly strong bounds on the degrees of models. For instance, if \mathcal{T} is a consistent axiomatisable theory, it has a countable model of degree $< \mathbf{0}'$, even of low degree. It follows, of course, that there are nonstandard models of PA which have low degree.

There are all sorts of interesting results concerning the connections between the degrees of nonstandard models and those of their theories.

If this very brief look at computable models has left you looking for more, I would recommend the survey by Valentina Harizanov, in the *Handbook of Recursive Mathematics*, listed in the Reading List.

Computable model theory can involve looking at complicated theories, and even cooking up unnatural theories with strange models. Whereas everyday mathematics is populated with just a few basic kinds of structure, describable via simple axioms.

There are interesting things to say about the constructive content of even the simplest structures. One cannot get much more basic than countable linear orderings, and even here we have a rich and interesting theory.

There are two closely related main approaches to such orderings:

(1) We say $\mathbf{A} = \langle \mathbb{N}, \leq_A \rangle$ is *computably presented* — or just *computable* when there is no ambiguity — if \leq_A is a computable relation. And \mathbf{A} is *computably presentable* if it is isomorphic to a computably presented ordering.

(2) A *computable subordering* of \mathbb{Q} is a computable subset of \mathbb{Q} with its usual ordering \leq .

Suborderings of \mathbb{Q} are more easily *related* to each other, while presented orderings of \mathbb{N} can be *built* more easily to bring out *differences* between notions. There are some very attractive results of Feiner which help us translate between the two settings, giving us the best of both worlds.

An effective version of an old result of Cantor tells us that if \mathbf{A} is computable, then \mathbf{A} is computably isomorphic to a computable subordering of \mathbb{Q} . Conversely, every computable subordering of \mathbb{Q} is computably isomorphic to a computably presented ordering \mathbf{A} of \mathbb{N} . The beauty of Feiner's

results lies in the tracing of the correspondences right up the arithmetical hierarchy. For instance:

- If \mathbf{A} is a Σ_1 presented linear ordering, then \mathbf{A} is Δ_2 -isomorphic to a Π_1 subordering of \mathbb{Q} .

One could approach structures such as Boolean algebras, groups, rings or fields with a similarly constructive outlook, and find interesting results and open questions. See the volume II of the *Handbook of Recursive Mathematics* for a plethora of interesting research.

READING LIST

- [1] S. B. Cooper. *Computability Theory*. Chapman & Hall/CRC, Boca Raton, London, New York, Washington, D.C., 2004.
- [2] S. B. Cooper and P. Odifreddi. Incomputability in Nature. In S.B. Cooper and S.S. Goncharov, *Computability and Models*. Kluwer Academic/Plenum, New York, Boston, Dordrecht, London, Moscow, 2003, pages 137–160.
- [3] M. Davis. *The Universal Computer: The Road from Leibniz to Turing*. W.W. Norton, New York, London, 2000.
- [4] Yu. L Ershov, S.S. Goncharov, A. Nerode, J.B. Remmel (Editors). *Handbook of Recursive Mathematics*, Volumes 1 and 2. Elsevier, Amsterdam, New York, Oxford, Tokyo, 1998.
- [5] E. R. Griffor (Editor). *Handbook of Computability Theory*. Elsevier, Amsterdam, New York, Oxford, Tokyo, 1999.
- [6] A. Hodges. *Alan Turing: The Enigma*. Vintage, London, Melbourne, Johannesburg, 1992.
- [7] Y. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, Mass., London, 1993.

- [8] P. Odifreddi. *Classical Recursion Theory*, Volumes I and II. North-Holland/Elsevier, Amsterdam, New York, Oxford, Tokyo, 1989 and 1999.
- [9] E. L. Post. *Solvability, Provability, Definability: The Collected Works of Emil L. Post* (Martin Davis, Editor). Birkhäuser, Boston, Basel, Berlin, 1994.
- [10] C. Reid. *Hilbert*. Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1986 (paperback edition).
- [11] R. Penrose. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press, Oxford, New York, Melbourne, 2002.
- [12] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, Mass., London, 1987.
- [13] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987.
- [14] A. M. Turing. *Collected Works: Mathematical Logic* (R.O. Gandy and C.E.M. Yates, Editors). Elsevier, Amsterdam, New York, Oxford, Tokyo, 2001.