COMPUTABILITY, ALGORITHMIC RANDOMNESS AND COMPLEXITY

ROD DOWNEY

What drew me to the study of computation and randomness

I think mathematical ability manifests itself in many different ways[1]. In particular, mathematicians can be drawn to space and geometry, can have strong analytic intuition, can be drawn to formalism, they can be drawn to counting arguments, etc. There is definitely no unique type of mathematician. Maybe those mathematicians who are drawn to algorithmic thinking have found a home in computer science. For myself, I have always found myself drawn to thinking algorithmically. As a student, I recall studying algebra. Naturally, we would be prescribed problems to solve and sit exams. Usually, I would find that instead of some short elegant proof I would grind out some longer, but more basic often algorithmic version. Likely this reflected lack of study, as I was pretty lazy as an undergraduate, so I had not read the notes mostly! Even when studying analysis, I saw this as an algorithmic game in that given ϵ, somehow I would try to compute δ, viewing this as a game of us versus an opponent.

In my honours year[2], at Queensland University, I recall studying Szmielew's decision procedure for the elementary theory of abelian groups and the word problem for groups. After this I moved to Monash to work on effective algebra which was very fashionable at the time. In effective or computable algebra, one tries to understand the effective content of mathematics. This is the extent to which mathematics can be made algorithmic. One imagines the data as being *presented* in some computable fashion, and then asks for the extent to which aspects or processes of the data can be made computable.

A nice illustrative example from combinatorics is Dilworth's decomposition theorem. The classical version says that if the size of the largest antichain in a partially ordered set is k, then the partially ordered set can be expressed as the union of k linearly ordered chains. The computable version asks whether this can be done computably. To wit, given a computable partially ordered set, $(P, \leq)$ (meaning that the domain is computable and the relation $\leq$ is computable), can we decompose this into k *computable* chains?

[1]In the following I have sort of combined the questions into a long essay. I have also made several comments which would have references, but I refer to the Downey-Hirschfeldt monograph as a suitable source for references.

[2]The British system has three years of undergraduate study for a degree and then one more year called the honours year, where if you achieve first class or second upper honours, you can then move directly on to do a Ph.D.

Kierstead showed that the answer is no, *but* we can always decompose into $f(w) = \frac{5^w-1}{4}$ many computable chains, using an *online* algorithm. For those of you looking for a nice problem, figure out what the correct bound is. We have this exponential upper bound, but don't even have polynomial lower bounds for $f(w)$.

What we have learned

Computability theory allows one to calibrate how difficult computational tasks are. To do this structures like the *degrees of unsolvability* (and various other hierarchies) were invented by Post, Turing and others. Here A and B have the same degree of unsolvability if using A as read only memory we can compute membership of B with an algorithm, (so that $A \leq_T B$) and conversely (so that $B \leq_T A$). These equivalence classes partition the world into collections of *equicomputability.* There is a basic spine generated by the *halting problem* which, in data n, m, asks if there is an algorithm that decides if the n'th algorithm halts on input m. A fundamental result is that the halting problem is algorithmically unsolvable, and this gives rise to a hierarchy, the degree of the halting problem, called $\mathbf{0}'$, the degree of the halting problem *if* I give you the halting problem as read only memory, called $\mathbf{0}''$, and so on.

As an illustration, the problem of deciding if a Diophantine equation in many variables with integer coefficients has a positive rational zero is famously known to be of degree $\mathbf{0}'$. Similarly, the word problem for finitely presented groups asks : what is the algorithmic complexity the complexity of determining whether two words are equal in a given finitely presented group. This is also of degree $\mathbf{0}'$. My favourite example is Conway's generalization of Collatz functions. Recall that the Collatz function is $f(x) = \frac{x}{2}$ if x is even and $f(x) = 3x + 1$ if x is odd. Then a famous conjecture is that if you look at the iterates $f(x), f(f(x)), \ldots$ then the sequence returns to 1 eventually. Conway asked what about if we choose a set of rational numbers $\langle q_1, \ldots, q_n\rangle$ and looked at a similar problem but with congruences chosen via these rationals. That is, define $f_{\langle} q_1, \ldots, q_n \rangle(x) = q_m \cdot x$ if $x \equiv m \mod n$, and choose the rationals so that the result is a integer whenever x is an integer. Then what can be predicted about the sequence for this f and a given x? Conway showed that we can choose a finite set of rationals so that this simulates precisely the action of a given algorithm. Each of these results is proven by a similar kind of simulation and hence we see that these disparate areas of mathematics can simulate computation.

The reader might note that I was a bit sloppy in that I defined *the* halting problem when the definition seems to depend on the coding of the programmes. This turns out to be of no consequence, since we can show that all reasonable versions of the problem are the same up to a very strong simulation, called an m-reduction.

Problems can be very much harder than the halting problem. At a far extreme, there is a notion called *analytic* where the core problem is not asking if there is some stage where a computation halts, but whether there is a function $F : A \to B$ where for all n some computable relation for $F(n)$ holds. A classic example of this is isomorphism between structures. Using computability theory we can show that certain problems are as hard as the hardest analytic problems, and hence there *cannot* be a set of invariants to simplify the problem in the way that, for instance, dimension is an invariant for vector space isomorphism. Recently, Montalbán and I used this method to show that the isomorphism problem for torsion free abelian groups is analytic complete, and hence no reasonable set of invariants (in the sense of simplifying the problem) can exist for this problem.

For my own interests, behind all of this is a preference for studying fundamental logical questions about mathematics. This again is something I seem to have been drawn to from an early age. At high school I was able to study logic and philosophy as that was available in Queensland.

Actually, I believe that our working arena for all mathematics is computable mathematics in the sense that any function which occurs in "real life" likely will be computable. By the same token, one definition of a continuous function is one that is computable relative to some A. Thus computability is the study of continuity! Of course in the "really real" world, we "really" only deal with finite things, and these are abstracted to methods allowing us to analyze them. So number theory advanced when it was converted into analysis. We regard things as continuous to enable tools from continuous mathematics to attack them. Certainly being infinite is often a great approximation to the finite! Computation is at the core of a lot of mathematics.

Anyway, I have worked in and about these areas for the last three decades. A some stage I became very interested in computer science, and in that time developed *parameterized complexity* (a kind of complexity we feel is more attuned to practical computation) with Mike Fellows. At the same time I was working through Li and Vitányi's proofs of lower bounds for computability results using Kolmogorov complexity. For example the proof that two tapes have more power than one tape on a Turing machine. I really understood these proofs only very formally and put them on my large pile of "must look at again at some stage".

In January 2000, Denis Hirschfeldt and I organized a conference at Kaikoura, a small town on the east coast of the south island of New Zealand. This was under the auspices of the Marsden Fund for basic science of New Zealand. What we did was to bring in several overseas experts to speak about recent developments in mathematics in short graduate level courses. The resulting volume is called *Aspects of Complexity* and contains a number of fine short courses in and about complexity, and is inexpensive! This method being a great initiative of the founder of the NZIMA, Sir Vaughan Jones, whose overall goal was to raise the standard of New Zealand mathematics. At the Kaikoura meeting, there were some very illuminating lectures by Lance Fortnow on Kolmogorov complexity, and I came back with some fire in my belly to know some more.

At more or less the same time, one of my ex-postdocs, Richard Coles, was working at Auckland with Cris Calude. Coles visited me and asked me one of Cris's questions. The question was whether the Solovay degrees of halting probabilities were dense. What does this question mean? Well, in algorithmic randomness the halting problem is replaced by the halting probability. The halting problem is the collection of indices of programmes that halt. For a prefix-free machine (as discussed a little later), the domain of the machine is a prefix free subset of the collection of finite strings, and thus considered as a collection of cylinders will have measure under the uniform measure. Instead of the haling set, we have the halting probability which can be considered as the sum of the probabilities that the machine halts over all strings. We will discuss more on this topic in a little bit. Solovay reducibility is a continuous version of m-reducibility. The technical question was whether the resulting degree structure was a dense partial ordering. Hirschfeldt, Nies and I began to think about this question in depth, and eventually solve the question. At the same time, we were began a lot of background reading, as questions lead to other questions and we really knew little about the area. In doing so, we discovered a large, and for us hitherto unknown, body of work on algorithmic randomness. As Denis and I say in the introduction to our book: "We also found that, while there is a truly classic text about *general* Kolmogorov Complexity, namely Li and Vitányi (book), most of the questions we were interested in either were open, were exercises in Li and Vitányi with difficulty ratings of about 40-something (out of 50), or necessitated an archaeological dig into the depths of a literature with few standards in notation or terminology, littered with relentless re-discovery of theorems and a significant amount of unpublished material. Particularly noteworthy amongst the unpublished material was the aforementioned set of notes by Solovay, which contained absolutely fundamental results about Kolmogorov complexity in general, and about initial segment complexities of reals in particular. As our interests broadened, we also became aware of seminal results from Stuart Kurtz's PhD Dissertation, which, like Solovay's results, seemed unlikely to ever be published in a journal."

Anyway, about this time, we were able to prove a number of results in and around trying to understand what it meant to calibrate randomness, and begin the work on K-triviality. As our knowledge grew, we became aware of other work of Kučera and early work of Levin, Schnorr and Chaitin, with particularly attractive popular works by Chaitin. Perhaps foolishly, Hirschfeldt and I then decided to write a book organizing this material. We estimated that it would take 2-3 years.

So here we are nearly a decade down the track, and the book is just finished at the time of my writing this article. This is not simply laziness on our part, but also because there has been an explosion of results in this area, notably by lots of my postdocs such as Barmpalias, Bienvenu, Greenberg, Griffiths, Hirschfeldt, LaForte, Miller, Montalbán, Yu, and by other gifted authors including Kučera, Muchnik, Nies, Reimann, Shen, Slaman, Stephan, and Vereshchagin (and many others) and this necessitated endless re-writes.

Randomness is important in calculation and the like, but for me the fascination lies in the intuition that something fundamental is going on here. It is a striking fact that something, a string say, which is *algorithmically* random (meaning that it passes a bunch of computable tests) will act in some mathematical situation like the *expected statistical outcome.* Why should this be? Well, if the situation is normal then it will be computable, and hence if it did not act in the expected way, the very computational nature of the situation would allow us to compress the string, and hence it would not be random.

Personally, I am not driven by the thought that the material might be useful, but to try to understand what randomness means. I deal with question like: When is one real or string more random than another? How powerful as computational resources are randomness sources? If one string is more random than another how does that align to computational power? What does independence mean? What happens if we vary the machines? What are the correct notions of compressibility? These all seem fundamental and my intuition tells me they are important.

The Downey-Hirschfeldt Monograph

I have been asked to try to summarize what is in the Downey-Hirschfeldt book. The paragraph above is a good beginning.

The tools we will use will be based around computability theory. So we begin with a condensed course in “advanced computability theory” where we develop the tools we will need. The idea is that we will study *algorithmic* randomness. The theme of the book is that for such a notion of randomness, there are three natural approaches. In some sense they all derived from the intuition of von Mises. In a remarkable early paper von Mises argued that if we had a random real $\alpha = a_0a_1\dots$ (which we will consider as an infinite sequence of 0's and 1's) then by the law of large numbers the first n bits of α, $\alpha \upharpoonright n$ should contain as many 0's as 1's as $n \to \infty$. Moreover, if we *selected* n bits $a_{i_0}, \dots, a_{i_{n-1}}$, with $i_0 < i_1 < \dots < i_{n-1}$, then also we ought to have as many 0's as 1's. What selections should be possible? Well, if we are interested in *algorithmic* randomness then presumably the correct notion of selection should be more or less *computable* selection. It is important to realize that von Mises intuition was well before the development of computability theory, and hence the notion of using computable selections was due to Church many years later. Unfortunately, in its classic formulation, Ville showed that *no* collection of selection functions (computable or not) would suffice to characterize a reasonable notion of randomness as there would be natural effective statistical tests failed by some real, random relative to the selection functions.

Now we return to the three approaches. The first approach is the statistical one due to Martin-Löf. This views statistical tests as null sets (sets of measure 0), and asks that a real be random iff it avoids all *effective* null sets.

The second approach to defining randomness observes that if a string has patterns then that allows for compression of the string by a programme that exploits the patterns. This is the way commercial compression packages work. Kolmogorov defined a string σ to be random relative to a machine M if the shortest M-programme to generate σ has length $|\sigma|$ or longer. The Kolmogorov complexity C_M of a string σ relative to M is the *length* of the shortest M-programme for σ. It turns out that there is a universal M for the definition in the sense that for any other $\hat{M}$ there is a fixed constant c, and any σ, $C_{\hat{M}}(\sigma) \leq C_M(\sigma) + c$. For such M, we write C for C_M. Now the second approach asks that a real be random iff all its initial segments are incompressible. Unfortunately, using Kolmogorov's definition, no real is random since all will have C-compressible initial segments. The problem is that C does not really capture one of the key intuitions of the notion of *information content.* Namely, if $M(\tau) = \sigma$ our intuition is that the bits of τ encode the information of the bits of σ. Thus $|\tau|$ should be a reasonable notion of the information content of σ, and the shortest τ would give the Kolmogorov complexity of σ. *But*, plainly τ gives more information than just the bits of τ, it gives also *the length of* τ as well. That is we get the bits of τ and some kind of termination symbol saying that that is the end of the programme. First Levin and then a little later Chaitin suggested ways around this. Perhaps the most popular method is to use machines which act like telephone numbers[3]: no programme is a prefix of another, and this gives a notion called *prefix-free* Kolmogorov complexity which we denote by K. It is also possible to use "continuous" complexity (of various types) where now if τ extends $\hat{\tau}$, then $M(\tau)$ should extend $M(\hat{\tau})$. This notion gives rise to several notions of complexity depending on the notion of allowable machine, and two main complexity are called Km and Km_P, monotone and process complexity.

Anyway our story has a happy ending, since for any of these notions, α is Martin-Löf random iff the all of the initial segments of α are random as strings. Even here we see the emergence another theme of the book: What are the appropriate measures of information content, and how do they relate. For example, we include the difficult proofs that for strings σ,

$$K(\sigma) = C(\sigma) + C(C(\sigma)) + O(C(C(C(\sigma)))),$$

and that this $C^{(3)}(\sigma)$ is *sharp* in the sense that it *cannot* be replaced by $O(C^{(4)}(\sigma))$. This remarkable result is due to Solovay and no proof of it has appeared in print. We include many results about the relationships of the

[3]Well, more or less. I know that in the US 0 is the operator, and 011 is the international prefix access code, but the spirit is correct.

differing complexities, and see how it emerges that differing complexities can be appropriate for differing notions of randomness.

The final approach is the closest in spirit to von Mises. What we will do is bet on the "next bit" of α from $\alpha \upharpoonright n$. Clearly, if α is random, if we use some kind of effective betting strategy, we should not be able to win infinite winnings. With the correct notion of effective betting, it turns out that we get the same notion of randomness. The notion of effective betting is one which corresponds to being effectively approximable from below.

Schnorr proved this coincidence of randomness notions, but then pointed out that if we were to intuitively think of effective betting strategies surely we would simply bet and then not later change our minds and maybe bet more, etc. This critique led to the development of other natural notions of effective randomness, such as computable, partial computable, weak and Schnorr randomness. I won't go into details (buy the book!), but these notions have complex interrelationships particularly when we look at them in relation to Turing degrees. For example, for a certain class of degrees with, in a quantifiable sense, little computational power, they all coincide; whereas in ones with high computational power they all separate.

Already we see the emergence of another theme. The computational complexity of a random set can affect its level of randomness and conversely. A classic illustration of this is Ω , the universal halting probability (the measure of the domain of a universal prefix-free machine). As shown by Chaitin, we know that this has very high computational power, like the halting problem and in a very compressed form. This fact does not accord with our intuition that random reals should have feeble computational power, in that whilst they might have lots of "information" none of it is usable.

What we now know in a very precise way is why this is all true. The following is a colourful way to think of this. Think of trying to pass a stupidity test. There are two ways to do this. One is to be the genuine article, but the other is to be so smart that you know the correct answers to appear stupid. We know that if we soup-up the randomness by asking that the real be (weakly) random given the halting problem as an oracle, then the computational power of the random real is very much weaker in many quantifiable ways. Moreover, Frank Stephan showed that random reals come in two varieties. There are those with very low computational power (in the sense that they cannot compute what is called a PA degree) which are the typical ones, and those that have some power in this sense, and all of these are the "false" randoms living above the halting problem. That is, almost all random reals have very little computational power and those that do are not very random, and all are basically the halting problem (or more) compressed. These PA degrees are those that are degrees of models of Peano arithmetic have remarkable interactions with random degrees, a fact first realized by Antonín Kuč era. One recent gem is the result of Barmpalias, Lewis and Ng who showed that every PA degree is the join of two random degrees.

A great part of the middle of the book looks at interactions of notions of randomness and notions of computational power. It has become clear that what are called domination properties are key notions. For example consider the class of degrees **a** which have the property that every function f computed by **a** should be dominated by a computable function. This would seem a class of almost computable degrees with low computational power in some sense. But they can be random. And, indeed, for such degrees weak, computable, Martin-Löf and even weak randomness relative to the halting problem all coincide!

There are many such investigations. For example, the power to compute a PA degree is related to the initial segment complexity by looking at the growth rate of the initial segment complexity.

One striking phenomenon was the analysis of the K-trivial degrees. It is an old result of Chaitin, building on work of Loveland that α is computable iff $C(\alpha \upharpoonright n) \leq C(1^n) + O(1)$, giving an information-theoretical definition of computability. Solovay showed that there are *noncomputable* reals β with $K(\beta \upharpoonright n) \leq K(1^n) + O(1)$. Thus the characterization of computability fails for prefix-free complexity. Such reals β are called *K-trivial.* They have remarkable properties. I am sure that this class is explored by Nies in his contribution, which is highly apt as many of the basic properties of this class were discovered by Nies and his co-authors. They are very easy to construct, and give natural solutions to Post's problem. They lead us to explore notions of *lowness.* For example it can be shown that α is K-trivial iff α is low for K in the sense that it does not compress anything: $K^\alpha(x) = K(x) + O(1)$ for all x. This is the tip of the iceberg. There are also many other lowness notions we explore. These ideas are also explored in Nies' recent monograph called *Computability and Randomness.*

We also include analyses along the themes of calibrating randomness by various pre-orderings. Solovay reducibility is one, but you could ask that $\alpha \leq \beta$ if the initial segment complexities align in some way. For example, we can define $\alpha \leq_K \beta$ iff $K(\alpha \upharpoonright n) \leq K(\beta \upharpoonright n) + O(1)$. It makes sense that if a real is random iff $K(\alpha \upharpoonright n) \geq n - O(1)$ for all n, then the pre-ordering above should define some notion of relative randomness. Indeed we now know, that $\leq K$ can also be used to define higher levels of randomness such as randomness relative to the halting set and its iterates. You can also try to calibrate relative randomness according to derandomization power. Now $\alpha \leq_{LR} \beta$ if everything β makes non-random is also made nonrandom by α. How do such measures relate to each other and how do they relate to computational complexity? Myriad questions suggest themselves and there have been a lot of deep and unexpected results proven here.

The last part of our monograph is devoted to looking at other related ideas. For example, in the same way that classical Hausdorff and other dimensions refine measure 0, we can look at effective versions following the lead of Jack Lutz. We have a long section devoted to understanding algorithmic dimensions. For example, we address questions like: I have some source of partial randomness, say a real of dimension $\frac{1}{2}$, and ask can I extract a real of dimension 1 from this by some effective means? This question has a beautiful negative answer by Joe Miller, and we include this and other related results in and around partial randomness as defined via notions of effective dimension like Hausdorff, packing, box counting and the like. We even include Lutz's notion of dimension for *finite* strings. Additionally, we look at the notion of the halting probability as an operator, showing that almost every random real is a halting probability relative to some oracle. This is a contrast to Stephan's result that suggests that halting probabilities should be rare as they have high computational power. The reconciliation is that relativization here does not involve coding. Finally we look at fundamental sets like the collection of non-random strings and ask what kind of computational power these sets of strings have as oracles, according to differing access mechanisms. Such results have impact in complexity theory as they seem related to separation questions of complexity classes following the work of Allender and his co-authors. The biggest part of the story missing in our book is the part of the randomness story low down in classes like EXP or the like. There simply was not space.

What we left out, and would have liked to include

Actually one reviewer asked us to give an account of what we did *not* include in the 870 or so pages which constitute the book. I did prepare an account and sent it to Denis, but in the end we decided that we did not have enough time to polish the material[4]. However, I will list what I had proposed below, and it should give you a perhaps idiosyncratic view of areas of randomness I would like to explore, and see not covered by our monograph.

The concept of randomness has held a central place in mathematics and computer science in recent years. Whilst Hirschfeldt and I feel somewhat (completely?) inadequate in our knowledge of all of the recent developments we offer a few. In most cases, there is, as yet, no applications to or of algorithmic randomness. In most practical applications of randomness in algorithms, it seems that only weak random sources seem to suffice *in practice.* For example, the advent of the Metropolis-Hastings algorithm in Markov Chain Monte Carlo algorithms has revolutionized Bayesian statistics in recent years, and is now the mainstay of applications of statical applications in science. A comprehensive account of these applications to wildly diverse areas of mathematics can be found in Diaconis (Bulletin of the AMS article), and it would take another monograph to describe applications of this methodology to physics and biology. There is certainly no current work in algorithmic randomness trying to speak to this material and it would rather nice to see such a development.

[4]For a mixed metaphor, we needed to kill the albatross.

In number theory, Terry Tao and others have made profound advances in the theory of distribution of the primes by the hypothesis that the primes are random. Now of course the primes are not random, but the intuition of generations of number theorists is that they are *random enough*, once you discount all the obvious facts about them. For instance, long ago van der Waerden proved that if you two colour the integers then one or the other colour will have arbitrarily long arithmetical progressions. In a very deep advance, Szemeredi proved that “random-ish” or “big” sets, sets of positive upper density[5], have arbitrarily long arithmetical progressions. Now all known proofs of this fact filter through a lemma which says that either a big set resembles a random set, or it is highly structured and in either case there will be long arithmetical progressions.

Using this idea, based in and around Szemeredi's techniques and their heirs, Green and Tao proved that the primes contain arbitrarily long arithmetical progressions. An abstract view of this methodology is provided by the *Dense Model Theorem* of Green, Tao and Ziegler; with an interesting view being presented by Trevisan, Tulsiani, and Vadhan in the Proceedings of the Annual Conference on Computational Complexity, 2009. It would be extremely interesting to try to make all of this precise, but the mathematics seems very deep. A very readable discussion of this can be found in Tao's talk at the Madrid International Conference of Mathematicians, and his “blog book”.

In some sense we have seen similar ideas applied in combinatorial group theory where people have looked at average and *generic* case behaviour of decision problems for groups. That is for generic complexity, we can look at algorithmic which give correct answers on a positive upper density version (usually density 1) of finitely generated groups. In generic case complexity, we ask that the algorithm only be partial, always give the correct answer, and halt with upper density 1. The original work can be found in a paper in the Journal of Algebra by Myasnikov, Kapovich, Schupp and Shpilrain. A typical theorem says that in a given finitely presented group, the word problem is linear time generically computable. This accords strongly with computer experiments. Much of this work relies on the fact that almost all groups are hyperbolic as shown by Gromov. Such groups are “almost free” in the sense of the work of Martin Bridson, and the logic has been analysed by the wonderful work of Sela. A small amount of the computability here has been developed. The general theory of such structures remains to be developed.

It is clear that it all seems related to Ergodic Theory, certainly in the case of additive number theory and Ramsey Theory. This insight was first realized by Furstenberg. One can obtain things like van der Waerden's Theorem using methods from ergodic theory which clearly can be viewed as involving randomness. A good discussion of this can be found in in the work

[5]That is sets of integers A such that $\lim_{n\to\infty}\frac{\{z : z\in A\wedge z\leq n\}}{n} > 0$.

of Avigad. There has been some work quantifying the amount of randomness needed for the Ergodic Theorem to work by Reimann and by Hoyrup and Rojas. It seems that it is sensitive to the formulation, and either involves Schnorr randomness or 2-randomness.

Again the harmononious interactions within mathematics are revealed in that the Tao work and the Ergodic Theorem are related to results in analysis, such as the Lebesgue theorem saying that monotone functions are differentiable almost everywhere as articulated by Tao in his blog. The constructivist, Oswald Demuth, had a programme where he argued that random functions should behave well. Theorems such as the Lebesgue theorem beg for analysis in terms of algorithmic randomness, and recently Demuth's programme has been extended Brattka, Miller and Nies. They show that the level of randomness needed to obtain differentiability of a computable continuous function on the reals is precisely computable randomness. That is, every computable continuous function of bounded variation on the real interval is differentiable at each computably random point, and conversely there are computable monotone continuous functions on the interval differentiable only at the computably random points.

Notice that to even state such theorems we need the language of computable analysis. To treat such interesting results would have required Hirschfeldt and I to to develop quite a bit of computable analysis. Certainly to develop at least as much as long initial segments of the classic books of Albeth, Pour-El and Richards or Weihrauch. It is interesting that our original work in the area of algorithmic randomness appeared in computable analysis conferences, yet so far there have not been many applications to analysis. It seems that Demuth's programme has a number of significant implications for computable analysis, and believe there will be a lot more work in this area. The fact that it connects to ergodic theory is the icing on the cake. Perhaps also related is the recent work of Braverman and Yampolsky on computability and Julia sets. Again in that work left c.e. reals play a prominent role, and hence it is all related to algorithmic randomness. We can only say that we did not have enough space to treat such material properly. Similar comments apply to applying algorithmic randomness to fundamental concepts of physics. A very nice recent example is the work of Kjos-Hanssen and Nerode, and Kjos-Hanssen and Gács on Brownian motion and thermodynamics. Again not only would Hirschfeldt and I have needed to develop the computable analysis, additionally we would be forced to develop some more high powered probability theory. All of this would clearly have taken many pages to develop and Hirschfeldt and I leave this to Volume 2 (joke).

In our book we treat algorithmic randomness on Cantor space. This is measure-theoretically identical with the real interval, but not homeomorphic to it. A great omission is the treatment of other spaces, particularly non-compact spaces. In the case of the reals, we have a natural measure we can use, namely Weiner measure. But in more general cases, the situation is reasonably murky. Again a rigorous development requires a substantial amount of analysis and computable analysis. It would go significantly beyond the scope of the book. Very recent work of Hoyrup, Rojas and Gács bases the whole thing on what are called Scott domains. This is another area which is ready for development.

As a "big picture" view of algorithmic randomness in more general spaces, it seems that analogs of the basic results seem to work. As a "big picture" view, it seems that analogs of the basic results seem to work. These results include existence of uniform tests, neutral measures, conservation of randomness. The article of Gács 2005 Theoretical Computer Science article has a good overview of the history. At Gács homepage there is a treatment of this material in a set of online lecture notes.

Given the focus within our book on the interactions of randomness and computability, Cantor space is a very natural domain. Even in the case of Cantor space, there is more to be said in terms of non-uniform measures. We allude to some of this when we discuss the work of Levin, Kautz, Reimann and Slaman. But the recent deep work of Reimann and Slaman on *never continuously random* reals would need a substantial treatment of set theory to do them justice and so in our book Denis and I only state some results, and but do prove some of those that don't need overtly set-theroretical methods. This is deep and important work I think.

In computational complexity theory there has been a number of major developments regarding randomness[6]. To even state the results would need quite a bit of development and we will give a breezy overview, referring the reader to the (mildly) cited works for more details. referring the reader to the cited works for more details. To treat these applications would need at least another book. One strand began with the work of Valiant-Vazirani who showed that SATISFIABILITY reduces to UNIQUE SATISFIABILITY (i.e. at most one satisfying assignment) under randomized polynomial time reductions. This attractive theorem led to Toda's Theorem showing that $\mathrm{P}^{\#P}$ contains the polynomial time hierarchy. Here $\#P$ is the counting analog of NP where we count the number accepting paths in a polynomial time Turing machine. Toda's Theorem states that if we can have the advice of such a counting oracle, then we can solve any problem in the polynomial time hierarchy. All known proofs of Toda's Theorem work through the operator BP, which is the probability version of NP, namely that most paths lead to the correct answer.

Another strand leads to two important generalizations of the notion of "proof": interactive proofs and probabilistically checkable proofs. NP can be viewed as the class of sets A such that a prover can provide a short "proof of membership" for any element of A, which can be verified deterministically. A more general notion arises if one allows the verifier to be a

[6]I thank Eric Allender for some thoughts and corrections in this account

probabilistic process; allowing this probabilistic verifier to hold a conversation with the prover (and allowing the verifier to be convinced by statistical evidence) leads to *interactive proofs* (and the class IP consisting of those languages where the prover can convince the verifier about membership). Initially, people suspected that IP would be a "small" augmentation to NP (for instance, there are oracles relative to which IP does not contain coNP by results of Fortnow and Sipser), and thus it came as a surprise when Babai, Fortnow, Lund, and Nisan showed that IP contains $P^{\#P}$ (and hence contains the polynomial hierarchy, by Toda's theorem). Shamir subsequently showed that IP is equal to PSPACE. The key step in the proof is to translate a PSPACE problem into a problem about polynomials over a finite field (via a process called *arithmetization*). The point of the large field is that two polynomials can intersect only rarely if they are not identical.

In an IP protocol, the prover is allowed to give different responses to identical queries from the verifier, along different computation paths. In the so-called multi-prover interactive proof model (MIP), this power is essentially taken away; this actually results in a *more* powerful system. Babai, Fortnow, and Lund showed that MIP = NEXP. This is extremely counterintuitive, because it means that an *exponential-size* proof can be verified by a probabilistic verifier who has only enough time to ask about a small part of the proof. The next step (in some sense, a "minimization" of the MIP=NEXP characterization) was an analogous characterization of NP in terms of Probabilistically Checkable Proofs; this is known as the PCP theorem of Arora, Safra, and Arora, Lund, Motwani, Sudan, Szegedy, and it is recognized as one of the crowning achievements of computational complexity theory (even thought it is only really a *reduction*!). The PCP Theorem states, roughly, that any problem in NP has a proof which is polynomial in length and can be verified in polynomial time with arbitrary high accuracy by checking a constant number of bits and running a logarithmic number of random tests on the proof. Stated another way, this says that *any* proof can be encoded efficiently in such a way, so that a referee need not read the entire proof, but can simply pick, say, 15 random bits in the paper and see if these bits satisfy some local consistency condition, and certify the proof as "correct" if no error is detected in these 15 bits; no correct proof will be rejected in this way, and the probability that an incorrect proof is accepted is less than the probability that an incorrect proof is accepted in the traditional method of journal refereeing.

The use of randomness has seen a lot of work to try to understand how it is possible to get sources of randomness. One idea is to take some weak source and try to extract nearly true randomness from it, or to extend the length of the source whilst still hoping to have the output "look random". This is an area of much deep work and Hirschfeldt and I briefly touch on this material when we look at packing and Hausdorff dimensions. The methods here are very sophisticated and again would need a lot of deep mathematics developed to treat them properly. Proper utilization of pseudo-random sources is especially important in cryptography.

Much work in probabilistic algorithms is centered on tasks for which we we have randomized algorithms, such as deciding polynomial identities, but for which we have absolutely no idea how to construct deterministic polynomial time algorithms. However, recently it has become apparent that perhaps this is merely because of our lack of brain power, rather than because deterministic algorithms are inherently weaker. That is, it is widely believed that BPP=P. Namely anything we have a randomized polynomial time algorithm for can actually be derandomized. This has happened in the case of primality testing, where the Solovay-Strassen algorithm gave a randomized algorithm for primality and subsequently Agrawal, Kayal, and Saxena gave a deterministic algorithm. Now it it is widely believed that not only is SATISFIABILITY hard, but *very hard,* in that it requires circuits of nearly exponential size (that is, for each input length, it is conjectured that the smallest circuit computing the function correctly for that input length has size $2^{\epsilon n}$ for some $\epsilon > 0$. Remarkably, Impagliazzo and Wigderson have shown that if there is any problem in Dtime(2^n) (such as SATISFIABILITY that requires circuits of size $2^{\epsilon n}$ then P=BPP; that is, *all* probabilistic algorithms can be derandomized.. Again it is not clear what the intuition really is connecting nonuniform complexity and randomness. Again this is way, way beyond the present book and we refer to Kozen's book for basic material and Wigderson's article in the proceedings of the Madrid International Congress of Mathematicians for an excellent overview.

Another nice application of the ideas of effective measure in computational complexity comes from the work of Lutz and his co-workers. The idea here is that if we believe, say, that P≠NP then we should quantify how big P is within NP. Using these ideas, those authors have shown that various measure theoretical hypotheses on the structure of complexity classes in terms of small or large measure have significant consequences for separations. Here we refer to Lutz's or Hitchcock's web page for references and more details.

There are myriad other applications in computational complexity, not the least of which are those where combinatorial arguments are replaced by Kolmogorov complexity ones. For instance, constructing "hard" inputs on which any simple algorithm must make an error can be difficult, but frequently it is easier to show that all simple algorithms must make an error on *random* inputs. I think the intuition here is the following: We have some algorithm we wish to show has certain behaviour. We use that fact that a string of high Kolmogorov complexity should behave in the expected way as the process is algorithmic. If it did not then it could be compressed.

For example, we can use this idea to prove worst case running times for various sorting algorithms, or to prove the Hastad switching lemma. There are many, many applications of this form as can bee seen in in Li and Vitányi's book. Li and Vitányi's book is also an excellent source of other applications of Kolmogorov complexity to things like thermodynamics, computational learning theory, inductive inference, biology and the like. Again Hirschfeldt and I though this was beyond our ken and space or time available.

Another area we did not develop is the time and space bounded versions. This goes back to work of Levin on Kt complexity, and we refer to the excellent surveys of Allender here for more details and to Li and Vitányi's book for background and historical results.

We remark that Vitányi and other authors have used the notion of Kolmogorov complexity to explore the common information between strings in analysis of things in real life. For example in computational biology to try to compare two phylogenetic trees we would invent metrics such as what is called Maximum Parsimony, and Maximum Likelihood. The idea is to replace this with a more general measure of normalized relative Kolmogorov complexity. Of course this is great, except that computing the complexities is undecidable as we see in the present book. I applications the idea has been to use commercial text compression packages like GZIP and see what happens. Vitányi and his co-authors have has some success with music evolution. Again more works need to be done.

Finally while Hirschfeldt and I did develop some aspects of the Kolmogorov complexity of finite strings they were in support of our aims. It is certainly the case that there is a lot more, especially from the Moscow school flowing from the students of Kolmogorov. We can only point at the work of Shen, Muchnik, Vereshchagin, Uspenskii, Zimand and others.

The most important open problems

What we have is a vast enterprise devoted to the themes above. The picture is still emerging with a number of very interesting technical questions still open. For example, if we allow "nonmonotonic" betting strategies, does computable randomness in this new sense coincide with Martin-Löf randomness? Is even a little bias allowable, etc? By this I mean suppose that I have a betting strategy which at some stage might favour some side, and if so is then committed to favouring that side. Is the resulting notion of randomness the same as Martin-Löf randomness[7]? As mentioned above, what about extensions to non-compact spaces such as the work of Peter Gács, and how should we understand independence as suggested by seminal works of Levin? (Actually, even reading some of those early works of Levin is challenging enough.) There are many questions in and around randomness, independence, computability and you should look at the survey of Miller and Nies or at either Nies' or our book here. Of course there are fundamental questions in complexity such as the conjecture that $BPP = P$. That

[7]The point here is that the usual proof that approximable effective betting strategies gives the same randomness as the Martin-Löf definition allows us first to favour $\sigma 0$ and the perhaps later we might choose to favour $\sigma 1$.

is, there are a host of algorithms which efficiently solve problems using randomness as a resource. It seems that all of these can be derandomized, or at least that is the current thinking of people working in complexity. How should this be proven? What are the languages efficiently reducible to the collection of non-random strings. We know it contains PSPACE but is the intersection of such languages the same? Is the definition even machine independent? There are a host of questions, and I would hope they would be articulated by other contributors to this volume.

The prospects for progress

In the future I would hope that this machinery would see applications in our understanding of things like physics and other sciences, as well as other parts of mathematics. We live in a world where most processes are finite and their approximations are computable. Thus the notion of expectation aligns itself to Kolmogorov complexity, as we mentioned in the beginning. Thus, one would think that algorithmic randomness should be an excellent approximation to "true" randomness whatever that means, quantum or otherwise. Surely we could use algorithmic randomness to better understand physical processes. As mentioned above, Kjos-Hanssen and Nerode have some initial forays on this in Brownian motion. There is also work by people like Vitányi and his group on things like phylogeny and sequence matching using approximations to Kolmogorov complexity as a measure of common information rather than using things like minimum distance etc. The idea is that rather than figuring out the best metric to see how similar two things are, use the fact that something like a normalized version of the relative Kolmogorov complexity is a universal notion of information distance. The problem, of course, is that it is not computable. But we might be able to use compression algorithms like ZIP or the like to approximate. This method apparently works well for evolution of music. Even biological processes would seem to need randomness to understand them. But here I am into the world of wild speculation. Information is everywhere, and we are developing tools to try to understand it. I think the future of this area is fascinating.

School of Mathematics, Statistics and Computer Science, Victoria University, PO Box 600 Wellington
http://homepages.ecs.vuw.ac.nz/~downey
E-mail address: Rod.downey@vuw.ac.nz