Saarland University

Faculty of Mathematics and Computer Science

Bachelor's Thesis

A Formalization of Kolmogorov Complexity in Synthetic Computability Theory

Author	**Supervisor**	**Advisor**
Nils Lauermann	Prof. Gert Smolka	Fabian Kunze

Reviewers
Prof. Gert Smolka
Prof. Markus Bläser

Submitted: August 24th, 2021

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, August 24th, 2021

Abstract

We present a formalization of Kolmogorov complexity and the non-random numbers in synthetic computability theory in the proof assistant Coq. In synthetic computability theory all functions $\mathbb{N} \to \mathbb{N}$ are considered computable and no external model of computation is required. The Kolmogorov complexity of an object is the size of its smallest description. A number whose Kolmogorov complexity is smaller than itself is defined as non-random or compressible.

We define Kolmogorov complexity in Coq using a universal interpreter and prove the invariance theorem. With the Berry paradox we prove the uncomputability of Kolmogorov complexity.

We formalize the many-one incompleteness of the non-random numbers. This is achieved by showing the simpleness of the non-random numbers. That means we show that the non-random numbers are enumerable, infinite and there exists no infinite, enumerable sub-predicate of the random numbers.

We mechanize Martin Kummer's proof of the existence of a minimal number of random numbers for every length. Additionally, Kummer proves the truth-table completeness of the non-random numbers. We give approaches to a potential formalization of this proof in Coq.

Except for Kummer's proof of the minimal count of random numbers, all of the major mechanized proofs use Markov's principle and do not require excluded middle.

Acknowledgements

First of all, I would like to thank Fabian for his outstanding support, week in, week out. His exceptional guidance and feedback was invaluable. Without his eye for the bigger picture I would still be working on some unimportant proof that leads nowhere.

I am extremely grateful to Professor Smolka for the opportunity to write this thesis at his chair. Attending his lectures was always incredibly interesting. Furthermore, I highly appreciate his extraordinary support throughout my bachelor studies.

In general, I want to thank all members of the Programming Systems Lab for their friendliness and advice.

I would like to express my sincere thanks to Professor Bläser for reviewing this thesis.

I am very grateful to Yannick for his valuable support and I also want to thank him and Jonas for their feedback on this thesis. I also thank Elliot Catt and Felix for their helpfulness.

I want to thank all my friends: Stay vital! Many thanks especially to Tobias, for his advice and our exciting chess games. Many thanks also to Mark for our wonderful collaborations.

Last but not least, I am incredibly grateful to my family. I want to thank them for their trust, support and willingness to even laugh at my awful jokes.

Contents

Chapter 1

Introduction

The scientific field called **algorithmic information theory** emerged in the early 1960s and deals with "the quantity of information in individual objects" [17]. We mostly associate the algorithmic information theory with **Kolmogorov complexity** today. Informally, the Kolmogorov complexity of a number is the size of its smallest description. This notion was accompanied by the idea to split up the numbers in two partitions: the **random** and the **non-random numbers**. A number is non-random if and only if its Kolmogorov complexity is smaller than itself. Another, arguably more intuitive, name for non-randomness is **compressibility**: If we can describe a number in such an efficient way so that the description is shorter than the number itself, then this number is compressible.

These concepts were accompanied by very interesting questions. Already in Andrei N. Kolmogorov's first publication on Kolmogorov complexity in 1965 he mentions its uncomputability [15]. In contrast to the random numbers, the non-random numbers are recursively enumerable so that the random numbers are undecidable. This engendered the question whether one can show the undecidability via a reduction from the halting problem to the non-random numbers. In other words, are the non-random numbers complete with regard to many-one, truth-table or Turing reductions? By 1970, the many-one-completeness of the non-random numbers could be refuted [42] and also the completeness with respect to Turing reductions was proven. The status of completeness with regard to truth-table reducibility was open for more than 30 years until Martin Kummer was able to prove the truth-table-completeness in 1996 [16].

In this thesis, we discuss these results and, to the best of our knowledge, give the first formalization of Kolmogorov complexity and the random numbers in the Coq Proof Assistant [37]. We also prove the many-one incompleteness of the non-random numbers. Notably, we give the first formalization of Kolmogorov complexity in the synthetic computability theory. Synthetic computability was proposed by Richman and Bridges [24, 2], and advanced by Bauer [1]. In contrast to analytic

computability, in synthetic computability no particular model of computation is the foundation of the work: Synthetic computability is set in constructive logic so that all functions are inherently computable and we do not need an external model of computation.

We are building upon the formalization of synthetic computability theory by Forster, Kirst, Smolka, and Jahn [10, 8, 9].

Kolmogorov complexity is also part of the popular culture in the form of code golf. Code golf is a competition in which people compete to find the smallest program in a particular programming language that has a desired output [6].

Related Work Catt and Norrish formalized Kolmogorov complexity in the *HOL4 interactive theorem prover* [3]. Unlike our work in the constructive logic of Coq, Catt and Norrish's formalization is set in classical logic. The focus of Catt and Norrish is mainly inequalities involving Kolmogorov complexity. In contrast to the synthetic approach we have chosen, Catt and Norrish use the λ-calculus and the general recursive functions as their model of computation.

Outline of this Thesis First, we define Kolmogorov complexity in Coq and prove many interesting properties, in particular the invariance theorem, in Chapter 3. With that, we will show the uncomputability in Chapter 4. In Chapter 5 we introduce the definition of the random and non-random numbers in Coq and go on to show that the non-random numbers are undecidable and many-one incomplete. Additionally, we prove in Coq that there is a lower bound for the count of random numbers (Chapter 6). Finally, in Chapter 7 we discuss Kummer's truth-table-completeness proof and although we do not provide a Coq mechanization, we will discuss the difficulties of a mechanization and potential ways to overcome them.

Mechanization As mentioned before, the results of Chapters 2 to 6 are mechanized in Coq. In the digital version of this thesis, definitions and proofs are linked to an online viewer for the Coq development. Every link directly leads to the relevant position in the Coq code.

The complete files may be accessed here:

`https://github.com/uds-psl/coq-kolmogorov-complexity`

Contributions We give the first formalization of Kolmogorov complexity in Coq, to the best of our knowledge. Furthermore, our work is set in synthetic computability theory. In this setting, we formalize the uncomputability of Kolmogorov complexity as well as the many-one incompleteness of the non-random numbers in Coq. Finally, we lay the foundation for a potential mechanization of the truth-table completeness of the non-random numbers.

1.1 The History of Kolmogorov Complexity

In the 1960s three people around the globe independently discovered the algorithmic information theory: Ray J. Solomonoff, Andrei N. Kolmogorov, and Gregory J. Chaitin.

The literature [17, 11] attributes this development to Alan Turing's discovery of the universal Turing machine in 1937 [38]. Kolmogorov complexity is defined depending on a machine: The Kolmogorov complexity of a number x with regard to a machine M is the size of the smallest input on which M returns x. The universal Turing machine can simulate any other Turing machine so that the Kolmogorov complexity with regard to the universal Turing machine is invariant. That means that in comparison to any other Turing machine, a number does not have a significantly greater Kolmogorov complexity with regard to the universal Turing machine, as the universal machine can simply simulate the other machine. This fact is called **Invariance Theorem** and its importance becomes apparent, when we consider that all three discoverers have independently proposed it almost identically.

The first to propose Kolmogorov complexity was not the eponym himself but Ray J. Solomonoff in 1960 with "A preliminary report on a general theory of inductive inference" [33]. Solomonoff introduced the *universal a priori probability* of a sequence and thereby as a by-product the first occurrence of Kolmogorov complexity. He states that a number with a small Kolmogorov complexity with respect to a universal Turing machine has a high a priori probability.

In 1964, he proved the invariance theorem [32]. Vitányi, co-author of the standard text "**An introduction to Kolmogorov complexity and its applications**", lauds this achievement for "governing Algorithmic Information Theory" [40].

Andrei N. Kolmogorov, was unaware of Solomonoff's work as he developed similar ideas in 1963 [14, 13]. According to Shiryaev, a student of Kolmogorov, he discussed his ideas in a report to the probability section of the Moscow mathematical society in the same year [27]. In 1965, Kolmogorov followed this up with the paper "Three approaches to the quantitative definition of information" [15], where he introduces his version of Kolmogorov complexity. In this paper, he also proves the invariance theorem.

The last person that independently started working on the field of Kolmogorov complexity is Gregory J. Chaitin in his 1966 paper [4]. In the successor paper from 1969 [5], Chaitin defines the typical Kolmogorov complexity alongside a proof of the invariance theorem. Chaitin refers to Kolmogorov, as he later became aware of his work.

Considering the timeline and the three inventors, one could argue that the name Kolmogorov complexity does not do the history justice. However, this name has

been widely established, so we follow the prevalent terminology.

This summary contains only a fraction of the whole story. For an extensive account we refer the reader to Li and Vitányi [17, 40] and Shiryaev [27].

Chapter 2

Preliminaries

In this chapter, we provide a brief overview of the type theory of Coq, important definitions for our work, and notations we use.

2.1 The Coq Proof Assistant

The results of this thesis are formalized in the Coq proof assistant. Coq is build upon the *Calculus of Inductive Constructions* (CIC) [22] which defines the type theory all definitions and lemmata must adhere to. Every term in Coq has a type; even types themselves. A statement is a type and in order to prove it, one needs to provide an inhabitant, i.e. a term with the corresponding type.

Coq's logic is constructive. That means, we can only prove a proposition by constructing an explicit proof. For example, that implies that we cannot prove the *law of excluded middle*

$$\mathrm{LEM} := \forall P : \mathbb{P}.\, P \vee \neg P$$

This is famously not the case in classical logic, where excluded middle is a staple.

Yet excluded middle is consistent in Coq which means that we can safely assume it [41]. Propositions like excluded middle that are neither provable nor disprovable are called **independent**.

A weaker, yet still independent [7], version of excluded middle is *Markov's principle*:

$$\mathrm{MP} := \forall f : \mathbb{N} \to \mathbb{B}.\, \neg\neg(\exists n.\, f\, n = \mathrm{true}) \to (\exists n.\, f\, n = \mathrm{true})$$

In essence, Markov's principle enables us to locally use excluded middle, when we are trying to prove the satisfiability of a boolean test. The following fact justifies this:

Fact 2.1 $\forall PQ : \mathbb{P}.\, ((Q \vee \neg Q) \to \neg\neg P) \to \neg\neg P$

Now, we can get the desired fact by instantiating P accordingly in the previous fact:

Fact 2.2

$$\mathrm{MP} \to \forall Q : \mathbb{P}.\, \forall f : \mathbb{N} \to \mathbb{B}.\, ((Q \vee \neg Q) \to \neg\neg(\exists n.\, f\, n = \mathrm{true})) \to (\exists n.\, f\, n = \mathrm{true})$$

In general, many statements in this thesis seem to be intuitionistically unprovable. Often, we will prove the double negated statement so that we can temporarily use excluded middle via Fact 2.1. We have multiple paths to resolving a non-trivial double negation in a hypothesis. If the proof goal itself is double negated it is straightforward:

Fact 2.3 $\forall P Q : \mathbb{P}.\, \neg\neg Q \to (Q \to \neg\neg P) \to \neg\neg P$

If we want to show falsity then we can also eliminate a double negation in a hypothesis:

Fact 2.4 $\forall P : \mathbb{P}.\, \neg\neg P \to (P \to \bot) \to \bot$

Otherwise, excluded middle or Markov's principle eliminate the double negation:

Fact 2.5 $\mathrm{LEM} \leftrightarrow \forall P : \mathbb{P}.\, \neg\neg P \to P$

With Fact 2.5 it becomes apparent, that Markov's principle is weaker than excluded middle. So, if possible we use MP over LEM to reduce the probability of introducing inconsistencies, when we are assuming additional axioms.

If we prove a proposition with a double negation, we say that we *classically* prove this proposition.

2.2 Important Types

In this section we discuss the standard types we use in this thesis.

The universes $\mathbb{T}$ and $\mathbb{P}$ Coq has a whole hierarchy of universes [29]. In this thesis we confine ourselves to a simplified version with the two universes $\mathbb{T}$ and $\mathbb{P}$. $\mathbb{T}$ is the type of all types and $\mathbb{P}$ is its impredicative subuniverse of propositions.

Basic Inductive Types The natural numbers $\mathbb{N}$ are defined with the two value constructors $0 : \mathbb{N}$ and the successor function $S : \mathbb{N} \to \mathbb{N}$. There exists a bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathtt{nat}$. We will use the notation $\langle a, b \rangle$ for the embedding of two numbers into one.

The boolean type $\mathbb{B}$ contains exactly the values true and false. The functions $\wedge_{\mathbb{B}}, \vee_{\mathbb{B}} : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$ compute the boolean conjunction and disjunction respectively.

With option types $\mathbb{O}X$ for any type X, we can either represent some value $x : X$ with $^\circ x$ or no value with $\emptyset$.

The type for lists containing elements of type X is given by $\mathbb{L}X$. A list is built by starting with the empty list $[]$ and successively adding one element to the front of a list with the cons constructor: For $x : X$ and $L : \mathbb{L}X$ the list $x :: L$ is the list L with x added at the front. The membership of $x : X$ in $L : \mathbb{L}X$ is denoted by $x \in L$. We write $L \subseteq L'$ ('list inclusion') when all elements of $L : \mathbb{L}X$ are members in $L' : \mathbb{L}X$. Two lists $L, L' : \mathbb{L}X$ are equivalent ($L \equiv L'$) if $L \subseteq L' \wedge L' \subseteq L$. For two lists $L, L' : \mathbb{L}X$, the concatenation is given by $L +\!\!+ L'$. We express the fact that a list L does not contain an element twice, i.e. is duplicate-free, with NoDup L. If all elements of the list $L : \mathbb{L}X$ satisfy a predicate $p : X \to \mathbb{P}$ we denote this by Forall p L. For an $x : X$ and $n : \mathbb{N}$ we define $x^n : \mathbb{L}X$ as the list which exactly contains n times x. The overlapping with the notation for exponentiation is no problem because the context makes the correct meaning apparent.

For $L, L' : \mathbb{L}X$, we can compute a list $L - L'$ containing all the elements from L which are not in L'. This is only possible if X is discrete, i.e. there exists a boolean equality decider for X (see below).

For case analyses on option types, numbers and lists we use the following notation:

$$\text{if } \ldots \text{ is } {}^\circ x \text{ then } \ldots \text{ else } \ldots$$

Sigma Types and Sum Types Coq separates between the computational ($\mathbb{T}$) and propositional level ($\mathbb{P}$): The *elimination restriction* disallows the extraction of information from existential quantifications $\exists$ and disjunctions $\vee$ in a computational context. Therefore, Coq provides computational equivalents that are situated in $\mathbb{T}$.

The computational analogue of existential quantifications are sigma types $(\Sigma x : X.\, px)$ with $p : X \to \mathbb{T}$ whose values are so-called dependent pairs (x, a) with $x : X$ and $a : px$. This name stems from the fact that the second component is depending on the value given in the first component. We can access both components of the pair. So, we can define projection functions π_1 and π_2: For $(x, a) : (\Sigma x : X.\, px)$ we have $\pi_1(x, a) = x$ and $\pi_2(x, a) = a$.

Sum types correspond to disjunctions. For two types X and Y, the sum type $X + Y : \mathbb{T}$ has the value constructors $L : X \to X + Y$ and $R : Y \to X + Y$. So a value of a sum type $X + Y$ can either contain a value $x : X$ or $y : Y$.

Discrete Types A type X is called discrete, if there exists a boolean equality decider $f : X \to X \to \mathbb{B}$, i.e. $\forall xy.\, x = y \leftrightarrow f\, x\, y = \text{true}$.

2.3 Synthetic Computability

In this section we recapitulate the most important definitions by Forster et al. [8, 10, 9].

2.3.1 Standard Notions of Computability Theory

Let $p : X \to \mathbb{P}$.

We say that p is subfinite if there exists a list that exhausts p, that is all values satisfying p are contained in this list.

Definition 2.6 (Subfinite Predicate) $\mathfrak{X}p := \exists l : \mathbb{L}X.\, \forall x : X.\, px \to x \in l$

We call p infinite iff p is not subfinite.

Definition 2.7 (Decidable Predicate) $\mathcal{D}p := \exists f : X \to \mathbb{B}.\, \forall x.\, px \leftrightarrow fx = \text{true}$

Definition 2.8 (Enumerable Predicate) $\mathcal{E}p := \exists f : \mathbb{N} \to \mathbb{O}X.\, \forall x.\, px \leftrightarrow \exists n.\, fn = {}^\circ x$

The function f in Definition 2.8 is called an enumerator for p.

Reductions

Let $p : X \to \mathbb{P}$, $q : Y \to \mathbb{P}$.

Definition 2.9 (Many-one Reduction) $p \preceq_m q := \exists f : X \to Y.\, \forall x.\, px \leftrightarrow q(fx)$

Truth-table reductions $\preceq_{tt}$ are weaker than many-one reductions, i.e. $\forall pq.\, p \preceq_m q \to p \preceq_{tt} q$.

We do not give Forster's completely formal definition of $\preceq_{tt}$ in Coq [8] because we do not formalize any truth-table reductions. This intuitive description is sufficient for our discussion of Kummer's truth-table reduction in Chapter 7.

A function $f : X \to \mathbb{L}Y \times (\mathbb{L}\mathbb{B} \to \mathbb{B})$ is a truth-table reduction from p to q if

$$\begin{aligned} fx = ([y_1, \ldots, y_n], t) \to (\forall i.\, 1 \leqslant i \leqslant n \to q(y_i) \leftrightarrow b_i = \text{true}) \\ \to p(x) \leftrightarrow t[b_1, \ldots, b_n] = \text{true} \end{aligned}$$

We denote the fact that there exists a truth-table reduction from p to q by $p \preceq_{tt} q$.

Completeness

Forster's completeness definition deviates from the usual definition: A complete predicate does not need to be enumerable. Additionally, we restrict complete predicates to predicates on numbers to simplify the definition.

Definition 2.10 (many-one complete) *Let* $p, q : \mathbb{N} \to \mathbb{P}$. q *is called many-one complete (m-complete) if* $\mathcal{E}p \to p \preceq_m q$.

Definition 2.11 (truth-table complete) *Let* $p, q : \mathbb{N} \to \mathbb{P}$. q *is called truth-table complete (tt-complete) if* $\mathcal{E}p \to p \preceq_{tt} q$.

Simple Sets

The notion of simple sets was first introduced by Emil L. Post in 1944 [23]. A simple set S is enumerable and infinite, and the complement of S is infinite and contains no infinite enumerable subset. Post proved that simple sets are not complete with regard to many-one reductions.

Definition 2.12 $\text{simple } p := \mathcal{E}p \wedge \neg\mathcal{X}\overline{p} \wedge \neg\exists q.(\forall x.qx \to \overline{p}x) \wedge \mathcal{E}q \wedge \neg\mathcal{X}q$

Fact 2.13 *Simple predicates are undecidable.*

Fact 2.14 *Simple predicates are m-incomplete.*

The following definition and fact are not formalized, but they will be interesting in the following chapters. Smullyan introduced the effectively simple sets whose complement's finite subsets additionally need to be bounded in size by a computable function [30]. Martin then showed that every effectively simple set is complete with respect to Turing reductions [18].

2.3.2 Partial Functions

It is only possible to define terminating, total functions in Coq. However, we will want to use partial functions in the following. For that, we use stationary sequences [8].

Definition 2.15

$$\text{stationary } (f : \mathbb{N} \to \mathbb{O}X) := \forall s_1 v.\, f\ s_1 = {}^\circ v \to \forall s_2.\, s_2 \geqslant s_1 \to f\ s_2 = {}^\circ v$$

We can interpret the input to f as a step count: If f returns ${}^\circ v$ for some step count then we want that a higher step count does not change the return value. Additionally, f can diverge if it returns $\emptyset$ for all step counts.

A partial function $f : X \rightharpoonup Y$ is represented in Coq by $f : X \to (\mathbb{N} \to \mathbb{O}Y)$ with $\forall x : X.\, \text{stationary } (f\ x)$.

Two partial functions $f, g : X \to (\mathbb{N} \to \mathbb{O}Y)$ compute the same partial function if f outputs y on input x at some point if and only if g does so too.

Definition 2.16 $f \simeq g := \forall x a.\, (\exists s.\, f\ x\ s = {}^\circ a) \leftrightarrow (\exists s.\, g\ x\ s = {}^\circ a)$

This predicate models an equivalence relation and in particular, it is transitive:

Fact 2.17 $\forall f g h.\, f \simeq g \to g \simeq h \to f \simeq h$

2.3.3 Church's Thesis

Our formalization does not follow the traditional approach of defining Kolmogorov complexity in one particular model of computation like the λ-calculus or the general recursive functions. Instead, our work uses the synthetic approach by Richman and Bridges [24, 2] and Bauer [1]. That means, we natively work in Coq without imposing an additional model of computation.

The definitions in this section are the work of Yannick Forster [8]. We assume a partial, abstract universal computation function $\phi : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{O}\mathbb{N}$. We follow Forster's notation $\phi_c^s\, x$ for the execution of an input x on the code c for s steps $(\phi\, c\, x\, s)$. As $\phi_c^s\, x$ is partial, we have $\forall cx.\, \text{stationary}\ \phi_c\, x$.

In reference to the Church-Turing thesis, Forster introduces Church's thesis: Church's thesis states that every (total) Coq function $f : \mathbb{N} \to \mathbb{N}$ is computed by ϕ via some code.

Definition 2.18 (Church's Thesis (CT))

$$\mathrm{CT}_\phi := \forall f : \mathbb{N} \to \mathbb{N}.\, \exists c : \mathbb{N}.\, \forall x : \mathbb{N}.\, \exists s : \mathbb{N}.\, \phi_c^s\, x = {}^\circ(f\, x)$$

The idea is, that every definable Coq function is by design computable, and accordingly the universal computation function must compute all of these.

Additionally, we require Church's thesis for partial, i.e. stationary, functions:

Definition 2.19 (Church's Thesis for partial functions (PCT))

$$\mathrm{PCT}_\phi := \forall f : \mathbb{N} \to \mathbb{N} \to \mathbb{O}\mathbb{N}.\, (\forall x.\, \text{stationary}\ (f\, x)) \to \exists c : \mathbb{N}.\, \phi_c \simeq f$$

Forster's name for a stronger version of PCT is "enumerability of partial functions" (EPF). EPF is stronger because it is applicable to whole families of partial functions [8]. For CT, Forster also introduces such a stronger version, synthetic Church's thesis (SCT), for families of total functions which can be regarded as a combination of CT and the S_n^m theorem. The weaker versions presented here suffice for our work.

An explicit consistency proof of CT in Coq is still outstanding but Forster gives many convincing arguments why it is safe to assume the consistency of CT [8]. CT is consistent with Markov's Principle according to Forster who refers to Swan and Uemura [35]. The situation for the law of excluded middle remains unclear. Forster speculates that CT and LEM are consistent. Whenever possible, we try to forgo using LEM which will succeed with the exception of the result of Chapter 6.

2.4 Existential and Least Witness Operator

The following facts are already proven in the mechanization by Forster et al. [9] which is why we do not prove them again.

Existential Witness Operator As discussed above, it is not possible to retrieve information from a propositional existential quantification in a computational context. Nevertheless, there is an exception for decidable predicates on numbers:

Fact 2.20 $\forall p : \mathbb{N} \to \mathbb{P}.\, \mathcal{D}p \to (\exists x.\, p\, x) \to (\Sigma x.\, p\, x)$

Least Witness The least witness of $p : \mathbb{N} \to \mathbb{P}$ is defined as the smallest number satisfying p.

Definition 2.21 (Least Witness) $\text{least}\, p\, n := pn \wedge \forall k.\, pk \to k \geqslant n$

Fact 2.22 $\forall pxy.\, \text{least}\, p\, x \to \text{least}\, p\, y \to x = y$

Proof By the antisymmetry of $\geqslant$. □

It is possible to compute the least witness of a satisfiable and decidable predicate:

Fact 2.23 (Least Witness Operator)

$$\forall p : \mathbb{N} \to \mathbb{P}.\, \mathcal{D}p \to (\Sigma x.\, p\, x) \to (\Sigma x.\, \text{least}\, p\, x)$$

A combination of existential and least witness operator provides the existential least witness operator:

Fact 2.24 (Existential Least Witness Operator)

$$\forall p : \mathbb{N} \to \mathbb{P}.\, \mathcal{D}p \to (\exists x.\, p\, x) \to (\Sigma x.\, \text{least}\, p\, x)$$

Additionally, we introduce a slightly altered, classical version for propositional existence:

Fact 2.25 (Constructive least witness operator)

$$\forall p : \mathbb{N} \to \mathbb{P}.\, \neg\neg(\exists x.\, p\, x) \to \neg\neg(\exists x.\, \text{least}\, p\, x)$$

For propositional existence, logical decidability suffices which is provided by the double negation. The double negation of the argument $\neg\neg(\exists x.\, p\, x)$ has no impact on this proof as it can be eliminated with Fact 2.3. With this double negation we can use the local excluded middle when we prove the satisfiability of p in an application of the constructive least witness operator.

2.5 Pigeonhole Principles

The following proofs stem from Forster et al. [9]. In the following, if we are using the pigeonhole principle, we are referring to one of the following facts.

Fact 2.26 $\forall l l'.\, (\forall xy.\, \mathcal{D}(x \neq y)) \to \text{NoDup}\, l \to |l| > |l'| \to \Sigma x.\, x \in l \wedge x \notin l'$

Alternatively, an existential quantification can be the return type instead of a sigma type. In this case logical decidability even suffices. Another version, with a double negated existential quantification does not require any decidability.

Fact 2.27 $\forall l.\, (\forall xy.\, (x \neq y) \vee \neg(x \neq y)) \to \text{NoDup}\, l \to \forall l'.\, l \subseteq l' \to |l| \leqslant |l'|$

2.6 Binary numbers

Our binary numbers are represented by lists of boolean values. Additionally, for our convenience we want that this binary encoding is bijective so that every number has a unique binary representation and leading zeros are not valueless. It is definitively possible to define such an encoding but for the sake of simplicity, we assume it as an axiom. Below, we give convincing arguments why it is indeed possible.

Axiom 2.28 (Binary encoding) *Let* $\lceil \cdot \rceil : \mathbb{N} \to \mathbb{LB}$ *and* $\lfloor \cdot \rfloor : \mathbb{LB} \to \mathbb{N}$ *have the following properties:*

1. $\forall l : \mathbb{LB}. \lceil \lfloor l \rfloor \rceil = l$
2. $\forall n : \mathbb{N}. \lfloor \lceil n \rceil \rfloor = n$
3. $\forall n : \mathbb{N}. |\lceil n \rceil| \leqslant \log_2(n) + 1$
4. $\forall xy : \mathbb{N}. x \leqslant y \to |\lceil x \rceil| \leqslant |\lceil y \rceil|$

To argue why it is possible to define this encoding, we want to refer to Catt and Norrish [3] who defined the *2-adic* binary representation by Smullyan [31] in HOL4 and proved properties 1, 2 and 4. Additionally, they remark that $|\lceil \cdot \rceil|$ "is essentially a natural number logarithm" with $|\lceil 0 \rceil| = 0$, $|\lceil 1 \rceil| = 1$ and for $0 < k$, $|\lceil 2^k \rceil| = k$. Hence, $|\lceil n \rceil| \leqslant \log_2(n)+1$ holds for $n < 2$. For $n \geqslant 2$ there exists a $k > 0$ with $2^k \leqslant n < 2^{k+1}$, so that for the natural number logarithm $\log_2(n) = k$. We know that property 4 is satisfied by Catt and Norrish's encoding, so that $|\lceil n \rceil| \leqslant |\lceil 2^{k+1} \rceil| = k+1$. That means we have $|\lceil n \rceil| \leqslant \log_2(n) + 1$ so that property 3 is fulfilled by their encoding.

For property 3 we want to give another, more intuitive justification. Smullyan states that the *2-adic* binary representation is identical to mapping every string in $\{0, 1\}^*$ to its position in the *shortlex* order, i.e. the lexicographical order with precedence for shorter strings. Intuitively, this encoding is just a denser version of the regular encoding, i.e. the length of a number in the bijective encoding is less or equal to the length of the same value (without leading zeros) in the regular encoding. The regular encoding satisfies property 3 (disregarding leading zeros), so that the bijective encoding also satisfies this property.

As logarithms are sublinear, so is the length of the binary representation:

Fact 2.29 $\neg \exists c. \forall n. n \leqslant |\lceil n \rceil| + c$

In the following, if we mention the length of a number $x : \mathbb{N}$ we refer to $|\lceil x \rceil|$. In general, as the encoding is bijective, we use the natural number and its binary representation often synonymously.

For all $n : \mathbb{N}$, we can compute a duplicate-free list containing all elements of length n. We denote this list by L_n. The duplicate-free list $L_{\leqslant n}$ contains all numbers of length $\leqslant n$.

Fact 2.30 *For all $n : \mathbb{N}$, there are exactly 2^n distinct numbers of length n.*

Proof By induction on n. □

Fact 2.31 *For all $n : \mathbb{N}$, there are exactly $2^{(n+1)} - 1$ distinct numbers of length $\leqslant n$.*

Proof By induction on n, using Fact 2.30. □

2.7 Auxiliary Lemmata

We use many auxiliary lemmata, e.g. for numbers and lists, throughout this thesis. Here, we list the most important ones.

The following fact was proven by Yannick Forster. We can classically decide every predicate for a finite number of values:

Fact 2.32 $\forall X.\, \forall p : X \to \mathtt{Prop}.\, \forall l.\, \neg\neg \exists l'.\, \forall x.\, x \in l' \leftrightarrow px \wedge x \in l$

Proof By induction on l. In the inductive step we have $x :: l$ and we use the double negation to decide if px holds. The inductive hypothesis gives us the desired list l' for l. If px holds then $x :: l'$, otherwise l'. □

The following fact is part of the Coq standard library:

Fact 2.33 $\forall X.\, \forall l l' : \mathbb{L}X.\, \mathrm{NoDup}\, l \to |l'| \leqslant |l| \to l \subseteq l' \to l' \subseteq l$

The next fact states, that two duplicate-free lists that form a partition of another duplicate-free list, contain together exactly the same elements as the partitioned list.

Fact 2.34

$$\begin{aligned}\forall X.\, \forall L l l' : \mathbb{L}X.\, &\mathrm{NoDup}\, L \to \mathrm{NoDup}\, l \to \mathrm{NoDup}\, l' \\ &\to l \subseteq L \to l' \subseteq L \\ &\to (\forall x.\, \neg(x \in l \wedge x \in l')) \to |L| = |l| + |l'| \\ &\to (l +\!\!+ l') \equiv L\end{aligned}$$

Proof By induction on l' with L and l quantified. □

Chapter 3

Kolmogorov Complexity in Coq

In this chapter, we define Kolmogorov complexity in Coq. We define universal codes and also construct a universal code. With these definitions we prove the invariance theorem and discuss its importance. Additionally, we show many facts involving Kolmogorov complexity. Finally, we give an account of Kolmogorov complexity in the literature and compare our definition with the definitions from the literature.

3.1 Kolmogorov Complexity

In the introduction we mentioned the uncomputability of Kolmogorov complexity, so consequently we cannot define Kolmogorov complexity as a Coq function. Hence, we define it as a predicate. For that, we use the least witness predicate (Definition 2.21) to capture the minimality aspect of Kolmogorov complexity. The predicate $\mathrm{KC} : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ receives a code for ϕ, the number whose Kolmogorov complexity is stated, and the corresponding Kolmogorov complexity:

Definition 3.1 (Kolmogorov complexity)

$$\mathrm{KC}\ (c\ x\ k : \mathbb{N}) := \mathrm{least}\ (\lambda k.\ \exists y s.\ |\lceil y \rceil| = k \wedge \phi_c^s\ y = {}^\circ x)\ k$$

For $\mathrm{KC}\ c\ x\ k$ we introduce the notation $\mathrm{KC}_c\ x = k$. We also abbreviate propositions like $(\mathrm{KC}_c\ x = k \to p\ k)$ to $(p\ (\mathrm{KC}_c\ x))$, for some predicate $p : \mathbb{N} \to \mathbb{P}$. For better readability, we often use the term complexity, when we are referring to Kolmogorov complexity in the following.

Because we do not have any immediate knowledge of the binary encoding of a number, it seems unnecessary to use it to define the size of a number. Alternatively, we could simply use the binary logarithm with identical results. However, it is much more convenient to explicitly use the lists, as this enables us to concatenate two binary numbers without having to deal with logarithms.

The following fact is straightforward because the least witness is functional (Fact 2.22).

Fact 3.2 (Kolmogorov complexity is functional)

$$\forall c\ x\ kc\ kc'.\, KC_c\ x = kc \to KC_c\ x = kc' \to kc = kc'$$

3.2 Universal Codes

In the introduction (Chapter 1.1) we briefly discussed the importance of the universal Turing machine. Therefore, we want an equivalent construct for ϕ. Hence, we define a code as universal if it can simulate any other code with some prefix to the input.

Definition 3.3 (Universal code)

$$\mathrm{univ}\ c := \forall c'.\, \exists p : \mathbb{LB}.\, \forall x : \mathbb{N}.\, \phi_{c'}\ x \simeq \phi_c\ \lfloor p +\!\!+ \lceil x \rceil \rfloor$$

Our definition does not capture every universal code, as the definition restricts the input to the form $\lfloor p +\!\!+ \lceil x \rceil \rfloor$. We follow Catt and Norrish [3] with this restriction. This constraint is not a problem, since we only need the existence of some code that can simulate other codes with a reasonable increase of input length. The latter is satisfied with this definition, since we have constant overhead in length of the input for each code that is simulated. So, we only need to prove the existence of a universal code, which we do now.

Lemma 3.4 (Existence of a universal code) *Assuming* PCT*, there exists a universal code.*

Proof We define a partial function $f : \mathbb{N} \to \mathbb{N} \to \mathbb{ON}$ that simulates any code in the way a universal code does. Then we can obtain a universal code by applying PCT to f.

We define f on inputs like $\lfloor \mathrm{false}^{|\lceil c \rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil c \rceil +\!\!+ \lceil x \rceil \rfloor$. This pattern enables us to recover c' as well as x from the list, because we know the length of $\lceil c' \rceil$ from the number of leading false and thus the start of $\lceil x \rceil$.

$$f\ \lfloor \mathrm{false}^{|\lceil c' \rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil c' \rceil +\!\!+ \lceil x \rceil \rfloor\ s := \phi^s_{c'}\ x$$

f is stationary because ϕ is stationary. This means we can apply PCT to f and receive a code c with $\forall x.\, f\ x \simeq \phi_c\ x$. The code c is universal: Given a code c' and an input x, pick the prefix $\mathrm{false}^{|\lceil c' \rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil c' \rceil$. We then have

$$\phi_c\ \lfloor \mathrm{false}^{|\lceil c' \rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil c' \rceil +\!\!+ \lceil x \rceil \rfloor \simeq f\ \lfloor \mathrm{false}^{|\lceil c' \rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil c' \rceil +\!\!+ \lceil x \rceil \rfloor \simeq \phi_{c'}\ x$$

Then, c is universal by the transitivity of $\simeq$ (Fact 2.17). □

In combination with CT, we can prove that a universal code c can also efficiently simulate any Coq function $f : \mathbb{N} \to \mathbb{N}$:

Fact 3.5

$$\mathrm{CT} \to \forall c.\, \mathrm{univ}\, c \to \forall f : \mathbb{N} \to \mathbb{N}.\, \exists d : \mathbb{N}.\, \forall x : \mathbb{N}.\, \exists i s : \mathbb{N}.\, \phi_c^s\, i = {}^\circ(f x) \wedge |\lceil i \rceil| \leqslant |\lceil x \rceil| + d$$

Proof With CT we receive a code c' that computes f. c' can be simulated by the universal code c with a prefix p. We pick $|p|$ for the desired d.

Assume an $x : \mathbb{N}$. We choose $\lfloor p +\!\!+ \lceil x \rceil \rfloor$ for i and we know by the definition of CT that there exists a step count s so that $\phi_c^s \lfloor p +\!\!+ \lceil x \rceil \rfloor = {}^\circ(f x)$. We also have $|p +\!\!+ \lceil x \rceil| = |\lceil x \rceil| + d$ so that the claim holds. □

3.3 Invariance Theorem

The definition of a universal code enables the simulation of any other code with an input that is only longer by a constant. This has an important implication for the complexity with regard to a universal code: The Kolmogorov complexity with regard to a universal code compared to an arbitrary code is only greater by a constant. This is because the universal code can always simulate the respective other code with constant overhead to the input so that the complexity must be smaller than the length of this input.

Theorem 3.6 (Invariance Theorem)

$$\forall c c'.\, \mathrm{univ}\, c \to \exists d.\, \forall x\, k c\, k c'.\, \mathrm{KC}_c\, x = k c \to \mathrm{KC}_{c'}\, x = k c' \to k c \leqslant k c' + d$$

Proof Let c be a universal code. Choose the length of the boolean list p obtained from the universality of c as the constant d. Now we receive a number x, kc and kc' where kc and kc' are the Kolmogorov complexity of x with regard to c and c' respectively. Let y and y' be the inputs for c and c' that justify the Kolmogorov complexities kc and kc'. We know that $\phi_c \lfloor p +\!\!+ \lceil y' \rceil \rfloor$ outputs x at some point. By the minimality of Kolmogorov complexity we then have $kc \leqslant |p +\!\!+ \lceil y' \rceil| = |p| + |\lceil y' \rceil| = kc' + |p|$. □

The importance of the invariance theorem for the field of Kolmogorov complexity cannot be understated. Kolmogorov called this the "main discovery" of Solomonoff and himself as this enabled them to define "complexity in an almost invariant way" [14]. Without this theorem the notion of Kolmogorov complexity would have no greater significance because the Kolmogorov complexity of a number differs vastly between machines. The invariance theorem introduces a greater order that justifies a universal measurement of complexity.

3.4 Properties of Kolmogorov Complexity

Due to the minimality aspect in the definition of Kolmogorov complexity, many Facts cannot be proven constructively without a double negation. Alternatively, we can assume the complexity of a number as a hypothesis to show that Kolmogorov complexity fulfils a certain property.

The Kolmogorov complexity of any function value is bound by the size of the respective function argument and some function-specific constant.

Fact 3.7

$$\mathrm{CT} \to \forall c.\, \mathrm{univ}\, c \to \forall f : \mathbb{N} \to \mathbb{N}.\, \exists d.\, \forall m\, kc.\ \mathrm{KC}_c\,(f\,m) = kc \to kc \leqslant |\lceil m \rceil| + d$$

Proof Follows by the minimality of Kolmogorov complexity and Fact 3.5. □

Fact 3.8 $\forall c y x s.\, \phi_c^s\, y = {}^\circ x \to \neg\neg \exists kc.\, \mathrm{KC}_c\, x = kc$

Proof Apply the constructive least witness operator (Fact 2.25) to $\lambda y.\, \exists s.\, \phi_c^s\, y = {}^\circ x$. We receive the smallest input y which returns x on c. By the monotonicity of the binary encoding (2.28.4) we have $\mathrm{KC}_c\, x = |\lceil y \rceil|$. □

Fact 3.9 $\mathrm{CT} \to \forall c.\, \mathrm{univ}\, c \to \forall x.\, \neg\neg \exists kc.\, \mathrm{KC}_c\, x = kc$

Proof Follows immediately with Fact 3.8, the code given by $(\mathrm{CT}_\phi\ (\lambda x.\, x))$, and the universality of c. □

3.5 Kolmogorov Complexity in the Literature

In this section we want to give an overview over the definitions of Kolmogorov complexity in the literature and want to compare these to the definition we use. All definitions agree on the fact that the Kolmogorov complexity of some object x is the size of the smallest description of x. Nevertheless, there are small differences in the concrete definitions.

Kolmogorov uses *general recursive functions* $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ as the model of computation [15]. For some general function $\varphi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, the complexity of x is defined as the length of the smallest program p that outputs x on input 1:

$$K_\varphi(x) = \begin{cases} \min\limits_{\varphi(p,1)=x} \mathrm{length}(p) \\ \infty \text{ if there is no such } p \text{ with } \varphi(p,1) = x \end{cases}$$

The input 1 is a remnant of the related conditional complexity. Conditional complexity is defined depending on the second input component of φ and measures the complexity of a number when this additional input is serves as an additional source of information. The unconditional complexity receives no additional information and hence this second component is fixed as 1.

Solomonoff's definitions rely on *Turing machines* and omit the second input component, as he did not introduce conditional complexity [32].

In contrast to Kolmogorov and Solomonoff, Michael F. Sipser also incorporates the size of the Turing machine in the definition of Kolmogorov complexity: Specifically, he defines the Kolmogorov complexity of x as $|M|+|y|$ where M is a Turing machine which returns x on input y and there is no other machine M' which returns x on input y' so that $|M'|+|y'|$ is smaller [28].[1] That means, he regards Kolmogorov complexity on a higher level, than most: The majority compare the complexity of objects between different machines in their chosen model of computation, whereas Sipser does not distinguish between Turing machines and defines Kolmogorov complexity in an absolute way. His invariance theorem compares with other models or rather "description languages", which are functions $p : \Sigma^* \to \Sigma^*$ with some alphabet Σ, that are computable by a Turing machine. Sipser defines the Kolmogorov complexity of x with respect to p as the length of the smallest $y \in \Sigma^*$, so that $p(y) = x$.

A slightly different perspective on Sipser's approach shows that this definition is very much in line with Kolmogorov and Solomonoff. Instead of saying that Sipser incorporates the machine length into the Kolmogorov complexity, we could also interpret the Kolmogorov complexity as being defined with regard to a fixed universal machine which receives a machine and an input on its input tape. The description languages are computed by some Turing machine so that we can simply omit this layer and directly address them as Turing machines. Hence, the invariance theorem states the usual relationship between a universal machine and all other machines.

In contrast to these definitions, we use the synthetic approach in our formalization so that we do not use an external model of computation.

Normally, the size function returns the length of the input number. Traditionally, many use the binary system [32, 20, 27], while others do not elaborate on the numeral system [15, 28]. Piergiorgio Odifreddi deviates from the usual definition by using the number itself as its size [21].

We follow the traditional approach by defining the size of a number as the length of its binary representation.

Instead, we could also adopt Odifreddi's [21] approach of using the number itself as its size. However, for the invariance theorem (Theorem 3.6) we require a bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$, where the size of the encoded pair increases at most by one if we increase the size of the second component by one. For the identity as size

[1] In reality Sipser does not use $|M| + |y|$ but instead $2 * |M| + |y| + 2$. This is due to the fact that he encodes the machine and the input into one string to measure their length.

function it is impossible to define such a bijection. It is known from Cantor's pairing function that, figuratively speaking, only diagonals in the cartesian coordinate system result in a bijection so that a straight line along one axis cannot work. This makes it impossible to prove the standard version of the invariance theorem which only adds an additive constant. Odifreddi does not require the invariance theorem for his short introduction to Kolmogorov complexity, so that this definition is sufficient.

Chapter 4

The Uncomputability of Kolmogorov Complexity

The standard proof of the uncomputability of Kolmogorov complexity is an application of the Berry paradox that predates the discovery of Kolmogorov complexity by more than 50 years. This paradox goes back to the librarian G.G. Berry (1867-1928) and was first published by Bertrand Russell in 1908 [26, 12]. The original wording of this paradox is "the least integer not nameable in fewer than nineteen syllables". However, this phrase only contains eighteen syllables, so that this 'defined' number would be nameable in less than nineteen syllables. Hence, there must be a contradiction and we cannot define a number this way.

The parallels to Kolmogorov complexity are evident. We could regard this measurement of syllables as an informal approach to Kolmogorov complexity. In our setting, we would rather state this paradox as "the least natural number with a Kolmogorov complexity greater than $\mathfrak{m}$" for some $\mathfrak{m} : \mathbb{N}$.

The origin of the first uncomputability proof is not completely certain. Vitányi [39] speculates that the first complete proof was published by Zvonkin and Levin in 1970 [42]. 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 [15].

Without explicit reference to the Barry paradox, Zvonkin and Levin take advantage of this idea. To this day, this proof is adopted practically unchanged in the textbooks. Catt and Norrish also follow it in their HOL4 formalization [3]. Our proof presented here does so, too.

4.1 The Unboundedness of Kolmogorov Complexity

Intuitively, it is easy to see that for a universal code there cannot exist a number that has the largest Kolmogorov complexity: Fact 3.9 guarantees that every number has a Kolmogorov complexity and it is obvious that there are only finitely many binary numbers of some length. The rigorous Coq proof is not as straightforward.

In order to prove the unboundedness, we need a list that contains all numbers with a Kolmogorov complexity smaller than some bound. First, we prove the existence of such a list containing all the numbers with one particular complexity:

Lemma 4.1

$$\forall c.\, \mathrm{univ}\, c \to \forall n.\, \neg\neg\exists L.\, \forall x.\, \mathrm{KC}_c\, x = n \to x \in L$$

Proof It suffices to find a list that contains all the outputs of c on numbers of length n, as all numbers with the Kolmogorov complexity n must be in this list. It is not possible to simply map ϕ to L_n, as we do not know if and when ϕ terminates for an element of L_n. That means, we need to consider every element on its own, so that we can use the double negation to decide whether ϕ_c terminates.

We can achieve this by proving that for all m there exists a list containing all outputs of ϕ_c on the first m elements of L_n. This is possible by induction on m. For the base case $m = 0$, the empty list $[]$ suffices. In the successor case, we have the desired list L for the first m elements by the inductive hypothesis. We use the double negation to decide whether ϕ_c terminates on the m-th element of L_n, if it exists.

If ϕ terminates with $^\circ o$ then $o :: L$ has the required properties. Otherwise, L contains all the desired numbers. □

From this Lemma, we can then derive the existence of the corresponding list for all lengths smaller or equal to the bound:

Lemma 4.2

$$\forall c.\, \mathrm{univ}\, c \to \forall n.\, \neg\neg\exists L.\, \forall m.\, m \leqslant n \to \forall x.\, \mathrm{KC}_c\, x = m \to x \in L$$

Proof By induction on n. For the inductive step, append the list for Sn obtained from Lemma 4.1 to the list of the inductive hypothesis. □

With this Lemma it is now straightforward to prove that there is no upper bound on the Kolmogorov complexity.

Theorem 4.3 (Unboundedness of KC)

$$\forall c.\, \mathrm{univ}\, c \to \forall n.\, \neg\neg\exists x\, kc.\, \mathrm{KC}_c\, x = kc \wedge n \leqslant kc$$

Proof By the infiniteness of the natural numbers, we can obtain a number x that is not an element of the list L from Lemma 4.2, i.e. that has a greater Kolmogorov complexity than n. With Fact 3.9 we receive the precise Kolmogorov complexity kc which must be greater than n, because else x would have been in L in the first place. □

Now it is obvious, why we did not require the lists from Lemma 4.1 and 4.2 to contain only the numbers with the relevant complexity. The additional elements do no harm, as we can always find an element not contained in this list. Therefore, it is sensible not to bother defining a list that satisfies the equivalence.

4.2 The Uncomputability of Kolmogorov Complexity

With the unboundedness of Kolmogorov complexity we have everything that is needed for the Berry paradox. Hence, we can now prove the uncomputability of Kolmogorov complexity.

Theorem 4.4 (The uncomputability of Kolmogorov complexity) *Given Markov's principle, there exists no function computing the Kolmogorov complexity of every number for some universal code.*

Proof We assume a universal code c, a function $kc : \mathbb{N} \to \mathbb{N}$ that computes Kolmogorov complexity, i.e. $\forall x.\, KC_c\, x = kc(x)$, and derive a contradiction. With the existential least witness operator 2.24 we can define the certifying function

$$H : \forall m.\, \Sigma x.\, \text{least}\, (\lambda x.\, m \leqslant kc(x))\, x$$

with the decidability of $\leqslant$, and the unboundedness of Kolmogorov complexity $(\forall m.\, \exists x.\, m \leqslant kc(x))$. In Theorem 4.3 we proved the unboundedness with a double negation. With Markov's principle we can derive the plain unboundedness.

Now, we can define $g := \lambda m.\, \pi_1\, (H\, m)$ with $\forall m.\, m \leqslant kc(g(m)))$. Additionally, Lemma 3.7 gives us a d such that $\forall m.\, kc(g(m)) \leqslant |\lceil m \rceil| + d$. Hence, the transitivity of $\leqslant$ gives us $\forall m.\, m \leqslant |\lceil m \rceil| + d$, which is a contradiction with Fact 2.29. □

The Berry paradox is very prominent in the proof: The hypothesis H is an immediate translation of the paradox into Coq.

4.3 Markov's Principle in the Uncomputability Proof

We firmly believe that the proof, in the way presented here, is only possible with Markov's principle. We use MP to obtain the unboundedness without a double negation. The unboundedness proof requires the double negation in order to decide the termination of ϕ.

Alternatively, we could preserve the double negation by using the constructive least witness operator (Fact 2.25) to obtain $H : \forall m.\, \neg\neg\exists x.\, \text{least}\, (\lambda x.\, m \leqslant kc(x))\, x$. As we are deriving a contradiction, i.e. proving falsity, we could remove the double negation for finitely many instances of m. This is not sufficient to derive a contradiction: Note, that we use $\neg\exists d.\, \forall m.\, m \leqslant |\lceil m \rceil| + d$ (Fact 2.29) in order to prove falsity. So, it is not sufficient to show $\exists d.\, m \leqslant |\lceil m \rceil| + d$ for one special m. That means we require the (total) computable function g with $\forall m.\, m \leqslant kc(g(m)))$. Hence, we need

to remove the double negation for all m, i.e. infinitely many instances, which is not possible without MP.

Yannick Forster suggested, that a different approach could obviate the need for Markov's principle: He hypothesized, that defining g as a partial function, although nettlesome, could succeed in eliminating Markov's principle.

4.4 Discussion

The uncomputability explains the profusion of double negations in Chapter 3. If we could compute Kolmogorov complexity, many of these facts would be trivially provable without the double negation. Hence, the double negation is required, because we need to logically decide the termination of ϕ.

Chapter 5

The Random Numbers

With the notion of Kolmogorov complexity it seems natural to distinguish between numbers that are 'easy' to compute and the ones that are not. With Lemma 3.7 we know that the size of a number and a constant form an upper bound on the Kolmogorov complexity of the number itself. Intuitively, a number with a Kolmogorov complexity smaller than the size of the number must be computable in an easier way than simply using a print machine. For example, there could be a repeating pattern or another characteristic so that an efficient algorithm could retrieve this information from a very small input.

If we compare the following string

126744325745103315

to strings like

333333333333333333

or even

112233445566778899

it is clear that the latter strings can be computed by a simple algorithm and a small input, whereas the computation of the former most likely needs to settle with a print machine.

The terminology that has developed for the differentiation of these kinds of strings is **random** and **non-random**. The benchmark for these classes is indeed the size of the number. That means, we call a number random, if its Kolmogorov complexity is greater or equal to its own size. Conversely, a number is non-random, if it is not random. Another name for randomness and non-randomness used in the literature is *incompressibility* and *compressibility* respectively [17]. These names may be more intuitive: A non-random number is compressible into a smaller input.

The first to discuss the concept of random numbers was Kolmogorov [15] in 1965. He informally proposed a slightly different approach: Simply put, he calls an element of a set of size N that can be computed by a program a lot smaller than $\log_2 N$ random, if its Kolmogorov complexity is close to $\log_2 N$. This idea is not as different to the approach stated above as one might think. For instance, the set of binary numbers of length k contains 2^k elements, so that a random number in that set would have a complexity of at least k which coincides with the approach used here. In general, these definitions do not always agree.

Li and Vitányi use the "compressibility" terminology because with "randomness" they refer to the work of Per Martin-Löf. Martin-Löf, a student of Kolmogorov, showed that incompressible strings are random in the way we look at it in probability theory [19]. He stated that for a random number with regard to Kolmogorov complexity "the number of ones in $\xi_1\xi_2\cdots\xi_n$ should be close to $n/2$". For the general case, he defined a universal test for randomness and extended his approach to infinite binary sequences.

We use the "random" terminology instead of "incompressible".

5.1 The Definitions in Coq

Like the complexity itself, the random and non-random numbers are defined depending on a code c.

Definition 5.1 (Random Numbers) $\mathrm{R}_c\, x := \forall y s.\, \phi_c^s\, y = {}^\circ x \to |\lceil y \rceil| \geqslant |\lceil x \rceil|$

Definition 5.2 (Non-random Numbers) $\overline{\mathrm{R}}_c\, x := \exists y s.\, \phi_c^s\, y = {}^\circ x \wedge |\lceil y \rceil| < |\lceil x \rceil|$

The canonical approach would be to define the non-random numbers as not random, i.e. $\overline{\mathrm{R}}_c\, x := \neg \mathrm{R}_c\, x$. We have decided on two separate definitions, as we work with the existential quantification instead of the negated universal quantification in the proofs anyway. This way we can omit the conversion in facts on non-random numbers.

The conversion between random and not non-random numbers is a consequence of the de Morgan law for existential quantification:

Fact 5.3 $\forall x.\mathrm{R}_c\, x \leftrightarrow \neg\overline{\mathrm{R}}_c\, x$

In the constructive setting of Coq, the de Morgan law for universal quantification in general only holds with excluded middle. That means, the equivalence of not random and non-random numbers does not follow immediately. The direction $\overline{\mathrm{R}}_c\, x \to \neg \mathrm{R}_c\, x$ works without additional axioms because for the order of numbers $\forall xy : \mathbb{N}.\, x < y + x \geqslant y$ holds.

Fact 5.4 $\forall x.\overline{\mathrm{R}}_c\, x \to \neg \mathrm{R}_c\, x$

For the converse, Markov's principle suffices because given an input and a step count we can define a boolean test for the non-randomness of the corresponding output.

Fact 5.5 $\mathrm{MP} \to \forall x.\neg \mathsf{R}_c\, x \to \overline{\mathsf{R}}_c\, x$

Our definition of (non-)randomness deviates from the usual definition found in the literature. The difference for the definition of non-randomness is, that it suffices for us to find *some* input that is smaller than the number it is outputting. This need not be necessarily the smallest one, i.e. we do not explicitly require the Kolmogorov complexity of x to be smaller than x itself. This definition is classically equivalent with Fact 3.9.

Fact 5.6 $\mathrm{LEM} \to \forall x.\, \overline{\mathsf{R}}_c\, x \leftrightarrow (\exists k.\, \mathrm{KC}_c\, x = k \land k < |\lceil x \rceil|)$

Similarly, for the random numbers we do not require the Kolmogorov complexity to be greater than the length of the number itself. A definition closer to the textbook also needs to capture that the Kolmogorov complexity can also be ∞. Classically, these definitions are also equivalent:

Fact 5.7 $\mathrm{LEM} \to \forall x.\, \mathsf{R}_c\, x \leftrightarrow (\exists k.\, \mathrm{KC}_c\, x = k \land k \geqslant |\lceil x \rceil|) \lor (\forall k.\, \neg \mathrm{KC}_c\, x = k)$

For our formalization we choose these definition, as it often renders the need for excluded middle unnecessary. For example, Fact 5.5 would also require excluded middle with the alternative definition. This is because a boolean test cannot capture the minimality of Kolmogorov complexity, as ϕ is partial.

5.2 The Unboundedness of the Random Numbers

For the unboundedness of the random numbers for a universal c, we cannot entirely reuse the approach for the unboundedness of Kolmogorov complexity (Chapter 4.1). The main idea of the latter proof is to use the fact that all numbers have a Kolmogorov complexity but there are only finitely many binary numbers of every respective length. Although not immediately apparent from the proof itself, this is an application of the pigeonhole principle. In the unboundedness proof for $\overline{\mathsf{R}}_c$ we also make use of the pigeonhole principle multiple times and even in a more explicit way.

Similarly to the previous unboundedness proof, we prove that for every list L containing non-random numbers of length k, there exists a duplicate-free list L′ containing inputs which justify the non-randomness of all members of L. In other words, L′ contains only elements with length smaller than k and their outputs on c are members of L.

Lemma 5.8 *For every code* $c : \mathbb{N}$, $k : \mathbb{N}$ *and duplicate-free* $L : \mathbb{L}\mathbb{N}$ *where* L

- *only contains numbers of length k*
- *only contains non-random numbers with regard to c*

there exists a duplicate-free list $L' : \mathbb{L}\mathbb{N}$ *which satisfies*

- $|L| = |L'|$
- $\forall i \in L'. \exists o s. o \in L \wedge \phi_c^s\, i = {}^\circ o \wedge |\lceil i \rceil| < |\lceil o \rceil|$

Proof By induction on L. In the base case $[]$ satisfies the properties. In the inductive step we have the list $o :: L$. We know o must be non-random and consequently we get i and s with $|\lceil i \rceil| < |\lceil o \rceil|$ and $\phi_c^s\, i = {}^\circ o$. Therefore, $i :: L'$ satisfies the properties. □

With this Lemma, we can now prove that there are $< 2^k$ non-random numbers of length k.

Lemma 5.9 *Any duplicate-free list* L *containing only non-random numbers of length* k *contains less than* 2^k *elements, for a universal c.*

Proof It suffices to show $|L| \neq 2^k$ because there are only 2^k distinct numbers of length k. So, we assume $|L| = 2^k$ and derive a contradiction. From Lemma 5.8 we receive a duplicate-free list L' with $|L'| = |L| = 2^k$ and Forall $(\lambda x. |\lceil x \rceil| < k)\ L'$. This is a contradiction by the pigeonhole principle, as we know there are only $2^k - 1$ numbers which have a length smaller than k (Fact 2.31), so that L' cannot contain 2^k different numbers. □

Corollary 5.10 *Given Markov's principle, any duplicate-free list* L *containing all random numbers of length* k *contains at least one element, for a universal c.*

Finally, we can prove the unboundedness of the random numbers: For every $k : \mathbb{N}$ there exists a number x of length k which is random with respect to a universal c.

Lemma 5.11 (Unboundedness of the Random Numbers)

$$\forall c.\, \mathrm{univ}\, c \to \forall k. \neg\neg \exists x. |\lceil x \rceil| = k \wedge R_c\, x$$

Proof With Fact 2.32 we receive a list l containing all non-random numbers of length k. There exists an equality decider for $\mathbb{N}$ so that we can compute a duplicate-free version of l that is equivalent to the original list. By the pigeonhole principle applied to L_k and l with Fact 2.30 and Lemma 5.9, we obtain a number that is not in l and thus random. □

5.3 The Non-Random Numbers are Simple

In this section, we prove that the predicate of non-random numbers is simple. This result is important as we are able to derive the many-one incompleteness of the non-random numbers from this (see Chapter 2). Furthermore, the non-random numbers are a natural example of a simple set in contrast to, for example, Post's simple set which seems very artificial [23]. This is remarkable, as there are only a few known natural sets which are simple, i.e. not decidable but the halting problem is not many-one reducible to it [8].

Zvonkin and Levin refer to Y.M. Barzdin as the original discoverer of the proof in the late 1960s [42]. He even proved the simpleness of a more general set: His set contains the numbers x with Kolmogorov complexity smaller or equal to the function value $f(x)$ of some unbounded general recursive function f.

Our proof generally follows the outline of Odifreddi [21], albeit there were several adjustments necessary. First and foremost, Odifreddi defined the Kolmogorov complexity with the number itself as its size, as discussed in Chapter 3.5. In the grand scheme of things, this is only minor, as the proof itself still follows the ideas of Barzdin's general proof.

In order to show that $\bar{R}_c$ is a simple predicate for a universal c, we need to prove three sub-results: $\bar{R}_c$ needs to be enumerable, there must be infinitely many random numbers and there must not exist a infinite, enumerable predicate subsumed by the random numbers.

Lemma 5.12 ($\bar{R}_c$ is enumerable) $\mathcal{E}\ \bar{R}_c$

Proof We define a function $f : \mathbb{N} \to \mathbb{ON}$ that enumerates $\bar{R}_c$. An input p is interpreted as a pair $\langle x, s\rangle$ by f. Then compute x on the code c for s steps. If $\emptyset$ is returned, then f also returns $\emptyset$. Otherwise, c outputs $^\circ y$. If y is guaranteed to be non-random, i.e. $|\lceil y\rceil| < |\lceil x\rceil|$, then $^\circ y$ is returned by f. In the opposite case $\emptyset$ is returned.

Due to the fact that the pairing function is bijective, we know that for every such pair $\langle x, s\rangle$ exists a number p that is mapped to the pair. By definition of $\bar{R}_c$ it is obvious that f enumerates every non-random number. □

Lemma 5.13 (R_c is infinite) $\neg\mathcal{X}\ R_c$

Proof We assume a list l containing all the random numbers and derive a contradiction. By the unboundedness (Lemma 5.11), we can get a random number x that has length $S(\max_{x\in l}|\lceil x\rceil|)$. x must be in l by the definition of l and consequently we have $|\lceil x\rceil| \leqslant \max_{x\in l}|\lceil x\rceil|$. This is a contradiction. □

Last, we need to prove that there is no infinite, enumerable sub-predicate of the random numbers:

Lemma 5.14 $\mathrm{MP} \to \neg(\exists q : \mathbb{N} \to \mathbb{P}.\, \mathcal{E}\, q \wedge \neg \mathcal{X}\, q \wedge (\forall x : \mathbb{N}.\, q\, x \to \mathrm{R_c}\, x)$.

Proof We assume the infinite, enumerable predicate q which is a sub-predicate of $\mathrm{R_c}$. Due to the fact that q is infinite and has an enumerator $f : \mathbb{N} \to \mathbb{ON}$, we know that for every number m, there classically exists a number n so that $f(n) = {}^\circ x$ with $|\lceil x \rceil| > m$ and qx.

With Markov's principle we can eliminate the double negation and get:

$$\forall m : \mathbb{N}.\, \exists n : \mathbb{N}.\, \text{if } (fn) \text{ is } ({}^\circ x) \text{ then } |\lceil x \rceil| > m \text{ else } \bot$$

With this we only have the propositional existence of such an n which we cannot use in a computational context. We can get a computational function with the existential witness operator (Fact 2.20), as the the order of numbers is decidable and hence the whole body of the existential quantification.

Now we can compute for all m an n on which f enumerates a number longer than m. Let $g : \mathbb{N} \to \mathbb{N}$ be the function which receives m and directly outputs a number longer than m which satisfies q. In other words, g satisfies $q(g(m))$ and $m < |\lceil g(m) \rceil|$, for all m.

With Fact 3.5 we have a k such that $\forall m.\exists i s.\phi_c^s\, i = {}^\circ(g(m)) \wedge |\lceil i \rceil| \leqslant |\lceil m \rceil| + k$. Furthermore, $q(g(m))$ implies $\mathrm{R_c}\,(g(m))$ and hence $|\lceil g(m) \rceil| \leqslant i$. Overall we have

$$m \overset{\text{Def. } g}{<} |\lceil g(m) \rceil| \overset{\mathrm{R_c}\,(g(m))}{\leqslant} |\lceil i \rceil| \leqslant |\lceil m \rceil| + k$$

and the transitivity of leq gives us $m \leqslant |\lceil m \rceil| + k$. This is a contradiction with Fact 2.29. □

It might seem unnecessary that we are looking for an index n so that the enumerator outputs a number x with $|\lceil x \rceil| > m$. However, if we are looking for a number x directly so that $qx \wedge |\lceil x \rceil| > m$ then Markov's principle would not suffice: q is not decidable and hence we cannot define a boolean test for $qx \wedge |\lceil x \rceil| > m$. By using the enumerator, we automatically get that the enumerated number satisfies q and if (fn) is $({}^\circ x)$ then $|\lceil x \rceil| > m$ else $\bot$ can be computed by a boolean test. Hence, this way Markov's principle suffices.

The idea of this lemma is similar to the proof of the uncomputability of Kolmogorov complexity, as it also makes use of the Berry paradox. Accordingly, the use of Markov's principle is necessary for the same reasons as in the uncomputability proof.

With these results, the simpleness of the non-random numbers follows immediately:

Theorem 5.15 MP $\to$ univ c $\to$ simple $\overline{R}_c$

Proof By Lemma 5.12, Lemma 5.13 and Lemma 5.14. □

With the simpleness of $\overline{R}_c$ we can conclude that the non-random numbers are undecidable and many-one incomplete:

Corollary 5.16 *The non-random numbers are undecidable.*

Proof Theorem 5.15 and Fact 2.13. □

Corollary 5.17 *The non-random numbers are many-one incomplete.*

Proof Theorem 5.15 and Fact 2.14. □

5.4 Comparison to the Textbook Proof

With Odifreddi's design decision to use the identity as size there are multiple minor differences between his proof and our formalization. Additionally, he uses the equivalent of our codes as the measurement for the Kolmogorov complexity. In contrast, we measure the complexity with the input for the code.

His approach enables Odifreddi to give a second proof for the non-existence of an infinite, recursively enumerable set of random numbers. The proof utilizes Roger's fixed-point theorem [25] besides the Berry paradox and crucially depends on the fact that the fixed-point theorem makes a statement about the input that is the basis for Kolmogorov complexity.

This proof is not feasible in our setting for two reasons: First, our version of Church's thesis for partial functions is not potent enough to prove the fixed-point theorem, as Forster's version for whole families of functions is needed [8]. Second, our definition of Kolmogorov complexity is based on the level of inputs, whereas the fixed-point theorem makes a statement about codes, so that it cannot manipulate the complexity in the way it would be required.

5.5 Discussion

The choice of our definition for (non-)randomness is essential for obviating the need for excluded middle. Alternatively, if we had followed the predominant approach of defining a number as non-random, if its Kolmogorov complexity is smaller than itself, then the non-random numbers would not be enumerable (Lemma 5.12) in constructive logic. To prove the enumerability in this scenario, we need to compute the Kolmogorov complexity of a number x, only given an input shorter than x and a step count so that ϕ outputs x. Not even Markov's principle suffices for that because we would need to define a boolean test which computes Kolmogorov complexity. This is not possible due to the uncomputability of Kolmogorov complexity (Theorem 4.4).

The non-random numbers are not only interesting because they are one of the few known simple sets which have an organic background. Additionally, we could prove their simpleness with solely Church's thesis. Forster's proofs of simple predicates always required the S_n^m theorem [8].

Odifreddi also discussed that the non-random numbers are effectively simple [21]. With that, the question for completeness with respect to Turing reductions of the non-random numbers can be answered affirmatively. In Chapter 7 we will discuss the situation for truth-table reductions.

Chapter 6

A Lower Bound for the Random Numbers

With the fact that there is at least one random number of every length (Lemma 5.9), the question of the ratio between random and non-random numbers arises. Naturally, no absolute answer is possible, as it highly depends on the respective code. Yet for universal codes we can still give a lower bound for the portion of random numbers for every length. Martin Kummer proved in 1996 that for every universal code c and length n there exists a constant k so that at least $\frac{1}{k}$ of the numbers with length n are random [16].

In this chapter we introduce the notation $R_c^n / \bar{R}_c^n$ for the random/non-random numbers of length n. We treat these like lists, but of course they are not computable. In reality, every proof that uses this notation assumes a duplicate-free list which contains exactly the random/non-random numbers of length n.

6.1 The Proof on Paper

We assume a universal code c. Kummer's idea is to define a partial function $\eta : \mathbb{N} \to \mathbb{N} \to \mathbb{ON}$ which only outputs non-random numbers. With a clever definition Kummer forces η to output a random number if there are less than $\frac{2^n}{k}$ random numbers of length n which leads to a contradiction. In the following we discuss Kummer's proof. Note that for this introduction we temporarily revert to the usual definition of binary numbers with valueless leading zeros to stay true to the original proof. In particular, we use the same notations ($\lceil \cdot \rceil : \mathbb{N} \to \mathbb{LB}$ and $\lfloor \cdot \rfloor : \mathbb{LB} \to \mathbb{N}$) but the functions are changed accordingly. Afterwards, we deal with the explicit Coq proof.

Making the outputs non-random If we consider the complexity with regard to η, it is easy to define η in such a way that every output is non-random: For an input i simply output any number o with $|\lceil o \rceil| < |\lceil i \rceil|$.[1] For the input 0, simply diverge.

[1] Note that this does not pose a contradiction to the previously discussed facts: η is not universal so that not all numbers need to have a (finite) complexity and hence we can only output non-random numbers.

However, this is not sufficient as the non-randomness for η does not imply the same for the universal code c. Yet it is possible to achieve this with the powerful invariance theorem (Theorem 3.6). Through PCT, we can obtain a code computing the partial function η. The invariance theorem then gives us a constant d so that $\forall x.\, \mathrm{KC}_c\, x \leqslant \mathrm{KC}_\eta\, x + d$. Hence, some x would be non-random with respect to c if

$$\mathrm{KC}_\eta\, x + d < |\lceil x \rceil| \;\Leftrightarrow\; \mathrm{KC}_\eta\, x < |\lceil x \rceil| - d \;\Leftrightarrow\; \mathrm{KC}_\eta\, x \leqslant |\lceil x \rceil| - d - 1$$

Consequently, η shall output some o on input i with $|\lceil i \rceil| + d + 1 \leqslant |\lceil o \rceil|$. There is one apparent problem with this proposal: We cannot possibly know the constant d when we are defining η considering d in turn depends on η, so that this would be a circular definition.

Kummer's solution to this problem is to incorporate the value of d in the input. That means we do not limit ourselves to one value for d but essentially cover all possible values of d. For that η interprets the number of leading false until the first true in the input as the value of d. With that all the information for the determination of the minimal length of the output is self-contained in every input. We still do not know the true d for η but we cover every possibly d so that one of these guesses must be correct. Hence, we can define η so that for the correct number of leading false in the input, η outputs only non-random numbers with respect to c. The construction is uniform in d but only the actual d is relevant for the proof. Because of that, we use d as if it is already the real one, as we necessarily cover it.

As a result of this addition of d in the input, our lower bound for the length of the output needs to be adapted. The inputs now have the form $\lceil i \rceil = \mathrm{false}^d +\!\!+ [\mathrm{true}] +\!\!+ \lceil i' \rceil$ so that $|\lceil i \rceil| = |\lceil i' \rceil| + d + 1$. Overall, we then have the following constraint for the output o:

$$|\lceil i' \rceil| + 2d + 2 \leqslant |\lceil o \rceil|$$

With this property, we ensure the non-randomness of every output of η for the correct d with regard to the universal c. To be more precise, the length n of the outputs o is chosen as exactly the lower bound, i.e. $n = |\lceil o \rceil| = |\lceil i' \rceil| + 2d + 2$. This exposes the problem that we can only ensure the non-randomness of numbers with length $n \geqslant 2d + 2$ as $|\lceil i' \rceil| \geqslant 0$. So, for $n \geqslant 2d + 2$ we know that η can output 2^{n-2d-2} different non-random numbers for every output length n. Kummer's concrete definition of η guarantees that 2^{n-2d-2} is also the lower bound for $|\mathrm{R}_c^n|$ and hence we have $k = 2^{2d+2}$ for the constant whose existence is asserted in this lemma.

The concrete definition of η We want to derive a contradiction if $|\mathrm{R}_c^n| < 2^{n-2d-2}$. We know $|\mathrm{R}_c^n| \geqslant 1$ and hence $n > 2d + 2$[2]. Consequently, η can output non-random numbers of length n.

[2] Corollary 5.10 proves this for the bijective binary encoding. The proof is identical for the ordinary binary encoding.

There are exactly 2^{n-2d-2} possible cardinalities of R_c^n that we want to prove impossible. The proof idea is based on the fact, that η can output exactly 2^{n-2d-2} non-random numbers of length n. Therefore, η is defined in such a way that we can explicitly lead every value of $|R_c^n|$ that is smaller than 2^{n-2d-2} to a contradiction. We construe i' as $|R_c^n|$ so that if $|R_c^n| < 2^{n-2d-2}$ then i' necessarily assumes the value $|R_c^n|$.

This is extremely valuable because for the correct $i' = |R_c^n|$ we can infer $|\bar{R}_c^n| = 2^n - 1 - i'$. From Fact 5.12, we have the enumerability of the non-random numbers and we know that exactly $2^n - 1 - i'$ distinct numbers from $\bar{R}_c^n$ can be enumerated. So, we wait until all $2^n - 1 - i'$ different non-random numbers of length n were enumerated and let η return any number of length n that was not enumerated. Hence, the output of η must be random which is a contradiction as every output of η (for the correct d) is non-random.

In the case that $i' \neq |R_c^n|$, η calculates the return value identically and thus diverges if $i' < |R_c^n|$.

6.2 The Proof in Coq

From here on we revert to the bijective binary encoding (Axiom 2.28).

There are some evident and some rather hidden hurdles we need to overcome for the formalization of this proof in Coq. To begin with, we want to enumerate m different elements of $\bar{R}_c^n$ into a list, if they exist. Second, we need to find a way to deal with the necessity of leading zeros, as the values of our binary numbers are changed by attaching any digit.

6.2.1 A List of enumerated Numbers of Length n

Given a step count it is easy to enumerate some non-random numbers of length n in a list. Then, when there a m distinct values, we would be done. Unfortunately, we have to consider a few more factors that need to be fulfilled.

First of all, η needs to be monotonic, i.e. not suddenly change the output with a higher step count. Consequently, we have to ensure that once we have enumerated m distinct elements, that these numbers do not change and no other number gets added with more steps. Otherwise, η might return a number that is not in this list for s steps, but might appear in it after $s + 1$ steps, so that it cannot be returned by η anymore.

Second, for our convenience we want this list to be duplicate-free by definition. This simplifies detecting when m distinct elements have been reached and makes it easier to compute a number of length n not in this list.

With these things in mind it is not too hard to compute this list. Nevertheless, we have tried two approaches to define a corresponding function and can conclude

that one of them seems to be strictly easier in terms of proving the desired properties.

Our first attempt, made use of predefined functions like *nodup* and *skipn* which remove duplicates and cut off a specified number of elements respectively. The main difficulty was to prove that the final m elements enumerated, if they exists, are unaltered for a higher step count. This was due to the fact that the function was a composition of five predefined functions and depended considerably on the internal details of their definitions. Therefore, the properties provided by the standard library were not sufficient, and unpleasant inductive proofs would have been necessary.

It turns out that it is easier to define this function from the ground up. This way we can explicitly ensure the desired properties in a straightforward way that combines the steps that would be 'executed' by the predefined functions in succession.

This is the final definition we settled on:

Definition 6.1 *Let* $f : \mathbb{N} \to \mathbb{ON}$ *be the enumerator of the non-random numbers.*

$$
\begin{array}{lll}
\mathsf{F}\ m\ n\ 0 & := & \text{if} \quad f(0) \text{ is } {}^{\circ}x \\
 & & \text{then} \quad \text{if } |\lceil x \rceil| = n \wedge m \neq 0 \text{ then } [x] \text{ else } [\,] \\
 & & \text{else} \quad [\,] \\
\mathsf{F}\ m\ n\ Ss & := & \text{if} \quad f(Ss) \text{ is } {}^{\circ}x \\
 & & \text{then} \quad \text{if } |\lceil x \rceil| = n \wedge m \neq |\mathsf{F}\ m\ n\ s| \wedge x \notin (\mathsf{F}\ m\ n\ s) \text{ then } x :: (\mathsf{F}\ m\ n\ s) \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{else } \mathsf{F}\ m\ n\ s \\
 & & \text{else} \quad \mathsf{F}\ m\ n\ s
\end{array}
$$

F evidently satisfies all the properties we demanded and proving most of them is straightforward. Nevertheless, some facts require a lot of work.

All the following facts are straightforward with induction on the step count s.

Fact 6.2 $\forall mns \in \mathbb{N}.\, \mathrm{NoDup}\,(\mathsf{F}\ m\ n\ s)$

Fact 6.3 $\forall mns \in \mathbb{N}.\, |\mathsf{F}\ m\ n\ s| \leqslant m$

Fact 6.4 $\forall mns \in \mathbb{N}.\, |\mathsf{F}\ m\ n\ s| \leqslant Ss$

Fact 6.5 $\forall mnsx \in \mathbb{N}.\, x \in (\mathsf{F}\ m\ n\ s) \to \overline{R}_c\, x$

Fact 6.6 $\forall mnsx \in \mathbb{N}.\, x \in (\mathsf{F}\ m\ n\ s) \to |\lceil x \rceil| = n$

Fact 6.7 $\forall mns \in \mathbb{N}.\, (\mathsf{F}\ m\ n\ s) \subseteq (\mathsf{F}\ m\ n\ (Ss))$

These properties are all trivial consequences of the definition of F so that the induction easily goes through with case analyses on the various conditions of the conditional.

Facts for the step count Next, we prove a generalized version of Fact 6.7 which is necessary for the monotonicity of η.

Lemma 6.8 $\forall mnss' \in \mathbb{N}.\, s' \geqslant s \to (F\ m\ n\ s) \subseteq (F\ m\ n\ s')$

Proof Induction on s'. In the base case also s must be equal to 0, so that the goal follows by the reflexivity of list inclusion.

In the successor case, we have $Ss' = s \vee s' \geqslant s$ from $Ss' \geqslant s$. The case $Ss' = s$ is also proved with the reflexivity of the inclusion. For $s' \geqslant s$, we use the transitivity of list inclusion with Fact 6.7 and the inductive hypothesis. □

Fact 6.9 $\forall mnss' \in \mathbb{N}.\, s' \geqslant s \to |(F\ m\ n\ s')| \geqslant |(F\ m\ n\ s)|$

Proof By the pigeonhole principle and Lemma 6.8. □

Fact 6.10 $\forall mnss' \in \mathbb{N}.\, |(F\ m\ n\ s')| = |(F\ m\ n\ s)| \to (F\ m\ n\ s') \equiv (F\ m\ n\ s)$

Proof We either have $s \leqslant s'$ or $s' \leqslant s$. Both cases are analogous so that we assume $s \leqslant s'$. $(F\ m\ n\ s) \subseteq (F\ m\ n\ s')$ is a consequence of Fact 6.8. We know $|(F\ m\ n\ s')| = |(F\ m\ n\ s)|$, NoDup $(F\ m\ n\ s)$ (Fact 6.2) and $(F\ m\ n\ s) \subseteq (F\ m\ n\ s')$ (Fact 6.8). Hence, we have $(F\ m\ n\ s') \subseteq (F\ m\ n\ s)$ with Fact 2.33. □

Facts for the desired length Now, we turn from facts focusing on the step count s, to similar properties for the desired size m. These proofs are a lot more tedious because F recurses on s and m is left unchanged so that induction on m is futile. All the main facts basically need to be proven by brute force: Induction on the step count and case analysis on the conditions of the conditionals.

Lemma 6.11

$$\forall mns \in \mathbb{N}.\, \exists L : \mathbb{L}\mathbb{N}.\, |L| \leqslant 1 \wedge F\ (Sm)\ n\ s \equiv L +\!\!+ (F\ m\ n\ s) \wedge \text{NoDup}\ (L +\!\!+ (F\ m\ n\ s))$$

Proof By induction on s and case analysis following the definition of F. In the inductive step the inductive hypothesis either gives us the desired list containing one element or $[]$. In the $[]$ case we check if an element is added to $F\ (Sm)\ n\ s$ that cannot be added to $F\ m\ n\ s$ anymore. If there is such an element x then $[x]$ is the desired list. Otherwise, $[]$ works because either an element is added to both $F\ (Sm)\ n\ s$ and $F\ m\ n\ s$ or to neither. In both cases, these lists are equivalent with the inductive hypothesis. □

Fact 6.12 $\forall mns \in \mathbb{N}.\, |F\ (Sm)\ n\ s| = Sm \to |F\ m\ n\ s| = m$

Proof With Lemma 6.11 and the fact that duplicate-free and equivalent lists have the same length. □

Fact 6.13 $\forall mm'ns \in \mathbb{N}.\, m' \geqslant m \to |F\ m'\ n\ s| = m' \to |F\ m\ n\ s| = m$

Proof By induction on m'. In the inductive step, we know $m' \geqslant m \vee Sm' = m$.

For $m' \geqslant m$, the goal follows with Fact 6.11 and the inductive hypothesis. For $Sm' = m$, the claim is exactly given by the hypothesis $|F\ (Sm')\ n\ s| = Sm'$. □

Lemma 6.14 $\forall mns \in \mathbb{N}. |F\ (Sm)\ n\ s| \leqslant m \to F\ m\ n\ s \equiv F\ (Sm)\ n\ s$

Proof By induction on s and case analysis following the definition of F. □

Lemma 6.15 $\forall mm'ns \in \mathbb{N}. |F\ m'\ n\ s| \geqslant m \to |F\ m\ n\ s| = m$

Proof By induction on m'. In the base case we have $m = 0$ by Fact 6.3. The goal also follows with Fact 6.3.

In the inductive step we either have $Sm' = m \vee m \leqslant m'$ by Fact 6.3. The first case follows with Fact 6.3.

For $m \leqslant m'$, it suffices to show $|F\ m'\ n\ s| = |F\ (Sm')\ n\ s| \vee |F\ m'\ n\ s| = m'$: If we have $|F\ m'\ n\ s| = |F\ (Sm')\ n\ s|$ then the goal follows immediately by the inductive hypothesis. With $|F\ m'\ n\ s| = m'$, the goal follows with Fact 6.13.

We need to show $|F\ m'\ n\ s| = |F\ (Sm')\ n\ s| \vee |F\ m'\ n\ s| = m'$. With Fact 6.3, we have $|F\ (Sm')\ n\ s| = Sm' \vee |F\ (Sm')\ n\ s| \leqslant m'$. In the first case $|F\ m'\ n\ s| = m'$ follows immediately with Fact 6.12. Given $|F\ (Sm')\ n\ s| \leqslant m'$, we know $F\ m'\ n\ s \equiv F\ (Sm')\ n\ s$ by Lemma 6.14. $|F\ m'\ n\ s| = |F\ (Sm')\ n\ s|$ is a consequence of the fact that both of these lists are duplicate-free (Fact 6.2). □

These facts all focused on the implications of an m' on a $m \leqslant m'$. Now we want to consider the opposite direction:

Fact 6.16 $\forall mns., (F\ m\ n\ s) \subseteq (F\ (Sm)\ n\ s)$

Proof By induction on s and case analysis following the definition of F. □

Lemma 6.17 $\forall mm'ns., m' \geqslant m \to (F\ m\ n\ s) \subseteq (F\ m'\ ns)$

Proof By induction on m' and Fact 6.16 in the successor case. □

The correctness of F Now, we also want to show that $(F\ m\ n\ s)$ at some point returns a list with m elements, if there are m different non-random elements of length n. For that, we need the following lemmata:

Lemma 6.18 *Let* $f : \mathbb{N} \to \mathbb{ON}$ *be the enumerator of the non-random numbers. For all* $n, m, s, x, y : \mathbb{N}$ *with* $fx = {}^{\circ}y$, $|\lceil y \rceil| = n$ *and* $x \leqslant s$, *it holds that* $y \in (F\ m\ n\ s)$ *or* $|F\ m\ n\ s| = m \wedge m \leqslant s$

Proof By induction on s. The base case follows by case analysis on m. For the successor case, we do a case analysis on $x \leqslant s \vee x = Ss$.

In the case $x \leqslant s$, we do case analysis on the inductive hypothesis. For $x \in (F\ m\ n\ s)$ we show $x \in (F\ m\ n\ (Ss))$ with Fact 6.7. If we have $|F\ m\ n\ s| = m \wedge m \leqslant s$, we easily get $|F\ m\ n\ Ss| = m \wedge m \leqslant Ss$ with Fact 6.9 and 6.3.

For $x = Ss$, we know that y fulfills all conditions to be in $(F\ m\ n\ (Ss))$ unless $|F\ m\ n\ s| = m$. In this case, Fact 6.9 and 6.3 gives us $|F\ m\ n\ (Ss)| = m$. $m > Ss$ results in a contradiction as $|F\ m\ n\ s| \leqslant Ss$ by Fact 6.4. □

Lemma 6.19 *Let $f : \mathbb{N} \to \mathbb{ON}$ be the enumerator of the non-random numbers. For duplicate-free lists L of non-random numbers, there exists a duplicate-free list L' so that $^\circ x \in (f @ L')$ if and only if $x \in L$.*

Proof Similarly to previous proofs, it suffices to find an L' for the first m elements of L. By the definition of enumerability (Definition 2.8), for all $x \in L$ there exists an $n : \mathbb{N}$ so that $fn = {}^\circ x$. In the inductive step, the list L' is obtained by attaching the corresponding n to the start of the list from the inductive hypothesis. □

Lemma 6.20 *Given a duplicate-free list L which contains m non-random elements of length n, there exists a step count s so that $|F\ m\ n\ s| = m$.*

Proof With Fact 6.19 we obtain a list $L' : \mathbb{LN}$ which contains the numbers on which f enumerates exactly the members of L. We claim that the step count $s = \max_{x \in L'}(x)$ satisfies $|F\ m\ n\ s| = m$.

If we could show that all elements from L are in $F\ m\ n\ (\max_{x \in L'}(x))$ then we would be done by the pigeonhole principle. Unfortunately, that must not necessarily be the case as there could be a non-random element not in L that fills up one of the m spaces. With Fact 6.15 it would suffice to show that for an $m' \geqslant m$ there would be at least m elements in $F\ m'\ n\ (\max_{x \in L'}(x))$. To ensure that all the desired elements enumerated in the $\max_{x \in L'}(x)$ steps end up in $F\ m'\ n\ (\max_{x \in L'}(x))$, we choose $m' = S(\max_{x \in L'}(x))$.

We need to prove $|F\ (S(\max_{x \in L'}(x)))\ n\ (\max_{x \in L'}(x))| \geqslant m = |L|$. For that we use the pigeonhole principle and must show $L \subseteq (F\ (S(\max_{x \in L'}(x)))\ n\ (\max_{x \in L'}(x)))$.

It suffices to show that for all $a : \mathbb{N}$ the first a elements of L are contained in

$$(F\ (S(\max_{x \in L'}(x)))\ n\ (\max_{x \in L'}(x)))$$

Use induction on a. The base case is trivial. For the inductive step, we know that the first a elements of L fulfill the property by the inductive hypothesis. If there is an a-th element x in L, then we know that there is also an element y in L' so that $fy = {}^\circ x$. With Lemma 6.18 and the fact that $y \leqslant (\max_{x \in L'}(x))$, we have the desired goal or $(S(\max_{x \in L'}(x))) \leqslant (\max_{x \in L'}(x))$. The latter is evidently a contradiction. □

6.2.2 Computing the Return Value

We can now enumerate $\bar{\mathrm{R}}_c$ into a list with the desired properties, if it exists. Then, η needs to return a number that is not present in the list but has the same length as all the elements. It is easy to prove that η always returns the same number for a higher step count if we return the least number not included in the list.

Lemma 6.21 $\forall mns \in \mathbb{N}.\, m < 2^n \to \Sigma x : \mathbb{N}.\, \mathrm{least}\,(\lambda x.\, x \notin (\mathrm{F}\ m\ n\ s) \wedge |\lceil x \rceil| = n)\ x$

Proof We apply the existential least witness operator (Fact 2.24). The decidability is easy because $\mathbb{N}$ is discrete hence list membership is also decidable for $\mathbb{LN}$. To prove the existence of a number of length n that is not in $(\mathrm{F}\ m\ n\ s)$, we use the pigeonhole principle. We have $m < 2^n$ and consequently $|\mathrm{F}\ m\ n\ s| < 2^n$ by Fact 6.3. This closes the proof. □

The argument $m < 2^n$ in the proof above is superfluous as we know that the non-random numbers always satisfy this property (Lemma 5.9). Nevertheless, it simplifies the proof as we do not need to prove that $|\mathrm{F}\ m\ n\ s|$ is always smaller than 2^n. We never choose $m \geqslant 2^n$, so that this decision is convenient.

For η, we only want to return this least element from Lemma 6.21 if $|\mathrm{F}\ m\ n\ s| = m$, as otherwise not enough elements have been enumerated yet. Additionally, we require $m < 2^n$ so that Lemma 6.21 can compute the number in the first place.

Lemma 6.22 *We can define a function* $\mathrm{G} : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{ON}$ *with* $\mathrm{G}\ m\ n\ s = {}^\circ x$ *if* $|\mathrm{F}\ m\ n\ s| = m$ *and* $m < 2^n$*, where x is the least element provided by Lemma 6.21, and* $\mathrm{G}\ m\ n\ s = \emptyset$ *otherwise.*

Proof By case analysis on $|\mathrm{F}\ m\ n\ s| = m \vee |\mathrm{F}\ m\ n\ s| < m$ and $m < 2^n \vee m \geqslant 2^n$ and Lemma 6.21. □

Finally, we discuss the monotonicity of G. The groundwork for this result was laid in the preceding facts.

Lemma 6.23 $\mathrm{G}\ m\ n$ *is monotonic, for all* $m, n : \mathbb{N}$.

Proof We are given $x, s, s' : \mathbb{N}$ with $\mathrm{G}\ m\ n\ s = {}^\circ x$ and $s \leqslant s'$ and need to show $\mathrm{G}\ m\ n\ s' = {}^\circ x$. $\mathrm{G}\ m\ n\ s' = \emptyset$ is a contradiction with Fact 6.9 and Lemma 6.22. Hence, $\mathrm{G}\ m\ n\ s' = {}^\circ x'$ and it suffices to show $x = x'$. Follows by the uniqueness of the least witness (Fact 2.22) and Fact 6.10. □

6.2.3 Defining η

In the previous sections, we have seen all tools necessary to define η. However, we still need to imitate the leading zeros Kummer used to obtain a longer number with

the same value. As discussed above, leading zeros that have no effect on the value do not exist in the bijective binary encoding. Consequently, we need to artificially increase the length of a binary number without changing the number itself.

We achieve this in a similar fashion to the pairing of two binary strings, which appends leading falses and one true as a separator to represent the length of the first component. Such a separator symbol marks the end of the valueless leading falses and the start of the real number. A problem we have to consider is that we lose one digit for the input because the true used as a separator is a mandatory part of the string. The proof expounded above relies on the fact that we can evidently express 2^{n-2d-2} values with $n-2d-2$ bits. However, if we subtract the separator bit we only have $n-2d-2-1$ bits left and hence $2^{n-2d-2}-1$ distinct numbers in the bijective encoding (Fact 2.31), so that we are off by one.

Recall that there is at least one random number of every length (Lemma 5.11) and that i' in the input of η can assume any possible value $|R_c^n| < 2^{n-2d-2}$. Hence, we do not actually require 2^{n-2d-2} values because we can rule out $|R_c^n| = 0$. Thus, we interpret i' as $|R_c^n| - 1$.

With that, we can define η in Coq.

Definition 6.24

$$\eta\ \lfloor \mathrm{false}^d +\!\!+ [\mathrm{true}] +\!\!+ \mathrm{false}^a +\!\!+ [\mathrm{true}] +\!\!+ i' \rfloor\ s := \mathrm{let}\ n := (Sa+|i'|+2d+2)\ \mathrm{in}\ G\ (2^n - S\lfloor i \rfloor)\ n\ s$$

Note that here n even cannot be smaller than $2d+2+1$ due to the additional separator bit. This does not pose a problem in the proof.

Lemma 6.25 *$\eta\ x$ is monotonic for all $x : \mathbb{N}$.*

Proof By Lemma 6.23. □

6.2.4 The Proof

The complete proof is now only a matter of putting the pieces together.

Theorem 6.26 *Given excluded middle and* PCT, *there exists a constant $k : \mathbb{N}$ so that every duplicate-free list containing all random numbers of some length $n : \mathbb{N}$ contains at least $\frac{2^n}{k}$ elements.*

Proof We assume a duplicate-free list $l : \mathbb{L}\mathbb{N}$ which contains all random numbers of length n. By PCT and the invariance theorem (Theorem 3.6) we can obtain a code c' with $\phi_{c'} \simeq \eta$ and a constant d so that $\forall x.\, KC_c\ x \leqslant KC_{c'}\ x + d$.

We choose $k = 2^{2d+2}$ and show that $|l| < 2^{n-2d-2}$ leads to a contradiction. From Corollary 5.10 we know $|l| \geqslant 1$ and hence $n \geqslant 2d+2+1$, so that η can output

numbers of length n. We use excluded middle to decide whether

$$\eta\ \lfloor \mathrm{false}^d +\!\!+ [\mathrm{true}] +\!\!+ \mathrm{false}^{(n-2d-2)-S|\lceil |l|-1\rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil |l|-1\rceil \rfloor$$

terminates. If η does not terminate we have a contradiction with $L_n - l'$ and Fact 6.20. So, we know η terminates in s steps with $^\circ o$. We want to prove falsity, hence we can eliminate double negations and obtain the complexity kc of o with regard to c and kc' with regard to c'. By the definition of Kolmogorov complexity, we know that

$$kc' \leqslant |\mathrm{false}^d +\!\!+ [\mathrm{true}] +\!\!+ \mathrm{false}^{(n-2d-2)-S|\lceil |l|-1\rceil|} +\!\!+ [\mathrm{true}] +\!\!+ \lceil |l|-1\rceil| = n-d-1$$

$\bar{R}_c\ o$ follows then by the invariance theorem

$$kc \leqslant kc' + d \leqslant n-d-1+d < n$$

We prove that o is in $F\ (2^n - |l|)\ n\ s$ which is a contradiction by the definition of η. It suffices to show that all non-random numbers of length n are in $F\ (2^n - |l|)\ n\ s$. So, let x be a non-random number of length n. We know with Fact 2.34 that x must be in $F\ (2^n - |l|)\ n\ s$ or in l. If $x \in l$ then x is random which contradicts $\bar{R}_c\ x$. □

We believe that excluded middle is not necessary for this proof as the termination of η can be proved with Lemma 6.20. First rough, incomplete sketches which only require Markov's principle already exist.

6.3 Discussion

Our formalization closely follows Kummer's proof albeit we deviate from the original as we enumerate $2^n - |\bar{R}_c^n|$ non-random elements instead of Kummer's proposed $|\bar{R}_c^n|$. We believe that Kummer's proof is erroneous here as we cannot comprehend how the output of η is necessarily random with his approach.

Instead of defining η with the separator symbol so that the input can assume the value $|R_c^n|$ naturally, we could also omit the separator symbol and subtract the smallest value of length $n-2d-2$ from the input value. This way, we would have 2^{n-2d-2} inputs while also getting all values smaller than 2^{n-2d-2} as an input. Unfortunately, we have no knowledge of the smallest number of length n, as we assume the binary encoding as an axiom. Hence, this approach is not possible for us.

The formalization of this result was very painful at times because handling these long input lists and hypotheses can become confusing quickly.

This result was not only interesting intrinsically but also as a case study for this proof method. Kummer's proof of the truth-table completeness of the non-random numbers uses an almost identical approach. We will discuss this in detail in the following chapter.

Chapter 7

Towards a Formalization of the tt-completeness of the Non-Random Numbers

As discussed in Chapter 5 the m-incompleteness of the non-random numbers was discovered relatively swiftly in the late 1960s after the introduction of Kolmogorov complexity at the start of the decade. The completeness with respect to Turing reductions is also established. However, the question for the situation with regard to truth-table reductions was open for a long time.

In 1996, Martin Kummer was able to finally solve this problem by proving the tt-completeness of the non-random numbers [16]. The core idea of this proof is very similar to the approach he used for proving the lower bound of the random numbers, that we discussed in the previous chapter. Nevertheless, his construction is much more intricate and very sophisticated. In this chapter, we discuss Kummer's proof. The foundation of a potential formalization has been laid in the previous chapter but there are many difficult problems left that need to be solved.

7.1 A high-level Overview of Kummer's Proof

In this section, we give a cursory overview of Kummer's proof on a high level. Afterwards, we discuss Kummer's elaborate construction in detail.

We assume a universal code c in this chapter. To prove the tt-completeness of $\bar{R}_c$ we assume an enumerable predicate $p : \mathbb{N} \to \mathbb{P}$ and show $p \preceq_{tt} \bar{R}_c$.

Kummer constructs for almost every $x : \mathbb{N}$ a list S_x so that $p(x) \leftrightarrow (\text{Forall } \bar{R}_c\ S_x)$. For that, Kummer utilizes the approach used in Chapter 6 to define a function $\eta : \mathbb{N} \to \mathbb{N} \to \mathbb{ON}$ that can output 2^{n-2d-2} non-random numbers of every length n. Likewise, the concrete d is unknown during the definition of η but all possible values of d are covered as the construction is uniform in d. In the following we assume that d is the true constant. Furthermore, we do not consider the internal mechanics of η in detail again, but instead we directly use the ability of η to make numbers non-random: For every length $n \geqslant 2d + 2$ we can use η to ensure the

non-randomness of exactly 2^{n-2d-2} numbers of length n. That means, any list that we construct which contains (no more than) 2^{n-2d-2} elements, all of which have length n, can be certified non-random by η. Note that for every length n this is only possible for a total of 2^{n-2d-2} numbers, so only for one such aforementioned list. For a justification of this fact, we refer the reader to Chapter 6.1.

Kummer associates every x with one exclusive $n \geqslant 2d+2$ so that S_x only contains numbers of length n, $|S_x| = 2^{n-2d-2}$ and no S_y with $y = x$ contains numbers of length n. That means with η we can ensure the non-randomness of the elements of S_x. With that we would necessarily have $p(x) \to (\text{Forall } \bar{R}_c\ S_x)$ but this would render $(\text{Forall } \bar{R}_c\ S_x) \to p(x)$ impossible for non-trivial predicates p. η needs to make sure that $p(x)$ holds before it makes the elements of S_x non-random. Therefore, we enumerate p, and if x gets enumerated then η terminates and outputs the elements of S_x.

With that, we have ensured that η makes the elements of S_x non-random if x satisfies p. Nevertheless, it is still possible that the elements of S_x are non-random on their own. To guarantee $(\text{Forall } \bar{R}_c\ S_x) \to p(x)$, Kummer makes clever arrangements. The following intuitive explanation is only indicative of the real proof. In the next section, we give a complete and detailed description of Kummer's construction.

To prevent Forall $\bar{R}_c\ S_x$ when $p(x)$ does not hold, we enumerate as many non-random numbers of length n as possible and only stop at some step s if and only if the enumerator of p has outputted x after s steps. The crux is that due to the construction we will know that there exists an $i : \mathbb{N}$ such that the number k of enumerated numbers of length n satisfies $i * 2^{n-2d-2} \leqslant k < (i+1) * 2^{n-2d-2}$. This k is agnostic towards the fact whether the enumeration stopped prematurely, because $p(x)$, holds or if all non-random numbers of length n are asymptotically enumerated, because $p(x)$ does not hold. So, we know at least $i * 2^{n-2d-2}$ non-random numbers get enumerated. We define S_x as the 2^{n-2d-2} smallest numbers of length n which are not among these $i * 2^{n-2d-2}$ enumerated numbers.

If $p(x)$ holds then the 2^{n-2d-2} numbers in S_x are non-random through η. If $p(x)$ does not hold then at least one number in S_x must be random. Otherwise, we know that in addition to the $i * 2^{n-2d-2}$ non-random numbers that were enumerated, the 2^{n-2d-2} numbers in S_x are non-random, too. S_x and the enumerated numbers are disjoint so that there would be at least $i * 2^{n-2d-2} + 2^{n-2d-2} = (i+1) * 2^{n-2d-2}$ non-random numbers. The enumeration does not stop because $p(x)$ does not hold and hence $(i+1) * 2^{n-2d-2}$ non-random numbers get enumerated at some point. This is a contradiction because we know that less than $(i+1) * 2^{n-2d-2}$ numbers get enumerated.

This cursory description gives an idea of the argument which serves as the founda-

tion of the proof. The main challenge the proof overcomes is to ensure $i*2^{n-2d-2} \leqslant k < (i+1)*2^{n-2d-2}$. We see that this problem is also solved non-constructively. That means we construct these sets for all possible values of i and one of these i must necessarily work.

7.2 Kummer's tt-completeness Proof

We will now discuss Kummer's proof in detail. Except for small details and different wording, this is exactly the proof Kummer has constructed [16].

7.2.1 Preliminaries

We assume an enumerable predicate $p : \mathbb{N} \to \mathbb{P}$ from which we truth-table reduce to $\overline{R}_c$.

In the construction we enumerate non-random numbers of length n in a list for some number of steps s. This list is denoted by $M_{n,s}$. For the computation of this list we can use the function F (Definition 6.1) from the previous chapter:

$$M_{n,s} := F\, 2^n\, n\, s$$

We define sequences of lists of numbers $S_0, \ldots, S_{2d+1}$. The current length of the sequence S_i is given by $l(i)$. We use the notation $S_{i,x}$ for the x-th list in the i-th sequence. If defined, $m_{i,x}$ indicates the length of $S_{i,x}$.

In addition to these sequences, we define lists E_n which each contain 2^{n-2d-2} numbers of length $n \geqslant 2d+2$. These lists are the ones which are made non-random by η. This works because the lists are enumerable with the construction below. So, η computes the construction below and if the desired E_n is defined then η makes its elements non-random. Otherwise, η diverges for the inputs which output numbers of length n.

In Kummer's proof all the occurrences of lists are sets. We will represent these sets with lists because we will see that in every concrete step these sets are all finite.

7.2.2 The Construction

Initially, all lengths n are available, all sequences are empty and $m_{i,x}$ is undefined for all i, x.

In step s, the greatest $i < 2^{2d+2}$ is selected for which an $2d+2 \leqslant n \leqslant s+2d+2$ exists such that

- n is available
- $\forall jx.\, , j \geqslant i \to n \neq m_{j,x}$
- $i*2^{n-2d-2} \leqslant |M_{n,s}| < (i+1)*2^{n-2d-2}$

For this i, the smallest such n is selected. Then the following is done:

- make all $m_{j,x}$ with $m_{j,x} = n$ undefined
- $S_{i,l(i)} :=$ smallest 2^{n-2d-2} elements from $L_n - M_{n,s}$
- $m_{i,l(i)} := n$
- $l(i) := l(i) + 1$

Finally, for all x that get enumerated by the enumerator of p within s steps and for all j so that $m_{j,x}$ is defined and available:

- $E_{m_{j,x}} := S_{j,x}$
- $m_{j,x}$ is now unavailable

7.2.3 The Proof

In the previous section, we presented Kummer's construction which is used to construct the truth-table reduction. Now, we discuss how this reduction is realized.

Fact 7.1 *There exists a greatest* i_0 *such that the sequence* S_{i_0} *is infinite*

Proof In every step s one sequence gets extended: $n = s+2d+2$ is always available in step s as it could not have been selected in an earlier step because n must be $\leqslant s+2d+2$. Additionally, we know $0 \leqslant |M_{n,s}| < 2^n$ from Lemma 5.9. Hence, there exists an i with $0 \leqslant i < 2^{2d+2}$ such that $i * 2^{n-2d-2} \leqslant |M_{n,s}| < (i+1) * 2^{n-2d-2}$

There are finitely many sequences and at least one sequence is infinite. Consequently, there must be a greatest i_0 such that S_{i_0} is infinite. □

With this fact we know that there also must exist a step count s_0 so that no sequence S_j with $j > i_0$ gets extended in a step $s \geqslant s_0$. Furthermore, any $m_{i_0,x}$ defined after step s_0 never becomes undefined again because this only happens if a sequence S_j with $j > i_0$ gets selected which does not happen after step s_0. Let x_0 be the smallest number so that $\forall x \geqslant x_0.\, m_{i_0,x}$ gets defined after step s_0.

Theorem 7.2 $\forall x \geqslant x_0.\, p(x) \leftrightarrow \text{Forall } \overline{R}_c\ (S_{i_0,x})$

Proof Let $x \geqslant x_0$.

For the $\rightarrow$ direction we assume $p(x)$. That means that the enumerator for p outputs x after some step s. We know $m_{i_0,x}$ gets defined in some step s' because S_{i_0} is infinite and $m_{i_0,x}$ never gets undefined again. Hence, in step $\max\{s, s'\}$ the construction defines $E_{m_{i_0,x}} := S_{i_0,x}$ and η ensures the non-randomness of $S_{i_0,x}$.

For $\leftarrow$ we assume Forall $\overline{R}_c\ (S_{i_0,x})$ and $\neg p(x)$, and derive a contradiction. $S_{i_0,x}$ was defined in a step $s \geqslant s_0$ as the 2^{n-2d-2} smallest elements in $L_n - M_{n,s}$ for some

n. We know that all elements in $S_{i_0,x}$ are non-random by the assumption and $M_{n,s}$ only contains non-random elements by definition. Additionally, $M_{n,s}$ and $S_{i_0,x}$ are disjoint by the definition of $S_{i_0,x}$. Lemma 6.20 applied to $S_{i_0,x} +\!\!+ M_{n,s}$ together with Lemma 6.17 and 6.9 gives us an $s' > s$ so that $|M_{n,s'}| = |S_{i_0,x}| + |M_{n,s}|$.

In the construction, i is selected so that $i * 2^{n-2d-2} \leqslant |M_{n,s}| < (i+1) * 2^{n-2d-2}$. Therefore, $|M_{n,s'}| \geqslant 2^{n-2d-2} + i_0 * 2^{n-2d-2} = (i_0+1) * 2^{n-2d-2}$. $\neg p(x)$ implies that $m_{i_0,x} = n$ never becomes unavailable so that in a step $s_1 > s$ an $i > i_0$ and n get selected. This is a contradiction because $s_1 > s_0$ which means no $i > i_0$ can get selected in step s_1. □

Corollary 7.3 $p \preceq_{tt} \overline{R}_c$

Proof For the finite number of $x < x_0$ we can decide if $p(x)$ holds. Therefore, if $x < x_0$ and $p(x)$ holds the reduction returns $([], (\lambda x.\, \text{true}))$. If $x < x_0$ and $p(x)$ does not hold the reduction returns $([], (\lambda x.\, \text{false}))$.

For $x \geqslant x_0$, the reduction uses Theorem 7.2. We need a function which computes the conjunction over all elements of a bool list because all elements of $S_{i_0,x}$ must be non-random.

$$\text{allTrue}\ (l : \mathbb{L}\mathbb{B}) : \mathbb{B} := \text{ if } l \text{ is } x :: lr \text{ then } (x \wedge_{\mathbb{B}} (\text{allTrue } lr)) \text{ else true}$$

With that, the reduction returns $(S_{i_0,x}, \text{allTrue})$.

The correctness follows with Theorem 7.2. □

7.3 Ideas for a Coq Formalization

The non-constructiveness of the construction already indicates that excluded middle or Markov's principle is required for this proof in Coq. In general, a Coq formalization of this tt-completeness proof will have to deal with a lot of difficult problems. We address many of these problems and if possible we try to give approaches to overcome them. Note that the following considerations are very schematic.

7.3.1 The Construction

Although the sequences are extended infinitely often, in each concrete step the current state is finite. That means the sequence S_i which contains lists of numbers can be represented in step s by a list of list of numbers, i.e. $\text{seq} : \mathtt{Type} := \mathbb{L}(\mathbb{L}\mathbb{N})$. Additionally, we have exactly 2^{2d+2} sequences so that we can represent all sequences in a step s with a list of sequences $\mathbb{L}(\text{seq})$.

The E_n lists can be represented by a list of list of numbers too, but they do not necessarily need to be defined in order and some E_n might never be defined. Hence, we use $\mathbb{L}(\mathbb{O}(\mathbb{L}\mathbb{N}))$ so that in the case that E_n is defined but E_m with $m < n$ does not exist, we can use the value $\emptyset$ as a placeholder for E_m.

For the $m_{i,x}$ values for the $i-th$ sequence we could also use lists of numbers, but we have to consider that these values can become undefined again. Hence, we could use lists of $\mathbb{ON}$, so that $\emptyset$ is used for values that were defined once but are undefined now. The representation of the m values for all sequences would then be $\mathbb{L}(\mathbb{L}(\mathbb{ON}))$.

The lengths that are unavailable are also finite in each step so that we could represent these with a list of numbers $\mathbb{LN}$.

We have already mentioned that the lists $M_{n,s}$ can be computed by $F\ 2^n\ n\ s$ (Definition 6.1).

Finding the greatest i and the corresponding smallest n is possible in each step: There are only finitely many choices for i ($i < 2^{2d+2}$) and n ($2d+2 \leqslant n \leqslant s+2d+2$) so that we can check for all possible i and n if they fulfil the (decidable) properties.

Consequently, we can define a function C that given s returns the state of the construction after s steps. C would have the following type in this encoding:

$$C : \underbrace{\mathbb{N}}_{\text{The steps}} \to \underbrace{\mathbb{L}(\text{seq})}_{\text{The sequences}} \times \underbrace{\mathbb{L}(\mathbb{L}(\mathbb{ON}))}_{\text{The } m \text{ values}} \times \underbrace{\mathbb{LN}}_{\text{The unavailable lengths}} \times \underbrace{\mathbb{L}(\mathbb{O}(\mathbb{LN}))}_{\text{The } E_n \text{ lists}}$$

7.3.2 The Definition of η

For $\eta : \mathbb{N} \to \mathbb{N} \to \mathbb{ON}$ we use the approach from Chapter 6.2.3.

The length n of the numbers which η outputs is encoded in the input and as a partial function, η receives a step count s. η computes $C(s)$ and if E_n is defined then η outputs the 2^{n-2d-2} elements of E_n and otherwise $\emptyset$.

We have to consider that η needs to output 2^{n-2d-2} numbers of length n but the definition in Chapter 6.2.3 only allows $2^{n-2d-2}-1$ outputs. In contrast to this proof, here we do not need to ignore leading zeros so that we can make use of all 2^{n-2d-2} inputs.

7.3.3 Finding i_0

Proving that there is a greatest i such that S_i is infinite, is one of the hardest challenges of a formalization of this proof. Specifying the infiniteness of a sequence is not trivial either. One possibility to express the infiniteness of the sequence i could be that for all a there exists a step s in which S_i is longer than a. If we could find some i so that S_i is infinite then we could find the greatest i_0 so that S_{i_0} is infinite. For that we could use a 'greatest witness operator' which provides the greatest number in a list which satisfies a predicate.

Additionally, we also need to find the step count s_0 after which only sequences S_i with $i \leqslant i_0$ are extended. These proofs are most likely very challenging and require

hard work.

7.4 Discussion

The experience from the previous Chapter shows that a tt-completeness proof of the non-random numbers in Coq would be very tedious. In the lower bound proof for the non-random numbers we have laid the groundwork for a tt-completeness proof but the complexity of Kummer's construction is considerable. Especially, showing the required properties for the constructed lists will be an enormous task.

Nevertheless, both proofs have a common underlying approach: Enumerate many non-random numbers so that (almost) all non-random numbers were enumerated. Then, define η so that numbers that were not enumerated become non-random. With that we either have derived a contradiction or ensured a property.

Chapter 8

Conclusion

In this chapter, we want to discuss our overall results, related work, and potential future work.

In this thesis, we gave the first formalization of Kolmogorov complexity in synthetic computability theory in Coq. With this foundation, we proved the uncomputability of Kolmogorov complexity as well as the m-incompleteness of the non-random numbers. Furthermore, we proved the existence of a lower bound for the count of random numbers.

The usage of synthetic computability throughout this thesis was extremely natural. Obtaining codes with Church's thesis is extremely convenient. Without this comfort the proof of the lower bound for the non-random numbers probably becomes completely impractical. Having to define the elaborate η function (Definition 6.24) in a foreign model of computation like the λ-calculus seems extremely intricate. The considerably more complex construction for the tt-completeness proof is already a difficult challenge for the synthetic approach so that the usage of the λ-calculus almost seems unfeasible.

8.1 The Coq Mechanization

External Code We use many proofs from the Coq standard library, for example proofs on lists or arithmetic facts. Additionally, we also use the Equations plugin [34] and the std++ library [36].

Our mechanization is build upon the definitions and proofs belonging to the paper by Forster, Jahn, and Smolka [9]. Besides the fundamental axioms, we also use lemmata like the pigeonhole principle and the least/existential witness operator from this work.

A few proofs are from "Modeling and Proving in Computational Type Theory Using the Coq Proof Assistant" by Gert Smolka [29]. These facts are annotated accordingly.

Size of the Proofs The sizes of the Coq proofs are in general manageable. Only the lower bound proof for the random numbers (Chapter 6) is bigger in size. The main reason for this is the elaborate construction which necessitates many auxiliary facts.

Content	Spec	Proof
Preliminaries	34	167
List Facts	78	557
Binary Encoding	65	453
Kolmogorov Complexity and Facts for KC	21	157
The Uncomputability of KC	14	162
Simpleness of the Non-Random Numbers	47	365
Lower Bound for the Random Numbers	64	838
Total	323	2699

8.2 Related Work

Catt and Norrish have formalized Kolmogorov complexity in the *HOL4 interactive theorem prover* [3]. HOL4's logic is classical in contrast to Coq's constructive logic.

Catt and Norrish prove the invariance theorem and the uncomputability of Kolmogorov complexity but their overall focus is more on inequalities involving Kolmogorov complexity. Their work also considers variations of Kolmogorov complexity, like the conditional Kolmogorov complexity [15] which is the complexity of a number given additional information.

Their formalization relies on the λ-calculus and in parts on the general recursive functions as their model of computation. λ-calculus terms alone make up about 200 lines of code of Catt and Norrish's whole HOL4 formalization which has approximately 6000 lines of code.[1] This estimation does not include the overhead for proving results for the λ-calculus.

We believe that our synthetic approach to Kolmogorov complexity is less demanding as we can use the usual comfort for function definitions provided by Coq. This is because we can obtain a code for any function by simply defining the function in Coq and applying Church's thesis to it. Most of Catt and Norrish's theorems and inequalities require the coding in the λ-calculus or the general recursive functions so that the extra effort is not negligible.

8.3 Future Work

A major part of this thesis is the completeness status of the non-random numbers with respect to different reductions. We have proven the many-one incompleteness

[1] Determined with `grep` by searching for the constructors of λ-terms and specific functions returning λ-terms.

of the non-random numbers in Chapter 5 A formalization of the truth-table completeness of the non-random numbers in Coq is still missing. In Chapter 7 we discussed approaches to a formalization of this proof and many necessary definitions and facts for this proof were discussed in Chapter 6. This proof is very interesting but definitely a major challenge that will require a lot of work.

The main results in this thesis all require additional axioms like Markov's principle or excluded middle. For example, the uncomputability proof of Kolmogorov complexity (Chapter 4) requires Markov's principle. Yannick Forster suggested that it could be possible to prove the uncomputability of Kolmogorov complexity without Markov's principle by using partial functions.

The lower bound proof for the non-random numbers uses excluded middle in the current state. Further investigations to see which axiom is really needed would be very interesting.

Bibliography

[1] Andrej Bauer. "First Steps in Synthetic Computability Theory". In: *Electronic Notes in Theoretical Computer Science* 155 (2006), pp. 5–31. DOI: 10.1016/j.entcs.2005.11.049.

[2] Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*. Vol. 97. London Mathematical Society Lecture Note Series. Cambridge University Press, 1987. DOI: 10.1017/CBO9780511565663.

[3] 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, 2021, pp. 291–299. DOI: 10.1145/3437992.3439921.

[4] Gregory J. Chaitin. "On the Length of Programs for Computing Finite Binary Sequences". In: *Journal of the ACM (JACM)* 13.4 (1966), pp. 547–569. DOI: 10.1145/321356.321363.

[5] Gregory J. Chaitin. "On the Length of Programs for Computing Finite Binary Sequences: Statistical Considerations". In: *Journal of the ACM (JACM)* 16.1 (1969), pp. 145–159. DOI: 10.1145/321495.321506.

[6] *Code Golf Stack Exchange*. Accessed: 10.08.2021. URL: https://codegolf.stackexchange.com/.

[7] Thierry Coquand and Bassel Mannaa. "The Independence of Markov's Principle in Type Theory". In: *Logical Methods in Computer Science ; Volume 13* (2017), Issue 3, 1860–5974. DOI: 10.23638/LMCS-13(3:10)2017.

[8] Yannick Forster. "Computability in Constructive Type Theory". PhD thesis. Saarland University, 2021. URL: https://ps.uni-saarland.de/~forster/thesis.php.

[9] Yannick Forster, Felix Jahn, and Gert Smolka. "A Constructive and Synthetic Theory of Reducibility". Unpublished draft. 2021.

[10] 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*. ACM, 2019, pp. 38–51. DOI: 10.1145/3293880.3294091.

[11] Alexander Gammerman and Vladimir Vovk. "Kolmogorov Complexity: Sources, Theory and Applications". In: *The Computer Journal* 42.4 (1999), pp. 252–255. DOI: 10.1093/comjnl/42.4.252.

[12] Nicholas Griffin. *The Cambridge Companion to Bertrand Russell*. Cambridge University Press, 2003. DOI: 10.1017/S1079898600003486.

[13] Andrei N. Kolmogorov. "On Tables of Random Numbers". In: *Sankhyā: The Indian Journal of Statistics, Series A* 25.4 (1963), pp. 369–376. DOI: 10.2307/25049284.

[14] Andrei N. Kolmogorov. "On Works in Information Theory and some of Its Applications". In: *Selected Works of A. N. Kolmogorov: Volume III: Information Theory and the Theory of Algorithms*. Ed. by Albert N. Shiryayev. Springer, 1993, pp. 219–221. DOI: 10.1007/978-94-017-2973-4_14.

[15] Andrei N. Kolmogorov. "Three approaches to the quantitative definition of information". In: *Problems of information transmission* 1.1 (1965), pp. 3–11.

[16] Martin Kummer. "On the complexity of random strings". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1996, pp. 25–36. DOI: 10.1007/3-540-60922-9_3.

[17] Ming Li and Paul M.B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. 4th ed. Springer, 2019. DOI: 10.1007/978-3-030-11298-1.

[18] Donald A. Martin. "Completeness, the recursion theorem, and effectively simple sets". In: *Proceedings of the American Mathematical Society* 17.4 (1966), pp. 838–842. DOI: 10.1090/S0002-9939-1966-0216950-5.

[19] Per Martin-Löf. "The definition of random sequences". In: *Information and Control* 9.6 (1966), pp. 602–619. DOI: 10.1016/S0019-9958(66)80018-9.

[20] André Nies. *Computability and randomness*. Vol. 51. Oxford University Press, 2009. DOI: 10.1093/acprof:oso/9780199230761.001.0001.

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

[22] Christine Paulin-Mohring. "Introduction to the Calculus of Inductive Constructions". In: *All about Proofs, Proofs for All*. Vol. 55. Studies in Logic (Mathematical logic and foundations). College Publications, 2015. URL: https://hal.inria.fr/hal-01094195.

[23] Emil L. Post. "Recursively enumerable sets of positive integers and their decision problems". In: *bulletin of the American Mathematical Society* 50.5 (1944), pp. 284–316. DOI: 10.1090/S0002-9904-1944-08111-1.

[24] Fred Richman. "Church's thesis without tears". In: *The Journal of symbolic logic* 48.3 (1983), pp. 797–803. DOI: 10.2307/2273473.

[25] Hartley Rogers Jr. *Theory of recursive functions and effective computability*. MIT press, 1987.

[26] Bertrand Russell. "Mathematical Logic as Based on the Theory of Types". In: *American Journal of Mathematics* 30.3 (1908), pp. 222–262. DOI: 10.2307/2369948.

[27] Albert N. Shiryaev. "Kolmogorov: Life and Creative Activities". In: *The Annals of Probability* 17.3 (1989), pp. 866–944. DOI: 10.1214/aop/1176991251.

[28] Michael F. Sipser. *Introduction to the Theory of Computation*. 3rd ed. Cengage, 2012.

[29] Gert Smolka. "Modeling and Proving in Computational Type Theory Using the Coq Proof Assistant". Accessed: 18.07.2021. July 2021. URL: www.ps.uni-saarland.de/~smolka/drafts/icl2021.pdf.

[30] Raymond M. Smullyan. "Effectively simple sets". In: *Proceedings of the American Mathematical Society* 15.6 (1964), pp. 893–895. DOI: 10.1090/S0002-9939-1964-0180485-7.

[31] Raymond M. Smullyan. *Theory of Formal Systems. (AM-47)*. Vol. 47. Princeton University Press, 2016. DOI: 10.1515/9781400882007.

[32] Ray J. Solomonoff. "A formal theory of inductive inference. Part I". In: *Information and Control* 7.1 (1964), pp. 1–22. DOI: 10.1016/S0019-9958(64)90223-2.

[33] Ray J. Solomonoff. "A preliminary report on a general theory of inductive inference". In: Citeseer. 1960.

[34] Matthieu Sozeau and Cyprien Mangin. "Equations reloaded: high-level dependently-typed functional programming and proving in Coq". In: *Proceedings of the ACM on Programming Languages* 3.ICFP (2019), pp. 1–29. DOI: 10.1145/3341690.

[35] Andrew Swan and Taichi Uemura. *On Church's Thesis in Cubical Assemblies*. 2019. arXiv: 1905.03014 [math.LO].

[36] The Coq std++ Team. *std++*. Accessed: 22.08.2021. URL: https://gitlab.mpi-sws.org/iris/stdpp.

[37] The Coq Development Team. *The Coq Proof Assistant*. Accessed: 10.08.2021. URL: https://coq.inria.fr/.

[38] Alan M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* s2-42.1 (1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[39] Paul M.B. Vitányi. "How Incomputable Is Kolmogorov Complexity?" In: *Entropy* 22.4 (2020). DOI: 10.3390/e22040408.

[40] Paul M.B. Vitányi. *Obituary: Ray Solomonoff, Founding Father of Algorithmic Information Theory*. Accessed: 16.07.2021. URL: http://raysolomonoff.com/tributes/vitanyi.html.

[41] Benjamin Werner. "Sets in types, types in sets". In: *International Symposium on Theoretical Aspects of Computer Software*. Springer. 1997, pp. 530–546. DOI: 10.1007/bfb0014566.

[42] Alexander K. Zvonkin and Leonid A. Levin. "The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms". In: *Russian Mathematical Surveys* 25.6 (1970), pp. 83–124. DOI: 10.1070/rm1970v025n06abeh001269.