Constructive and Synthetic Reducibility Degrees: Post's Problem for Many-one and Truth-table Reducibility in Coq

Yannick Forster, Felix Jahn

Constructive and Synthetic Reducibility Degrees

Post's Problem for Many-one and Truth-table Reducibility in Coq

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

Felix Jahn
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

We present a constructive analysis and machine-checked theory of one-one, many-one, and truth-table reductions based on synthetic computability theory in the Calculus of Inductive Constructions, the type theory underlying the proof assistant Coq. We give elegant, synthetic, and machine-checked proofs of Post's landmark results that a simple predicate exists, is enumerable, undecidable, but many-one incomplete (Post's problem for many-one reducibility), and a hypersimple predicate exists, is enumerable, undecidable, but truth-table incomplete (Post's problem for truth-table reducibility).

In synthetic computability, one assumes axioms allowing to carry out computability theory with all definitions and proofs purely in terms of functions of the type theory with no mention of a model of computation. Proofs can focus on the essence of the argument, without having to sacrifice formality. Synthetic computability also clears the lense for constructivisation.

Our constructively careful definition of simple and hypersimple predicates allows us to not assume classical axioms, not even Markov's principle, still yielding the expected strong results.

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

Keywords and phrases type theory, computability theory, constructive mathematics, Coq

Digital Object Identifier 10.4230/LIPIcs.CSL.2023.16

Supplementary Material `github.com/uds-psl/coq-synthetic-computability/tree/reddegrees`.

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

Acknowledgements We thank Dominik Kirst, Gert Smolka, Lennard Gäher, and Andrej Dudenhefner for discussions and feedback on the drafts of this paper, as well as the reviewers for their comments.

1 Introduction

The founding moment of "computability theory" deserving of the suffix "theory" was maybe Emil L. Post's 1944 paper [25]. Post introduced the concepts of one-one, many-one, and truth-table reducibility and identified and answered important questions on the structure of the reducibility degrees induced by these relations. Centrally, Post was interested in the question whether there are enumerable but undecidable degrees such that an undecidability proof cannot be done by reduction from the halting problem. For many-one and truth-table reducibility, Post was able to construct such degrees by introducing simple and hypersimple sets, which are still taught in modern textbook presentations of the field. The question whether an enumerable, undecidable problem which is not Turing-reducible from the halting problem exists became known as *Post's problem*, and we reuse the terminology for *Post's problem for many-one reducibility* ($\preceq_{\mathsf{m}}$) and *Post's problem for truth-table reducibility* ($\preceq_{\mathsf{tt}}$).

Early in his paper, Post remarks *'That mathematicians generally are oblivious to the importance of this work of Gödel, Church, Turing, Kleene, Rosser and others as it affects the subject of their own interest is in part due to the forbidding, diverse and alien formalisms in*

which this work is embodied.' The evolution of "computability" to "computability theory" started by Post was enabled by a presentation of the lead questions in a more appealing, intuitive way, abstracting away from the "forbidding, alien formalisms" constituted by general recursive functions, Turing machines, the λ-calculus, or Post's own tag systems. Like in modern textbook presentations and research papers on computability (and complexity) theory, concrete formalisations are avoided by universal application of Church's thesis already to prove basic results such as the *s-m-n* theorem, fixpoint theorems, etc.

This circumstance is remarkable because it turns computability theory into a field not directly applicable to machine-checked proofs. While the use of proof assistants for cutting-edge research in e.g. programming language theory is already omnipresent and is even entering mainstream mathematics with the Lean proof assistant's `mathlib` [20], computability is unlikely to just catch up: Manually filling in machine-checked proofs for every use of (informal) Church's thesis is just infeasible.

We thus work in a different setting which renders computability theory subject to machine-checked proofs: synthetic computability using as logical foundation the Calculus of Inductive Constructions (CIC, the type theory underlying the Coq proof assistant) as set up by the first author of the present paper [9]. The setting has several historic predecessors: In the most basic form of synthetic computability, morally present in the Russian school of constructivism due to Markov [19], one assumes an axiom stating that for every function $\mathbb{N} \to \mathbb{N}$ there exists a μ-recursive function computing it, known as CT (Church's thesis, [17]). In synthetic computability due to Richman [27], a function $\phi\colon \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ is *abstractly* assumed to be universal for all partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ (the axiom EPF, stating that $\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\exists c.\ \phi_c \equiv f$). Richman assumes the axiom of countable choice to be able to prove the *s-m-n* theorem, and in his book with Bridges [3] they remark that assuming just an *s-m-n* operator abstractly would also suffice, without elaborating further. Abstracting even more, Bauer [1] just works with enumerable sets and assumes that the set of enumerable sets is enumerable, together with Markov's principle and the axiom of countable choice.

As a consequence of assuming axioms of synthetic computability together with the axioms of countable choice, the law of excluded middle becomes disprovable since it entails the existence of a non-computable function. This renders synthetic computability in these settings a constructivist, anti-classical endeavour. Textbook presentations of computability theory are however explicit in that they use classical logic freely. Thus, to translate textbook presentations into an anti-classical synthetic setting, one has to constructivise proofs on the go. However, this is not always completely possible: It is well-known that Post's theorem that a set is decidable if and only if both it and its complement are enumerable is equivalent to MP (*Markov's Principle*) [32]. For other results, e.g. the ones covered in this paper, to the best of our knowledge it was unknown which logical assumptions such as LEM (*Excluded Middle*) or MP are necessary.

The first author [9, 7] has observed that in CIC, assuming a universal function ϕ abstractly together with an abstract *s-m-n* operator suffices to carry out synthetic computability theory. Equivalently, several axioms can be used: The axiom EPF, concerned with partial functions and more akin to the one used by Richman, and the axiom EA, concerned with enumerable predicates and more akin to Bauer's axiom. Since in CIC – contrary to other constructive foundations – the axiom of countable choice is not provable, it seems that one can consistently assume classical logical axioms even as strong as the law of excluded middle if needed.

The setting is fully in the spirit of Post's abstraction away from the "forbidding, alien formalisms" of machine models and forms a suitable setting for the analysis of the constructive status of theorems in computability theory. The present paper

1. presents a fully synthetic development of most results from Post's 1944 paper [25], with

no reference to any model of computation,
2. uses intuitionistic logic enriched with the assumption of a (parametrically) universal enumerator but *no* additional axioms, and
3. is fully mechanised in the Coq proof assistant as a further step in the overarching goal of mechanising the pillar-stones of mathematics and computer science. The results of the paper are hyperlinked with Coq code as indicated by the 🐓-symbols.

We believe that machine-checked proofs are also on its own a contribution to "classical" computability theory, i.e. without considering synthetic and constructive aspects: It enables the creation of a machine-checked library of the central theorems of computability theory, ensuring that no odd corner cases are left out in proofs or are hidden under uses of the Church Turing thesis. For the present paper, converting the well-known proofs from a set-theoretic foundation present in textbooks to type theory was not problematic, and modelling choices did not play a crucial role.

For Post's problem for $\preceq_{\mathsf{m}}$, i.e. that an enumerable, undecidable, many-one incomplete, simple predicate exists, the definition of simple in the synthetic setting is most interesting. The complement of a simple predicate has to be infinite, but is not allowed to have an infinite, enumerable subpredicate. In classical mathematics, p is infinite if and only if it is Cantor-infinite, i.e. if there is an injective function $\mathbb{N} \hookrightarrow p$. However, Cantor-infinite predicates (defined via functions) have a (synthetically) enumerable, infinite subpredicate – enumerated by said function. In constructive mathematics, a predicate p is called infinite if for any sequence $[x_1, \dots, x_n]$ there exists a y different from all x_i but s.t. py. However, any fully constructive proof of infinity in this sense can be turned into a proof of Cantor infinity, meaning there can be no such proof for the complement of a simple predicate. It is thus crucial that infinity is defined as non-finiteness free and thereby void of any computational content. The complement of a simple predicate is then non-finite, but not (synthetically) Cantor-infinite. Only with this definition of infinity Post's problem for $\preceq_{\mathsf{m}}$ can be settled constructively.

For Post's problem for $\preceq_{\mathsf{tt}}$, i.e. that an enumerable, undecidable, truth-table incomplete, hypersimple predicate exists, several interesting aspects appear: First, the definition of hypersimple predicates has to be chosen carefully to ensure that hypersimple predicates are tt-incomplete. The construction of a hypersimple predicate H is then easier. But since conventional proofs of its undecidability factor via simpleness (and since the textbook proofs showing that hypersimple predicates are simple seem to be inherently classical and we only manage to weaken the assumption to MP), we give a direct, fully constructive undecidability proof for H, which however does not generalise to arbitrary hypersimple predicates.

We try to give intuitive conceptual outlines in the paper, but focus on the interesting synthetic and constructive aspects more than the proof ideas. We largely follow the excellent book by Rogers [28], supplemented by the books by Cutland [4], Soare [29], and Odifreddi [22]. The Bachelor's thesis of the second author [15], containing preliminary results covered in this paper, discusses some aspects in more detail. The PhD thesis of the first author also discusses the material presented in this paper [8]. In general, it might be helpful for non-experts in computability theory to consult one of the books in cases where the presented intuition is not sufficient.

2 The Calculus of Inductive Constructions

We work in the calculus of inductive constructions (CIC) as implemented by the Coq proof assistant [30]. The calculus is a constructive type theory with a (cumulative hierarchy of) type universe(s) $\mathbb{T}$ and an impredicative universe of propositions $\mathbb{P} \subseteq \mathbb{T}$. The universe of propositions $\mathbb{P}$ is impredicative, so e.g. $(\forall X : \mathbb{P}.\ X) : \mathbb{P}$, and a sub-universe of $\mathbb{T}$, i.e. whenever $P : \mathbb{P}$ we also have $P : \mathbb{T}$. Both aspects however only play a minor role in this paper, important is that the universe $\mathbb{P}$ is separated from the universe $\mathbb{T}$. That means that in general, computations cannot inspect proofs to return a computational value.

Most instructively, the difference can be explained by looking at dependent pairs $\Sigma x.\ Ax$ – which we verbalise as "one can construct x s.t. Ax" – and existential quantification $\exists x.\ Ax$ – "there exists x s.t. Ax". Both are implemented as an inductive type of (dependent) pairs (x, y), with the only difference that $(\exists x.\ Ax) : \mathbb{P}$ but $(\Sigma x.\ Ax) : \mathbb{T}$. Dependent pairs can be eliminated in arbitrary contexts, i.e. there is an elimination function of the following type[1]

$\forall p : (\Sigma x.\ Ax) \to \mathbb{T}.\ (\forall xy.\ p(x, y)) \to \forall (s : \Sigma x.\ Ax).\ ps.$

We call principles eliminating a proposition into types *large elimination principles*, following the terminology "large elimination" for Coq's case analysis construct `match` [23]. Crucially, CIC proves a large elimination principle for the falsity proposition $\bot$, i.e. explosion applies to arbitrary types: $\forall A : \mathbb{T}.\ \bot \to A$. For existential quantification, a large elimination principle is not provable, the only principle one can obtain is the following, where the return type of p is $\mathbb{P}$:

$\forall p : (\exists x.\ Ax) \to \mathbb{P}.\ (\forall xy.\ p(x, y)) \to \forall (s : \exists x.\ Ax).\ ps.$

There are two exceptions to this observation: First, whenever $\exists x.\ Ax$ (also in the relational form $\forall x_1.\exists x_2.\ Rx_1x_2$), can be proved *without assumptions*, one could instead prove the stronger result that $\Sigma x.\ Ax$ (for the relational form $\forall x_1.\Sigma x_2.\ Rx_1x_2$ or equivalently $\Sigma f.\forall x_1.Rx_1(fx_1)$). Secondly, for the case where p has a boolean decision function, an elimination principle is provable, reminiscent of the μ operator of general recursive functions.

Fact 1. *There is a guarded minimisation function*

$\mu_{\mathbb{N}} : \forall f : \mathbb{N} \to \mathbb{B}.\ (\exists n.\ fn = \mathsf{true}) \to \Sigma n.\ fn = \mathsf{true} \wedge \forall m.\ fm = \mathsf{true} \to m \geq n.$

The inductive types of interest in this paper are all standard, but listed for reference:

$n : \mathbb{N} ::= 0 \mid \mathsf{S}\, n$	(natural numbers)	$A + B := \mathsf{inl}\, a \mid \mathsf{inr}\, b$ *with* $a{:}\, A,\ b{:}\, B$	(sums)
$b : \mathbb{B} ::= \mathsf{false} \mid \mathsf{true}$	(booleans)	$A \times B := (a, b)$ *with* $a{:}\, A,\ b{:}\, B$	(pairs)
$l{:}\, \mathbb{L}A ::= [] \mid a :: l$ *with* $a{:}\, A$	(lists)	$\Sigma x{:}\, X.Ax := (x_0, a)$ *with* $A{:}\, X \to \mathbb{T},\ x_0{:}\, X, a{:}\, Ax_0$	(dependent pairs)
$o{:}\, \mathbb{O}A ::= \mathsf{None} \mid \mathsf{Some}\, a \ \mathit{w/}\ a{:}\, A$	(options)		

We mention the following notations here: We use pairing function on natural numbers written as $\langle \_, \_ \rangle : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ and for all $f : \mathbb{N} \to \mathbb{N} \to X$ an inverse construction $\lambda\langle n, m\rangle.\ fnm$ of type $\mathbb{N} \to X$ s.t. $(\lambda\langle n, m\rangle.fnm)\langle n, m\rangle = fnm$. We use the list operations $\mathsf{map} : (X \to Y) \to \mathbb{L}X \to \mathbb{L}Y$, $\_ +\!\!+ \_ : \mathbb{L}X \to \mathbb{L}X \to \mathbb{L}X$, $\mathsf{filter} : (X \to \mathbb{B}) \to \mathbb{L}X \to \mathbb{L}X$, and $\_[\_] : \mathbb{L}X \to \mathbb{N} \to \mathbb{O}X$. We write $|l|$ for the length of $l : \mathbb{L}X$. For $p : X \to \mathbb{P}$, we denote the complement as $\overline{p}$ and the Cartesian product with $q : Y \to \mathbb{P}$ as $p \times q$. p is called *inhabited* if $\exists x.px$. When defining a predicate we identify a type X with $\lambda(x : X).\top$. We write $\mathsf{Forall}_2\, p\, l_1\, l_2$ for $l_1 : \mathbb{L}X$, $l_2 : \mathbb{L}Y$, and $p : X \to Y \to \mathbb{P}$ if p holds pointwise on the lists l_1 and l_2, i.e. for the inductive predicate with constructors of type $\mathsf{Forall}_2\ []\ []$ and $\forall x\ y\ l_1\ l_2.\ pxy \to \mathsf{Forall}_2\ l_1\ l_2 \to \mathsf{Forall}_2\ (x :: l_1)\ (y :: l_2)$.

[1] Note that, as customary in type theory, quantifications range over the rest of the expression, i.e. the above formula is equivalent to $\forall (p : (\Sigma x.\ Ax) \to \mathbb{T}).\ ((\forall xy.\ (p(x, y))) \to \forall (s : \Sigma x.\ Ax).\ (ps))$.

3 Constructive proofs

A proposition $P : \mathbb{P}$ is *stable* if it is unchanged under double negation, i.e. $\neg\neg P \to P$. Furthermore, we say that P is *logically decidable*, if $P \vee \neg P$ holds.

Fact 2. *Logically decidable propositions are stable.*

Stability properties are useful if they apply to a goal, whereas logically decidability is also useful to establish assumptions. The *law of excluded middle* LEM states that every proposition is logically decidable, and thus enables strengthening assumptions. *Markov's Principle* (MP) accepted for instance in Russian constructivism states that satisfiability of a boolean test on natural numbers is stable and is a consequence of LEM:

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

LEM is routinely used in textbooks, often in the form of double negation elimination:

Fact 3. $\mathsf{LEM} \leftrightarrow \forall P : \mathbb{P}.\ \neg\neg P \to P$

It is folklore that LEM is independent in CIC: It can be consistently assumed, but not proved. However, when the conclusion is stable, LEM is not needed for case analysis:

Fact 4. *Let Q be stable. We have that*

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

Note that negated propositions are always stable, and in fact the following holds:

Fact 5. *P is stable if and only if P is equivalent to $\neg Q$ for some Q.*

MP is also independent in CIC [24, 18]. For classical theorems on enumerable predicates, often full LEM is not needed and MP suffices, see Fact 9 in the next section.

4 Synthetic Computability Theory

The key ingredients for synthetic computability theory are synthetic definitions of central notions and suitable synthetic axioms. We define synthetic decidability, enumerability, semi-decidability, and many-one reducibility [11, 6], mirroring the textbook definitions without the requirement that the witnessing function is computable in a model of computation. We recall the synthetic setting due to Forster [9] in which we work. A predicate $p : X \to \mathbb{P}$ is

- *decidable* if there exists a decider: $\mathcal{D}p := \exists f : X \to \mathbb{B}. \forall x.\ px \leftrightarrow fx = \mathsf{true}$
- *semi-decidable* if there exists a semi-decider: $\mathcal{S}p := \exists f. \forall x.\ px \leftrightarrow \exists n. fxn = \mathsf{true}$
- *enumerable* if there exists an enumerator: $\mathcal{E}p := \exists f : \mathbb{N} \to \mathbb{O}X. \forall x.\ px \leftrightarrow \exists n.\ fn = \mathsf{Some}\, x$
- *strongly enumerable* if the following holds: $\mathcal{E}_+p := \exists f : \mathbb{N} \to X. \forall x.\ px \leftrightarrow \exists n.\ fn = x$

We treat n-ary predicates as unary via (implicit) uncurrying. A type X is *discrete* if $\mathcal{D}(\lambda xy : X. x = y)$ and *enumerable* if $\mathcal{E}(\lambda x : X. \top)$. We sum up the connections of the notions:

Fact 6.
1. *Decidable predicates are semi-decidable.*
2. *Decidability is closed under (pointwise) $\wedge$, $\vee$, and $\neg$.*
3. *Finite predicates are decidable.*
4. *Semi-decidable predicates on enumerable types are enumerable.*
5. *Semi-decidability is closed under (pointwise) $\wedge$ and $\vee$.*
6. *The projections $\lambda y. \exists x.\ pxy$ and $\lambda x. \exists y.\ pxy$ of an enumerable predicate $p : X \to Y \to \mathbb{P}$ are enumerable.*

7. *Enumerable predicates on discrete types are semi-decidable.*
8. *Enumerability is closed under (pointwise)* $\wedge$ *and* $\vee$*.*
9. *Strongly enumerable predicates are enumerable.*
10. *Enumerable inhabited predicates are strongly enumerable.*

Fact 7. 1. $\mathbb{N}$ *and* $\mathbb{B}$ *are discrete and enumerable.*
2. *Both discrete and enumerable types are closed under pairs, sums, options, and lists.*

A *retraction* is an injective function with explicit inverse. X is a retract of Y if there are $I : X \to Y$ and $R : Y \to \mathbb{O}X$ s.t. $R(Ix) = \mathsf{Some}\, x$ and $\forall xy.\ Ry = \mathsf{Some}\, x \to y = Ix$.

Fact 8. *A type is discrete and enumerable if and only if it is a retract of* $\mathbb{N}$*.*

This fact enables most of our results to only mention predicates $\mathbb{N} \to \mathbb{P}$, and then transport for free to predicates on infinite, enumerable, discrete types (such as $\mathbb{LB} \times \mathbb{ON}$).

We can characterise MP in terms of enumerable and semi-decidable predicates:

Fact 9 ([11] 2.17, [6] 41)**.** *The following are equivalent:*
1. MP 2. $\forall p : \mathbb{N} \to \mathbb{P}.\ \mathcal{S}p \to \forall x.\ \neg\neg px \to px$ 3. $\forall p : X \to \mathbb{P}.\ \mathcal{E}p \to \neg\neg(\exists n.\ pn) \to \exists n.\ pn$

The following strengthening of $\mu_{\mathbb{N}}$ (Fact 1) will be crucial later:

Fact 10. *Let* $p : \mathbb{N} \to X \to \mathbb{P}$ *be enumerable and* X *discrete. Then there is* $\mu_{\mathcal{E}} : \forall n.(\exists x.pnx) \to \Sigma x.pnx$*.*

Note that the result of $\mu_{\mathcal{E}}$ is independent of the proof $H : \exists x.\ pnx$, even without assumptions.

In this paper we will define many-one, one-one, and truth-table reducibility ($\preceq_{\mathsf{m}}, \preceq_1, \preceq_{\mathsf{tt}}$). In general, for a reducibility notion $\preceq_{\mathsf{r}}$ we define predicates $p : X \to \mathbb{P}$ and $q : Y \to \mathbb{P}$ to be r-*equivalent* if $p \equiv_{\mathsf{r}} q := p \preceq_{\mathsf{r}} q \wedge q \preceq_{\mathsf{r}} p$. Informally, we might also refer to the class of predicates q s.t. $p \equiv_{\mathsf{r}} q$ as the r-degree of p, but will not make degrees formal to avoid extensionality assumptions. A predicate $p : \mathbb{N} \to \mathbb{P}$ is called r-*complete* if $\mathcal{E}p \wedge \forall q : \mathbb{N} \to \mathbb{P}.\ \mathcal{E}q \to q \preceq_r p$. We freely use the notion as well for predicates $p : X \to \mathbb{P}$ if X is in bijection to $\mathbb{N}$.

All notions of reducibility have in common that they form pre-orders (i.e. are reflexive and transitive) and that they transport decidability backwards, i.e. $p \preceq_r q \to \mathcal{D}q \to \mathcal{D}p$.

We now start with the first notion of synthetic reducibility: A function $f : X \to Y$ is a *many-one reduction* from $p : X \to \mathbb{P}$ to $q : Y \to \mathbb{P}$ if it expresses p in terms of q:

$$p \preceq_{\mathsf{m}} q := \exists f : X \to Y. \forall x.\ px \leftrightarrow q(fx)$$

Fact 11. 1. *Many-one reducibility forms a pre-order.*
2. *If* $p \preceq_{\mathsf{m}} q$ *and* q *is decidable then* p *is decidable, resp. for semi-decidability and stability.*
3. $p \preceq_{\mathsf{m}} q \to \overline{p} \preceq_{\mathsf{m}} \overline{q}$*.*

Regarding the order structure of $\preceq_{\mathsf{m}}$, one can prove that decidable predicates constitute minima for the class of non-trivial predicates, and that $\preceq_{\mathsf{m}}$ forms an upper semi-lattice:

Fact 12. *Let* p *be decidable. If* $\exists x_1 x_2.\ qx_1 \wedge \neg qx_2$*, then* $p \preceq_{\mathsf{m}} q$*.*

Fact 13. *Let* $p : X \to \mathbb{P}$ *and* $q : Y \to \mathbb{P}$*. Then there is a lowest upper bound* $p + q : X + Y \to \mathbb{P}$ *w.r.t.* $\preceq_{\mathsf{m}}$*: If* $(p + q)(\mathsf{inl}\, x) := px$ *and* $(p + q)(\mathsf{inr}\, y) := qy$*, then* $p + q$ *is the join of* p *and* q *w.r.t* $\preceq_{\mathsf{m}}$*, i.e.* $p \preceq_{\mathsf{m}} p + q$*,* $q \preceq_{\mathsf{m}} p + q$*, and for all* r *if* $p \preceq_{\mathsf{m}} r$ *and* $q \preceq_{\mathsf{m}} r$ *then* $p + q \preceq_{\mathsf{m}} r$*.*

In traditional computability theory, a universal machine for the chosen model of computation is central to almost all interesting results. In a synthetic approach to computability where there is no explicit model of computation and the notion of function and computable

function are identified, a universal function cannot be defined. Instead, its existence has to be axiomatically assumed.

To develop synthetic computability theory agnostic towards classical axioms like LEM, we assume the enumerability axiom EA [9, 7]. The axiom EA postulates a *parametrically universal enumerator* $\varphi\colon \mathbb{N}\to(\mathbb{N}\to\mathbb{ON})$, i.e. that for all $p\colon \mathbb{N}\to\mathbb{N}\to\mathbb{P}$

$$(\exists f\colon \mathbb{N}\to(\mathbb{N}\to\mathbb{ON}).\forall i.\ f_i \text{ enumerates } p_i) \to \exists\gamma\colon \mathbb{N}\to\mathbb{N}.\forall i.\ \varphi_{\gamma i} \text{ enumerates } p_i.$$

To ease language, we often refer to just the universal property as EA in this paper.

Fact 14. $\forall p\colon \mathbb{N}\to\mathbb{P}.\ \mathcal{E}p \to \exists c.\ \varphi_c \text{ enumerates } p$

Fact 15. *Let X be enumerable and discrete and $p\colon X\times\mathbb{N}\to\mathbb{P}$ be enumerable. Then*

$$\exists\gamma\colon X\to\mathbb{N}.\forall x.\ \varphi_{\gamma x} \text{ enumerates } \lambda y.p(x,y).$$

Fact 16. *Let I be enumerable and discrete and $p\colon I\to\mathbb{N}\to\mathbb{P}$. We have*

$$(\exists f\colon I\to(\mathbb{N}\to\mathbb{ON}).\forall i.\ f_i \text{ enumerates } p_i) \to \exists\gamma\colon I\to\mathbb{N}.\forall i.\ \varphi_{\gamma i} \text{ enumerates } p_i$$

As is common in developments of computability, we start by defining an enumerable, undecidable, m-complete predicate and its diagonal:

$$\mathcal{W}_c x := \exists n.\ \varphi_c n = \mathsf{Some}\, x \qquad \mathcal{K}c := \mathcal{W}_c c$$

We call $\mathcal{W}$ the *universal table* of enumerable predicates, justified by the following property:

Fact 17. $\forall p\colon \mathbb{N}\to\mathbb{P}.\ \mathcal{E}p \leftrightarrow \exists c.\ \forall x.\ \mathcal{W}_c x \leftrightarrow px$

The universal table $\mathcal{W}$ is enumerable, undecidable and m-complete, i.e. takes the role of the halting problem from textbook computability. $\mathcal{K}$ plays a similar role as the self-halting problem, instead of the codes halting on themselves, $\mathcal{K}c$ holds if c is in the range of φ_c.

We start by showing that the complement of $\mathcal{K}$ is not enumerable. Thus $\mathcal{K}$ is undecidable, and undecidability can be transported to $\mathcal{W}$ via a many-one reduction.

Fact 18. $\neg\mathcal{E}\overline{\mathcal{K}}$ *and thus* $\neg\mathcal{D}\overline{\mathcal{K}}$ *and* $\neg\mathcal{D}\mathcal{K}$

Fact 19. $\mathcal{K}\preceq_{\mathsf{m}}\mathcal{W}$*, and thus* $\neg\mathcal{E}\overline{\mathcal{W}}$*,* $\neg\mathcal{D}\overline{\mathcal{W}}$*,* $\neg\mathcal{D}\mathcal{W}$*.*

To show the enumerability of both $\mathcal{K}$ and $\mathcal{W}$, we show the enumerability of $\mathcal{W}$ and again transport via the above many-one reduction, this time positively.

Fact 20. $\mathcal{E}\mathcal{W}$ *and thus* $\mathcal{E}\mathcal{K}$*.*

We now turn towards m-completeness of $\mathcal{W}$ and $\mathcal{K}$, i.e. that all $p\colon X\to\mathbb{P}$ for enumerable, discrete X many-one reduce to $\mathcal{W}$ and $\mathcal{K}$. To establish this for $\mathcal{K}$ for the first time requires the full strength of EA, whereas before the non-parametric Fact 14 would have sufficed.

Lemma 21. $\mathcal{W}$ *is m-complete.*

Proof. Let p be enumerable by φ_c via Fact 14. Then $\lambda x.(c,x)$ reduces p to $\mathcal{W}$. ◀

Lemma 22. $\mathcal{W}\preceq_{\mathsf{m}}\mathcal{K}$

Proof. We obtain the reduction function γ from Fact 16 with $p(x,y)z := \mathcal{W}_x y$. Since $\forall xyz.\ \mathcal{W}_{\gamma(x,y)}z \leftrightarrow \mathcal{W}_x y$ we have $\mathcal{W}_x y \leftrightarrow \mathcal{W}_{\gamma(x,y)}(\gamma(x,y)) \leftrightarrow \mathcal{K}(\gamma(x,y))$. ◀

Fact 23. $\mathcal{W}\equiv_{\mathsf{m}}\mathcal{K}$ *and $\mathcal{K}$ is m-complete.*

We can recover the characterisation of MP as stability of halting:

Fact 24. MP *is equivalent to the stability of* **1.** $\mathcal{W}$*,* **2.** $\mathcal{K}$*,* **3.** *any enumerable, m-complete predicate on* $\mathbb{N}$*, and* **4.** *every enumerable, m-complete predicate on* $\mathbb{N}$*.*

5 Finite predicates

We discuss two notions of finiteness: Finite (often Bishop- or $\mathcal{B}$-finite) and subfinite (also $\widetilde{\mathcal{B}}$-finite) predicates. We say that a list $l : \mathbb{L}X$ *lists* a predicate $p\colon X\to\mathbb{P}$ if $px \leftrightarrow x \in l$. Note that l is not allowed to contain elements not fulfilled by the predicate. A list $l : \mathbb{L}X$ *exhausts* a predicate $p\colon X\to\mathbb{P}$ if $px \to x \in l$: We write $\mathcal{L}p$ for finite and $\mathcal{X}p$ for subfinite predicates p.

$$\mathcal{L}p := \exists l : \mathbb{L}X.\forall x : X.\ px \leftrightarrow x \in l \qquad\qquad \mathcal{X}p := \exists l : \mathbb{L}X.\ \forall x : X.\ px \to x \in l$$

Clearly, finiteness implies subfiniteness, but the converse is equivalent to LEM:

Fact 25. *Finite predicates are subfinite.*

Lemma 26. *Every subfinite predicate is not not finite.*

Proof. We first prove the lemma that $\forall l_0 : \mathbb{L}X.\forall p\colon X\to\mathbb{P}.\neg\neg\exists l'.\forall x.\ x \in l' \leftrightarrow x \in l_0 \wedge px$ which follows by induction on l_0. Now, let l exhaust p and let p be not finite. We have to prove falsity. Using the lemma, we obtain l' s.t. $\forall x.\ x \in l' \leftrightarrow x \in l \wedge px$ and still have to prove falsity. Now l' lists p. Contradiction. ◀

Lemma 27. *If every subfinite predicate is finite,* LEM *holds.*

Proof. For $P : \mathbb{P}$ we define $p(x : \mathbb{B}) := P$. p is subfinite because it is exhausted by $[\mathsf{true}, \mathsf{false}]$. If p is listed by l, case analysis on l allows proving $P \vee \neg P$. ◀

Corollary 28. LEM *holds if and only if every subfinite predicate is finite.*

6 Pigeonhole principles

Pigeonhole principles are omnipresent in discrete mathematics. We will prove three variants of the pigeonhole principle based on duplicate-free lists: Our formulation of the principle is that for any duplicate-free list l_1 longer than a list l_2 one can obtain an element x which is in l_1 but not in l_2 – for different formalisations of "obtain" in CIC[2]: **1.** x is computable $(\forall l_1 l_2.\ \cdots \to \Sigma x.\ \ldots)$, **2.** x constructively exists $(\forall l_1 l_2.\ \cdots \to \exists x.\ \ldots)$, **3.** x classically exists $(\forall l_1 l_2.\ \cdots \to \neg\neg\exists x.\ \ldots)$. These are exactly the three possible formalisations of "obtain" in CIC: A function returning a dependent pair (Σ), a proof of an existential proposition ($\exists$), or of a double-negated existential proposition ($\neg\neg\exists$, equivalently $\neg\neg\Sigma$). Formulating existence as Σ inherently means that the result has to be *computable*, a property unchanged by the assumption of logical axioms like LEM. $\exists$ has to be proved using a (computable) function, but the function cannot be used computationally after the proof. This means that any constructive proof of $\forall x.\exists y.\ldots$ not using assumptions could always be turned into a proof of $\forall x.\Sigma y$ and vice versa, but under assumptions and as assumptions the two behave differently, see Section 2 for more details.

We now turn towards proving the principles, which will vary in the requirements on the underlying type X. For the formalisation, we define $\#l : \mathbb{P}$ for a list $l : \mathbb{L}X$ inductively in the expected way to state that l does not contain any duplicates.

Given an equality decider for the base type X it is straightforward to prove the $\forall\Sigma$-version of the pigeonhole principle:

[2] Admittedly, this formulation is more a "pigeon-less hole principle", but we use the well-known terminology.

Lemma 29. *Let d decide equality on X and $l_1, l_2 : \mathbb{L}X$. If $\#l_1$ and $|l_1| > |l_2|$, then $\Sigma x.\, x \in l_1 \wedge x \notin l_2$.*

Proof. By induction on $\#l_1$, with l_2 generalised. The first case is contradictory, since $|[]| = 0 > |l_2|$ is impossible. Let $x \notin l_1$ and $\#l_1$. Using d allows a case analysis whether $x \in l_2$ or $x \notin l_2$: If $x \notin l_2$, the claim is immediate. If $x \in l_2$, the claim follows from the induction hypothesis for $l_2 := \mathsf{filter}(\lambda y.\mathbf{if}\ dxy\ \mathbf{then}\ \mathsf{false}\ \mathbf{else}\ \mathsf{true})\ l_2$ (i.e. for l_2 with x removed). ◀

Note how the $\forall\Sigma$ version depends on computationally removing an element from a list, and thus on an equality decider d. If no such d is available, a removal function is not definable. However, for the $\forall\exists$ and $\forall\neg\neg\exists$ forms of the pigeonhole principle, a removal *function* is not needed. Instead, for the $\forall\exists$ version it suffices to prove that for any list l_0 and any element x_0, there exists ($\exists$) a list with the same elements of l_0, just x_0 removed. This becomes possible provided $x_1 \neq x_2$ is logically decidable for all x_1, x_2.For the $\forall\neg\neg\exists$ version of the pigeonhole principle, consequently a removal principle of the form $\neg\neg\exists l \ldots$ suffices, which can be proved fully constructively without assumptions.

To prove the two removal principles,we define a generalised filter predicate $l_0 \supseteq_p l$ w.r.t. a predicate $p\colon X \to \mathbb{P}$ stating that l is exactly the sublist of l_0 with all elements which fulfil p:

$$\frac{}{[] \supseteq_p []} \qquad \frac{px \qquad l_0 \supseteq_p l}{(x :: l_0) \supseteq_p (x :: l)} \qquad \frac{\neg px \qquad l_0 \supseteq_p l}{(x :: l_0) \supseteq_p l}$$

We can prove two existence principles, one assuming that p is logically decidable, and one proving a double negation:

Fact 30. *Let $l_0 : \mathbb{L}X$. Then (1) $\neg\neg\exists l.\, l_0 \supseteq_p l$ and (2) $(\forall x.px \vee \neg px) \to \exists l.\, l_0 \supseteq_p l$.*

The $\forall\exists$ and $\forall\neg\neg\exists$ forms of the pigeonhole principle follow:

Lemma 31. *If $\#l_1$ and $|l_1| > |l_2|$, then $\neg\neg\exists x.x \in l_1 \wedge x \notin l_2$.*

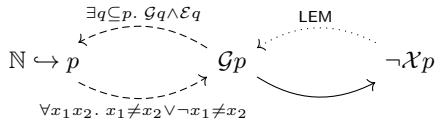
Proof. By induction on $\#l_1$, with l_2 generalised. The case $l_1 = []$ is contradictory since then $|[]| = 0 > |l_2|$. Thus let $x \notin l_1$ and $\#l_1$. Since the claim is negative, we can do a case analysis on $x \in l_2$ by Fact 4 **3**. If $x \notin l_2$, the claim is immediate. If $x \in l_2$, we obtain l s.t. $l_2 \supseteq_{(\lambda y.x \neq y)} l$ from Fact 30 (1). The claim follows by induction for l. ◀

Lemma 32. *If $\forall x_1 x_2\colon X.x_1 \neq x_2 \vee \neg x_1 \neq x_2$, $\#l_1$ and $|l_1| > |l_2|$, then $\exists x.\, x \in l_1 \wedge x \notin l_2$.*

Proof. Similar to the last proof, using Fact 30 (2) and the assumption. ◀

7 Infinite Predicates

The central notions of simple and hypersimple predicates depend on a formalisation of infinity. Similar to finite predicates, there are several classically equivalent but constructively different definitions of infinite predicates. We discuss three possible definitions of infinity, where a predicate $p\colon X \to \mathbb{P}$ is non-finite if it is not finite ($\neg\mathcal{L}p$), or equivalently not subfinite ($\neg\mathcal{X}p$), both stable propositions, generative ($\mathcal{G}p$) if for every list there exists a further element fulfilling p not in the list (a $\forall\exists$ proposition), and Cantor-infinite ($\mathbb{N} \hookrightarrow p$) if there exists an injection returning only elements in p (equivalent to a $\forall\Sigma$ proposition). Those three notions of infinite predicates mirror exactly the three different formulations of pigeonhole principles in the previous chapter. We obtain the following graph, where the dashed (dotted) lines are annotated with sufficient (and necessary) conditions:



Formally, a predicate $p\colon X \to \mathbb{P}$ is called *non-finite* if is not finite. A predicate $p\colon X \to \mathbb{P}$ is *finite* (written $\mathcal{L}p$) if $\exists l\colon \mathbb{L}X.\ \forall x.\ px \leftrightarrow x \in l$.

Due to the negation, non-finiteness is equivalent to non-sub-finiteness. A predicate $p\colon X \to \mathbb{P}$ is *subfinite* (written $\mathcal{X}p$) if $\exists l\colon \mathbb{L}X.\ \forall x.\ px \to x \in l$. We discuss more properties on finite predicates in Section 5.

Fact 33. $\neg\mathcal{X}p \leftrightarrow \neg\mathcal{L}p$.

A predicate $p\colon X \to \mathbb{P}$ is called *generative* if for every list there exists a further element fulfilling p which is not in the list:

$$\mathcal{G}p := \forall l : \mathbb{L}X.\exists x.\ px \wedge x \notin l$$

Fact 34. *Generative predicates are inhabited, and non-finite predicates not not inhabited.*

Generative predicates on $\mathbb{N}$ can be characterised as follows:

Fact 35. $\forall p\colon \mathbb{N} \to \mathbb{P}.\ \mathcal{G}p \leftrightarrow \forall n.\exists m \geq n.\ pm$

The following characterisation of non-finiteness as $\forall\neg\neg\exists$-formula connects the two notions:

Fact 36. *If* $\forall x_1 x_2 : X.\ x_1 = x_2 \vee x_1 \neq x_2$, *then* $\neg\mathcal{X}p \leftrightarrow \forall l : \mathbb{L}X.\neg\neg\exists x.\ px \wedge x \notin l$.

Corollary 37. $\forall p\colon \mathbb{N} \to \mathbb{P}.\ \neg\mathcal{X}p \leftrightarrow \forall n.\neg\neg\exists m \geq n.pm$

Since the proof of the direction from right to left of Fact 36 does not depend on logical decidability of equality we can prove the following:

Fact 38. *Generative predicates are non-finite.*

Fact 39. *If* LEM *holds and p is non-finite, p is generative.*

We introduce a second $\forall\exists$-notion of infinite predicates: $p\colon X \to \mathbb{P}$ is called *unbounded* if there exist duplicate-free lists of arbitrary length containing only elements from p:

$$\mathcal{U}p := \forall n : \mathbb{N}.\exists l.\ |l| = n \wedge \#l \wedge \forall x \in l.\ px$$

Fact 40. $\neg\mathcal{X}p \leftrightarrow \forall n : \mathbb{N}.\neg\neg\exists l.\ |l| = n \wedge \#l \wedge \forall x \in l.\ px$

Lemma 41. *Generative predicates are unbounded.*

Proof. Let p be generative and $n : \mathbb{N}$. We construct l by induction on n. For $n = 0$ we pick $l = []$. For $n = \mathsf{S}n'$ we use the inductive hypothesis to obtain l, and use generativity to obtain x s.t. px and $x \notin l$. Now $x :: l$ is the wanted list. ◀

We then can prove the following using Lemmas 31 and 32.

Fact 42. *Unbounded predicates are non-finite, and are generative for types X with* $\forall x_1 x_2 : X.\ x_1 \neq x_2 \vee \neg x_1 \neq x_2$.

LEM is necessary for non-finite predicates to be unbounded:

Lemma 43. *If all non-finite $p\colon \mathbb{N} \to \mathbb{P}$ are unbounded,* LEM *holds.*

Proof. Let $P : \mathbb{P}$ and $pn := P \leftrightarrow n$ *is even*. If p is inhabited, LEM holds: Given pn we can analysing whether n is even. Since unbounded predicates are inhabited, by assumption it suffices to prove that p is non-finite. So let p be exhaustible, we have to prove falsity. We can thus decide P and it suffices to prove that p is generative to obtain a contradiction.

If P holds, $pn \leftrightarrow n$ *is even* and if $\neg P$, $pn \leftrightarrow n$ *is odd*. Both are generative. ◀

Corollary 44. *Non-finite predicates are unbounded or generative if and only if* LEM *holds.*

Lastly, we introduce Cantor infinity, often used in textbooks. A predicate $p\colon X\to\mathbb{P}$ is *Cantor-infinite* if there exists an injection $f\colon\mathbb{N}\to X$ only returning elements in p:

$$\mathbb{N} \hookrightarrow p := \exists f\colon\mathbb{N}\to X.\ f \text{ is injective} \land \forall n.\ p(fn),$$

whereby we define as also in the remainder of the paper a function $f : A \to B$ to be injective if $\forall a_1, a_2.\ f(a_1) = f(a_2) \to a_1 = a_2$.

Fact 45. *Cantor-infinite predicates are unbounded.*

Lemma 46. *Let X be discrete and $p\colon X\to\mathbb{P}$. Then $\mathbb{N} \hookrightarrow p \leftrightarrow \forall l : \mathbb{L}X.\Sigma x.\ px \land x \notin l$.*

Proof. The forward direction is easy using Lemma 29. Conversely, let $F : \forall l : \mathbb{L}X.\Sigma x.\ px \land x \notin l$ and $fl := \pi_1(Fl)$. Let $g0 := []$ and $g(\mathsf{S}n) := gn +\!\!+ [f(gn)]$. Then pick $\lambda n.f(gn)$. ◀

We have proven that non-finite predicates are generative if and only if LEM holds. However, non-finite, enumerable predicates on $\mathbb{N}$ are generative already given MP:

Lemma 47. *Assume* MP *and let $p\colon\mathbb{N}\to\mathbb{P}$. If p is enumerable and non-finite, p is generative.*

Proof. Let p be enumerable and non-finite, and let l be given. We have to prove $\exists x.\ x \notin l \land px$. Since p is enumerable, so is $\lambda x.\ x \notin l \land px$. Using Fact 9 it suffices to prove $\neg\neg\exists x.\ x \notin l \land px$, which holds by Fact 36. ◀

We adapt [11, Th. 2.28] to prove that generative enumerable predicates are Cantor-infinite:

Fact 48. *Let X be discrete. Generative, enumerable predicates over X have a strong, injective enumerator: $\forall p\colon X\to\mathbb{P}.\ \mathcal{G}p \to \mathcal{E}p \to \exists f.\ f \text{injective} \land \forall x.\ px \leftrightarrow \exists n.\ fn = x$.*

Corollary 49. *Generative, enumerable predicates over discrete types are Cantor-infinite.*

In synthetic computability, Cantor-infinite predicates are problematic because the function $f\colon\mathbb{N}\to X$ can be turned into an enumerator. From $\mathbb{N} \hookrightarrow p$ we cannot conclude that p is enumerable, but that p has a Cantor-infinite, enumerable subpredicate:

Lemma 50. $\mathbb{N} \hookrightarrow p \to \exists q.\ \mathcal{E}q \land (\forall x.\ qx \to px) \land \mathbb{N} \hookrightarrow q$

Proof. Given p and an injection f witnessing $\mathbb{N} \hookrightarrow p$, define $qx := \exists n.\ fn = x$. Clearly, $\forall x.\ qx \to px$ since $\forall n.\ p(fn)$, and $\lambda n.\ \mathsf{Some}\,(fn)$ enumerates q. f still proves $\mathbb{N} \hookrightarrow q$. ◀

Recall that a predicate is simple if it is enumerable, and its complement is infinite but does not have an infinite enumerable subpredicate. Given the last lemma, we have that under the assumption of a universal enumerator φ, defining infinity as...

1. Cantor infinity proves there is no simple predicate.
2. generativity makes the existence of simple predicates logically independent, since any constructive proof of generativity could be turned into a proof of Cantor-infinity by Lemma 46, but this is not provable,
3. non-finiteness allows to construct a simple predicate.

8 One-one reducibility

We now introduce our second notion of reducibility: One-one reducibility is a special case of many-one reducibility. A function $f : X \to Y$ is a *one-one reduction* from p to q if f is an injective many-one reduction from p to q:

$$p \preceq_1 q := \exists f : X \to Y.\ f \text{ is injective} \land \forall x.\ px \leftrightarrow q(fx)$$

Fact 51. *1. One-one reducibility forms a pre-order, 2. implies many-one reducibility, and 3. transports decidability, semi-decidability, and stability backwards.*

It is easy to prove that one-one and many-one reducibility are not equivalent:

Fact 52. *If $px := \top$ and $qx := x = 0$, $p \preceq_{\mathsf{m}} q$ but $q \not\preceq_1 p$.*

Note that the lemma leaves open whether there are two enumerable but *undecidable* predicates p and q s.t. $p \preceq_{\mathsf{m}} q$ but $p \not\preceq_1 q$. We settle this more interesting case in Corollary 68 via simple predicates. In Appendix A, we characterise m-reducibility in terms of 1-reducibility via so-called cylindrification, adapting the proof by Rogers [28, §7.6 Th. VIII].

9 Simple predicates

We now turn to Post's problem for $\preceq_{\mathsf{m}}$, i.e. to finding an enumerable, undecidable, m-incomplete predicate S i.e. $\mathcal{W} \not\preceq_{\mathsf{m}} S$. Post's observation was that the complements of m-complete predicates are productive. It thus suffices to find an enumerable predicate with non-productive complement. A predicate p is productive if its non-enumerability is witnessed by a function f s.t. for every enumerable subpredicate $\mathcal{W}_c$ of p, p and $\mathcal{W}_c$ differ on fc.

$$\mathsf{productive}\,(p : \mathbb{N} \to \mathbb{P}) := \exists f : \mathbb{N} \to \mathbb{N}. \forall c.\ (\forall x.\ \mathcal{W}_c x \to px) \to p(fc) \land \neg \mathcal{W}_c(fc)$$

Lemma 53. *Productive predicates are not enumerable.*

Proof. Let p be enumerable, i.e. $px \leftrightarrow \mathcal{W}_c x$ by Fact 17. Let f be a productive function for p, i.e. $p(fc)$ and thus $\mathcal{W}_c(fc)$, and $\neg\mathcal{W}_c(fc)$. Contradiction. ◀

Lemma 54. *$\overline{\mathcal{K}}$ is productive.*

Proof. Pick $\lambda n.n$. Let c be s.t. $\forall x.\ \mathcal{W}_c x \to \overline{\mathcal{K}} x$. Then $\mathcal{W}_c c \to \neg \mathcal{W}_c c$. Contradiction. ◀

Every productive predicate is Cantor-infinite and thus has an enumerable, Cantor-infinite subpredicate.

Lemma 55. *Every productive predicate is Cantor-infinite.*

Proof. From Fact 16 and since $x \in l$ is decidable, we obtain $c : \mathbb{L}\mathbb{N} \to \mathbb{N}$ s.t. $\mathcal{W}_{cl} x \leftrightarrow x \in l$ for every $l : \mathbb{L}\mathbb{N}$. Let p have a productive function f. We prove $\forall l : \mathbb{L}\mathbb{N}. \Sigma x.\ (\forall x_0 \in l. px_0) \to px \land x \notin l$, which suffices by a slight adaption of Lemma 46. Given $l : \mathbb{L}\mathbb{N}$, pick $x := f(cl)$. ◀

Corollary 56. *Every productive predicate has an enumerable, Cantor-infinite subpredicate.*

We can now show that the complement of m-complete predicates contains an enumerable, Cantor-infinite subpredicate since productiveness transports along many-one reductions:

Lemma 57. *Let $p \preceq_{\mathsf{m}} q$. If p is productive, q is productive.*

Proof. Let f many-one reduce p to q, and let g be a productive function for p. Fact 16 yields k s.t. $\mathcal{W}_{(kc)}x \leftrightarrow \mathcal{W}_c(fx)$. Then $\lambda c.\, f(g(kc))$ is a productive function for q. ◀

Lemma 58. *Let p be m-complete. Then $\overline{p}$ has an enumerable, Cantor-infinite subpredicate.*

Proof. Since productiveness of $\overline{p}$ follows from $\overline{\mathcal{K}} \preceq_{\mathsf{m}} \overline{p}$ and productiveness of $\overline{\mathcal{K}}$. ◀

In particular, then the complement of an m-complete predicate has a non-finite, enumerable subpredicate. Post's idea to find an enumerable, but undecidable, many-one *incomplete* predicate hinges on exactly this observation. We follow Post and define a predicate $p\colon X \to \mathbb{P}$ to be *simple* if it is enumerable, and its complement is both non-finite and does *not* contain a non-finite, enumerable predicate:

$$\mathsf{simple}\, p := \mathcal{E}p \land \neg\mathcal{X}\overline{p} \land \neg\exists q.\, (\forall x.\, qx \to \overline{p}x) \land \neg\mathcal{X}q \land \mathcal{E}q$$

Fact 59. *Complements of simple predicates are not enumerable.*

Corollary 60. *Simple predicates are undecidable.*

Lemma 61. *Simple predicates are m-incomplete.*

We follow the presentation of Rogers [28, §8.1 Th. II] to construct the same simple predicate as Post [25, §5]. The latter two properties of a simple predicate (non-finite complement and may not contain a non-finite, enumerable subpredicate) will drive the construction, since either one is easy to establish on their own, but the combination needs care. Post's idea was to construct a predicate S containing an element from every non-finite (enumerable) predicate $\mathcal{W}_c$. Thus, $\overline{S}$ cannot have a non-finite, enumerable subpredicate. To ensure that $\overline{S}$ is still non-finite, S contains only a *unique* $x > 2c$ with $\mathcal{W}_c x$ for every large enough $\mathcal{W}_c$. The condition $x > 2c$ ensures that there are at least n elements less or equal $2n$ in $\overline{S}$, and thus $\overline{S}$ is non-finite.

The only technical difficulty in the definition of S is to obtain a unique x satisfying $x > 2c$ and $\mathcal{W}_c x$ for every large enough $\mathcal{W}_c$. Post ensures this by choosing the x enumerated first by φ_c (i.e. the x with the least index n s.t. $\varphi_c n = \mathsf{Some}\, x$) satisfying $x > 2c$. We abstract away from this property, and observe that any function mapping c to an $x > 2c$ does the job.

We fix a function $\psi : \forall c.\, (\exists x.\, \mathcal{W}_c x \land x > 2c) \to \mathbb{N}$ s.t. $\psi c H = x \to \mathcal{W}_c x \land x > 2c$ and $\psi c H_1 = \psi c H_2$, i.e. a proof-irrelevant choice function for the predicate $\lambda x.\, \mathcal{W}_c x \land x > 2c$. Such a choice function ψ can be constructed by using for instance $\mu_{\mathcal{E}}$ (Fact 10).

We then define S as the image of ψ and prove that S is a simple predicate:

$$Sx := \exists c\colon \mathbb{N}. \exists H : (\exists y.\, \mathcal{W}_c y \land y > 2c).\, \psi c H = x$$

Lemma 62. *S is enumerable.*

Proof. There is a strong enumerator $E : \mathbb{N} \to \mathbb{N}$ for $\lambda c. \exists x. \mathcal{W}_c x \land x > 2c$ and thus we have $H\colon \forall n. \exists x. \mathcal{W}_{(En)} x \land x > 2 \cdot (En)$. Then, $\lambda n. \psi(En)(Hn)$ strongly enumerates S. ◀

Lemma 63. *$\overline{S}$ is non-finite.*

Proof. By Fact 40 it suffices to prove $\forall n. \neg\neg \exists L. |L| = n \land \# L \land \forall x \in L. \overline{S}x$. Given n, let $px := Sx \land x \leq 2n$. p is exhausted by $[0, \dots, 2n]$, thus it is not not finite. Since the claim is negative, we can assume some duplicate-free L_p listing p.

Now $|L_p| \leq n$: by decomposing Sx for every $x \in L_p$, we obtain L'_p with $\# L'_p$, $|L'_p| = |L_p|$, and $\forall c \in L'_p.\, c < n \land \exists H.\, \psi c H \in L_p$. Hence, $|L_p| \leq n$. Now the first n elements of $\mathsf{filter}(\lambda x.\, x \notin L_p)\, [0, \dots, 2n]$ form the wanted list. ◀

Lemma 64. *$\overline{S}$ contains no non-finite, enumerable subset.*

Proof. Let q be non-finite, contained in $\overline{S}$ and enumerated by φ_c via Fact 10. We derive a contradiction by showing $[0,\dots,2c]$ to exhaust q: Assume qx. Then $\mathcal{W}_c x$ since φ_c enumerates q. We have to prove $x \in [0,\dots,2c]$, which is stable. So let $x \notin [0,\dots,2c]$ and derive a contradiction. Since $\mathcal{W}_c x \land x > 2c$ holds, there is in particular a proof $H : \exists x.\, \mathcal{W}_c x \land x > 2c$. By definition of ψ, we have $\mathcal{W}_c(\psi c H)$. By definition of φ_c, $q(\psi c H)$, and thus $\neg S(\psi c H)$, i.e. $\neg\exists c' H'.\psi c' H' = \psi c H$ – contradiction. ◀

Theorem 65. *S is a simple predicate.*

Proof. Direct by Lemmas 62, 63, and 64. ◀

Based on our definitions and work, Forster, Kunze, and Lauermann [13] prove that nonrandom numbers, defined via Kolmogorov complexity, also are simple. The construction of any of these simple predicates settles Post's problem for $\preceq_{\mathsf{m}}$:

Theorem 66. *There exists an enumerable, undecidable and m-incomplete predicate.*

Proof. Direct by Corollary 60, Lemma 61, and Theorem 65. ◀

In the following we write $p \times \mathbb{N}$ for the predicate $\lambda(x,n){:}\,\mathbb{N}\times\mathbb{N}.px$ and use the following observation to prove that $\preceq_{\mathsf{m}}$ and $\preceq_1$ differ on enumerable, undecidable predicates.

Lemma 67. *Given $p{:}\,\mathbb{N}\to\mathbb{P}$ and $p \times \mathbb{N} \preceq_1 p$, p is not simple.*

Proof. Let p be simple and f a one-one reduction from $p \times (\lambda x : \mathbb{N}.\top)$ to p. We have to prove falsity, thus we can assume an element x_0 s.t. $\overline{p}x_0$ since $\overline{p}$ is non-finite by Fact 34. Now $\lambda x.\exists n.f(x_0,n) = x$ is a non-finite, enumerable subpredicate of $\overline{p}$. ◀

Corollary 68. *$\preceq_1$ and $\preceq_{\mathsf{m}}$ differ on enumerable, undecidable predicates.*

Proof. Let S be a simple predicate. Now $S \times \mathbb{N} \preceq_{\mathsf{m}} S$ with $\lambda(x,n).\, x$, but $S \times \mathbb{N} \not\preceq_1 S$ by Lemma 67. ◀

10 Truth-table reducibility

Recall that p is many-one reducible to q if a decision for px can be computed from one instance of q, namely $q(fx)$. Truth-table reducibility generalises this intuition: A predicate p is truth-table reducible to q if a decision for px can be computed by evaluating a boolean formula with atoms of the form qy_i for finitely many queries y_i. Equivalently, boolean formulas with n inputs can also be expressed as truth-tables with 2^n rows, explaining the name "truth-table reducibility". We fix a canonical listing function $\mathsf{gen}{:}\,\mathbb{N}\to\mathbb{L}(\mathbb{LB})$of boolean lists of length n, with $|\mathsf{gen}\,n| = 2^n$ and $\forall l{:}\,\mathbb{LB}.\, l \in \mathsf{gen}\,n \leftrightarrow |l| = n$.

We model truth-tables T with n inputs as boolean lists and define the value of T on input l as the i-th element of T if l is the i-th element of $\mathsf{gen}\,n$. Formally, $\mathsf{truthtable} := \mathbb{LB}$ and for $l{:}\,\mathbb{LB}$ of length n:

$$l \vDash T := \exists i.\, T[i] = \mathsf{Some}\,\mathsf{true} \land (\mathsf{gen}\,|l|)[i] = \mathsf{Some}\,l$$

When convenient, we assume $(l \vDash T){:}\,\mathbb{B}$ since $\lambda l T.\, l \vDash T$ is decidable. Any $f{:}\,\mathbb{LB}\to\mathbb{B}$ can be converted into the truth-table $\mathsf{map}f(\mathsf{gen}\,n)$ with n inputs s.t. $l \vDash \mathsf{map}f(\mathsf{gen}\,n) \leftrightarrow fl = \mathsf{true}$ provided $|l| = n$. On paper, we abuse notation and treat any function $f{:}\,\mathbb{LB}\to\mathbb{B}$ as truth-table.

A function $f : X \to \mathbb{L}Y \times \mathsf{truthtable}$ is a *truth-table reduction* from p to q if $\forall x.px \leftrightarrow l \vDash \pi_2(fx)$ for all lists l which reflect q pointwise on the query list $\pi_1(fx)$. A predicate $p : X \to \mathbb{P}$ is *truth-table reducible* to a predicate $q : Y \to \mathbb{P}$ if there is a truth-table reduction:

$$p \preceq_{\mathsf{tt}} q := \exists f : X \to \mathbb{L}Y \times \mathsf{truthtable}. \forall x\, l.\ \mathsf{Forall}_2\,(\lambda yb.\ qy \leftrightarrow b = \mathsf{true})\,(\pi_1(fx))\,l \to px \leftrightarrow l \vDash \pi_2(fx)$$

Lemma 69. 1. *Truth-table reducibility forms a pre-order.*
2. *If $p \preceq_{\mathsf{tt}} q$ and q is decidable then p is decidable.*
3. *If $p \preceq_{\mathsf{m}} q$ then $p \preceq_{\mathsf{tt}} q$.*

Truth-table reducibility can employ negation and thus does not transport enumerability.

Fact 70. $\overline{p} \preceq_{\mathsf{tt}} p$

Fact 71. *Truth-table reducibility is an upper semi-lattice.*

11 Hypersimple predicates

For settling Post's problem w.r.t. $\preceq_{\mathsf{tt}}$ in our synthetic setting we once more follow Rogers [28, §9.5] and introduce majorising functions: $f : \mathbb{N} \to \mathbb{N}$ *majorises* a predicate $p : \mathbb{N} \to \mathbb{P}$ if

$$\forall n. \neg\neg \exists l : \mathbb{L}\mathbb{N}.\ \#l \wedge |l| = n \wedge \forall m \in l. pm \wedge m \leq fn.$$

Note that we slightly adapted the definition of majorising in order to be more suitable for constructive proofs. We immediately introduce a strengthening of the notion following Odifreddi [22]: A function $f : \mathbb{N} \to \mathbb{N}$ *exceeds* a predicate $p : \mathbb{N} \to \mathbb{P}$ if $\forall n. \neg\neg \exists i.\ n < i \leq fn \wedge pi$.

Fact 72. *If f exceeds p, $\lambda n. f^n 0$ majorises p.*

Lemma 73. *If $\forall x.qx \to px$ and q is Cantor-infinite, there exists f exceeding p.*

Proof. Take $\lambda n.\ \pi_1(F[0, \dots, n])$, where F is obtained from the $\forall\Sigma$ formula in Lemma 46. ◀

Corollary 74. *Given* MP*, if there is no f majorising p, then $\overline{p}$ does not have a non-finite, enumerable subpredicate.*

Proof. Let no f majorise p and let q be a non-finite, enumerable subpredicate of $\overline{p}$. By MP and Lemma 47, q is generative. By Corollary 49, q is Cantor-infinite. By Lemma 73 there is f exceeding p. By Fact 72, p is majorised – contradiction. ◀

The following is an adaption of Th. III.3.10 in the book by Odifreddi [22], with

$$p \vDash (P, T) := \forall l.\ \mathsf{Forall}_2\,(\lambda xb.\ px \leftrightarrow b = \mathsf{true})\,P\,l \to l \vDash T.$$

Lemma 75. *If $\mathcal{K} \preceq_{\mathsf{tt}} p$ there exists a function exceeding $\overline{p}$.*

Proof. Let g be a tt-reduction from $\mathcal{K}$ to p. By Fact 16 there is $c : \mathbb{L}\mathbb{N} \to \mathbb{N}$ s.t. $\forall lx.\ \mathcal{W}(cl)x \leftrightarrow \neg((\lambda y.y \notin l) \vDash gx)$. Let $\mathsf{gen}_n : \mathbb{L}(\mathbb{L}\mathbb{N})$ contain exactly all duplicate-free lists l s.t. $\mathsf{max}\ l \leq n$. We define $a_{n,i} := c(\mathsf{gen}_n[i])$ for $i < 2^n$ and $a_{n,i} := c[]$ otherwise. Then $\lambda n.\ 1 + \mathsf{max}\,[\pi_1(ga_{n,i}) \mid i < 2^n]$ exceeds $\overline{p}$. To show this, let n be given and assume

$$(*) : \forall j.\ n < j < \mathsf{max}\,[\pi_1(ga_{n,i}) \mid i < 2^n] \to pj$$

We have to prove falsity. Note that $\neg\neg\exists.i < 2^n.\ \forall z.\ z \notin \mathsf{gen}_n[i] \leftrightarrow p^*z$ where $p^*z := (\neg\neg pz \wedge z \leq n) \vee z > n$. Since we have to prove falsity, we can assume such an i. Now by $(*)$ we have $\forall j.n < j \in \pi_1(ga_{n,i}) \to pj$ since $\pi_1(ga_{n,i}) < \mathsf{max}\,[\pi_1(ga_{n,i}) \mid i < 2^n]$ follows from $i < 2^n$. Since for $x \leq n$, $px \leftrightarrow p^*x$ by definition, we have $\forall x \in \pi_1(ga_{n,i}).\ px \leftrightarrow p^*x$. But now we obtain the following contradiction:

$$\begin{aligned} p \vDash ga_{n,i} &\leftrightarrow \mathcal{W}\, a_{n,i}\, a_{n,i} \leftrightarrow \mathcal{W}\,(c(\mathsf{gen}_n[i]))\, a_{n,i} \\ &\leftrightarrow \neg(\overline{\mathsf{gen}_n[i]} \vDash ga_{n,i}) \leftrightarrow \neg(p^* \vDash ga_{n,i}) \\ &\leftrightarrow \neg(p \vDash ga_{n,i}) \end{aligned}$$

□

A predicate p is *hypersimple* if it is enumerable and $\overline{p}$ is non-finite and not majorised:

$$\mathsf{hypersimple}\,(p\colon \mathbb{N}\to\mathbb{P}) := \mathcal{E}p \wedge \neg\mathcal{X}\overline{p} \wedge \neg\exists f.f \textit{ majorises } \overline{p}$$

Fact 76. *Hypersimple predicates are not* tt*-complete.*

Fact 77. *Given* MP*, hypersimple predicates are simple.*

12 Construction of a hypersimple predicate

We now construct and verify a hypersimple predicate, a result due to Post [25, §9]. We however follow Rogers [28, §8.1 Th. II], who presents a (more general) construction due to Dekker [5], defining a hypersimple predicate H_I for an arbitrary undecidable $I\colon\mathbb{N}\to\mathbb{P}$ with a strong, injective enumerator $E_I\colon\mathbb{N}\to\mathbb{N}$. By instantiating with e.g. $\mathcal{W}' := \lambda\langle c, x\rangle.\mathcal{W}_c x$, we obtain that $H_{\mathcal{W}'}$ is hypersimple, and thus enumerable, undecidable and tt-incomplete. The predicate $H_I\colon\mathbb{N}\to\mathbb{P}$ is defined as the so-called "deficiency predicate" of I:

$$H_I x := \exists x_0 > x.\ E_I x_0 < E_I x$$

Lemma 78. *$\overline{H_I}$ is non-finite.*

Proof. By Corollary 37 it suffices to prove $\forall x.\neg\neg\exists y \geq x.\overline{H_I}y$. We use complete induction on $E_I x$. Given $x\colon\mathbb{N}$ assume $(*)\colon\neg\exists y{\geq}x.\overline{H_I}y$. The claim is negative. Case analysis:

1. If $H_I x$, there exists $x_0 > x$ with $E_I x_0 < E_I x$. Hence, induction for x_0 yields $\neg\neg\exists y \geq x_0.\overline{H_I}y$ contradicting $(*)$.
2. If $\overline{H_I}x$ holds, x is an element as required. ◀

Lemma 79. *If f majorises $\overline{H_I}$, I is decidable.*

Proof. We prove that $g := \lambda x.x \in_{\mathbb{B}} \mathsf{map}\ E_I\ [0,\dots,f(\mathsf{S}\,x)]$ decides I if f majorises $\overline{H_I}$, i.e. $\forall x.\ Ix \leftrightarrow gx = \mathsf{true}$. The direction from right to left is easy, since E_I enumerates I. For the converse direction let $E_I n = x$ for some n. We show $n \in [0,\dots,f(\mathsf{S}\,x)]$, which is stable. Thus let $n \notin [0,\dots,f(\mathsf{S}\,x)]$, i.e. $n > f(\mathsf{S}\,x)$. Since f majorises $\overline{H_I}$ and the goal is $\bot$, we can assume l with $\#l$, $|l| = \mathsf{S}\,x$ and $\forall y \in l.\overline{H_I}y \wedge y \leq f(\mathsf{S}\,x)$. Let $m := \mathsf{max}(\mathsf{map}E_I\,l)$. We have that $m \geq x$, since $|\mathsf{map}E_I l| = |l| > x$ and $\#(\mathsf{map}E_I l)$, and that $m = E_I m_0$ for some $m_0 \in l$ and therefore $m_0 \leq f(\mathsf{S}\,x)$ and $\overline{H_I}m_0$ by the properties of l. We now prove $H_I m_0$ to obtain a contradiction. We have $n > f(\mathsf{S}\,x) \geq m_0$ and $E_I n = x \leq m = E_I m_0$. By the injectivity of E_I, $E_I n = E_I m_0$ implies $n = m_0$ (a contradiction), i.e. $E_I n < E_I m_0$ as required. ◀

Theorem 80. *H_I is a hypersimple predicate, and in particular $H_{\mathcal{W}'}$ is.*

Proof. $\mathcal{E}H_I$ follows by (6) of Fact 6. ◀

We have not yet proved H_I to be undecidable. A proof that H_I is simple seems to require MP. We instead constructively prove $\mathcal{D}H_I \to \mathcal{D}I$ and thus H_I undecidable for undecidable I:

To do so, we define $\mathcal{B}\colon \mathbb{N} \to \mathbb{T}$ describing a certain kind of guardedness with guards in $\overline{H_I}$:

$\mathcal{B}n := \Sigma x. E_I x \geq n \wedge \overline{H_I} x$

Lemma 81. $\forall n x. \Sigma x' \geq x. E_I x' > n.$

Proof. By the injectivity of E_I using Lemma 29. ◀

Lemma 82. *Let H_I be decidable by some f. Then, $\forall n x'. (\mathcal{B}n) + (\Sigma x > x'. E_I x < n)$.*

Proof. Let H_I be decidable. Given n and x', we show

$$\forall n_0 \geq x'. E_I n_0 > n \to (\mathcal{B}n) + (\Sigma x > x'. E_I x < n)$$

by complete induction on $E_I n_0$:

Let $n_0 \geq x'$ with $E_I n_0 > n$. If $\overline{H_I} n_0$, $\mathcal{B}n$ holds and we are done. If $H_I n_0$ holds, we have $\exists n_1 > n_0.\ E_I n_1 < E_I n_0$. Applying $\mu_{\mathbb{N}}$ gives the stronger assumption $\Sigma n_1 > n_0.\ E_I n_1 < E_I n_0$. Then, induction for $E_I n_1$ implies the claim.

Finally, Lemma 81 yields $n_0 \geq x'$ as assumed in the claim. ◀

Lemma 83. *Let H_I be decidable by some f. Then $\forall n. \mathcal{B}n$.*

Proof. Let H_I be decidable. Given n, we first show

$$\forall d. (\mathcal{B}n) + (\Sigma L. \# L \wedge |L| = d \wedge \forall x \in L. E_I x \leq n)$$

by induction on d:

The base case is immediate by picking $L := []$. In the step case, the inductive hypothesis either yields $\mathcal{B}n$ which shows the claim or a duplicate-free list L with $|L| = d$ and $\forall x \in L. E_I x \leq n$ and an element $x > \mathsf{max}\, L$. Now, Lemma 82 for $x' := \mathsf{max}\, L$ yields again either $\mathcal{B}n$ or an element $x > \mathsf{max}\, L$ with $E_I x \leq n$. Then, $(x :: L)$ is a list as required which.

For $d := 2 + n$ however, $\Sigma L. \# L \wedge |L| = 2 + n \wedge \forall x \in L. E_I x \leq n$ is contradictory: By the injectivity of E_I, $\mathsf{map}\, E_I\, L$ is duplicate-free, $|\mathsf{map}\, E_I\, L| = 2 + n > |[0, \dots, n]$ but $\forall x \in \mathsf{map}\, E_I\, L. x \in [0, \dots, n]$. Lemma 29 implies the contradiction. Hence, we have $\mathcal{B}n$. ◀

Lemma 84. $(\forall n. \mathcal{B}n) \to \mathcal{D}I.$

Proof. $\forall n. \mathcal{B}n$ yields $\forall n. \Sigma x. E_I x \geq n \wedge \forall x_0 > x. E_I x_0 > E_I x$ which implies $H : \forall n. \Sigma x. \forall x_0 > x. E_I x_0 > n$.

Then, $\lambda n. n \in_{\mathbb{B}} \mathsf{map}\, E_I\, [0, \dots, \pi_1(Hn)]$ decides I. ◀

Corollary 85. $\mathcal{D}H_I \to \mathcal{D}I.$

Proof. Directly by Lemma 83 and Lemma 84. ◀

Corollary 86. *H_I is undecidable, and in particular $H_{\mathcal{W}'}$ is.*

It seems that this fully constructive proof of $\mathcal{D}H_I \to \mathcal{D}I$ can be turned into an again fully constructive Turing reduction from I to H_I, provided a reasonable notion of synthetic Turing reductions: Instead of applying $\mu_{\mathbb{N}}$ to the proof of $H_I n_0$ in Lemma 82, partial functions (that must be crucially present in Turing reductions) would allow to find $n_1 > n_0$ with $E_I n_1 < E_I n_0$ via an unbounded search.

The above results settle Post's problem for $\preceq_{\mathsf{tt}}$:

Theorem 87. *There exists an enumerable, undecidable and* tt*-incomplete predicate.*

Proof. By Fact 76, Corollary 86 and Theorem 80. ◀

13 Discussion

This paper provides formalised, mechanised, constructive, synthetic proofs of major well-known results in the area of reducibility theory. The synthetic perspective tremendously helped in all three other aspects: Synthetic proofs are easier to formalise, easier to mechanise, and easier to constructivise since it clears the view by avoiding a hard to handle concrete model of computation.

We would argue that a synthetic formalisation is an almost necessary pre-requisite for a mechanised proof of advanced computability-theoretic results: working in a model of computation in the phase of mechanisation where details are still unclear seems too tedious, since defining and verifying programs in a model of computation is time-consuming and should be delayed as much as possible. The concretisation of all synthetically developed results to a model of computation in a second step is then routine. Even in the mechanised proofs, no principal obstacles should occur. For Coq specifically, the automatic extraction of functions to a λ-calculus by Forster and Kunze [12] would simplify the task further.

Choosing CIC rather than other systems is crucial for projects in constructive reverse mathematics concerning synthetic computability theory: the textbook proofs this paper is based on are often heavily classical, not only in the regard that different, constructively non-equivalent definitions of infinity are used interchangeably. It was thus conceivable that some results would depend on a classical logical axiom like LEM or the limited principle of omniscience LPO. Due to the separated universe of propositions CIC features, $\mathsf{CT} \wedge \mathsf{LEM}$ seems to be consistent, which is however inconsistent in e.g. HoTT [6]. However, in the end all our results do not require any classical assumptions. This means that the results can be transported to other type theories. Compatibility with classical logic is however still desirable both for possible investigations of classical synthetic computability theory as well as for purposes of constructive reverse mathematics.

As follow-up work to the present paper, Forster, Kunze, and Lauermann [13] prove that nonrandom numbers, defined via Kolmogorov complexity, form a simple predicate. Crucially, they started by working in a classical setting under the assumption of LEM, and gradually constructivise proofs until not even MP is required.

For future work, two questions seem most prominent: First, we would like to analyse whether our assumptions are minimal. It would be interesting to see whether MP is necessary to prove that hypersimple predicates are simple and undecidable. Secondly, Turing reducibility is the next natural step. The first author and Kirst propose a synthetic definition of Turing reducibility in CIC in [10], based on an idea by Bauer [2]. Several open questions remain, most prominently how to construct a universal oracle functional. We would like to develop synthetic versions of the Kleene-Post theorem [16], stating that there are incomparable Turing-reducibility degrees, Post's theorem [26], connecting Turing reducibility via the Turing jump to the arithmetical hierarchy, and directly subsequent to the present paper the Friedberg-Mučnik theorem [14, 21], settling Post's problem by proving that there exists an enumerable, undecidable, but Turing-reducibility incomplete predicate.

The Coq development accompanying the paper is sizeable due to the number of results covered, but due to the synthetic approach it is still concise. It is relatively elementary: No advanced dependent types are needed, we do not rely on complex hand-written automation.

The preparations regarding the axioms for synthetic computability, finiteness, Pigeonhole principles, and infinity, take 900 lines of code (LOC). Defining $\preceq_{\mathsf{m}}$, $\preceq_{1}$, and $\preceq_{\mathsf{tt}}$ and proving related facts needs 700 LOC. General results regarding simple and hypersimple predicates take 100 and 230 LOC respectively. The construction of S and S^* needs 700 LOC, of H_I only 250. The `std++` library [31] is used for several convenient auxiliary functions regarding lists missing in Coq's standard library.

A Cylindrification proofs

We characterise many-one reducibility and truth-table reducibility in terms of one-one reducibility.

For the former, the proofs by Rogers [28, §7.6 Th. VIII] work via so-called cylindrification and we generalise them slightly to apply to base types other than $\mathbb{N}$. In our setting, a cylindrification of a predicate $p : X \to \mathbb{P}$ is $p \times Z$ for an inhabited type Z, and predicates of the form $p \times Z$ in general are called cylinders.

We fix two predicates $p: X \to \mathbb{P}$, $q: Y \to \mathbb{P}$ and $z: Z$.

Lemma 88. *Let $f: Y \times Z \to Z$ be an injection. Then $q \preceq_{\mathsf{m}} p \leftrightarrow q \times Z \preceq_1 p \times Z$.*

Proof. We first prove:

1. $p \preceq_1 p \times Z$ given $z : Z$.
2. $p \times Z \preceq_{\mathsf{m}} p$.
3. If $q \preceq_{\mathsf{m}} p \times Z$ then $q \preceq_1 p \times Z$, provided an injection $g: Y \times X \to Z$.

The only interesting proof is (3): Let f reduce q to $p \times Z$. Then $\lambda y.(\pi_1(fy), g(y, \pi_1(fy)))$ is proves $q \preceq_1 p \times Z$. It is injective since g is injective and correct since f is correct.

Now, the direction from left to right follows from (1), (2), and (3). The converse direction from (1) and (2). ◀

Corollary 89. *Let $f: Y \to Z$ be injective. Then $p \preceq_1 p \times Z$ and for q s.t. $q \equiv_{\mathsf{m}} p$ and $p \preceq_1 q$, $q \preceq_1 p \times Z$.*

To characterise truth-table reducibility in terms of one-one reduciblity, we introduce so-called truth-table cylinders, following Rogers [28, §8.4 Th. IX]. Given a predicate $p: X \to \mathbb{P}$, we define the predicate $p^{\mathsf{tt}}: \mathbb{L}X \times \mathsf{truthtable} \to \mathbb{P}$:

$$p^{\mathsf{tt}} := \lambda z.\ \forall l.\ \mathsf{Forall}_2\ (\lambda x b.\ px \leftrightarrow b = \mathsf{true})(\pi_1 z)\ l \to l \vdash \pi_2 z$$

The characterisation seems to be inherently non-constructive, we thus assume p to be stable.

Lemma 90. *Let $p: X \to \mathbb{P}$, $q: Y \to \mathbb{P}$, $x_0: X, y_0: Y$, and $g: \mathbb{L}X \times \mathsf{truthtable} \to Y$ be an injection. If p is stable, then $p \preceq_{\mathsf{tt}} q \leftrightarrow p^{\mathsf{tt}} \preceq_1 q^{\mathsf{tt}}$.*

Proof. We loosely follow the proof by Rogers [28, §8.4 Th. IX] and prove the following: (1) If p is stable then $p \preceq_1 p^{\mathsf{tt}}$. (2) $p^{\mathsf{tt}} \preceq_{\mathsf{tt}} p$. (3) Every reduction $p \preceq_{\mathsf{tt}} q$ can be given via an injective reduction function, provided an injection $f: X \to Y$ exists. (4) If p is stable, $f: X \to Y$ injective then $p \preceq_{\mathsf{tt}} q \to p \preceq_1 q^{\mathsf{tt}}$.

(1) and (2) are straightforward. For (3) assume $p \preceq_{\mathsf{tt}} q$ via a reduction function g and construct $g'x := (fx :: \pi_1(gx), \lambda\, b :: L.\ \pi_2(gx)L)$. g' is injective since f is injective. For (4), let $p \preceq_{\mathsf{tt}} q$ via an injection f by (3). Then f 1-reduces p to q^{tt}. The claim from left to right follows using (1), (2), and (4), the direction from right to left needs (1) and (4). ◀

Corollary 91. *If $p, q: \mathbb{N} \to \mathbb{P}$ and p stable, $p \preceq_{\mathsf{tt}} q \leftrightarrow p^{\mathsf{tt}} \preceq_1 q^{\mathsf{tt}}$.*

References

1 Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006.

2 Andrej Bauer. Synthetic mathematics with an excursion into computability theory. University of Wisconsin Logic seminar, 2021. URL: `http://math.andrej.com/asset/data/madison-synthetic-computability-talk.pdf`.

3 Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*, volume 97. Cambridge University Press, 1987.

4 Nigel Cutland. *Computability: An introduction to recursive function theory*. Cambridge university press, 1980.

5 James C. E. Dekker. A theorem on hypersimple sets. *Proceedings of the American Mathematical Society*, 5:791–796, 1954.

6 Yannick Forster. Church's Thesis and Related Axioms in Coq's Type Theory. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13455, doi:10.4230/LIPIcs.CSL.2021.21.

7 Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. URL: https://ps.uni-saarland.de/~forster/thesis.

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

9 Yannick Forster. Parametric Church's Thesis: Synthetic computability without choice. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, pages 70–89, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-93100-1_6.

10 Yannick Forster and Dominik Kirst. Synthetic Turing reducibility in constructive type theory. 28th International Conference on Types for Proofs and Programs (TYPES 2022), 2022.

11 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*, pages 38–51, 2019.

12 Yannick Forster and Fabian Kunze. A Certifying Extraction with Time Bounds from Coq to Call-By-Value Lambda Calculus. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11072, doi:10.4230/LIPIcs.ITP.2019.17.

13 Yannick Forster, Fabian Kunze, and Nils Lauermann. Synthetic Kolmogorov Complexity in Coq. working paper or preprint, March 2022. URL: https://hal.inria.fr/hal-03596267.

14 Richard M Friedberg and Hartley Rogers Jr. Reducibility and completeness for sets of integers. *Mathematical Logic Quarterly*, 5(7-13):117–125, 1959. doi:10.1002/malq.19590050703.

15 Felix Jahn. *Synthetic One-One, Many-One, and Truth-Table Reducibility in Coq*. Bachelor's thesis, Saarland University, 2020.

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

17 Georg Kreisel. Mathematical logic. *Lectures in modern mathematics*, 3:95–195, 1965. doi:10.2307/2315573.

18 Bassel Mannaa and Thierry Coquand. The independence of markov's principle in type theory. *Logical Methods in Computer Science*, 13, 2017.

19 Andrei Andreevich Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375, 1954.

20 The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 367–381, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3372885.3373824.

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

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

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

24 Pierre-Marie Pédrot and Nicolas Tabareau. Failure is not an option. In *European Symposium on Programming*, pages 245–271. Springer, 2018.

25 Emil L Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944.

26 Emil L. Post. Degrees of recursive unsolvability - preliminary report. In *Bulletin of the American Mathematical Society*, volume 54, pages 641–642. American Mathematical Society (AMS), 1948.

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

28 Hartley Rogers. *Theory of recursive functions and effective computability*. CUMINCAD, 1987.

29 Robert I Soare. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets*. Springer Science & Business Media, 1999.

30 The Coq Development Team. The Coq Proof Assistant, version 8.11.0. `https://doi.org/10.5281/zenodo.3744225`, Jan 2020. `doi:10.5281/zenodo.3744225`.

31 The Coq std++ Team. An extended "standard library" for Coq. `https://gitlab.mpi-sws.org/iris/stdpp`, 2020.

32 Anne Sjerp Troelstra and Dirk van Dalen. Constructivism in mathematics. vol. i, volume 121 of. *Studies in Logic and the Foundations of Mathematics*, 26, 1988.