Synthetic Kolmogorov Complexity in Coq

Yannick Forster
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Inria, Gallinette Project-Team, Nantes, France

Fabian Kunze
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Nils Lauermann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
University of Cambridge, Cambridge, United Kingdom

Abstract

We present a generalised, constructive, and machine-checked approach to Kolmogorov complexity in the constructive type theory underlying the Coq proof assistant. By proving that nonrandom numbers form a simple predicate, we obtain elegant proofs of undecidability for random and nonrandom numbers and a proof of uncomputability of Kolmogorov complexity.

We use a general and abstract definition of Kolmogorov complexity and subsequently instantiate it to several definitions frequently found in the literature.

Whereas textbook treatments of Kolmogorov complexity usually rely heavily on classical logic and the axiom of choice, we put emphasis on the constructiveness of all our arguments, however without blurring their essence. We first give a high-level proof idea using classical logic, which can be formalised with Markov's principle via folklore techniques we subsequently explain. Lastly, we show a strategy how to eliminate Markov's principle from a certain class of computability proofs, rendering all our results fully constructive.

All our results are machine-checked by the Coq proof assistant, which is enabled by using a synthetic approach to computability: rather than formalising a model of computation, which is well known to introduce a considerable overhead, we abstractly assume a universal function, allowing the proofs to focus on the mathematical essence.

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Type theory

Keywords and phrases Kolmogorov complexity, computability theory, random numbers, constructive matemathics, synthetic computability theory, constructive type theory, Coq

Digital Object Identifier 10.4230/LIPIcs.ITP.2022.12

Supplementary Material *Software (Source Code)*:
`https://github.com/uds-psl/coq-synthetic-computability/tree/kolmogorov`
archived at `swh:1:dir:172c753027deabef87ce303e73dc98fc14745a89`

Funding *Yannick Forster*: received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101024493.

Acknowledgements We want to thank Dominik Kirst and Gert Smolka for many fruitful discussions.

1 Introduction

Kolmogorov complexity[1] as a subject of computability and information theory is usually presented in the abstract rather than using spelled-out definitions (e.g. [42, 30]). This is no oversight: Kolmogorov complexity is a general concept, invariant in the concrete definition as long as some structural properties are fulfilled. The Kolmogorov complexity of a number x is the length of the shortest bitstring s s.t. decoding s terminates with x, based on a certain decoding function called *description mode* (which can and will be left abstract at first).

[1] The origins of the complexity notion are complicated, see Li and Vitányi's book for an overview [30, §1.13]. It is often also called Solomonoff-Kolmogorov-Chaitin complexity, crediting the influential work by Ray Solmonoff [44, 45, 46], Andrej Kolmogorov [24, 25], and Gregory Chaitin [6]

In terms of (the abstraction away from) details, presentations of Kolmogorov complexity are well in line with presentation of other areas of computability theory, which are also usually given conceptually rather than with spelled out definitions. This is due to extensive use of the Church-Turing thesis, stating that every intuitively computable function is in fact computable in a fixed (and thus any) Turing-complete model of computation. Thus, presentations of computability theory can abstract away from the chosen model, and definitions could be but are not spelled out using many different models like the λ-calculus, Turing machines, μ-recursive functions, counter machines, or more elaborate models like `while`-based simply typed programming languages.

Presentations of Kolmogorov complexity take generality one step further: Not only could they be instantiated to various models of computation, even the concrete definition of a description mode can be left abstract and instantiated to multiple possible definitions per model of computation.

Recently, one such particular definition for the model of the full λ-calculus was given by Catt and Norrish [5] in HOL4. They prove various inequalities w.r.t. Kolmogorov complexity and that Kolmogorov complexity is not a computable function. Even the formal definition of Kolmogorov complexity in their setting is non-trivial and requires both classical logic (i.e. a variant of the law of excluded middle) and a unique choice operator. Subsequently, various constructions on the λ-calculus become relevant, which they acknowledge to be tedious.

This tedium is well known amongst different machine-checked proofs in computability theory: Results like Rice's theorem [39] or the existence of universal machines are feasible to machine check, but more involved results like the Myhill isomorphism theorem [32] (one-one equivalence gives rise to a recursive isomorphism), the existence of Post's simple and hypersimple sets [37] (there are enumerable, undecidable many-one/truth-table degrees different from the halting problem), the Kleene-Post theorem [22] (stating that there are incomparable Turing degrees), solutions to Post's problem [31, 18] (there is an enumerable, undecidable Turing degree different from the halting problem) or Post's theorem [38] (relating Turing jumps and the arithmetical hierarchy[2]) are out of reach for direct translations of the textbook setting based on models of computation to proof assistants: Dealing with concrete programs in low-level models of computations outweighs the (also involved) mathematical arguments underlying the proofs to a level that mechanisation becomes infeasible.

To circumvent the tedium of models of computation for machine-checked proofs, *synthetic* approaches have proved successful: Synthetic undecidability has allowed machine-checked undecidability proofs for various landmark results [15, 28, 21, 9, 8], based on the observation that a problem is undecidable if an assumed Coq function deciding the problem could be used to construct e.g. a decider of the Turing machine halting problem. For abstract results in computability, Forster [13] presents an axiomatic setting allowing to prove Rice's theorem. Forster, Smolka, and Jahn [14] work in this setting and construct simple and hypersimple sets, allowing to prove that one-one, many-one, and truth-table reducibility all differ.

In this setting, based on the axiom *Synthetic Church's Thesis*, one assumes a function $\phi\colon \mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{N}$, where a call $\phi_c x$ morally represents the c-th program in a fixed model of computation evaluated on input x, but the definition of ϕ is not explicitly given, and a *universality property* of ϕ w.r.t. all functions of type $\mathbb{N} \rightharpoonup \mathbb{N}$, i.e. that $\forall f\colon \mathbb{N} \rightharpoonup \mathbb{N}.\exists c.\forall x.\phi_c x \equiv fx$, and an S_n^m operator for ϕ like in the S_n^m theorem.

The observation that computability theory can be developed in such a setting goes back to Richman and Bridges [40, 3], who develop synthetic computable analysis in Bishop-style constructive logic, and has also been employed by Bauer [1, 2] in the internal logic of

[2] The given reference is usually used for Post's theorem, but just constitutes a short abstract containing neither the theorem or a proof. Standard textbook references give modern proofs [41, 34].

the effective topos. In contrast to the work of Richman, Bridges, and Bauer, developing synthetic computability in the constructive type theory underlying the Coq proof assistant is compatible with classical logic, i.e. the law of excluded middle can be assumed alongside Synthetic Church's Thesis without immediately introducing an inconsistency.

Synthetic computability in the Calculus of Inductive Constructions (CIC [7, 35, 36]), the constructive type theory underlying the Coq proof assistant [47], is based on two facts: First, that all functions definable in CIC can be shown computable in CIC (thus allowing the assumption of a universal function, implementable as the universal μ-recursive function). Secondly, that CIC has a strict separation between function spaces such as $\mathbb{N} \to \mathbb{N}$ or $\mathbb{N} \rightharpoonup \mathbb{N}$ and logic, carried out in the impredicative universe of propositions $\mathbb{P}$. Thus, assuming the law of excluded middle (LEM) does not leak into the function space and does not allow the definition of an uncomputable function. In this setting, one has the choice between three equivalent axioms to develop results: SCT, based on total functions $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$, EPF, based on partial functions $\mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{N}$, and EA, based on parametrically enumerable predicates $\mathbb{N} \to \mathbb{N} \to \mathbb{P}$. Forster, Jahn, and Smolka [14] use EA to construct Post's simple predicate.

Coming back to Kolmogorov complexity, the noncomputability of Kolmogorov complexity can also be proved by showing that nonrandom numbers (numbers which can be efficiently described, i.e. which are longer than their Kolmogorov complexity) form a simple predicate. We give a machine-checked proof of this result using the definition of simple predicates by Forster, Jahn, and Smolka [14] based on the axiom EPF [13], which seems most natural for Kolmogorov complexity due to the frequent use of partial functions.

We structure the paper to be instructive for similar future endeavours: First, we give the high-level proof idea in Section 2, without formal definitions and without bothering about the use of classical logic. Then, we introduce the relevant aspects of CIC in Section 3: Total and partial functions (treated as computable functions), partial relations (which are not necessarily computable), total relations (also not provably computable, but where a constructive totality proof is possible if and only if the relation is computable), and classically total relations (which can be proved total using classical logic and are not necessarily computable).

We recap the formalities of synthetic computability in CIC in Section 4 and discuss provable versions of choice principles in Section 5. We then observe that Kolmogorov complexity has to be formalised as a partial, classically total relation and introduce the notion formally in Section 6. In Section 7 we introduce nonrandom and random numbers, suitably formalised to allow a constructive enumerability proof, and show that the nonrandom numbers form a simple predicate. For infinity of random numbers, we rely on the definition by Forster, Jahn, Smolka [14] and explain how to insert double negations in the formalised statements of the intuitive proofs to get rid of uses of the law of excluded middle. To show that nonrandom numbers are immune (a necessary condition to show nonrandom numbers simple) we rely on Markov's principle (MP), which simplifies the proof.

We then discuss the role of MP in Section 8 and show a general technique how the specific way how MP is used in the proof, which also occurs in other proofs, can be eliminated to obtain a fully constructive proof that nonrandom numbers form a simple predicate.

In Section 9 we then instantiate our general definition of Kolmogorov complexity to the definition used by Catt and Norrish (while still staying in the synthetic setting). We also instantiate to a definition similar to the definition used in Odifreddi's book [34] in Section 10.

We claim that we obtain elegant proofs of all results and that our proofs do not lose elegance either due to formalising, due to mechanising them in a proof assistant, or due to presenting them in constructive logic. The machine-checked development[3] is accessible to any Coq user and no knowledge of tedious details regarding models of computation is necessary.

[3] The development is hyperlinked with the present paper, always indicated by a clickable -symbol.

2 Intuition: Nonrandom numbers form a simple predicate

The Kolmogorov complexity $\mathcal{C}(x)$ of a number x is the size of its shortest description. We call a partial function $\delta : \mathbb{N} \rightharpoonup \mathbb{N}$ a *description mode*. We call y a description of x if $\delta y \triangleright x$.

The size of descriptions is the length of its interpretation as bitstring. We write $|n|$ for the length of the bit string representing n. We will assume that the interpretation as bitstring works such that the size of n is at most as long as the logarithm (by 2) of n plus 1.

As a consequence, we obtain the following.

▶ **Fact 1.** *For all d, there is n with $n > |n| + d$.*

▶ **Fact 2.** *There are exactly 2^k descriptions with size k and, $2^k - 1$ descriptions with size less than k.*

Proof. The first follows since there are exactly 2^k bit strings of length k. The second is by usual arithmetical arguments. ◀

We only assume one property of δ, namely that it is optimal up to a constant. I.e. for every computable function δ' there is a constant d s.t.

$$\forall y' x.\ \delta' y' \triangleright x \to \exists y.\ \delta y \triangleright x \wedge |y| < |y'| + d.$$

Using the optimality on $\delta' y := y$ we obtain that δ is surjective:

▶ **Fact 3.** *Every number has a description.*

A number is nonrandom if its shortest description is shorter than the number itself. Formally, $\mathcal{N}(x) := \mathcal{C}(x) < x$. Intuitively (but because we do not go into detail how δ is defined, not formally), the following number is nonrandom:

001 001 001 001 001 001 001 001 001 001 001 001 001 001 001 001 001

This is because it can be written more compactly as "repeat 001 for 17 times". On the other hand, the terminology random seems intuitive when looking at a number like

833 256 052 501 225 207 711 246 773 401 191 432 051 794 357 674 718

We now prove that nonrandom numbers form a simple predicate, a result that first appeared in [50] and is there attributed to Janis Barzdins. A predicate is simple if it is enumerable and its complement is infinite but does not contain an enumerable, infinite subpredicate.

▶ **Lemma 4.** *Nonrandom numbers are enumerable.*

Proof. Try all possible inputs y and evaluate δy for all possible step numbers n. If δy evaluates to a value x in n steps and $|y| < |x|$, then output x. ◀

▶ **Lemma 5.** *Random numbers are infinite.*

Proof. Suppose all 2^k many numbers x_i of size k would be nonrandom. This means that they would have 2^k many distinct descriptions y_i, all with size smaller than k. But there are at most $2^k - 1$ numbers of size smaller than k. Contradiction. ◀

The third part of the definition of simpleness, which is often called immunity, is easiest to explain with some more formulas.

▶ **Lemma 6.** *There is no enumerable infinite sub-predicate of random numbers.*

Proof. Assume such a predicate q. Since q is infinite, in particular, for every k it contains x with $|x| > k$. Now since q is enumerable, we can define a computable function f computing these x, i.e. for all k, fk is in q. Since q is a subpredicate of random numbers, in particular fk is random.

$$\forall k.\ k < |fk|.$$

Since δ is optimal, we have a constant d and descriptions y_k for every fk at most as long as $|k| + d$, i.e.:

$$\forall k.\ |y_k| \leq |k| + d.$$

Since fk is random, its size is shorter than its shortest description, and in particular it is shorter than its description y_k, i.e.

$$\forall k.\ |fk| \leq |y_k|.$$

But now $\forall k.\ k < |fk| \leq |y_k| \leq |k| + d$, a contradiction to Fact 1. ◀

▶ **Corollary 7.**

1. *Nonrandom numbers are simple.*
2. *Both random and nonrandom numbers are undecidable.*
3. *Kolmogorov complexity is noncomputable.*

Proof. Simple predicates can be shown undecidable [14]. If there would be a function f computing Kolmogorov complexity, $\lambda x.fx < |x|$ would decide nonrandomness. ◀

3 Constructive Logic in Coq's Type Theory

In the previous section, we informally defined Kolmogorov complexity as the size of the *least* description. Since every number has a description, Kolmogorov complexity was treated as a total function. Later, we showed that this total function is however not computable.

When we now want to define Kolmogorov complexity in CIC, we cannot define it as a function, because all definable functions in CIC always are, by definition, computable. Thus, we will define Kolmogorov complexity as a functional relation. While in set theory the notion of a function and a functional relation are conflated they are strictly separate in CIC.

We make use of this separation and introduce the following general leastness-predicate transformer: For every predicate $p : \mathbb{N} \to \mathbb{P}$ one can define the predicate $y\ is\ (\boldsymbol{\mu} x.\ px)$ being satisfied exactly if y is the least number satisfying p.

$$y\ is\ (\boldsymbol{\mu} x.\ px) := py \wedge \forall y'.\ py' \to y \leq y'$$

We can then prove that the predicate is functional.

Fact 8. $y_1\ is\ (\boldsymbol{\mu} x.\ px) \to y_2\ is\ (\boldsymbol{\mu} x.\ px) \to y_1 = y_2$

As we have defined it, we cannot prove that whenever p is satisfied there is a least element satisfying it. To see this, we can look again at Kolmogorov complexity: After defining Kolmogorov complexity using $\boldsymbol{\mu}$, we could then show that Kolmogorov complexity is a total relation. This, however, is impossible: A constructive totality proof $\forall x.\exists y.$ would correspond via the Curry-Howard interpretation to a definable function computing Kolmogorov complexity, which we know cannot exist because Kolmogorov complexity is uncomputable.

Since we know that a totality proof cannot be constructive, we will use classical totality: A relation $X \to Y \to \mathbb{P}$ is classically total if $\forall x.\neg\neg\exists y.\ Rxy$.

When proving the $\neg\neg\exists$ part of classical totality, one can use classical logic. This is because one can think of double negation as a modality with the following proof rules.

Fact 9. *The following hold:*

1. $P \to \neg\neg P$
2. $\neg\neg P \to (P \to \neg\neg Q) \to \neg\neg Q$
3. $(P \vee \neg P \to \neg\neg Q) \to \neg\neg Q$

Item 1 states that the double negation modality can be left at any time (it has the type of the return of a monad). 2 states how to combine different double negated proofs (it has the type of the bind of a monad). 3 states that in the double negation modality, arbitrary case disctinctions are allowed.

We can indeed prove that whenever p is satisfied it does not not have a least element:

Fact 10. $py \to \neg\neg\exists y.\ y\ is\ (\boldsymbol{\mu} x.\ px)$

The well-known axioms of the law of excluded middle LEM and Markov's principle MP can be thought of as extending the proof rules for the double negation modality.

$$\mathsf{LEM} := \forall P{:}\mathbb{P}.\ P \vee \neg P \qquad \mathsf{MP} := \forall f{:}\mathbb{N} \to \mathbb{B}.\ \neg\neg(\exists n.\ fn = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$$

MP allows entering the double negation modality whenever the goal is a Σ^0_1 formula. The following states that LEM allows entering the double negation modality for every goal.

Fact 11. $\mathsf{LEM} \to \neg\neg Q \to Q$

Using LEM, one can prove that least elements for inhabited predicates always exist. Furthermore, in constructive mathematics, it is standard practice to also analyse whether made classical assumptions are necessary, and in this case one can indeed prove that the existence of least elements for inhabited predicates implies LEM.

Lastly, we need a good, constructive definition of infinite predicates. The definition cannot be too weak, which might mean that we cannot show that simple predicates are undecidable. The definition cannot be too strong either, which might mean that it is impossible to establish random numbers to be infinite. We can here just follow Forster, Jahn, and Smolka [14] in using non-finite predicates. We use the type of lists over X, written $\mathbb{L}X$, with [] being the empty list and $x :: l$ being the list with element x added to a list l. A predicate $p{:}X \to \mathbb{P}$ is *finite* if $\exists l{:}\mathbb{L}X.\ \forall x.\ px \leftrightarrow x \in l$. A predicate $p{:}X \to \mathbb{P}$ is *subfinite* if $\exists l{:}\mathbb{L}X.\ \forall x.\ px \to x \in l$. A predicate is *non-finite* if it is not finite.

Fact 12. *A predicate is non-finite if and only if it is not subfinite.*

Non-finiteness for predicates on natural numbers can be expressed as a $\forall\neg\neg\exists$ version of having elements of arbitrary size.

Fact 13. *Let $f{:}\mathbb{N} \to \mathbb{N}$ be a size function which assigns only finitely many elements the same size.*

A predicate $p : \mathbb{N} \to \mathbb{P}$ is infinite if and only if

$$\forall n.\ \exists x.\ fx \geq n \wedge px.$$

4 Synthetic Computability Theory

Computability theory is usually presented with respect to an effective enumeration ϕ of all computable function, i.e. ϕ_c denotes the c-th computable function, and for every computable function f there is a code c s.t. f and ϕ_c agree. By appeal to the (informal) Church Turing thesis, ϕ has the following informal universal property:

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\ f \text{ is intuitively computable} \to \exists c. \forall x v.\ \phi_c x \rhd v \leftrightarrow f x \rhd v.$$

In synthetic computability theory [13], one uses essentially the same approach, but (1) abstracts against an abstract function ϕ rather than a concrete enumeration, and (2) assumes all functions $\mathbb{N} \rightharpoonup \mathbb{N}$ of the meta-theory to be intuitively computable. Thus, we obtain an axiom assuming a function $\phi : \mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{N}$ with the following formal universality property:

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\ \exists c. \forall x v.\ \phi_c x \rhd v \leftrightarrow f x \rhd v$$

This axiom is called EPF in [11] and is a consequence of the axiom EPF in [13].[4] Consistency of such an axiom is a consequence of the consistency of the axiom CT in CIC, see [13, Appendix A] and [11].

The first simplification entailed by synthetic computability becomes obvious when defining standard notions of computability, which do not need to rely on a model of computation [15]. A predicate $p : X \to \mathbb{P}$ is

- *decidable* if there exists a decider: $\exists f : X \to \mathbb{B}. \forall x.\ px \leftrightarrow fx = \mathsf{true}$
- *enumerable* if there exists an enumerator: $\exists f : \mathbb{N} \to \mathbb{O}X. \forall x.\ px \leftrightarrow \exists n.\ fn = \mathsf{Some}\, x$

Here, $\mathbb{O}X$ is the type with constructors None and Some x where $x : X$ and $\mathbb{B}$ has elements true and false.

The intuition behind decidability should be immediately clear. Enumerability captures the definition that a predicate is "r.e." ("recursively enumerable") if it is the co-domain of a function. Equivalent domain-based definitions can be given [12] but we do not require them.

If a predicate p is enumerable we write $\mathcal{E}p$. We write $\mathcal{E}R$ for a relation $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ if the predicate $\lambda(x, y).\ Rxy$ is enumerable. We call a type X *discrete* if $\lambda(x_1, x_1).\ x_1 = x_2$ is decidable and *enumerable* if $\lambda x : X.\ \top$ is enumerable.

One can give various definitions of partial functions in CIC with enumerable graph [12, §4.5]. We abstractly use a function type $\mathbb{N} \rightharpoonup \mathbb{N}$ and write $fx \rhd y$ if $f : \mathbb{N} \rightharpoonup \mathbb{N}$ on input x evaluates to y. This type can for instance be inhabited by stationary functions $\mathbb{N} \to (\mathbb{N} \to \mathbb{O}\mathbb{N})$, i.e. where $fxn_1 = \mathsf{Some}\, x \to \forall n_2 \geq n_1.\ fxn_2 = \mathsf{Some}\, x$. We just need the following fact and only need more constructions for partial functions in Fact 37.

Fact 14. *One can define an enumeration function* graph : $(\mathbb{N} \rightharpoonup \mathbb{N}) \to \mathbb{N} \to \mathbb{O}(\mathbb{N} \times \mathbb{N})$ *s.t.*

$$\forall f x y.\ fx \rhd y \leftrightarrow \exists n.\ \mathsf{graph}\, f n = \mathsf{Some}\, (x, y).$$

Following Forster, Jahn, and Smolka [14] one can also define reducibility, e.g. many-one, one-one, or truth-table reducibility. We refrain from doing so here and just use their definition of simpleness. A predicate $p : \mathbb{N} \to \mathbb{P}$ is *simple* if

1. p is enumerable,
2. the complement of p is not finite,
3. the complement of p has no non-finite, enumerable sub-predicate q (i.e. $\forall x.\ qx \to \neg px$).

[4] This is not a typo, both axioms are unfortunately indeed named the same.

5 Choice Functions

In computability theory, unbounded search is a crucial tool to define functions. In CIC, we have two options for unbounded search: First, we can implement an unbounded search operator yielding partial functions (e.g. of type $\mathbb{N} \rightharpoonup \mathbb{N}$). Secondly, we can make use of the fact that a choice principle for enumerable relations is provable. This means that for enumerable relations (often also called Σ_1^0-relations) one can extract a function from a totality proof.

Fact 15. *Every total enumerable relation $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ has a choice function:*

$$\mathcal{E}R \to (\forall x.\exists y.\ Rxy) \to \exists f.\forall x.\ Rx(fx)$$

Proof. Due to the fact that whenever $\exists n : \mathbb{N}.\ fn = \mathsf{true}$ also $\Sigma n : \mathbb{N}.\ fn = \mathsf{true}$, discovered independently by Benjamin Werner and Jean-François Monin and part of Coq's standard library as *constructive ground description* operator. ◀

As discussed it is however not always possible to prove totality of a relation without classical logic. For enumerable relations, MP suffices to obtain full classical power for the totality proof.

Fact 16. *Given* MP*, every classically total enumerable relation $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ is total:*

$$\mathsf{MP} \to \mathcal{E}R \to (\forall x.\neg\neg\exists y.\ Rxy) \to \forall x.\exists y.\ Rxy$$

Corollary 17. *Given* MP*, every classically total enumerable relation $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ has a choice function:*

$$\mathsf{MP} \to \mathcal{E}R \to (\forall x.\neg\neg\exists y.\ Rxy) \to \exists f.\forall x.\ Rx(fx)$$

6 Kolmogorov Complexity

Recall that the Kolmogorov complexity of a number x is the size of the least description of x. To formally capture the notion, we need to formally capture *size* and *description*.

We will model descriptions using the type of natural numbers. To define the size of a description we parameterise our development against a bijection between natural numbers and binary strings, i.e. against functions $\lceil\cdot\rceil : \mathbb{N} \to \mathbb{LB}$ and $\lfloor\cdot\rfloor : \mathbb{LB} \to \mathbb{N}$ s.t.

1. $\forall l.\ \lceil\lfloor l \rfloor\rceil = l$
2. $\forall n.\ \lfloor\lceil n \rceil\rfloor = n$
3. $\forall n.\ \mathsf{len}(\lceil n \rceil) \leq \mathsf{log}_2(n) + 1$
4. $\forall xy.\ x \leq y \to \mathsf{len}(\lceil x \rceil) \leq \mathsf{len}(\lceil y \rceil)$

We model binary strings formally as lists of booleans $\mathbb{LB}$, for instance $[]$, $[\mathsf{true}, \mathsf{false}]$, $[\mathsf{false}, \mathsf{false}, \mathsf{false}]$ are then binary strings. We write $\mathsf{len}(l)$ for the length of a binary string.

To see that such a bijection exists, we can define one which identifies a natural number n with the binary expansion of $n + 1$ with the most significant bit left out. I.e. this bijection associates the following numbers and lists:

$$0 \sim [] \qquad 1 \sim [\mathsf{false}] \qquad 2 \sim [\mathsf{true}] \qquad 3 \sim [\mathsf{false}, \mathsf{false}] \qquad 4 \sim [\mathsf{false}, \mathsf{true}] \qquad \ldots$$

It is then routine to prove the above properties, we do not go into detail here.

We now turn back to working with any bijection fulfilling the above four properties and define the *size* of a number n as the length of its interpretation as binary string, i.e.

$$|n| := \mathsf{len}(\lceil n \rceil).$$

We can use concatenation to define a pairing function on natural numbers:

$$\langle x, y \rangle := \lfloor \lceil x \rceil +\!\!+ \lceil y \rceil \rfloor$$

Note that $\langle x, y \rangle$ is surjective, but injectivity or the existence of a (left-)inverse function depend on the concrete definition of $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$, because x and y are not necessarily reconstructible from $\lceil x \rceil +\!\!+ \lceil y \rceil$. We however do not rely on either.

Fact 18. $|\langle x, y \rangle| = |x| + |y|$

Since the size behaves logarithmically, there (classically) exist numbers n which are larger than $|n|$. Formally:

Fact 19. *For all d, $\neg\neg\exists n.\ n > |n| + d$. Equivalently: For all d, $\neg\forall n. n \leq |n| + d$.*

The following fact will become important later, because it restricts the amount of possible descriptions with a certain size.

Fact 20. *There are exactly 2^k distinct lists of booleans x with $\mathsf{len} x = k$:*

$$\exists l{:}\,\mathbb{L}(\mathbb{LB}).\ (\forall x.\ x \in l \leftrightarrow \mathsf{len}(x) = k) \wedge \#l \wedge \mathsf{len}(l) = 2^k$$

We write $\#l$ to indicate that l does not contain any duplicates. One can find various definitions of what constitutes a description of a number x. We will just assume a *description mode* $\delta{:}\,\mathbb{N} \rightharpoonup \mathbb{N}$. As in the intuitive explanation, we will assume that δ is optimal, i.e. that

$$\forall \delta'{:}\,\mathbb{N} \rightharpoonup \mathbb{N}.\exists d{:}\,\mathbb{N}.\forall y' x.\ \delta' y' \triangleright x \to \exists y.\ \delta y \triangleright x \wedge |y| < |y'| + d.$$

As before, using optimality on the identity function, we can obtain that every number has a description.

Fact 21. $\forall x.\exists y.\ \delta y \triangleright x$

In Sections 9 and 10 we give several possible definitions for description modes frequently found in the literature.

We define the Kolmogorov complexity $\mathcal{C}(x)$ of a number x to be the least number s s.t. s is a size of a description of x, i.e. we define a functional relation $\mathcal{C}{:}\,\mathbb{N} \to \mathbb{N} \to \mathbb{P}$ as follows

$$\mathcal{C}(x)s := s \mathit{\ is\ } \boldsymbol{\mu} s.\ \exists y.\ s = |y| \wedge \delta y \triangleright x$$

We also write $s \mathit{\ is\ } \mathcal{C}(x)$ for $\mathcal{C}(x)s$.

Surjectivity of δ suffices to prove that $\mathcal{C}(x)$ is classically total.

Fact 22. $\forall x.\ \neg\neg\exists s.\ s \mathit{\ is\ } \mathcal{C}(x)$

We can also use optimality of δ to prove that a definition of $\mathcal{C}$ using any other description operator δ' yields a similar definition of Kolmogorov complexity, only differing in a constant. In this so-called *invariance theorem* we write $\mathcal{C}_\delta$ and $\mathcal{C}_{\delta'}$ for the two respective definitions of Kolmogorov complexity.

Theorem 23. *If δ is an optimal description mode we have that for every $\delta'{:}\,\mathbb{N} \rightharpoonup \mathbb{N}$ there exists $d{:}\,\mathbb{N}$ s.t. whenever s is $\mathcal{C}_\delta(x)$ and s' is $\mathcal{C}_{\delta'}(x)$ we have $s \leq s' + d$.*

7 Nonrandom Numbers

In this section, we introduce nonrandom numbers and prove that they form a simple predicate. Recall that a number is nonrandom if it has a description shorter than its own size. Formally, we have three options to define nonrandom numbers:

$$\mathcal{N}_1(x) := \exists s.\ s\ is\, \mathcal{C}_\delta(x)\ \wedge s < |x|$$
$$\mathcal{N}_2(x) := \exists y.\ \delta y \triangleright x \qquad \wedge |y| < |x|$$
$$\mathcal{N}_3(x) := \forall s.\ s\ is\, \mathcal{C}_\delta(x) \to\ \ s < |x|$$

Note that we cannot simply use $\mathcal{C}_\delta(x) < |x|$, because $\mathcal{C}$ is not a total function. Options $\mathcal{N}_1$ and $\mathcal{N}_3$ capture the two possible ways to talk about the values of a functional relation. Option $\mathcal{N}_2$ circumvents mentioning $\mathcal{C}$, because if some description y has smaller size than x, then surely the smallest description (i.e. $\mathcal{C}_\delta(x)$) has smaller size as well. All definitions have different constructive strength, but are classically equivalent.

Fact 24. $\mathcal{N}_1(x) \to \mathcal{N}_2(x)$, $\mathcal{N}_2(x) \to \mathcal{N}_3(x)$, $\mathcal{N}_3(x) \to \neg\neg\mathcal{N}_1(x)$.

Proof. Relying on surjectivity of δ. ◀

Since only $\mathcal{N}_2$ can be proved enumerable constructively, we choose it as definition.

$$\mathcal{N}(x) := \exists y.\ \delta y \triangleright x \wedge |y| < |x|$$

Fact 25. *$\mathcal{N}$ is enumerable.*

Proof. Let $f : \mathbb{N} \to \mathbb{O}\mathbb{N}$ enumerate the graph of δ, i.e. $\delta y \triangleright x \leftrightarrow \exists n.\ fn = \mathsf{Some}\,(x, y)$. Then $\lambda n.$ **if** fn **is** $\mathsf{Some}\,(x', y)$ **then if** $x' = x \wedge |y| < |x|$ **then** $\mathsf{Some}\, y$ **else** None **else** None enumerates $\mathcal{N}$. ◀

The complement of nonrandom numbers is formed by *random* or *incompressible* numbers:

$$\mathcal{R}(x) := \forall y.\ \delta y \triangleright x \to\ \ |x| \le |y|$$

We emphasise that, counter-intuitively perhaps given the naming, random numbers are constructively the complement of nonrandom numbers (rather than the other way around):

Fact 26. $\mathcal{R}(x) \leftrightarrow \neg\mathcal{N}(x)$.

Classically, a number is random if it is shorter than or as long as its shortest description, again for surjective description functions:

Fact 27. *Given $\forall x.\exists y.\ \delta y \triangleright x$ we have Let $\mathcal{R}_1(x) := \exists s.\ s\ is\, \mathcal{C}_\delta(x) \wedge |x| \le s$ and $\mathcal{R}_3(x) := \forall s.\ s\ is\, \mathcal{C}_\delta(x) \to |x| \le s$. Then $\mathcal{R}_1(x) \to \mathcal{R}(x)$, $\mathcal{R}(x) \to \mathcal{R}_3(x)$ and $\mathcal{R}_3(x) \to \neg\neg\mathcal{R}_1(x)$.*

We have already proved that $\mathcal{N}$ is enumerable. To prove that it is simple, we need to prove that $\mathcal{R}$ (being the complement of $\mathcal{N}$) is not finite and to prove that $\mathcal{R}$ has no enumerable, non-finite subpredicate. We will do so in the following sections.

7.1 The Random Numbers are Non-Finite

Recall Fact 20 stating that there are exactly 2^k distinct lists of numbers $l : \mathbb{LB}$ with length k. We can immediately lift this to the following:

Corollary 28. *There are exactly 2^k distinct numbers y with $|y| = k$:*

$$\exists l : \mathbb{LN}.\ (\forall y.\ y \in l \leftrightarrow |y| = k) \land \# l \land \mathsf{len}(l) = 2^k$$

Corollary 29. *There are at most $2^k - 1$ distinct numbers y with $|y| < k$:*

$$\forall l : \mathbb{LN}.\ (\forall x.\ x \in l \to |y| < k) \to \# l \mathsf{len}(l) < 2^k - 1$$

Lemma 30. *Not every number of size k is nonrandom:*

$$\forall k : \mathbb{N}. \neg \forall x : \mathbb{N}.\ |x| = k \to \mathcal{N}(x)$$

Proof. Suppose all 2^k many numbers of size k would be nonrandom, i.e. we have distinct x_i for $0 \le i < 2^k$ s.t. there is y_i with $|y_i| < |x_i| = k$ and $\delta y_i \triangleright x_i$.

Note that we have $|y|_i < k$ and all y_i are distinct (because $y_i = y_j$ implies $x_i = x_j$). But by the last corollary, there are at most $2^k - 1$ numbers y with $|y| < k$. Contradiction. ◀

Corollary 31. *There classically exist random numbers of every size:*

$$\forall k. \neg\neg \exists x.\ |x| = k \land \mathcal{R}(x)$$

Corollary 32. *$\mathcal{R}$ is not finite.*

7.2 The Nonrandom Numbers are Simple

Recall that we have assumed δ to be an optimal description function. Restricting optimality to total functions yields the following.

Fact 33. $\forall f : \mathbb{N} \to \mathbb{N}. \exists d : \mathbb{N}. \forall x : \mathbb{N}. \exists y_x.\ \delta y_x \triangleright f x \land |y_x| < |x| + d$

We only need this fact, not full optimality, for the following central result.

Lemma 34. *Given* MP*, $\mathcal{R}$ does not have an enumerable, non-finite subpredicate.*

Proof. Let q be enumerable, non-finite, and $\forall x.\ qx \to \mathcal{R}(x)$. We have to obtain a contradiction.

Since q is non-finite the relation $\lambda kx.\ k < |x| \land qx$ is classically total, i.e. $\forall k. \neg\neg \exists x.\ k < |x| \land qx$. Thus, by MP and Corollary 17 there is a function $f : \mathbb{N} \to \mathbb{N}$ s.t. for all $k < |fk|$ (1) and $q(fk)$. In particular, due to the latter and since q is a subpredicate of random numbers, $\forall ky.\ \delta y \triangleright fk \to |fk| \le |y|$ (2).

By optimality for total functions there exists d such that for all k there is y_k with $\delta y_k \triangleright fk$ and $|y_k| < |k| + d$ (3). Thus we have for all k:

$$k \overset{(1)}{<} |fk| \overset{(2)}{\le} |y_k| \overset{(3)}{\le} |k| + d.$$

This is a contradiction to Fact 19. ◀

Theorem 35. *Given* MP*, $\mathcal{N}$ is simple.*

▶ **Corollary 36.** *The following hold given* MP*:*

1. *$\mathcal{R}$ is undecidable.*
2. *$\mathcal{N}$ is undecidable.*
3. *$\mathcal{C}$ is uncomputable.*

8 The Role of Markov's Principle

Markov's principle MP is used to prove that $\mathcal{R}$ has no enumerable, non-finite subpredicate in the proof of simpleness of $\mathcal{R}$.

The simpleness proof of nonrandom numbers uses MP to obtain a choice function f for the classically total, enumerable relation $\lambda kx.\ k < |x| \wedge qx$. Then optimality is applied to f to obtain an upper bound for $\mathcal{C}_\delta(fx)$, i.e. d with $\forall x.\ \mathcal{C}_\delta(fx) < |x| + d$. However, optimality would readily apply to partial functions f.

Enumerable relations can equivalently be characterised using partial choice functions. This approach is not very common in type theory because partial functions are usually avoided, but in the setting of Kolmogorov complexity we have been relying on partial functions all along. We say that a partial function $f : X \rightharpoonup Y$ is a partial choice function for a relation $R : X \to Y \to \mathbb{P}$ if values computed by f are in the relation. A partial choice function has the same domain as R if $\forall x.(\exists y.\ Rxy) \leftrightarrow \exists y.\ fx \rhd y$. For functional relations, the conditions are equivalent to $Rxy \leftrightarrow fx \rhd y$. A relation is enumerable if and only if it has a partial choice function with the same domain.

Fact 37. *Every enumerable relation $R{:}\,\mathbb{N} \to \mathbb{N} \to \mathbb{P}$ has a partial choice function with the same domain, and vice versa:*

$$\mathcal{E}R \leftrightarrow \exists f{:}\,\mathbb{N} \rightharpoonup \mathbb{N}.\ (\forall xy.\ fx \rhd y \to Rxy) \wedge \forall x.\ (\exists y.\ Rxy) \to (\exists y.\ fx \rhd y)$$

If R is functional, then $\mathcal{E}R \leftrightarrow \exists f{:}\,\mathbb{N} \rightharpoonup \mathbb{N}.\ \forall xy.\ fx \rhd y \leftrightarrow Rxy$.

We can then use optimality to prove the following result applying to every classically total, enumerable relation R.

Lemma 38. *Let $R{:}\,\mathbb{N} \to \mathbb{N} \to \mathbb{P}$ be a classically total, enumerable relation. Then*

$$\exists d{:}\,\mathbb{N}.\forall x{:}\,\mathbb{N}.\exists y_x.\neg\neg\exists v_x.\ Rxv_x \wedge \delta y_x \rhd v_x \wedge |y|_x < |x| + d.$$

Proof. By Fact 37 R has a partial choice function $f{:}\,\mathbb{N} \rightharpoonup \mathbb{N}$ s.t. $fx \rhd v_x \to Rxv_x$ and $Rxv_x \to \exists v'_x.\ fx \rhd v'_x$. Optimality of δ used on f immediately yields the result. ◀

The rest of the proof then proceeds exactly as before:

Lemma 39. *$\mathcal{R}$ does not have an enumerable, non-finite sub-predicate.*

Proof. Let q be enumerable, non-finite, and $\forall x.\ qx \to \mathcal{R}(x)$. We have to obtain a contradiction.

Since q is non-finite $\lambda ky.\ y\ is\ \boldsymbol{\mu} x.\ \ k < |x| \wedge qx$ is classically total, i.e. for all k, there classically exists x_k s.t. $k < |x_k|$ (1) and qx_k. In particular, due to the latter, x_k is random: $\forall y.\ \delta y \rhd x_k \to |x_k| \leq |y|$.

By Lemma 38 there exists d such that for all k there exist y_k and there classically exists x_k with $k < |x_k|$ (1), qx_k, thus $\forall y.\ \delta y \rhd x_k \to |x_k| \leq |y|$ (2), $\delta y_k \rhd x_k$ and $|y_k| < |k| + d$ (3).

We now prove $\forall k.\ k < |k| + d$, a contradiction by Fact 19.

Let k be given. We prove $k < |k| + d$. Since $<$ is decidable, we can obtain y_k and x_k with properties (1) - (3) from above and have the wanted contradiction to Fact 19:

$$k \overset{(1)}{<} |x_k| \overset{(2)}{\leq} |y_k| \overset{(3)}{<} |k| + d$$ ◀

Note that overall, it would have been shorter to just explain the proof without MP in the paper. However, the proof becomes arguably a little more indirect by eliminating MP. We consciously decided to show both versions, because we think that being able to transparently compare both approaches can be helpful to eliminate uses of MP from other theorems not exclusively but especially in computability theory.

9 Universal codes

We now instantiate the definition of Kolmogorov complexity to a definition based on universal codes as used by Catt and Norrish [5]. We call a code u *universal* if $\forall c.\ \exists x.\ \forall y.\ \phi_c y \equiv \phi_u \langle x, y\rangle$ and define

$$\delta_u x := \phi_u x \qquad \mathcal{C}_u(x) s := s \text{ is } \boldsymbol{\mu} s.\ s = |y| \wedge \delta_u y \triangleright x \qquad \mathcal{N}_u x := \exists y.\ \delta_u y \triangleright x \wedge |y| < |x|$$

We need to prove optimality of δ_u for partial functions to deduce simpleness of $\mathcal{N}$:

Lemma 40. *If u is universal then δ_u is optimal:*

$$\forall f\colon \mathbb{N} \rightharpoonup \mathbb{N}.\exists d\colon \mathbb{N}.\forall x\colon \mathbb{N}.\ \exists y_x.\ \forall v_x.\ f x \triangleright v_x \to \phi_u y_x \triangleright v_x \wedge |y_x| < |x| + d.$$

Proof. By universality of u we have c s.t. $\lambda y.\ \phi_u \langle c, y\rangle$ computes f.

Pick $d := |c| + 1$. Now let x be given. Pick $y_x := \langle c, x\rangle$. Let v_x be given and $f x \triangleright v_x$. We have $\phi_u y_x = \phi_u \langle c, x\rangle \triangleright v_x$ and $|y_x| = |\langle c, x\rangle| = |c| + |x| < |x| + d$ as wanted. ◀

Lastly, it is easy to show that there exists a universal code based on the universality of ϕ:

Lemma 41. *If ϕ is universal, there exists a universal code u.*

Proof. Recall that $\langle x, y\rangle := \lfloor \lceil x \rceil +\!\!+ \lceil y \rceil \rfloor$. We can define

$$I(x\colon \mathbb{N}) := \lfloor [\underbrace{\mathsf{true}, \dots, \mathsf{true}}_{x \text{ times}}, \mathsf{false}] \rfloor$$

$$D_1 n := \lfloor \mathsf{first}\ i\ (\mathsf{skip}\ (i+1)\ \lceil n \rceil) \rfloor \text{ where } i \text{ is the index of the first } \mathsf{false} \text{ in } \lceil l \rceil$$

$$D_2 n := \lfloor \mathsf{skip}\ (2 \cdot i + 1)\ \lceil n \rceil \rfloor \text{ where } i \text{ is the index of the first } \mathsf{false} \text{ in } \lceil l \rceil$$

We have that $D_1 \langle \langle Ix, x\rangle, y\rangle = x$ and $D_2 \langle \langle Ix, x\rangle, y\rangle = y$.

Now let $f x := \phi_{D_1 x}(D_2 x)$ and u be its code w.r.t. ϕ. u is universal because given c we can pick $x := \langle Ic, c\rangle$ and have that $\phi_c y \equiv \phi_u \langle x, y\rangle$. ◀

We can construct a universal code u s.t. this definition of Kolmogorov complexity and nonrandom numbers agree with the ones used by Sipser [43], who treats both the code and the input to ϕ as the description.

Lemma 42. *If pairing is computabily invertible (i.e. given a function* inv *with* $\mathsf{inv}\langle x, y\rangle = (x, y)$*), there is a universal code u s.t. $\mathcal{C}_u(x)$ is equivalent to $\lambda y.\ y$ is $\boldsymbol{\mu} s.\ \exists c e.\ s = |c| + |e| \wedge \phi_c e \triangleright x$ and $\mathcal{N}_u x$ is equivalent to $\exists c\ e.\ \phi_c e \triangleright x \wedge |c| + |e| < |x|$.*

Proof. Let a code u be strongly universal if $\forall c i.\ \phi_c i \equiv \phi_u \langle c, i\rangle$. It is immediate that every strongly universal code is universal and that every strongly universal code fulfils the properties.

Thus, it suffices to prove that there exists a strongly universal code: Define

$$f x := \textbf{let}\ (c, i) := \mathsf{inv}\ x\ \textbf{in}\ \phi_c i.$$

By universality of ϕ there exists a code u s.t. $\forall x.\ \phi_u x \equiv f x$. I.e. $\phi_c i \equiv f \langle c, i\rangle \equiv \phi_u \langle c, i\rangle$. ◀

Catt and Norrish define pairing as $\lfloor Ix +\!\!+ \lceil x \rceil +\!\!+ \lceil y \rceil \rfloor$ with I being the function from the proof of Lemma 41, which makes pairing immediately computationally invertible. For our definition of pairing, we do not know how to define such an inversion operation, but the whole development in this paper does not depend on the concrete definition of pairing and could as well be carried out with Catt and Norrish's definition.

10 Fixed inputs

Odifreddi [34] defines the Kolmogorov complexity of x as the smallest c such that the c-th Turing machine (w.r.t. an effective enumeration of Turing machines) outputs x when ran on a blank tape. That is, he defines the description operator to simulate on input c the c-th Turing machine on blank tape, and then defines the size of a Turing machine as exactly its code. Thus, there is exactly one Turing machine of size k, rather than exponentially many.

This non-conventional approach allows Odifreddi to obtain an elegant proof of simpleness of nonrandom numbers using Rogers' fixed-point theorem. However, Odifreddi's definition is not invariant w.r.t. other definitions (in the sense of Theorem 23) since there are strings with Kolmogorov complexity exponentially larger than using our definition.

However, one can use the first part of the definition to instantiate our definition as well. That is, we can define δ in terms of ϕ for an arbitrary fixed input i.

$$\delta c := \phi_c i \qquad \mathcal{C}(x)s := s \; is \; \boldsymbol{\mu} s.\; \exists c.\; s = |c| \land \phi_c i \rhd x \qquad \mathcal{N}x := \exists c.\; \phi_c i \rhd x \land |c| < |x|$$

The assumed universality of ϕ does not suffice to prove that $\mathcal{N}$ is simple using this definition. We need to adapt it in order to make statements on the size of γx for certain functions as follows.

Theorem 43. *If* $\forall f\colon \mathbb{N} \rightharpoonup \mathbb{N}.\ \exists \gamma\colon \mathbb{N} \to \mathbb{N}. \exists d\colon \mathbb{N}.\ \forall x i.\ \phi_{\gamma x} i \equiv f x \land (\exists y.\ f x \rhd y) \to |\gamma x| < |x| + d$, *then* $\mathcal{N}$ *is simple.*

Note that this is strictly stronger than the universality condition, but it can similarly be proved consistent by showing that a formulation of CT_{L} as discussed by Forster [13, §7, Appendix A] also implies this version.

11 Related Work

Vitanyi [48] attributes the first complete, spelled-out undecidability proof to Zvonkin and Levin in 1970 [50]. Zvonkin and Levin themselves refer to Kolmogorov who claimed the uncomputability of Kolmogorov complexity in 1965 by pointing to the undecidability of the halting problem [25], however without a real proof. Furthermore, the paper by Zvonkin and Levin seems to contain the first proof that that the nonrandom numbers are simple, but attribute the result to Janis Barzdins.

Other definitions of Kolmogorov complexity. Odifreddi [34] defines Kolmogorov complexity as $\mathcal{C}(x) := \boldsymbol{\mu} c.\ \phi_c 0 \rhd x$, i.e. assuming $|c| = c$. Our theory cannot be instantiated to such definitions of Kolmogorov complexity, because it contradicts property **3** of the encoding of numbers as binary strings. In particular, there are just k many numbers x with $|x| < k$ and just one number with exactly size k, so our proof cannot be used directly. Odifreddis' proof seems to be about as complicated as our proof, but not considering a size function allows a very elegant use of Roger's fixed-point theorem [41], of which Forster [13] gives a machine-checked proof. Thus, everything would be in place to also give a machine-checked proof that nonrandom numbers form a simple predicate based on Odifreddi's definition in our setting, but we leave this to future work.

Other machine-checked proofs regarding Kolmogorov complexity. In Section 9 we discuss a synthetic version of the definition used by Catt and Norrish [5].

There are various differences in the approaches: First, Catt and Norrish define ϕ in terms of the λ-calculus. In order to obtain c s.t. ϕ_c computes f they have to explicitly construct a λ-term computing f. Terms of the λ-calculus constitute about 200 lines of code of their HOL4 formalisation (not accounting for proofs about these terms) which has approximately 6000 lines in total. For comparison of the magnitude, our whole development has around 800 lines of Coq code. Note that we just give these numbers for comparison of magnitude and do not want to relate the complexity of Coq code and HOL4 code via line numbers.

Catt and Norrish use an operation MIN_SET computing the least element of a non-empty set to define a Kolmogorov complexity function. In general, such an operation requires at least a unique choice operator and the law of excluded middle. We instead work with functional relations, which poses no problems and avoids the operator. They also use classical logic, i.e. rely on LEM. While classical logic would be available in our setting, our result also shows that it is not needed. Classical logic is only available in our synthetic setting because we avoid (unique) choice operators [11]. Note that Catt and Norrish do not consider random or nonrandom numbers and prove instead that $\mathcal{C}$ is uncomputable. This is a direct consequence of the proof that $\mathcal{N}$ is simple, but the proof idea is exactly the same.

Other results on Kolmogorov complexity. Catt and Norrish [5] also prove the Kraft and Kolmogorov-Kraft inequalities and various other information-theoretic properties from the book by Hutter [20, Theorem 2.10]. We do not see any hurdles in adding them on top of our development, but leave them for future work as we focus on computability-theoretic properties. Regarding computability Kummer [26] proves that $\mathcal{N}$ is truth-table complete, i.e. that every enumerable predicate p, and in particular the halting problem $\mathcal{K}$ truth-table reduces to $\mathcal{N}$. The proof is highly involved and would pose an interesting challenge for interactive theorem proving. Forster, Jahn, and Smolka [14] give a machine-checked construction of a simple but truth-table complete predicate following Post, which is easier.

Other machine-checked proofs in computability and complexity theory. Norrish [33] proves Rice's theorem in HOL4 based on the full λ calculus, the development is now underlying the work on Kolmogorov complexity with Catt.

Forster and Smolka [17] prove Rice's theorem in Coq based on the weak call-by-value λ calculus. Forster, Kunze, Smolka, and Wuttke [16] prove that the weak call-by-value λ calculus and Turing machines can simulate each other with polynomial overhead in time. Gäher and Kunze [19] prove the Cook-Levin theorem stating that SAT is NP-complete with a definition of NP in the weak call-by-value λ-calculus, but by employing the simulation of the λ-calculus on Turing machines to encode computation as boolean circuits.

Xu, Zhang, and Urban [49] prove that the halting problem of Turing machines is undecidable in Isabelle. Larchey-Wendling [27] proves that every total μ-recursive function gives rise to a Coq function. Carneiro [4] proves Rice's theorem in Lean based on μ-recursive functions. Ramos et al. [10] prove Rice's theorem in PVS based on PVS0 and assuming a fixed-point theorem. Forster, Jahn, and Smolka [14] prove a synthetic version of Myhill's isomorphism theorem and a synthetic computational version of the Cantor-Bernstein theorem without assuming any axioms. Lastly, Forster [13, 12] proves Rice's theorem and Forster, Jahn, and Smolka [14] develop the theory of one-one, many-one, and truth-table degrees and construct simple and hypersimple sets, based on the axiom EA, equivalent to EPF.

12 Discussion

We conclude by discussing several more general points concerning machine-checked proofs, synthetic computability, and constructive mathematics.

For the scope of this project, translating from textbook proofs to the synthetic settings was barely noticable: The look and feel is like working in a textbook setting. Classical logic is available and the axiom of choice, albeit being explicitly acknowledged as foundation e.g. by Rogers [41], does not play any practical role. Instead of using (unique) choice operators to obtain functions one can work with the – often anyways more convenient – (classically) total functional relation directly.

The original textbook proofs on Kolmogorov complexity are heavily classical. The first steps of the present paper were undertaken as part of the third author's Bachelor's thesis [29], and explicitly used the law of excluded middle. The agnosticism of constructive type theory and Coq w.r.t classical provided very helpful in this regard: `Require Import Classical.` turns Coq into a classical proof assistant and basic logical automation tactics like `tauto` use classical logic immediately.

In a second pass, also part of the mentioned Bachelor's thesis, the uses of LEM were analysed one by one and either circumvented by a constructivisation of the statement, or by a use of Markov's principle where no immediate constructivisation was possible. For instance, the usual definition "There exist members of every size." $(\forall s.\exists n \geq s.\ pn)$ to define infinite sets has to be translated to the constructively weaker definition of infinity, "For every size, the subset of members of this size is not non-empty." $(\forall n.\neg(\neg\exists n.\ n \geq s \wedge pn))$. Here, Coq serves as a true proof *assistant*: It helps in keeping track of details and assumptions and keeps its user honest by forcing them to apply classical axioms explicitly. Manually constructivising proofs on paper would be more time-intensive, as proofs have to be checked repeatedly on axiom usage to be sure.

In the case of Kolmogorov complexity, obtaining fully constructive proofs directly from textbook proofs seems unfeasible. Even obtaining proofs based solely on MP requires a great deal of experience with constructive proofs. Thus it is crucial that both Coq and our setting of synthetic computability are compatible with classical reasoning: First working up to LEM, then up to MP and only then re-working the proofs to not use MP turned out to be feasible.

Overall, our paper can be seen as a case study substantiating various points often found in papers on interactive theorem proving:

First, giving a machine-checked proof has benefits distinct from the question of correctness. The ITP forces users to think about the most concise and elegant ways to express definitions, statements, and proofs and to isolate the properties of structures necessary to prove the wanted results. In our case, we can isolate that the assumption of an optimal description function suffices to prove the existence of a simple predicate, no other properties of the description function or the subsequent definition of Kolmogorov complexity are needed.

Secondly, thinking about constructive proofs has benefits distinct from the question which axioms are necessary for a result. Constructive logic enforces direct arguments, thus making presentations easier to follow, and makes proofs by contradiction explicit already in theorem statements, thus making presentations more transparent.

Thirdly, having the *possibility* to assume classical axioms is vital. Nowadays, few (if any) constructive mathematicians outright reject axioms like the law of excluded middle, but rather emphasise the benefits for presentations on the one hand, and on the other hand the academic curiosity motivating the analysis of which assumptions constitute the minimal set of assumptions necessary for any given theorem. The circumstance that our setting of

synthetic computability theory allows classical axioms, in contrast to previous approaches forming an anti-classical endeavour, was crucial for the present paper, which as already mentioned started out as a Bachelor's thesis project: Due to the pioneering work by Catt and Norrish it was clear that a machine-checked proof based on classical axioms is in reach, the first task was just to translate it to the simpler setting of synthetic computability theory. Probably, we would not have dared posing a simultaneous constructivisation and translation to the synthetic setting for a thesis project on Bachelor's level.

References

1 Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006. doi:10.1016/j.entcs.2005.11.049.

2 Andrej Bauer. On fixed-point theorems in synthetic computability. *Tbilisi Mathematical Journal*, 10(3):167–181, 2017. doi:10.1515/tmj-2017-0107.

3 Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*, volume 97. Cambridge University Press, 1987. doi:10.1017/CBO9780511565663.

4 Mario Carneiro. Formalizing Computability Theory via Partial Recursive Functions. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITP.2019.12.

5 Elliot Catt and Michael Norrish. On the formalisation of Kolmogorov complexity. In *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM, January 2021. doi:10.1145/3437992.3439921.

6 Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *Journal of the ACM*, 16(3):407–422, July 1969. doi:10.1145/321526.321530.

7 Thierry Coquand and Gérard P Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988. doi:10.1016/0890-5401(88)90005-3.

8 Andrej Dudenhefner. The undecidability of system F typability and type checking for reductionists. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–10. IEEE, 2021. doi:10.1109/LICS52264.2021.9470520.

9 Andrej Dudenhefner. Constructive many-one reduction from the halting problem to semi-unification. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 18:1–18:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.18.

10 Thiago Mendonça Ferreira Ramos, Ariane Alves Almeida, and Mauricio Ayala-Rincón. Formalization of rice's theorem over a functional language model. Technical report, University of Brasília, 2020. URL: https://www.mat.unb.br/~ayala/RiceThFormalization.pdf.

11 Yannick Forster. Church's Thesis and Related Axioms in Coq's Type Theory. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2021.21.

12 Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. doi:10.22028/D291-35758.

13 Yannick Forster. Parametric church's thesis: Synthetic computability without choice. In Sergei N. Artëmov and Anil Nerode, editors, *Logical Foundations of Computer Science - International Symposium, LFCS 2022, Deerfield Beach, FL, USA, January 10-13, 2022, Proceedings*, volume 13137 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2022. doi:10.1007/978-3-030-93100-1_6.

14 Yannick Forster, Felix Jahn, and Gert Smolka. A Constructive and Synthetic Theory of Reducibility: Myhill's Isomorphism Theorem and Post's Problem for Many-one and Truth-table Reducibility in Coq (Full Version), February 2022. preprint. URL: https://hal.inria.fr/hal-03580081.

15 Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in Coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs - CPP 2019*. ACM Press, 2019. doi:10.1145/3293880.3294091.

16 Yannick Forster, Fabian Kunze, Gert Smolka, and Maximilian Wuttke. A mechanised proof of the time invariance thesis for the weak call-by-value λ-calculus. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

17 Yannick Forster and Gert Smolka. Weak call-by-value lambda calculus as a model of computation in Coq. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*, pages 189–206. Springer, 2017. doi:10.1007/978-3-319-66107-0_13.

18 R. M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsovlability (solution of post's problem, 1944. *Proceedings of the National Academy of Sciences*, 43(2):236–238, February 1957. doi:10.1073/pnas.43.2.236.

19 Lennard Gäher and Fabian Kunze. Mechanising complexity theory: The cook-levin theorem in Coq. In *12th International Conference on Interactive Theorem Proving, ITP 2021, June 29 to July 1, 2021, Rome, Italy (Virtual Conference)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ITP.2021.20.

20 Marcus Hutter. *Universal Artificial Intellegence*. Springer Berlin Heidelberg, 2005. doi:10.1007/b138233.

21 Dominik Kirst and Dominique Larchey-Wendling. Trakhtenbrot's theorem in Coq. In *Automated Reasoning*, pages 79–96. Springer International Publishing, 2020. doi:10.1007/978-3-030-51054-1_5.

22 Steven C. Kleene and Emil L. Post. The upper semi-lattice of degrees of recursive unsolvability. *The Annals of Mathematics*, 59(3):379, May 1954. doi:10.2307/1969708.

23 John R. Kline. The April meeting in New York. *Bulletin of the American Mathematical Society*, 54(7):622–648, July 1948. doi:10.1090/s0002-9904-1948-09030-9.

24 Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963. doi:10.1016/S0304-3975(98)00075-9.

25 Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):3–11, 1965.

26 Martin Kummer. On the complexity of random strings. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 25–36. Springer, 1996. doi:10.1007/3-540-60922-9_3.

27 Dominique Larchey-Wendling. Typing total recursive functions in Coq. In *International Conference on Interactive Theorem Proving*, pages 371–388. Springer, 2017. doi:10.1007/978-3-319-66107-0_24.

28 Dominique Larchey-Wendling and Yannick Forster. Hilbert's Tenth Problem in Coq. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.FSCD.2019.27.

29 Nils Lauermann. A formalization of Kolmogorov complexity in synthetic computability theory. Bachelor's thesis, Saarland University, 2021. URL: https://ps.uni-saarland.de/~lauermann/bachelor.php.

30 Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer New York, 2008. doi:10.1007/978-0-387-49820-1.

31 Albert Abramovich Muchnik. On strong and weak reducibility of algorithmic problems. *Sibirskii Matematicheskii Zhurnal*, 4(6):1328–1341, 1963.

32 John Myhill. Creative sets. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 1957. doi:10.1002/malq.19550010205.

33 Michael Norrish. Mechanised computability theory. In *ITP 2011*, volume 6898 of *LNCS*, pages 297–311. Springer, 2011. doi:10.1007/978-3-642-22863-6_22.

34 Piergiorgio Odifreddi. *Classical recursion theory: The theory of functions and sets of natural numbers*. Elsevier, 1992.

35 Christine Paulin-Mohring. Inductive definitions in the system Coq rules and properties. In *International Conference on Typed Lambda Calculi and Applications*, pages 328–345. Springer, 1993. doi:10.1007/BFb0037116.

36 Christine Paulin-Mohring. Introduction to the Calculus of Inductive Constructions, January 2015. URL: https://hal.inria.fr/hal-01094195.

37 Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944. doi:10.1090/S0002-9904-1944-08111-1.

38 Emil L. Post. Degrees of recursive unsolvability - preliminary report. In *Bulletin of the American Mathematical Society*, volume 54(7), pages 641–642. American Mathematical Society (AMS), July 1948. doi:10.1090/s0002-9904-1948-09030-9.

39 Henry G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953. doi:10.1090/S0002-9947-1953-0053041-6.

40 Fred Richman. Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803, 1983. doi:10.2307/2273473.

41 Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.

42 A. Shen, V. Uspensky, and N. Vereshchagin. *Kolmogorov Complexity and Algorithmic Randomness*. American Mathematical Society, November 2017. doi:10.1090/surv/220.

43 Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006. doi:10.1145/230514.571645.

44 Ray J. Solomonoff. A preliminary report on a general theory of inductive inference (revision of report v-131). *Contract AF*, 49(639):376, 1960.

45 R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, March 1964. doi:10.1016/s0019-9958(64)90223-2.

46 R.J. Solomonoff. A formal theory of inductive inference. part II. *Information and Control*, 7(2):224–254, June 1964. doi:10.1016/s0019-9958(64)90131-7.

47 The Coq Development Team. The Coq proof assistant version 8.13.2, January 2021. doi:10.5281/zenodo.4501022.

48 Paul M.B. Vitányi. How incomputable is Kolmogorov complexity? *Entropy*, 22(4):408, April 2020. doi:10.3390/e22040408.

49 Jian Xu, Xingyuan Zhang, and Christian Urban. Mechanising Turing machines and computability theory in Isabelle/HOL. In *International Conference on Interactive Theorem Proving*, pages 147–162. Springer, 2013. doi:10.1007/978-3-642-39634-2_13.

50 A K Zvonkin and L A Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, December 1970. doi:10.1070/rm1970v025n06abeh001269.