The Kleene-Post and Post's Theorem in the Calculus of Inductive Constructions

Yannick Forster
Inria, Nantes Université, LS2N, Nantes, France

Dominik Kirst
Ben-Gurion University of the Negev, Beer-Sheva, Israel
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Niklas Mück
Saarland University and MPI-SWS, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

The Kleene-Post theorem and Post's theorem are two central and historically important results in the development of oracle computability theory, clarifying the structure of Turing reducibility degrees. They state, respectively, that there are incomparable Turing degrees and that the arithmetical hierarchy is connected to the relativised form of the halting problem defined via Turing jumps.

We study these two results in the calculus of inductive constructions (CIC), the constructive type theory underlying the Coq proof assistant. CIC constitutes an ideal foundation for the formalisation of computability theory for two reasons: First, like in other constructive foundations, computable functions can be treated via axioms as a purely synthetic notion rather than being defined in terms of a concrete analytic model of computation such as Turing machines. Furthermore and uniquely, CIC allows consistently assuming classical logic via the law of excluded middle or weaker variants on top of axioms for synthetic computability, enabling both fully classical developments and taking the perspective of constructive reverse mathematics on computability theory.

In the present paper, we give a fully constructive construction of two Turing-incomparable degrees à la Kleene-Post and observe that the classical content of Post's theorem seems to be related to the arithmetical hierarchy of the law of excluded middle due to Akama et. al. Technically, we base our investigation on a previously studied notion of synthetic oracle computability and contribute the first consistency proof of a suitable enumeration axiom. All results discussed in the paper are mechanised and contributed to the Coq library of synthetic computability.

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

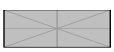
Keywords and phrases Constructive mathematics, Computability theory, Logical foundations, Constructive type theory, Interactive theorem proving, Coq proof assistant

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.37

Supplementary Material `github.com/uds-psl/coq-synthetic-computability/tree/code-paper-kleene-post-post`.

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.
Dominik Kirst: is supported by a Minerva Fellowship of the Minerva Stiftung Gesellschaft für die Forschung mbH.

Acknowledgements We want to thank Felix Jahn, Gert Smolka, Dominique Larchey-Wendling, and the participants of the TYPES '22 conference for many fruitful discussions about Turing reducibility, Ian Shilito and the anonymous reviewers of this paper for helpful feedback, as well as Martin Baillon, Yann Leray, Assia Mahboubi, Pierre-Marie Pédrot, and Matthieu Piquerez for discussions about notions of continuity. The central inspiration to start working on Turing reducibility in type theory is due to Andrej Bauer's talk at the Wisconsin logic seminar in February 2021. Furthermore, the first two authors want to thank Benjamin Kaminski, his research group, and the royals of the castle for hosting their nostalgic stay in Saarbrücken to finish writing this paper.



1 Introduction

We study two well-known results in computability theory from the perspective of synthetic mathematics: the Kleene-Post theorem [31], stating that there are incomparable Turing degrees,[1] and Post's theorem [41], establishing a close link between Turing jumps and the arithmetical hierarchy due to Kleene [30] and Mostowski [35]. Both have been historically important: The former clarifies that Turing degrees are not linearly ordered, whereas the latter links a purely logical characterisation of sets of numbers with oracle computations.

Although still covered in the standard canon of computability textbooks [43, 38, 45], these theorems pose an interesting benchmark for the synthetic approach since they involve higher-order notions like Turing reductions and relative semi-decidability, that are not obvious to represent synthetically, see e.g. the discussion in the PhD thesis of the first author [12, §9.2]. Furthermore, both results also crucially rely on an enumeration of oracle machines, which has not been established as a consistent axiom in synthetic computability yet. Therefore in previous work [15], we have first suggested a careful definition of oracle computability and conjectured that it allows to derive the consistency of such an enumeration. In the present paper, we confirm this conjecture by constructing an enumeration from the well-known axiom Church's thesis (CT) [33] and then use this enumeration to prove and analyse the Kleene-Post and Post's theorem.

Synthetic computability exploits the fact that in a constructive foundation of mathematics only computable functions are definable a priori. Non-computable functions arise a posteriori when combining function existence principles such as countable choice or unique choice with classical axioms like the law of excluded middle (LEM) or weaker counter-parts such as the limited principle of omniscience (LPO) or the weak limited principle of omniscience (WLPO). In constructive settings, where such classical principles would have to be explicitly assumed, the theory of computable functions can thus be studied by considering the whole function space as computable: a so-called synthetic approach allowing for a concise but precise mathematical development. In contrast, in classical settings such as ZFC set theory one has to resort to an analytic model of computability like Turing machines or one of its equivalents, cluttering formal definitions and proofs with computability conditions.

Synthetic approaches to computability have been expressed in several dialects of constructive mathematics: Markov's work in the Russian school of constructivism relies explicitly on a computational background theory [34]. Kreisel's formulation of the axiom CT internalises the fact that every (definable) function is computable in a model of computation [33], e.g. working over intuitionistic Heyting arithmetic. Working in Bishop-style constructive mathematics, Richman [42] suggests an axiom stating that the partial function space is enumerable, which can be stated without even defining models of computation. Bauer [2, 3] works in the effective topos [23] where the set of enumerable sets is enumerable, formulated as the axiom EA, and Swan and Uemura [46] establish the consistency of CT for univalent type theory.

For the specific case of Turing reductions, Bauer [4] characterises an oracle computation by a higher-order functional with certain continuity and computability conditions. However, due to countable choice being present, his setting based on the effective topos is inherently anti-classical, i.e. no axioms like LPO or even WLPO can be assumed consistently on top of EA,

[1] The seminal 1954 paper by Kleene and Post establishes various other results besides this one. In particular, it also proves that both constructed degrees Turing reduce to the halting problem. We follow the terminology to only use the incomparability part of the result used e.g. in the textbooks by Odifreddi [38] and Cooper [6] as well as more recent work on (classical) reverse mathematics by Sanders [44] and Brattka et. al [5].

so one is bound to fully constructive reasoning. Recently proposing an alternative definition, Swan [47] works around the incompatibility of univalent mathematics with classical axioms in synthetic computability due to the presence of unique choice [11] by characterising oracle computations via 0-truncated $\neg\neg$-sheafification. As this implements a negative translation making constructive distinctions invisible, one is bound to fully classical reasoning. Thus, both settings are unusable for a sub-classical logical analysis of computability theory in the style of constructive reverse mathematics [24, 10]. Offering a solution, such an analysis is possible in our setting since the calculus of inductive constructions (CIC) [7, 8, 39] provides a universe of (possibly classical) propositions mostly disconnected from the (possibly computable) function spaces, so we can freely assume and distinguish classical axioms together with the base axiom for synthetic computability.

In the case of the Kleene-Post theorem, we report on a fully constructive proof that can be obtained by standard techniques of modelling mathematics in a constructive foundation. In the case of Post's theorem, we localise the use of classical logic: Based on Akama et. al's arithmetical hierarchy of the law of excluded middle [1] we derive that Σ_n-LEM is sufficient to obtain Post's theorem up to the same level n. This remains a preliminary analysis, however, since we do not prove that Post's theorem at level n in turn implies Σ_n-LEM. For many auxiliary results, e.g. for closure properties of the arithmetical hierarchy, we conjecture that weaker axioms would suffice. In particular, we observe a seemingly weaker variant of Markov's principle at play that appears not to have been treated in the literature before.

A preliminary proof of the Kleene-Post theorem with an assumed enumerator for a weaker definition of Turing reductions has been discussed in an extended abstract at TYPES '22 [26]. The same abstract discusses a proof of Post's theorem with a similar assumption and using the full law of excluded middle, based on the Bachelor's thesis of the third author [37].

Contributions We contribute a consistency proof of an enumeration axiom for oracle computable functionals, derived from the well-known axiom CT and its fully synthetic variant EPF [13, 12]. Based on this axiom, we give a fully constructive synthetic proof of the Kleene-Post theorem [31] following Odifreddi [38] and a synthetic definition of the Turing jump. We then give the first formal definition of the arithmetical hierarchy in constructive type theory and mechanise several proofs due to Akama et. al [1] about the arithmetical hierarchy of classical axioms. We use axioms from the Σ_n-level of this hierarchy to prove Post's theorem [41] and discuss perspectives regarding a reverse analysis. Lastly, we give a more traditional definition of the arithmetical hierarchy via a syntactic modeling of first-order logic and prove that these hierarchies are equivalent if and only if CT holds. All proofs are machine-checked using the Coq proof assistant [48] and all statements in this PDF are hyperlinked with the HTML version of the proofs.

Outline After collecting some preliminary definitions and notations in Section 2, we recall the concept and basic properties of synthetic oracle computability of [15] in Section 3. We next derive an enumerator of oracle computations from established axioms for synthetic computability (Section 4), followed by a first application of the enumerator to derive the Kleene-Post theorem (Section 5). To prepare the second application regarding Post's theorem (Section 9), we first study Turing jumps (Section 6), the arithmetical hierarchy (Section 7), and classical assumptions characterising the structure of arithmetical sets (Section 8). We complement the technical development with a purely syntactic definition of the arithmetical hierarchy in Section 10 and conclude in Section 11.

2 Preliminaries

We use inductive types of natural numbers $\mathbb{N}$ and booleans $\mathbb{B}$ with constructors true and false, lists X^* with constructors $[]$ and $x :: l$ and concatenation operation $l_1 +\!\!+ l_2$, the sum type $X + Y$ with constructors $\mathsf{inl}\, x$ and $\mathsf{inr}\, x$, and vectors X^n with same notations as for lists.

We use a type of partial functions $X \rightharpoonup Y$, and write $fx \triangleright y$ if fx is defined with value y. The concrete implementation of partial functions is not important. Mathematically, we abstract away from details, whereas in the Coq formalisation we work against an abstract interface of partial functions, that can for instance be instantiated using step-indexing.

Two crucial properties are that the graph of partial functions should be testable via step-indexing and that one can perform unbounded search:

Lemma 1. *Partial functions have the following properties:*

1. *There is a function* $\epsilon\colon (X \rightharpoonup Y) \to \mathbb{N} \to X \to Y \to \mathbb{B}$ *with* $fx \triangleright y \leftrightarrow \exists n.\ \epsilon\ f\ n\ x\ y = \mathsf{true}$.
2. *There is a function* $\mu\colon (\mathbb{N} \to \mathbb{B}) \rightharpoonup \mathbb{N}$ *with* $\mu f \triangleright n \leftrightarrow fn \triangleright \mathsf{true} \wedge \forall m < n.\ fm \triangleright \mathsf{false}$.

As is common in type theory we work with predicates instead of sets, a formality that does not introduce any mathematical overhead or relevant change of presentation. Predicates are defined as functions into the universe of propositions, i.e. $p\colon X \to \mathbb{P}$. We define the complement of a predicate as $\overline{p}x := \neg px$. The universe of propositions is impredicative in CIC, which plays no essential role. More relevantly, propositions in CIC cannot in general be analysed in computations, meaning that e.g. projection functions of type $(\exists n : \mathbb{N}.\ px) \to \mathbb{N}$ can only be defined in special circumstances, not in general.

We now define basic notions of synthetic computability theory [16, 2]: decidability, semi-decidability, and many-one reducibility.
A predicate $p\colon X \to \mathbb{P}$ is decidable if it is reflected by a boolean function:

$$\mathcal{D}(p) := \exists f\colon X \to \mathbb{B}.\ \forall x\colon X.\ px \leftrightarrow fx = \mathsf{true}$$

We define semi-decidability using partial functions into the type $\mathbb{1}$ with only element $\star$:

$$\mathcal{S}(p) := \exists g\colon X \rightharpoonup \mathbb{1}.\ \forall x\colon X.\ px \leftrightarrow gx \triangleright \star$$

Lastly, a predicate $p\colon X \to \mathbb{P}$ is many-one reducible to $q\colon Y \to \mathbb{P}$ if p can be encoded into q:

$$p \preceq_m q := \exists f\colon X \to Y.\ \forall x.\ px \leftrightarrow q(fx)$$

The biggest deviation from paper presentations of computability theory is that we do not consider equivalences classes w.r.t. any reducibility, which are notoriously hard to treat in formalised approaches, but rather talk about concrete representatives of equivalence classes.

3 Oracle Computability and Turing Reducibility

We use the synthetic definition of oracle computability introduced in prior work [15]. It is based on a notion of computability of functionals $F\colon (Q \to A \to \mathbb{P}) \to I \to O \to \mathbb{P}$. The argument $R\colon Q \to A \to \mathbb{P}$ is to be read as the oracle relating questions $q : Q$ to answers $a : A$, $i\colon I$ is the input to the computation, and $o\colon O$ is the output. Technically, synthetic oracle computability of such functionals is based on a notion of *continuity* via a partial and more extensional variant of dialogue trees [49]. Conceptually, considering oracle computations via continuous functionals was introduced by Kleene [29] and Kreisel [32]. Kleene calls such functionals *countable*, since they can analytically be represented in a model of computation. Synthetically, they become countable using axioms for synthetic computability as discussed in Section 4.

A functional $F\colon (Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$. is considered (oracle)-computable if there is an underlying computation tree $\tau\colon I\to A^*\rightharpoonup Q+O$ capturing the extensional behaviour of F by

$$\forall Rxb.\ FRxb \leftrightarrow \exists qs\ as.\ \tau x\ ;\ R\vdash\ qs\ ;\ as\wedge \tau\ x\ as\triangleright \mathsf{out}\ b$$

where the *interrogation relation* $\sigma; R\vdash qs; as$ is inductively defined for $\sigma\colon A^*\rightharpoonup Q+O$ as

$$\frac{}{\sigma\ ;\ R\vdash\ []\ ;\ []} \qquad \frac{\sigma\ ;\ R\vdash\ qs\ ;\ as \qquad \sigma\, as\triangleright \mathsf{ask}\ q \qquad Rqa}{\sigma\ ;\ R\vdash\ qs +\!\!+ [q]\ ;\ as +\!\!+ [a]}$$

and where we write $\mathsf{ask}\ q$ and $\mathsf{out}\ o$ for the respective injections into the sum type $Q+O$.

Intuitively, a computation tree takes as input a list of answers given already by the oracle. It can then (1) ask another question to the oracle, (2) return an output, or (3) compute forever while neither asking a question or returning an output. The computation on an input then is described by a sequence of runs of the tree, first given the empty list $[\,]$ as input, and then subsequently the list of all answers produced by the oracle. We use τ as letter for trees that take input and σ for trees that do not.

We now define Turing reducibility from p to q as computable functionals that map the characteristic relation of q to the characteristic relation of p. To this end, given $r\colon Z\to\mathbb{P}$, we define its characteristic relation $\hat{r}\colon Z\to\mathbb{B}\to\mathbb{P}$ as

$$\hat{r}zb := \textbf{if}\ b\ \textbf{then}\ rz\ \textbf{else}\ \neg rz$$

Then a Turing reduction from $p\colon X\to\mathbb{P}$ to $q\colon Y\to\mathbb{P}$ is an oracle-computable functional $F\colon (Y\to\mathbb{B}\to\mathbb{P})\to X\to\mathbb{B}\to\mathbb{P}$ such that $\forall xb.\,\hat{p}xb\leftrightarrow F\hat{q}xb$ and we write $p\preceq_{\mathsf{T}} q$ if such F exists.

Technically we do not work explicitly with trees in this paper, but rely on the following properties [15], establishing essentially the closure under a certain function algebra, containing variants of applications, constants, identity, branching, composition, and unbounded search.

Lemma 2. *The following functionals are computable provided the given functionals are:*

1. $\lambda Rio.\ F(gi)o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $g\colon I\to I'$ *and* $F\colon (Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$,
2. $\lambda Rio.\ \bot$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$,
3. $\lambda Rio.\ fi\triangleright o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $f\colon I\rightharpoonup O$,
4. $\lambda Rio.\ fi = o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $f\colon I\to O$,
5. $\lambda Rio.\ o = v$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $v\colon O$,
6. $\lambda Rio.\ Rio$ *of type* $(I\to O\to\mathbb{P})\to I\to O\to\mathbb{P}$,
7. $\lambda Rio.$ **if** fi **then** F_1Rio **else** F_2Rio *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* F_1, F_2 *of the same type and* $f : I\to\mathbb{B}$,
8. $\lambda Rio.\ \exists o'\colon O'.\ F_1\ R\ i\ o'\wedge F_2\ R\ (i,o')\ o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $F_1\colon (Q\to A\to\mathbb{P})\to (I\to O'\to\mathbb{P})$ *and* $F_2\colon (Q\to A\to\mathbb{P})\to ((I\times O')\to O\to\mathbb{P})$,
9. $\lambda Rin.\ R\ (i,n)\ \mathsf{true}\wedge\forall m<n.\ R\ (i,m)\ \mathsf{false}$ *of type* $((I\times\mathbb{N})\to\mathbb{B}\to\mathbb{P})\to I\to\mathbb{N}\to\mathbb{P}$.

Lemma 3. *If* $p\preceq_m q$, *then* $p\preceq_{\mathsf{T}} q$.

Proof. Let f be given such that $\forall x.\ px\leftrightarrow q(fx)$. Define $FRxb := R(fx)b$, which is computable with Lemma 2 (**1**) and (**6**). We have that $\hat{p}xb\leftrightarrow F\hat{q}xb$ by case analysis on b. ◀

Turing reducibility can be seen as decidability of p relative to q. Similarly, we also define the central notion of relative semi-decidability. A predicate $p\colon X\to\mathbb{P}$ is semi-decidable in $q\colon Y\to\mathbb{P}$ if there is an oracle-computable functional $F\colon (Y\to\mathbb{B}\to\mathbb{P})\to (X\to\mathbb{1}\to\mathbb{P})$ such that $\forall x.\ px\leftrightarrow F\hat{q}x\star$ and we write $\mathcal{S}_q(p)$ if such F exists.

Lemma 4. *If* $p\preceq_{\mathsf{T}} q$, *then* $\mathcal{S}_q(p)$ *and* $\mathcal{S}_q(\overline{p})$ *for the complement* $\overline{p}\,x := \neg p\,x$.

Proof. See Lemma 3 in [15]. ◀

4 Enumeration Axiom for Synthetic Oracle Computability

Non-relative synthetic computability uses an axiomatic assumption of an enumerator of all partial functions [17], or equivalently of all enumerable predicates [14, 2], or equivalently a step-indexed interpreter enumerating all total functions [13]. All are consequences of the well-known axiom CT [33] stating that all total functions of type $\mathbb{N}\to\mathbb{N}$ are computable, which is consistent in type theory [46, 13].

For relative synthetic computability, we introduce a novel axiom and derive its consistency from the consistency of CT by constructing an enumerator of computable functionals based on an enumerator of partial functions.

The axiom EPF, itself a consequence of CT [13] and thus consistent, states that there is an enumerator $\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$ universal for partial functions.

$$\mathsf{EPF} := \exists\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\ \forall f\colon\mathbb{N}\rightharpoonup\mathbb{N}.\exists c\colon\mathbb{N}.\ \forall xv.\ \theta_c x\rhd v \leftrightarrow fx\rhd v$$

Note that EPF implies the existence and undecidability of a synthetic variant of the self-halting problem $\mathcal{K}x := \exists v.\ \theta_x x\rhd v$ [12], and thus in particular we have e.g. $\mathcal{K}\not\leq_{\mathsf{T}}\emptyset$.

For various results we will need a version working with families of functions $f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$ and consequently coding functions $\gamma\colon\mathbb{N}\to\mathbb{N}$ such that f_n and $\theta_{\gamma n}$ agree, in line with the parametric version of the axiom EA used by Forster and Jahn [14]. Intuitively, this is related to having an *s-m-n* operator for θ. We can construct such a stronger form of θ directly by using a bijection between $\mathbb{N}$ and $\mathbb{N}\times\mathbb{N}$. This pairing function is written as $\langle x,y\rangle$ and we use the notation $f\langle x,y\rangle := \dots$ to define a function which takes as argument one single natural number, and uses the inverse of the pairing function to decompose it into x and y implicitly.

Lemma 5. EPF *is equivalent to the following parametric form:*

$$\exists\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\ \forall f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\exists\gamma\colon\mathbb{N}\to\mathbb{N}.\ \forall nxv.\ \theta_{\gamma n}x\rhd v\leftrightarrow f_n x\rhd v$$

Proof. The direction from right to left is immediate. From left to right, take θ' which enumerates all partial functions and define $\theta_{\langle c,n\rangle}x := \theta'_c\langle n,x\rangle$. Now given a family $f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$, use universality of θ' on the function $g\langle n,x\rangle := f_n x$ to obtain c with $*:\forall av.\ \theta'_c a\rhd v\leftrightarrow ga\rhd v$. Then for $\gamma n := \langle c,n\rangle$ we have $\theta_{\gamma n}x\rhd v \overset{\gamma}{\leftrightarrow} \theta_{\langle c,n\rangle}x\rhd v \overset{\theta}{\leftrightarrow} \theta'_c\langle n,x\rangle\rhd v \overset{*}{\leftrightarrow} g\langle n,x\rangle\rhd v \overset{g}{\leftrightarrow} f_n x\rhd v$. ◀

We now explicitly explain the construction for an enumeration of all relative semi-deciders, but the same construction works for arbitrary computable functionals based on retracts of $\mathbb{N}$. To define an enumeration of all semi-deciders, we use retractions $\iota_{1,2}\colon X_{1,2}\to\mathbb{N}$ and $\rho_{1,2}\colon\mathbb{N}\to X_{1,2}$ with $\forall n.\ \iota_{1,2}(\rho_{1,2}n)=n$ for $X_1 := \mathbb{N}\times\mathbb{B}^*$ and $X_2 := \mathbb{N}+\mathbb{1}$.

We define $\xi\colon\mathbb{N}\to(\mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1})$ which then consequently parametrically enumerates every family of trees τ by:

$$\xi_c xl := \theta_c(\iota_1(x,l)) \gg\!= \lambda v.\ \mathsf{ret}\ (\rho_2 v) \quad \textit{with} \quad \forall\tau.\ \exists\gamma.\ \forall n\,x\,l\,v.\ \xi_{\gamma n}xl\rhd v\leftrightarrow\tau_n xl\rhd v$$

The bind notation $\gg\!=$ evaluates the left hand side to a value if possible, and then passes its value to the function on the right hand side. Given a tree $\tau\colon\mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1}$ we define $\hat{\tau}Rx := \exists qs\,as.\ \sigma\,;R\vdash qs\,;as\wedge\tau\ x\ as\rhd\mathsf{out}\star$ and finally $\Xi_c Rx := \widehat{\xi_c}Rx$.

Lemma 6. *The relation* $\lambda R(c,x)o.\ \Xi_c Rx$ *is computable.*

Proof. Use $\lambda(c,x)l.\ \xi_c xl$. ◀

Lemma 7. *Given a family of trees, i.e.* $\tau_i\colon\mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1}$, *there exists a function* $\gamma\colon\mathbb{N}\to\mathbb{N}$ *such that* $\forall iRx.\ \Xi_{\gamma i}Rx\leftrightarrow\widehat{\tau_i}Rx$.

Corollary 8. *Given a computable F there exists a code c such that $\forall Rx.\, FRx\star \leftrightarrow \Xi_c Rx$.*

We can give a similar enumerator for Turing reductions, which will be necessary for the Kleene-Post theorem. Note that we do not require the enumerator there to be parametric.

Theorem 9. *There is an enumerator of functionals $\chi\colon \mathbb{N}\to(\mathbb{N}\to\mathbb{B}\to\mathbb{P})\to\mathbb{N}\to\mathbb{B}\to\mathbb{P}$ such that*

1. *χ_c is computable and*
2. *given a computable F there exists a code c such that $\forall Rxb.\, FRxb \leftrightarrow \chi_c Rxb$.*

In the remainder of this paper, we just need to assume the enumerators Ξ and χ without any knowledge of their implementation. So their availability can be treated as an axiom for synthetic oracle computability and the construction provided in this section amounts to a consistency proof. In fact, this axiom likewise implies EPF:

Lemma 10. *The statement of Theorem 9 implies* EPF.

Proof. Instead of proving EPF, we prove the following version, which is equivalent [12]:

$$\mathsf{EPF}_{\mathbb{B}} := \exists\theta\colon \mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{B}).\ \forall f\colon \mathbb{N}\rightharpoonup\mathbb{B}.\exists c\colon \mathbb{N}.\ \forall xv.\ \theta_c x \triangleright v \leftrightarrow fx \triangleright v$$

The proof is straightforwad by using that one can turn any function $f\colon \mathbb{N}\rightharpoonup\mathbb{B}$ into an oracle-computable F, such that $F\ (\lambda xv.\bot)$ agrees with f. ◀

5 The Kleene-Post Theorem

To establish incomparable Turing degrees, we adapt the proof given in Odifreddi's textbook [38] to our synthetic setting. Compared to the next sections, we here focus more on the intuition of the synthetic and constructive setting and omit some formal details. The usual strategy is to obtain said degrees as the unions $A := \bigcup_{n:\mathbb{N}} s_n$ and $B := \bigcup_{n:\mathbb{N}} t_n$ of cumulative increasing sequences s_n and t_n of boolean strings (interpreted implicitly as the relation arising from relating i to b for a string $b_0, \dots b_i \dots b_k$) such that the former take care that no χ_n induces a reduction $B \preceq_T A$ and the latter conversely rule out $A \preceq_T B$. Naturally, in our synthetic setting we are not able to define these sequences as functions $\mathbb{N}\to\mathbb{B}^*$, as this would force A and B decidable. Instead, we characterise both sequences simultaneously with an inductive predicate $\rightsquigarrow\colon \mathbb{N}\to\mathbb{B}^*\to\mathbb{B}^*\to\mathbb{P}$ such that $n \rightsquigarrow (s,t)$ represents s_n as s and t_n as t, by adding to the base case $0 \rightsquigarrow ([],[])$ the following inductive rules:

$$\frac{2n \rightsquigarrow (s,t) \qquad s' \sqsupseteq s \qquad \chi_n\, s' \,|t|\, b \qquad \forall u < s'.\, \neg\chi_n\, u \,|t|\, b}{2n+1 \rightsquigarrow (s', t +\!\!+ [\neg b])}\,\mathrm{E1} \qquad \frac{2n \rightsquigarrow (s,t) \qquad \neg(\exists s' b.\, s' \sqsupseteq s \wedge \chi_n\, s' \,|t|\, b)}{2n+1 \rightsquigarrow (s, t +\!\!+ [\mathsf{false}])}\,\mathrm{E2}$$

$$\frac{2n+1 \rightsquigarrow (s,t) \qquad t' \sqsupseteq t \qquad \chi_n\, t' \,|s|\, b \qquad \forall u < t'.\, \neg\chi_n\, u \,|s|\, b}{2n+2 \rightsquigarrow (s +\!\!+ [\neg b], t')}\,\mathrm{O1} \qquad \frac{2n+1 \rightsquigarrow (s,t) \qquad \neg(\exists t' b.\, t' \sqsupseteq t \wedge \chi_n\, t' \,|s|\, b)}{2n+2 \rightsquigarrow (s +\!\!+ [\mathsf{false}], t)}\,\mathrm{O2}$$

In every even step with $2n \rightsquigarrow (s,t)$ the sequences are extended such that χ_n applied to any prefix of A differs from any prefix of B at position $|t|$, either by flipping the result if χ_n already converges on some extension $s' \sqsupseteq s$ least with respect to some ordering $u < u'$ of strings (E1) or by setting a dummy value if χ_n diverges on all extensions (E2). Dually, in every odd step with $2n+1 \rightsquigarrow (s',t')$ it is taken care that χ_e applied to any prefix of B differs from any prefix of A.

We first show that the relation $n \rightsquigarrow (s,t)$ indeed captures a (classically total) cumulative increasing sequence of boolean strings:

Lemma 11. *The following properties of $n \rightsquigarrow (s,t)$ hold.*

1. *For every n there not not exist s and t with $n \rightsquigarrow (s,t)$.*
2. *If $n \rightsquigarrow (s,t)$ and $n' \rightsquigarrow (s',t')$ for $n \leq n'$, then $s \sqsubseteq s'$ and $t \sqsubseteq t'$.*
3. *If $2n \rightsquigarrow (s,t)$ then $n \leq |s|$ and $n \leq |t|$.*

Proof. We give a detailed proof of (1) since it relies on careful constructive reasoning, the proofs of (2) and (3) are by routine arguments. The proof of (1) is by induction on n, in the case of 0 we just choose $s := []$ and $t := []$ and conclude with $0 \rightsquigarrow ([],[])$. In the case of $n+1$, we assume that there are no s' and t' with $n+1 \rightsquigarrow (s',t')$ and, given that we then want to derive a contradiction, may use the inductive hypothesis to assume positively that there are s and t with $n \rightsquigarrow (s,t)$. Since we still derive a contradiction, we can perform a case analysis whether or not there are $s' \sqsupseteq s$ and b with $\chi_n\, s' \,|t|\, b$, given that $\neg\neg(P \vee \neg P)$ holds constructively for every $P : \mathbb{P}$. If not, we just set $s' := s$ and $t' := t +\!\!+ [\mathsf{false}]$ and conclude $n+1 \rightsquigarrow (s',t')$ with (E2). If so, given the negative goal we can actually find a least such s', given that

$$(\exists n.\, p\, n) \to \neg\neg\exists n.\, p\, n \wedge \forall n' < n.\, p\, n$$

holds constructively for every $p : \mathbb{N} \to \mathbb{P}$. So for the least s', we set $t' := t +\!\!+ [\neg b]$ and conclude $n+1 \rightsquigarrow (s',t')$ with (E1). ◀

We next formally define the incomparable degrees A and B by

$$A\, x := \exists n\, s\, t.\, n \rightsquigarrow (s,t) \wedge s_x = \mathsf{true} \quad B\, x := \exists n\, s\, t.\, n \rightsquigarrow (s,t) \wedge t_x = \mathsf{true}$$

where s_x denotes the x-th element in s and is false other otherwise and state the central lemma used to show $B \not\preceq_T A$, a dual version yields $A \not\preceq_T B$.

Lemma 12. *If $2n \rightsquigarrow (s,t)$ and $2n+1 \rightsquigarrow (s',t')$, then $\hat{B}\,|t|\,b$ implies $\neg\chi_n\, \hat{A}\,|t|\,b$.*

Proof. We analyse how $2n+1 \rightsquigarrow (s',t')$ could have been derived from $2n \rightsquigarrow (s,t)$.

- In the case (E1), we have that $t' = t +\!\!+ [\neg b']$ and $\chi_n\, s' \,|t|\, b'$ for s' being the least such extension of s. By the former and the assumption $\hat{B}\,|t|\,b$ we derive $b = \neg b'$ using $t' \subseteq \hat{B}$ and $t'_{|t|} = \neg b'$. But then the further assumption $\chi_n\, \hat{A}\,|t|\,\neg b'$ is in conflict with $\chi_n\, s' \,|t|\, b'$ via monotonicity of oracle computations [15, Lemma 41], using $s' \subseteq \hat{A}$.
- In the case (E2), we have that $t' = t +\!\!+ [\mathsf{false}]$ and there is no extension $s' \sqsupseteq s$ with $\chi_n\, s' \,|t|\, b'$. However, if we now assume that $\chi_n\, \hat{A}\,|t|\,b$, then by modulus-continuity of oracle computations [15, Lemma 1] there is a finite prefix of $\hat{A}$ determining the outcome of $\chi_n\, \hat{A}$, in fact there is N such that $\chi_n\, s_N \,|t|\, b$ for all $N \rightsquigarrow (s_N, t_N)$. Now given the negative goal, we can use (1) of Lemma 11 to actually obtain such s_N and t_N. Then by comparing $2n$ with N we either obtain $s \sqsubseteq s_N$ or $s_N \sqsubseteq s$ by (2) of Lemma 11, so in either case we find an extension $s' \sqsupseteq s$ with $\chi_n\, s' \,|t|\, b'$, in contradiction to the assumption. ◀

From this lemma the Kleene-Post theorem then follows immediately.

Theorem 13. *There are predicates A and B such that neither $A \preceq_T B$ nor $B \preceq_T A$.*

Proof. Suppose that $B \preceq_T A$, so $\chi_c\, \hat{A}\, x\, b \leftrightarrow \hat{B} x b$ for some c. Given that we have to derive a contradiction, we can argue classically enough to obtain $2n \rightsquigarrow (s,t)$, $2n+1 \rightsquigarrow (s',t')$, and $\hat{B}\,|t|\,b$. Then by Lemma 12 we obtain $\neg\chi_c\, \hat{A}\,|t|\,b$, contradicting $\chi_c\, \hat{A}\,|t|\,b \leftrightarrow \hat{B}|t|b$. That also $A \not\preceq_T B$ follows similarly. ◀

6 The Turing Jump

The Turing jump is the relativised equivalent to the self-halting problem: a number c is contained in the Turing jump of a predicate q if the c-th oracle machine halts on input c while given q as oracle. Formally, we define the Turing jump q' of a predicate $q\colon \mathbb{N}\to\mathbb{P}$ as

$$q'c := \Xi_c\, \hat{q}\, c\, \star\,.$$

Crucially, the jump is semi-decidable in the predicate, but its complement is not.

Lemma 14. *$\mathcal{S}_q(q')$ and $\neg\mathcal{S}_q(\overline{q'})$.*

Proof. Take $\lambda Rco.\ \Xi_c Rc$ for $\mathcal{S}_q(q')$. For $\neg\mathcal{S}_q(\overline{q'})$, let F be computable and $\forall x.\ \neg q'x \leftrightarrow F\hat{q}x\star$. By definition, $\forall x.\ \neg q'x \leftrightarrow \neg\Xi_x\hat{q}x$. Using Corollary 8 we have c such that $\forall x.\ F\hat{q}x\star \leftrightarrow \Xi_c\hat{q}x$ and thus in particular $\neg\Xi_c\hat{q}c \leftrightarrow \Xi_c\hat{q}c$ – a contradiction. ◀

We now prove two standard results: First, that the Turing jump of a predicate is strictly higher in the order of Turing reducibility than the predicate itself, and secondly, that semi-decidability of p in q can be expressed as many-one reducibility of p to q'. To do so, we define an alternative Turing jump q° given $q\colon \mathbb{N}\to\mathbb{P}$ corresponding to a relativised halting problem, rather than a relativised self-halting problem q', as $q^\circ\langle c, x\rangle := \Xi_c\, \hat{q}\, x$.

Lemma 15. *$q' \preceq_m q^\circ$ and $q^\circ \preceq_m q'$.*

Proof. The first is by $\lambda c.\langle c, c\rangle$. For the second, use Lemma 7 for $\tau_{\langle c,x\rangle} nl := \xi_c x l$. ◀

Lemma 16. *$q \preceq_m q^\circ$*

Proof. Note that $\lambda Rxo.\ R\,x\,\mathsf{true}$ is computable. Via Corollary 8 this means we have c with $\forall Rxo.\ \Xi_c Rx \leftrightarrow Rx\mathsf{true}$. Now $\lambda x.\ \langle c, x\rangle$ is the wanted many-one reduction. ◀

Lemma 17. *$q \preceq_\mathsf{T} q'$ and $q' \not\preceq_\mathsf{T} q$.*

Proof. The first part is straightforward with the last two lemmas. For the second part, if $q' \preceq_\mathsf{T} q$ we have with Lemma 4 that $\overline{q'}$ is semi-decidable in q, contradicting Lemma 14. ◀

This lemma summarises the mentioned fact Turing jumps are strictly higher in terms of Turing reducibility. We next establish the announced connection between relative semi-decidability, (many-one) reducibility, and Turing jumps.

Lemma 18. *If $\mathcal{S}_q(p)$, then $p \preceq_m q^\circ$.*

Proof. Let F be computable and $\forall x.\ px \leftrightarrow F\hat{q}x\star$. Using Corollary 8, we have c such that $\Xi_c\hat{q}x \leftrightarrow F\hat{q}x\star$. Now take $\lambda x.\ \langle c, x\rangle$ as reduction. ◀

Lemma 19. *If $p \preceq_m q'$, then $\mathcal{S}_q(p)$.*

Proof. Let f be the many-one reduction. It then suffices to prove that $\lambda x.\ q'(fx)$ is semi-decidable in q. Take $\lambda Rco.\ \Xi_{fc}\ R\ (fc)$. ◀

Corollary 20. *$\mathcal{S}_q(p)$ if and only if $p \preceq_m q'$.*

Corollary 21. *If $p \preceq_\mathsf{T} q$, then $p' \preceq_m q'$.*

We now define iterated Turing jumps from a predicate q:

$$q^{(0)} := q \qquad\qquad q^{(n+1)} := (q^{(n)})'$$

In textbooks, one usually uses the empty predicate $\emptyset$ as basis, since it is Turing equivalent to any other decidable predicate. In the context of many-one reductions, however not all decidable predicates are equivalent: only non-trivial ones are. To obtain a more uniform treatment that does not have to consider special cases for decidable predicates, we use as basis the predicate that has sole element 0:

$$\mathbb{0} := \lambda x.\ x = 0$$

We also remark that, working explicitly with the previously discussed axiom EPF would allow us to prove that $\mathcal{K}$ is many-one equivalent to $\mathbb{0}^{(1)}$.

7 The Arithmetical Hierarchy

The arithmetical hierarchy was developed independently by Kleene [30] and Mostowski [35], using models of computation to define the 0 level. In line with the rest of this paper, we here define a synthetic variant relying on type-theoretic functions. In Section 10 we discuss the connection to an alternative definition where the 0 level uses quantifier-free logical formulas of a formal first-order syntax. Informally, a predicate $p\colon \mathbb{N}^k \to \mathbb{P}$ is in Σ_n if

$$\forall v.\ pv \leftrightarrow \exists x_1 . \forall x_2 . \ldots . Q x_n .\ f([x_n, \ldots, x_2, x_1] +\!\!+ v) = \mathsf{true}$$

where Q_n is either $\exists$ or $\forall$ depending on n being odd or even, i.e. if p can be characterised by a first-order formula with n quantifier alternations before a boolean function application, starting with an existential quantification. The definition of Π_n is dual. Formally, we use mutually defined inductive predicates:

$$\frac{\forall v : \mathbb{N}^k.\ pv \leftrightarrow fv = \mathsf{true}}{\Sigma_0^k p} \qquad \frac{\Pi_n^{k+1} q \qquad \forall v : \mathbb{N}^k.\ pv \leftrightarrow \exists x.\ q(x :: v)}{\Sigma_{n+1}^k p}$$

$$\frac{\forall v : \mathbb{N}^k.\ pv \leftrightarrow fv = \mathsf{true}}{\Pi_0^k p} \qquad \frac{\Sigma_n^{k+1} q \qquad \forall v : \mathbb{N}^k.\ pv \leftrightarrow \forall x.\ q(x :: v)}{\Pi_{n+1}^k p}$$

One usually defines $\Delta_n^k p := \Sigma_n^k p \wedge \Pi_n^k p$, but we do not use this notion technically. From now on, we leave out the arity k since it is always clear from context. In the remainder of this section we collect closure properties that hold constructively, in the next section we continue with some non-constructive closure properties.

The hierarchy treats predicates extensionally:

Lemma 22. *Whenever* $\forall v.\ p_1 v \leftrightarrow p_2 v$*, then* $\Sigma_n p_1$ *implies* $\Sigma_n p_2$ *and* $\Pi_n p_1$ *implies* $\Pi_n p_2$*.*

The 1 level of the hierarchy can be characterised using semi-decidability:

Lemma 23. *A predicate* p *is semi-decidable if and only if* $\Sigma_1 p$*.*

Proof. From left to right, let p be semi-decidable, i.e. $\forall v : \mathbb{N}^k.\ pv \leftrightarrow fv \triangleright \star$. The result follows using Lemma 1 (**1**), because $fv \triangleright \star \leftrightarrow \exists n.\ \epsilon\, f\, n\, v \star = \mathsf{true}$. From right to left, we use unbounded search μ from Lemma 1 (**2**). ◀

By straightforward induction, we also obtain that the hierarchy is cumulative:

Lemma 24. *If $n \leq m$ then $\Sigma_n \subseteq \Sigma_m$ as well as $\Pi_n \subseteq \Pi_m$.*

Furthermore, it is closed under many-one reductions:

Lemma 25. *If $p_1 \preceq_m p_2$ then $\Sigma_n p_2$ implies $\Sigma_n p_1$ and $\Pi_n p_2$ implies $\Pi_n p_1$.*

Proof. By mutual induction. The base cases are immediate. We prove that if $p_1 \preceq_m p_2$ via f, $\forall v.\ p_2 v \leftrightarrow \exists x.\ q(x :: v)$, and $\Pi_n q$, then $\Sigma_{n+1} p_1$, the other case is similar.

We have that $\forall v.\ p_1 v \leftrightarrow \exists x.\ q(x :: fv)$. By the induction hypothesis, it thus suffices to prove $(\lambda(x :: v).\ q(x :: fv)) \preceq_m q$, which is trivial. ◀

Note that in this proof, we use the notation $\lambda(x :: v) \ldots$, which defines a function taking as argument a non-empty vector, i.e. an argument of type X^{n+1}. We now show further closure properties of the levels, crucially making use of Lemma 25 and pairing $\langle \cdot, \cdot \rangle$.

Lemma 26. *$\Sigma_n(\lambda v.\ \exists xy.\ p(x :: y :: v))$ if $\Pi_n p$, and $\Pi_n(\lambda v.\ \forall xy.\ q(x :: y :: v))$ if $\Sigma_n q$.*

Corollary 27. *$\Sigma_n p$ implies $\Sigma_n(\lambda v.\ \exists x.\ p(x :: v))$ and $\Pi_n q$ implies $\Pi_n(\lambda v.\ \forall x.\ q(x :: v))$.*

By induction we then have:

Lemma 28. *$\Sigma_n \subseteq \Pi_{n+1}$ and $\Pi_n \subseteq \Sigma_{n+1}$.*

Lemma 29. *If $\Sigma_n p_1$ and $\Sigma_n p_2$ then $\Sigma_n(\lambda v. p_1 v \wedge p_2 v)$. If $\Pi_n q_1$ and $\Pi_n q_2$ then $\Pi_n(\lambda v. q_1 v \wedge q_2 v)$.*

Proof. By mutual induction. The base cases are easy.

We just show one inductive case, the other one is dual. Let $\forall v.\ p_1 v \leftrightarrow \exists x.\ q_1'(x :: v)$, $\forall v.\ p_2 v \leftrightarrow \exists y.\ q_2'(y :: v)$, and $\Pi_n q_1'$ as well as $\Pi_n q_2'$, then $\Sigma_{n+1}(\lambda v.\ p_1 v \wedge p_2 v)$.

Note that we have that $\lambda v.\ p_1 v \wedge p_2 v \leftrightarrow \exists xy.\ q_1'(x :: v) \wedge q_2'(y :: v)$. Using Lemma 26 and the induction hypothesis, it suffices to prove that $\Pi_n \lambda(x :: y :: v).\ q_1'(x :: v)$ and $\Pi_n \lambda(x :: y :: v).\ q_2'(x :: v)$, which follows from $\Pi_n q_1'$, $\Pi_n q_2'$, and Lemma 25. ◀

The following two lemmas have similarly technical proofs, we omit the details.

Lemma 30. *Let $f : \mathbb{N} \to \mathbb{B}$. If $\Sigma_n p_1$ and $\Sigma_n p_2$, then $\Sigma_n(\lambda(x :: v).$ **if** fx **then** $p_1 v$ **else** $p_2 v)$. If $\Pi_n q_1$ and $\Pi_n q_2$ then $\Pi_n(\lambda(x :: v).$ **if** fx **then** $q_1 v$ **else** $q_2 v)$.*

Lemma 31. *If $\Sigma_n p$, then $\Sigma_n(\lambda(N :: v).\ \forall x < N.\ p(x :: v))$. Moreover, if $\Pi_n q$, then $\Pi_n(\lambda(N :: v).\ \forall x < N.\ q(x :: v))$.*

We prove closure under disjunction for Σ_n now:

Lemma 32. *If $\Sigma_n p_1$ and $\Sigma_n p_2$ then $\Sigma_n(\lambda v. p_1 v \vee p_2 v)$.*

Proof. With $p_1 v \vee p_2 v \leftrightarrow \exists n.$ **if** $n = 0$ **then** $p_1 v$ **else** $p_2 v$ from Lemma 30 and corollary 27. ◀

The proof for Π_n requires classical logic. We thus only give it in the next section, where we introduce a fine-grained characterisation of classical logic for the arithmetical hierarchy, allowing to state a stronger theorem for closure of Π_n under disjunction.

8 An Arithmetical Hierarchy of Classical Axioms

The most common axiom to enable classical logical reasoning in constructive foundations is the law of excluded middle (LEM), or, equivalently, double negation elimination (DNE). Both LEM and DNE can be weakened to apply to certain propositions only. If restricted to apply to Σ_1 propositions only, one obtains the limited principles of omniscience (LPO) and Markov's principle (MP), respectively.

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

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

LPO implies MP, but the converse is well-known to not be provable [9].

Akama, Berardi, Hayashi, and Kohlenbach [1] introduce relativisations of the law of excluded middle and double negation elimination to the arithmetical hierarchy, and prove that they are in relation as displayed in Figure 1. We prove these implications formally in CIC and using Coq, with the following definitions:

$$\Sigma_n\text{-LEM} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Sigma_n p \to \forall v.pv \vee \neg pv \qquad \Sigma_n\text{-DNE} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Sigma_n p \to \forall v.\neg\neg pv \to pv$$

$$\Pi_n\text{-LEM} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Pi_n p \to \forall v.pv \vee \neg pv \qquad \Pi_n\text{-DNE} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Pi_n p \to \forall v.\neg\neg pv \to pv$$

On the 0 and 1 levels these axioms have well-known connections to LPO and LEM:

Lemma 33. *The following hold*

1. Σ_0-LEM *holds constructively, and thus all* 0 *levels of the axioms,*
2. Σ_1-LEM $\leftrightarrow$ LPO*,*
3. Σ_1-DNE $\leftrightarrow$ MP*,*
4. Π_1-LEM $\leftrightarrow$ WLPO *with* $\mathsf{WLPO} := \forall f{:}\,\mathbb{N}{\to}\mathbb{B}.\ (\forall n.\ fn = \mathsf{false}) \vee \neg(\forall n.\ fn = \mathsf{false})$*,*
5. Π_1-DNE *holds constructively.*

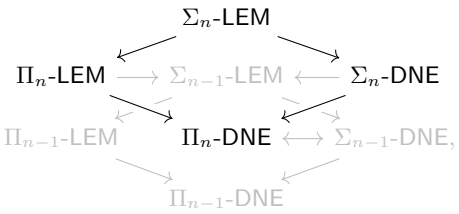


Figure 1 Arithmetical hierarchy of the law of excluded middle and related principles [1].

To prove the implications from Figure 1, we need that the arithmetical hierarchy is closed under complements and that Π_n is closed under disjunction (which holds constructively for Σ_n, see Lemma 32). We begin with the closures under complement:

Lemma 34. *If* $\Sigma_n p$ *and* Π_n-DNE*, then* $\Pi_n \overline{p}$*, and if* $\Pi_n p$ *and* Σ_n-DNE*, then* $\Sigma_n \overline{p}$*.*

Proof. By mutual induction. The base cases hold constructively.

For the first inductive case, we prove that if $\forall v.\ pv \leftrightarrow \exists x.\ p'(x :: v)$, $\Sigma_n p'$, and Σ_{n+1}-DNE, then $\Sigma_{n+1}\overline{p}$. By the inductive hypothesis we have $\Pi_n \overline{p}$ provided Π_n-DNE, which is a consequence of Σ_{n+1}-DNE. It thus suffices to prove that $\forall v.\ \neg pv \leftrightarrow \forall x.\ \neg p'(x :: v)$, which holds constructively.

For the other case, we prove that if $\forall v.\ pv \leftrightarrow \forall x.\ p'(x :: v)$, $\Sigma_n p'$, and Σ_{n+1}-DNE, then $\Sigma_{n+1}\overline{p}$. By the induction hypothesis we have $\Pi_n\overline{p}$ provided Π_n-DNE, which is a consequence of Σ_{n+1}-DNE. It thus suffices to prove that $\forall v.\ \neg pv \leftrightarrow \exists x.\ \neg p'(x :: v)$. The direction from right to left holds constructively. For the direction from left to right, assume $\neg pv$ and prove $\exists x.\ \neg p(x :: v)$. Since by the inductive hypothesis we have that $\Sigma_{n+1}(\lambda v.\ \exists x.\ \neg p'(x :: v)$ we can use Σ_{n+1}-DNE and it suffices to prove $\neg\neg\exists x.\ \neg p(x :: v)$, which follows constructively from $\neg pv$. ◀

It is straightforward to prove that Π_n is closed under disjunction assuming Π_n-LEM:

Lemma 35. *If* Π_n-LEM, $\Pi_n p$, *and* $\Pi_n q$ *then* $\Pi_n(\lambda v. pv \vee qv)$.

We now prove the implications from Figure 1.

Lemma 36.
1. Σ_n-LEM $\to$ Σ_n-DNE
2. Π_n-LEM $\to$ Π_n-DNE
3. Σ_n-DNE $\leftrightarrow$ Π_{n+1}-DNE
4. Π_{n+1}-LEM $\to$ Σ_n-LEM
5. Σ_n-LEM $\to$ Π_n-LEM
6. Σ_{n+1}-DNE $\to$ Σ_n-LEM

Proof. (1) and (2) are immediate, because in general $P \vee \neg P \to \neg\neg P \to P$. (3) is by induction. (4) follows by $\Sigma_n \subseteq \Pi_{n+1}$.

(5) has a more interesting proof: Assume Σ_n-LEM. By the previous implications, we can then use Σ_n-DNE and Π_n-DNE. Let $\Pi_n p$ and $v : \mathbb{N}^k$. We have to prove $pv \vee \neg pv$. By Lemma 34 and Σ_n-DNE, we have $\Sigma_n\overline{p}$. Using Σ_n-LEM we have $\neg pv \vee \neg\neg pv$. Using Π_n-DNE, we have $pv \vee \neg pv$.

For (6), assume Σ_{n+1}-DNE and $\Sigma_n p$. We prove $pv \vee \neg pv$ by applying Σ_{n+1}-DNE. Because Σ_{n+1} is closed under disjunction by Lemma 32, it suffices to prove that p is in Σ_{n+1}, which is trivial, and that $\overline{p}$ is, which follows from Lemma 34. ◀

Note that the converses of the implications are not provable: For (1), MP does not imply LPO (see e.g. [21]). For (2), Π_1-LEM is WLPO, which is not provable, and Π_1-DNE is provable. For (4), since Σ_0-LEM is provable, but Π_1-LEM is WLPO. For (5), since (at level 1) WLPO is strictly weaker than LPO [21]. We furthermore cannot prove that Σ_1-DNE implies Π_1-LEM, because MP does not imply WLPO. For (6), since Σ_0-LEM is provable, but Σ_1-DNE is MP.

It seems that for both closure properties we proved the assumptions are stronger than necessary. To prove closure under disjunction, it would suffice to assume DNE for disjunctions where both sides of the disjunct are Π_n formulas [1], but we avoid introducing this axiom.

We conjecture that for the purpose of proving that the arithmetical hierarchy is closed under negation, Σ_n-DNE is also strictly stronger than necessary. This is because at level 1, the theorem is equivalent to an axiom which seems to be weaker than MP (i.e. Σ_1-DNE).

Lemma 37. $\Pi_1 p \to \Sigma_1\overline{p}$ *iff* $\forall f : \mathbb{N}\to\mathbb{B}.\exists g : \mathbb{N}\to\mathbb{B}.\ \neg\neg(\exists n.\ fn = \mathsf{true}) \leftrightarrow (\exists n.\ gn = \mathsf{true})$.

This principle can be seen as an "anonymised" Markov's principle. It is an obvious consequence of MP (with $g := f$), but it seems to be strictly weaker. We are not aware of this axiom appearing in the literature, and we conjecture it to be non-provable. It can also be seen as the level 1 instance of closure under double negation for the hierarchy.

We say that the arithmetical hierarchy is closed under double negations at level n if $\Sigma_n p$, then $\Sigma_n\overline{\overline{p}}$, and if $\Pi_n p$, then $\Pi_n\overline{\overline{p}}$. And, as before, it is closed under complements at level n if $\Sigma_n p$, then $\Pi_n\overline{p}$, and if $\Pi_n p$, then $\Sigma_n\overline{p}$.

Lemma 38. *If the arithmetical hierarchy is closed under complements at level n, it is closed under double negations at level n.*

For the converse, we need to assume that the hierarchy is closed under double negations for all levels $m \leq n$, because closure at level $n+1$ seems not to imply closure at level n. Furthermore, the proof seems to require Π_n-DNE – a principle strictly weaker than Σ_n-DNE which we used to prove closure under negation.

Lemma 39. *Given Π_n-DNE, if the arithmetical hierarchy is closed under double negation at all levels $m \leq n$, it is closed under complements at level n.*

We conjecture that above level 1, this is not an equivalence, i.e. that some axiom potentially weaker than Π_n-DNE is needed.

9 Post's Theorem

By composition of previous results, we now derive the statements comprising Post's theorem. We first explicitly capture the connection of relative semi-decidability and the arithmetical hierarchy in the next two lemmas.

Lemma 40. *Assume Π_n-LEM. If $\Sigma_{n+1}p$, then p is semi-decidable relative to some q in Π_n.*

Proof. Let Σ_{n+1}, i.e. there is q in Π_n such that $pv \leftrightarrow \exists x.q(x :: v)$. We show that p is semi-decidable in q by linearly searching for x, i.e. pick

$$FRvo := \exists x.R\ (x :: v)\ \mathsf{true} \wedge \forall x' < x.R\ (x' :: v)\ \mathsf{false}$$

which is oracle-computable by Lemma 2 (**9**). We need to prove that $(\exists x.q(x :: v)) \leftrightarrow F\hat{q}v\star$. The direction from right to left is immediate. For the direction from left to right we have to prove that given x with $q(x :: v)$, i.e. $\hat{q}(x :: v)\mathsf{true}$, there is a *least* x such that $\hat{q}(x :: v)\mathsf{true}$. This follows from Π_n-LEM and $\Pi_n q$. ◀

Lemma 41. *Assume Σ_n-DNE. If p is semi-decidable relative to some q in Π_n, then $\Sigma_{n+1}p$.*

Proof. Let $\forall v.\ pv \leftrightarrow F\hat{q}v\star$ for an oracle-computable F. This means we have τ such that

$$\forall v.\ pv \leftrightarrow \exists qs\ as.\ \tau v\ ;\hat{q} \vdash\ qs\ ;\ as \wedge \tau\ v\ as \triangleright \mathsf{out}\ \star\ .$$

Note that $\lambda(as :: v).\ \tau\ v\ as \triangleright \mathsf{out}\ \star$ is in $\Sigma_1 \subseteq \Sigma_{n+1}$, because it is trivially semi-decidable, which makes Lemma 23 applicable. Using Corollary 27, it then suffices to prove that $\lambda(qs :: as :: v).\ \tau v\ ;\hat{q} \vdash\ qs\ ;\ as$ is in Σ_{n+1}.

To do so, we prove that

$$\tau v\ ;\hat{q} \vdash\ qs\ ;\ as \leftrightarrow \forall n < |as|.\ \tau\ v\ (as\ \lfloor_n) \triangleright \mathsf{ask}\ qs_n \wedge \hat{q}(qs_n)(as_n)$$

where $as\ \lfloor_n$ is the list with the first n elements of as, and qs_n and as_n are the n-th element of qs and as, respectively. The direction from left to right is by induction on the interrogation, from right to left by induction on as.

Using Lemma 31 and once again that $\triangleright$ is semi-decidable, it suffices to prove that $\lambda(y :: b :: [\,]).\ \hat{q}yb$ is in Σ_{n+1}. Using Lemma 30, we have to prove that both q and its complement are in Σ_{n+1}. The former holds because q is in Π_n and Lemma 28. The latter holds because q is in Π_n, and thus its complement is in Σ_n using Σ_n-DNE and Lemma 34, and thus in Σ_{n+1} using Lemma 24. ◀

Incidentally, apart from the 0 case, the inductive structure of all remaining proofs for Theorem 43 is unchanged as long as the following holds:

Lemma 42. $\Sigma_0\mathbb{0}$

Then finally, Post's theorem can be stated as follows:

Theorem 43. *The following hold assuming* Σ_n-LEM*:*

1. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for some* q *in* Π_n*,*
2. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for some* q *in* Σ_n*,*
3. $\Sigma_n\mathbb{0}^{(n)}$*,*
4. *if* $\Sigma_n p$ *then* $p \preceq_m \mathbb{0}^{(n)}$*, so in particular* $p \preceq_\mathsf{T} \mathbb{0}^{(n)}$*,*
5. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_{\mathbb{0}^{(n)}}(p)$*.*

Note that, in light of Lemma 42, by further relativising the 0 level of the arithmetical hierarchy to relative decidability one can obtain a relativised form of Post's theorem from an arbitrary base predicate, that does not even necessarily have to be arithmetical.

10 The Syntactic Arithmetical Hierarchy

We have defined the arithmetical hierarchy in a synthetic way, by defining the 0 level using boolean functions rather than predicates decidable in a model of computation. Another natural definition is syntactic, purely in terms of first-order logic, where the 0 level is characterised by quantifier-free formulas, and where the Δ_1 class can alternatively be characterised by predicates that are decidable in a model of computation.

We define this syntactic arithmetical hierarchy now, show that it is included in the previously defined synthetic arithmetical hierarchy, and show that the converse inclusion is equivalent to the well-known constructive axiom CT, a strengthening of the axiom EPF.

As a basis, we employ the Coq library of first-order logic [27, 25] and recall the basic framework. We use a de-Bruijn representation of first-order formulas φ over the term language of arithmetic (i.e. with constant 0, unary successor function S, and binary operation symbols for addition and multiplication). We use a type of variables V on paper, which we implement using de Bruijn indices as $V := \mathbb{N}$ in Coq.

$$t : \mathsf{term} ::= x \mid n \mid t_1 \mathbin{\dot{+}} t_2 \mid t_1 \mathbin{\dot{\times}} t_2 \qquad x{:}\,V, n{:}\,\mathbb{N}$$

$$\varphi : \mathsf{form} ::= \dot{\bot} \mid t_1 \doteq t_2 \mid \varphi_1 \mathbin{\dot{\wedge}} \varphi_2 \mid \varphi_1 \mathbin{\dot{\vee}} \varphi_2 \mid \varphi_1 \mathbin{\dot{\to}} \varphi_2 \mid \dot{\forall}\varphi \mid \dot{\exists}\varphi$$

A formula is quantifier-free if it does not use the constructors $\dot{\exists}$ and $\dot{\forall}$.

We furthermore use a Tarski-style satisfaction predicate $\rho \vDash \varphi$ for $\rho < : V \to \mathbb{N}$ being an assignment from de Bruijn indices to natural numbers which maps the terms to their respective interpretation of natural numbers, and formulas to their respective interpretations in the meta-logic.

$$[\![x]\!]_\rho := \rho x \qquad [\![n]\!]_\rho := n \qquad [\![t_1 \mathbin{\dot{+}} t_2]\!]_\rho := [\![t_1]\!]_\rho + [\![t_2]\!]_\rho \qquad [\![t_1 \mathbin{\dot{\times}} t_2]\!]_\rho := [\![t_1]\!]_\rho \cdot [\![t_2]\!]_\rho$$

$$\rho \vDash \dot{\bot} := \bot \qquad \rho \vDash t_1 \doteq t_2 := [\![t_1]\!]_\rho = [\![t_2]\!]_\rho \qquad \rho \vDash \varphi_1 \mathbin{\dot{\wedge}} \varphi_2 := (\rho \vDash \varphi_1) \wedge (\rho \vDash \varphi_2)$$

$$\rho \vDash \varphi_1 \mathbin{\dot{\vee}} \varphi_2 := (\rho \vDash \varphi_1) \wedge (\rho \vDash \varphi_2) \qquad \rho \vDash \varphi_1 \mathbin{\dot{\to}} \varphi_2 := (\rho \vDash \varphi_1) \to (\rho \vDash \varphi_2)$$

$$\rho \vDash \dot{\forall}\varphi := \forall d.(d;\rho) \vDash \varphi \qquad \rho \vDash \dot{\exists}\varphi := \exists d.(d;\rho) \vDash \varphi$$

We then define when a formula φ is $\dot{\mathbf{\Sigma}}_n$ or respectively $\dot{\mathbf{\Pi}}_n$.[2]

$$\frac{\varphi \text{ is quantifier-free}}{\dot{\mathbf{\Sigma}}_n \varphi} \qquad \frac{\dot{\mathbf{\Pi}}_n \varphi}{\dot{\mathbf{\Sigma}}_{n+1} \dot{\exists} \varphi} \qquad \frac{\dot{\mathbf{\Sigma}}_{n+1} \varphi}{\dot{\mathbf{\Sigma}}_{n+1} \dot{\exists} \varphi}$$

$$\frac{\varphi \text{ is quantifier-free}}{\dot{\mathbf{\Pi}}_n \varphi} \qquad \frac{\dot{\mathbf{\Sigma}}_n \varphi}{\dot{\mathbf{\Pi}}_{n+1} \dot{\forall} \varphi} \qquad \frac{\dot{\mathbf{\Pi}}_{n+1} \varphi}{\dot{\mathbf{\Pi}}_{n+1} \dot{\forall} \varphi}$$

Based on this, we define

$$\dot{\mathbf{\Sigma}}_n^k p := \exists \varphi.\ \dot{\mathbf{\Sigma}}_n \varphi \land \forall v{:}\,\mathbb{N}^k.\ pv \leftrightarrow \rho_v \vDash \varphi \qquad \dot{\mathbf{\Pi}}_n^k p := \exists \varphi.\ \dot{\mathbf{\Pi}}_n \varphi \land \forall v{:}\,\mathbb{N}^k.\ pv \leftrightarrow \rho_v \vDash \varphi$$

where ρ_v is a function mapping i to the i-th value in v (and 0 if v is not long enough). As before, we leave out k from here on.

Lemma 44. *Satisfaction of quantifier-free formulas is decidable.*

Lemma 45. *The syntactic arithmetical hierarchy is included in the synthetic arithmetical hierarchy.*

Proof. We prove by mutual induction that

$$(\dot{\mathbf{\Sigma}}_n \varphi \to \Sigma_n(\lambda v.\ \rho_v \vDash \varphi)) \land (\dot{\mathbf{\Pi}}_n \varphi \to \Pi_n(\lambda v.\ \rho_v \vDash \varphi)).$$

The base cases follows by the last lemma. The cases of adding $\dot{\exists}$ to a $\dot{\mathbf{\Pi}}_n$ formula, or $\dot{\forall}$ to a $\dot{\mathbf{\Sigma}}_n$ formula follow by definition of Σ and Π. The cases of adding $\dot{\exists}$ to a $\dot{\mathbf{\Sigma}}_n$ formula, or $\dot{\forall}$ to a $\dot{\mathbf{\Pi}}_n$ formula follow by Corollary 27. ◀

The reverse direction is not provable without axioms. In fact, it is equivalent to the axiom Church's thesis (CT), stating that all functions are computable in a model of computation. The axiom EPF we have used is a direct consequence of CT [11].

Let $\phi_c^n x$ be the execution of the c-th μ-recursive function according to an enumeration, on input x for n steps.

$$\mathsf{CT} := \forall f{:}\,\mathbb{N}{\to}\mathbb{N}.\exists c{:}\,\mathbb{N}.\ \forall x.\exists n.\ \phi_c^n x = 1 + (fx)$$

Lemma 46. *Assuming* CT, $\dot{\mathbf{\Sigma}}_1(\lambda v.\ fv = \mathsf{true})$.

Proof. See Hermes and Kirst [22] and Kirst and Peters [28]. ◀

Lemma 47. *Assuming* CT, $\dot{\mathbf{\Pi}}_1(\lambda v.\ gv = \mathsf{true})$.

Proof. Let g be given. Using the last lemma with $fv := \neg_{\mathbb{B}} gv$ we have that $\dot{\mathbf{\Sigma}}_1(\lambda v.\neg_{\mathbb{B}} gv = \mathsf{true})$. The claim follows by proving that $\dot{\mathbf{\Sigma}}_1 p \to \dot{\mathbf{\Pi}}_n(\lambda v.\ \neg pv)$ in general, which holds constructively without axioms. ◀

Corollary 48. *Assuming* CT, $\Sigma_1 \subseteq \dot{\mathbf{\Sigma}}_1$ *and* $\Pi_1 \subseteq \dot{\mathbf{\Pi}}_1$.

[2] We use boldface fonts with a dot here to distinguish from the synthetic arithmetical hierarchy as clearly as possible. Note that there is no connection to the so-called boldface and lightface pointclasses from descriptive set theory, which are also based on hierarchies of formulas classified via quantifier alternations.

Note that for $n = 0$, the hiearchies are not equivalent, because not every decidable predicate can be presented as quantifier-free formula[3] This however does not affect Post's theorem, since it only talks about $n > 0$.

Lemma 49. *Assuming* CT*, the synthetic arithmetical hierarchy is included in the syntactic arithmetical hierarchy on every level* $n > 0$*.*

Proof. By induction, using the last lemma. ◀

To avoid reintroducing classical axioms for the syntactic arithmetical hierarchy, we re-state Post's theorem by using classical logic in general:

▶ **Theorem 50.** *The following hold assuming the law of excluded middle and* CT*:*

1. $\dot{\mathbf{\Sigma}}_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for a* q *in* $\dot{\mathbf{\Pi}}_n$*,*
2. $\dot{\mathbf{\Sigma}}_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for a* q *in* $\dot{\mathbf{\Sigma}}_n$*,*
3. $\dot{\mathbf{\Sigma}}_n\mathbb{0}^{(n)}$*,*
4. *if* $\dot{\mathbf{\Sigma}}_n p$ *then* $p \preceq_m \mathbb{0}^{(n)}$*, so in particular* $p \preceq_{\mathsf{T}} \mathbb{0}^{(n)}$*,*
5. $\dot{\mathbf{\Sigma}}_{n+1}p$ *if and only if* $\mathcal{S}_{\mathbb{0}^{(n)}}(p)$*.*

▶ **Lemma 51.** *If the synthetic arithmetical hierarchy is included in the syntactic arithmetical hierarchy on level* $1 > 0$*, then* CT *holds.*

Proof. Assume $\Sigma_1 \subseteq \dot{\mathbf{\Sigma}}_1$. The axiom that semi-decidable predicates are semi-decided by a μ-recursive function is equivalent to CT [12]. Let $p\colon \mathbb{N}\to\mathbb{P}$ be semi-decidable. Then it is in Σ_1. By assumption, it then is in $\dot{\mathbf{\Sigma}}_1$. Thus, it can be semi-decided by a μ-recursive function. ◀

Lemma 52. *Given* LEM*, whenever a predicate* p *is definable via a first-order formula* φ *one can compute* n *and either a proof of* $\dot{\mathbf{\Sigma}}_n p$ *or* $\dot{\mathbf{\Pi}}_n p$*.*

Note that the assumption of LEM could be weakened to $\dot{\mathbf{\Sigma}}_n$-LEM [20, 19].

11 Conclusion

In this paper, we use the definition of oracle computability of [15] mostly abstractly, i.e. through the closure properties in Lemma 2 as interface. While the invariants for the constructions underlying these properties are intricate and often tedious, utilising the properties themselves is straightforward. The only explicit uses of the underlying computation trees are Lemma 41 and the construction of the enumerators Ξ and χ from EPF. The proof of Post's theorem can then be seen as a sanity check for our synthetic definition of Turing reducibility: It agrees with analytic Turing reducibility on arithmetical predicates.

The Coq mechanisation, contributing roughly 3k lines of code (2k for the Kleene-Post and Post's theorem, 1k for the syntactic arithmetical hierarchy) to the Coq library of synthetic computability, has proven beneficial besides the goals of formalising the foundations of theoretical computer science and a constructive reverse analysis: Having a mechanisation allows for quickly changing definitions without the need for extensive manual re-checking. We have utilised this e.g. when changing the base of the hierarchy to $\mathbb{0}$ instead of $\emptyset$, which was almost automatic. Furthermore, identifying and tracking classical assumptions manually would be cumbersome, and is almost for free using a proof assistant.

[3] The situation would be different if we would allow for bounded quantification on the 0 level of the hierarchy, but we syntactically disallow any quantification to appear.

Bauer [4] and Swan [47] suggest definitions of oracle computability and Turing reducibility in unpublished work. Our definition of oracle computability is stronger than Bauer's [15, Lemmas 2 and 42]. However, the resulting notion of Turing reducibility might still be the same – it hinges on the constructability of an enumerator for Bauer's notion. Additionally, our definition can be adapted to Swan's setting in univalent type theory, potentially implying his definition of Turing computability. Vice versa we expect the implication to be unprovable because it would require a form of double negation elimination stronger than MP, which seemingly implies LPO and thus is inconsistent in univalent synthetic computability. Swan's definition cannot be exactly reproduced in our setting, since it requires higher inductive types. A version of CIC with higher inductives but without univalence could be suitable.

For future work, it first would be interesting to frame more results from the 1954 Kleene-Post paper, e.g. that there are countably many incomparable Turing degrees with the order structure of the rationals, in our setting of synthetic computability. Secondly, we would like to carry out the full analysis of what classical axioms are implied by, and thus equivalent to, Post's theorem. Lastly, it would be intriguing to give a synthetic solution to Post's problem [40], e.g. via the Friedberg-Muchnik theorem and the priority method [18, 36].

References

1 Yohji Akama, Stefano Berardi, Susumu Hayashi, and Ulrich Kohlenbach. An arithmetical hierarchy of the law of excluded middle and related principles. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 192–201. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319613.

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

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

4 Andrej Bauer. Synthetic mathematics with an excursion into computability theory (slide set). *University of Wisconsin Logic seminar*, 2020. URL: http://math.andrej.com/asset/data/madison-synthetic-computability-talk.pdf.

5 Vasco Brattka, Matthew Hendtlass, and Alexander P. Kreuzer. On the uniform computational content of computability theory. *Theory of Computing Systems*, 61(4):1376–1426, August 2017. doi:10.1007/s00224-017-9798-1.

6 S Barry Cooper. *Computability theory*. CRC Press, 2003. doi:10.1201/9781315275789.

7 Thierry Coquand. Metamathematical investigations of a calculus of constructions. Technical Report RR-1088, INRIA, 1989. URL: https://hal.inria.fr/inria-00075471.

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

9 Hannes Diener. Constructive Reverse Mathematics. *arXiv:1804.05495 [math]*, 2020. URL: https://arxiv.org/abs/1804.05495, arXiv:1804.05495.

10 Hannes Diener and Hajime Ishihara. Bishop-style constructive reverse mathematics. In *Theory and Applications of Computability*, pages 347–365. Springer International Publishing, 2021. doi:10.1007/978-3-030-59234-9_10.

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

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

13 Yannick Forster. Parametric Church's Thesis: Synthetic computability without choice. In *International Symposium on Logical Foundations of Computer Science*, pages 70–89. Springer, 2022. doi:10.1007/978-3-030-93100-1_6.

14 Yannick Forster and Felix Jahn. Constructive and Synthetic Reducibility Degrees: Post's Problem for Many-One and Truth-Table Reducibility in Coq. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2023/17482, doi:10.4230/LIPIcs.CSL.2023.21.

15 Yannick Forster, Dominik Kirst, and Niklas Mück. Oracle computability and turing reducibility in the calculus of inductive constructions, 2023. arXiv:2307.15543.

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

17 Yannick Forster, Fabian Kunze, and Nils Lauermann. Synthetic Kolmogorov Complexity in Coq. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving (ITP 2022)*, volume 237 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16721, doi:10.4230/LIPIcs.ITP.2022.12.

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

19 Makoto Fujiwara and Taishi Kurahashi. Prenex normal form theorems in semi-classical arithmetic. *The Journal of Symbolic Logic*, 86(3):1124–1153, 2021. doi:10.1017/jsl.2021.47.

20 Makoto Fujiwara and Taishi Kurahashi. Prenex normalization and the hierarchical classification of formulas, 2023. arXiv:2302.11808.

21 Matt Hendtlass and Robert Lubarsky. Separating fragments of WLEM, LPO, and MP. *The Journal of Symbolic Logic*, 81(4):1315–1343, 2016. doi:10.1017/jsl.2016.38.

22 Marc Hermes and Dominik Kirst. An analysis of Tennenbaum's theorem in constructive type theory. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSCD.2022.9.

23 J. Martin E. Hyland. The effective topos. In *The L. E. J. Brouwer Centenary Symposium, Proceedings of the Conference held in Noordwijkerhout*, pages 165–216. Elsevier, 1982. doi:10.1016/s0049-237x(09)70129-6.

24 Hajime Ishihara. Reverse mathematics in Bishop's constructive mathematics. *Philosophia Scientae*, (CS 6):43–59, September 2006. doi:10.4000/philosophiascientiae.406.

25 Dominik Kirst. *Mechanised Metamathematics: An Investigation of First-Order Logic and Set Theory in Constructive Type Theory*. PhD thesis, Saarland University, 2022. doi:10.22028/D291-39150.

26 Dominik Kirst, Yannick Forster, and Niklas Mück. Synthetic Versions of the Kleene-Post and Post's Theorem. 28th International Conference on Types for Proofs and Programs (TYPES 2022), 2022. URL: https://types22.inria.fr/files/2022/06/TYPES_2022_paper_65.pdf.

27 Dominik Kirst, Johannes Hostert, Andrej Dudenhefner, Yannick Forster, Marc Hermes, Mark Koch, Dominique Larchey-Wendling, Niklas Mück, Benjamin Peters, Gert Smolka, and Dominik Wehr. A Coq library for mechanised first-order logic. In *Coq Workshop*, 2022. URL: https://coq-workshop.gitlab.io/2022/abstracts/Coq2022-01-01-first-order-logic.pdf.

28 Dominik Kirst and Benjamin Peters. Gödel's theorem without tears - essential incompleteness in synthetic computability. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual*

Conference on Computer Science Logic, CSL 2023, February 13-16, 2023, Warsaw, Poland, volume 252 of *LIPIcs*, pages 30:1–30:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.30.

29 S. C. Kleene. Countable functionals. *Journal of Symbolic Logic*, 27(3):81–100, 1959. doi:10.2307/2964658.

30 Stephen C. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53(1):41–73, 1943. doi:10.1090/S0002-9947-1943-0007371-8.

31 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.

32 Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland Pub. Co., 1959.

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

34 Andrei A. Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375, 1954. doi:10.1007/978-94-017-3477-6.

35 Andrzej Mostowski. On definable sets of positive integers. *Fundamenta Mathematicae*, 34:81–112, 1947. doi:10.4064/fm-34-1-81-112.

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

37 Niklas Mück. *The Arithmetical Hierarchy, Oracle Computability, and Post's Theorem in Synthetic Computability*. Bachelor's thesis, Saarland University, 2022. URL: https://ps.uni-saarland.de/~mueck/bachelor.php.

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

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

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

41 Emil L. Post. Degrees of recursive unsolvability - preliminary report. In *Bulletin of the American Mathematical Society*, volume 54:7, pages 641–642. American Mathematical Society (AMS), 1948.

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

43 Hartley Rogers. Theory of recursive functions and effective computability. 1987.

44 Sam Sanders. Refining the taming of the reverse mathematics zoo. *Notre Dame Journal of Formal Logic*, 59(4), January 2018. doi:10.1215/00294527-2018-0015.

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

46 Andrew Swan and Taichi Uemura. On Church's thesis in cubical assemblies. *arXiv preprint arXiv:1905.03014*, 2019. URL: https://arxiv.org/abs/1905.03014.

47 Andrew W Swan. Oracle modalities. *Second International Conference on Homotopy Type Theory (HoTT 2023)*, 2023. URL: https://hott.github.io/HoTT-2023/abstracts/HoTT-2023_abstract_35.pdf.

48 The Coq Development Team. The Coq proof assistant, June 2023. doi:10.5281/zenodo.8161141.

49 Jaap van Oosten. Partial combinatory algebras of functions. *Notre Dame Journal of Formal Logic*, 52(4):431–448, 2011. doi:10.1215/00294527-1499381.