Church's Thesis and Related Axioms in Coq's Type Theory

Yannick Forster

Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany
forster@cs.uni-saarland.de

Abstract

"Church's thesis" (CT) as an axiom in constructive logic states that every total function of type $\mathbb{N} \to \mathbb{N}$ is computable, i.e. definable in a model of computation. CT is inconsistent both in classical mathematics and in Brouwer's intuitionism since it contradicts weak Kőnig's lemma and the fan theorem, respectively. Recently, CT was proved consistent for (univalent) constructive type theory.

Since neither weak Kőnig's lemma nor the fan theorem is a consequence of just logical axioms or just choice-like axioms assumed in constructive logic, it seems likely that CT is inconsistent only with a combination of classical logic and choice axioms. We study consequences of CT and its relation to several classes of axioms in Coq's type theory, a constructive type theory with a universe of propositions which proves neither classical logical axioms nor strong choice axioms.

We thereby provide a partial answer to the question as to which axioms may preserve computational intuitions inherent to type theory, and which certainly do not. The paper can also be read as a broad survey of axioms in type theory, with all results mechanised in the Coq proof assistant.

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

Keywords and phrases Church's thesis, constructive type theory, constructive reverse mathematics, synthetic computability theory, Coq

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.21

Supplementary Material `https://github.com/uds-psl/churchs-thesis-coq`

Acknowledgements I want to thank Gert Smolka, Andrej Dudenhefner, Dominik Kirst, and Dominique Larchey-Wendling for discussions and feedback on drafts of this paper. Special thanks go to the anonymous reviewers for their helpful ideas, constructive comments, and editorial suggestions.

1 Introduction

The intuition that the concept of a constructively defined function and a computable function can be identified is prevalent in intuitionistic logic since the advent of recursion theory and is maybe most natural in constructive type theory, where computation is primitive.

A formalisation of the intuition is the axiom CT ("Church's thesis"), stating that every function is computable, i.e. definable in a model of computation. CT is well-studied as part of Russian constructivism [34] and in the field of constructive reverse mathematics [11, 25].

CT allows proving results of recursion theory without extensive references to a model of computation, since one can reason with functions instead. While such synthetic developments of computability theory [1, 7, 37] can be carried out in principle without assuming any axioms [14], assuming CT allows stronger results: CT essentially provides a universal machine w.r.t. all functions in the logic, allowing to show the non-existence of certain deciding functions – whose existence is logically independent with no axioms present.

It is easy to see that CT is in conflict with traditional classical mathematics, since the law of excluded middle LEM together with a form of the axiom of countable choice $\mathsf{AC}_{\mathbb{N},\mathbb{N}}$ allows the definition of non-computable functions [46]. This observation can be sharpened in various ways: To define a non-computable function directly, the weak limited principle of omniscience WLPO and the countable unique choice axiom $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$ suffice. Alternatively,

29th EACSL Annual Conference on Computer Science Logic (CSL 2021).
Editors: Christel Baier and Jean Goubault-Larrecq; Article No. 21; pp. 21:1–21:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Kleene noticed that there is a decidable tree predicate with infinitely many nodes but no computable infinite path [28]. If functions and computable functions are identified via CT, a Kleene tree is in conflict with weak Kőnig's lemma WKL and with Brouwer's fan theorem.

It is however well-known that CT is consistent in Heyting arithmetic with Markov's principle MP [27] which given CT states that termination of computation is stable under double negation. Recently, Swan and Uemura [43] proved that CT is consistent in univalent type theory with propositional truncation and MP.

While predicative Martin-Löf type theory as formalisation of Bishop's constructive mathematics proves the full axiom of choice AC, univalent type theory usually only proves the axiom of unique choice AUC. But since $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$ suffices to show that LEM implies $\neg$CT, classical logic is incompatible with CT in both predicative and in univalent type theory.

In the (polymorphic) calculus of (cumulative) inductive constructions, a constructive type theory with a separate, impredicative universe of propositions as implemented by the proof assistant Coq [44], none of AC, AUC, and $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$ are provable. This is because large eliminations on existential quantifications are not allowed in general [35], meaning one can not recover a function in general from a proof of $\forall x.\exists y.\ Rxy$. However, choice axioms as well al LEM can be consistently assumed in Coq's type theory [47]. Furthermore, it seems likely that the consistency proof for CT in [43] can be adapted for Coq's type theory.

This puts Coq's type theory in a special position: Since to disprove CT one needs a (weak) classical logical axiom and a (weak) choice axiom, assuming just classical logical axioms or just choice axioms might be consistent with CT. This paper is intended to serve as a preliminary report towards this consistency question, approximating it by surveying results from intuitionistic logic and constructive reverse mathematics in constructive type theory with a separate universe of propositions, with a special focus on CT and other axioms based on notions from computability theory. Specifically, we discuss these propositional axioms:

- computational enumerability axioms (EA, EPF) and Kleene trees (KT) in Section 5
- extensionality axioms like functional extensionality (Fext), propositional extensionality (Pext), and proof irrelevance (PI) in Section 6
- classical logical axioms like the principle of excluded middle (LEM, WLEM), independence of premises (IP), and limited principles of omniscience (LPO, WLPO, LLPO) in Section 7
- axioms of Russian constructivism like Markov's principle (MP) in Section 8
- choice axioms like the axiom of choice (AC), countable choice (ACC, $\mathsf{AC}_{\mathbb{N},\mathbb{N}}$, $\mathsf{AC}_{\mathbb{N},\mathbb{B}}$), dependent choice (ADC), and unique choice (AUC, $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$) in Section 9
- axioms on trees like weak Kőnig's lemma (WKL) and the fan theorem (FAN) in Section 10
- axioms regarding continuity and Brouwerian principles (Homeo, Cont, WC-N) in Section 11

The following hyper-linked diagram displays provable implications and incompatible axioms.

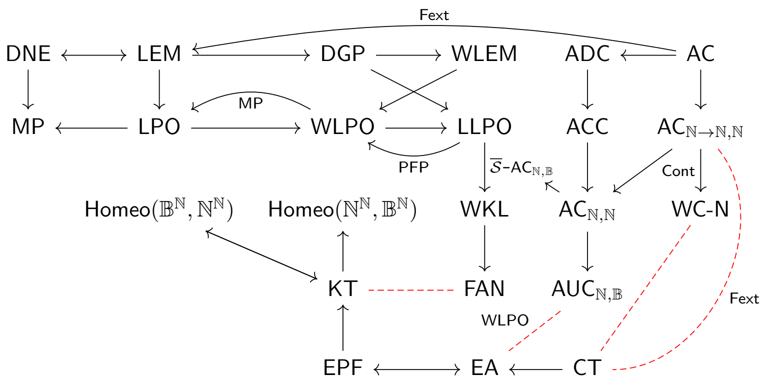

Figure 1 Overview of results. $\to$ are implications, - - - denotes incompatible axioms.

All results in this paper are mechanised in the Coq proof assistant and the proof scripts are accessible at `https://github.com/uds-psl/churchs-thesis-coq`. The statements in this document are hyperlinked to their Coq proof, indicated by a 🐓-symbol.

Outline. Section 2 establishes necessary preliminaries regarding Coq's type theory and introduces the notions of (synthetic) decidability, enumerability, and semi-decidability. Section 3 introduces CT formally, together with the related synthetic axioms EA and EPF. Section 4 contains undecidability proofs based on CT. Section 5 introduces decidable binary trees and constructs a Kleene tree. The connection of CT to the classes of axioms as listed above is surveyed in Sections 6 to 11. Section 12 contains concluding remarks.

2 Preliminaries

We work in the polymorphic calculus of cumulative inductive constructions as implemented by the Coq proof assistant [44], which we will refer to as "Coq's type theory". The calculus is a constructive type theory with a cumulative hierarchy of types $\mathbb{T}_i$ (where i is a natural number, but we leave out the index from now on), an impredicative universe of propositions $\mathbb{P} \subseteq \mathbb{T}$, and inductive types in every universe. The inductive types of interest in this paper are

$$n : \mathbb{N} ::= 0 \mid \mathsf{S}\, n \qquad\qquad b : \mathbb{B} ::= \mathsf{false} \mid \mathsf{true}$$

$$o : \mathbb{O}A ::= \mathsf{None} \mid \mathsf{Some}\, a \quad \textit{where } a : A \qquad\qquad l : \mathbb{L}A ::= [\,] \mid a :: l \quad \textit{where } a : A$$

$$A + B := \mathsf{inl}\, a \mid \mathsf{inr}\, b \quad \textit{where } a : A \textit{ and } b : B \qquad A \times B := (a, b) \quad \textit{where } a : A \textit{ and } b : B$$

One can easily construct a pairing function $\langle \_ , \_ \rangle : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ 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 write $n =_{\mathbb{B}} m$ for the boolean equality decider on $\mathbb{N}$, and $\neg_{\mathbb{B}}$ for boolean negation.

If $l : \mathbb{L}A$ then $l[n] : \mathbb{O}A$ denotes the n-th element of l. If $n < |l|$ we can assume $l[n] : A$.

We write $\forall x : X.\ Ax$ for both dependent functions and logical universal quantification, $\exists x : X.\ Ax$ where $A : X \to \mathbb{P}$ for existential quantification and $\Sigma x : X.\ Ax$ where $A : X \to \mathbb{T}$ for dependent pairs, with elements (x, y). Dependent pairs can be eliminated into arbitrary types, i.e. there is an elimination principle of type $\forall p : (\Sigma x.\ Ax) \to \mathbb{T}.\ (\forall (x : X)(y : Ax).\ p(x, y)) \to \forall (s : \Sigma x.\ Ax).\ ps$. We call such a principle eliminating a proposition into arbitrary types a *large elimination principle*, following the terminology "large elimination" for Coq's case analysis construct `match` [35]. Crucially, Coq's type theory proves a large elimination principle for the falsity proposition $\bot$, i.e. explosion applies to arbitrary types: $\forall A : \mathbb{T}.\ \bot \to A$. In contrast, existential quantification can only be eliminated for $p : (\exists x.\ Ax) \to \mathbb{P}$, but the following more specific large elimination principle is provable:

🐓 **Lemma 1.** *There is a guarded minimisation function $\mu_{\mathbb{N}}$ of the following type:*

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

There are various implementations of such a minimisation function in Coq's Standard Library.[1] One uses a (recursive) large elimination principle for the accessibility predicate, see e.g. [32, §2.7, §4.1, §4.2] and [6, §14.2.3, §15.4] for a contemporary overview how to implement large eliminations principles. We will not need any other large elimination principle in this paper. A restriction of large elimination in general is necessary for consistency of Coq [8]. As a by-product, the computational universe $\mathbb{T}$ is separated from the logical universe $\mathbb{P}$, allowing classical logic in $\mathbb{P}$ to be assumed while the computational intuitions for $\mathbb{T}$ remain intact.

[1] The idea was conceived independently by Benjamin Werner and Jean-François Monin in the 1990s.

$\mathsf{part}\,A : \mathbb{T}$	partial values over $A : \mathbb{T}$	
$\overset{!}{=} : \mathsf{part}\,A \to A \to \mathbb{P}$	definedness of values	$x \overset{!}{=} a_1 \to x \overset{!}{=} a_2 \to a_1 = a_2$
$(x : \mathsf{part}\,A)\downarrow : \mathbb{P}$		$x\downarrow := \exists a.\ x \overset{!}{=} a$
$\equiv_{\mathsf{part}\,A} : \mathsf{part}\,A \to \mathsf{part}\,A \to \mathbb{P}$	equivalence	$x \equiv_{\mathsf{part}\,A} y := (\forall a.\ x \overset{!}{=} a \leftrightarrow y \overset{!}{=} a)$
$\mathsf{ret} : A \to \mathsf{part}\,A$	monadic return	$\mathsf{ret}\,a \overset{!}{=} a$
$\mathsf{undef} : \mathsf{part}\,A$	undefined value	$\nexists a. \mathsf{undef} \overset{!}{=} a$
$\gg\!= : \mathsf{part}\,A \to (A \to \mathsf{part}\,B) \to \mathsf{part}\,B$	monadic bind	$x \gg\!= f \overset{!}{=} b \leftrightarrow (\exists a.\ x \overset{!}{=} a \land fa \overset{!}{=} b)$
$\mu : (\mathbb{N} \to \mathbb{B}) \to \mathsf{part}\,\mathbb{N}$	unbounded search	$\mu f \overset{!}{=} n \leftrightarrow fn = \mathsf{true} \land$ $\forall m < n.\ fm = \mathsf{false}$
$\mathsf{seval} : \mathsf{part}\,A \to \mathbb{N} \to \mathbb{O}A$	step-indexed evaluation	$x \overset{!}{=} a \leftrightarrow \exists n.\ \mathsf{seval}\,xn = \mathsf{Some}\,a$

Figure 2 A monad for partial values.

2.1 Partial Functions

All definable functions in type theory are total by definition. To model partiality, one often resorts to functional relations $R : A \to B \to \mathbb{P}$ or step-indexed functions $A \to \mathbb{N} \to \mathbb{O}B$, as for instance pioneered by Richman [37] in constructive logic, see e.g. [12] for a comprehensive overview.

For our purpose, we simply assume a type $\mathsf{part}\,A$ for $A : \mathbb{T}$ and a definedness relation $\overset{!}{=} : \mathsf{part}\,A \to A \to \mathbb{P}$ and write $A \nrightarrow B$ for $A \to \mathsf{part}\,B$. We assume monadic structure for part (ret and $\gg\!=$), an undefined value (undef), a minimisation operation (μ), and a step-indexed evaluator (seval). The operations and their specifications are listed in Figure 2.

2.2 Equivalence relations on functions

Besides intensional equality ($=$), we will consider other more extensional equivalence relations in this paper. For instance, extensional equality of functions f, g ($\forall x.\ fx = gx$), extensional equivalence of predicates p, q ($\forall x.\ px \leftrightarrow qx$), or range equivalence of functions f, g ($\forall x.\ (\exists y.\ fy = x) \leftrightarrow (\exists y.\ gy = x)$). We will denote all of these equivalence relations with the symbol $\equiv$ and indicate what is meant by an index. For discrete X (e.g. $\mathbb{N}$, $\mathbb{O}\mathbb{N}$, $\mathbb{L}\mathbb{B}$, $\dots$), $\equiv_X$ denotes equality, $\equiv_{\mathbb{P}}$ denotes logical equivalence, $\equiv_{A \to B}$ denotes an extensional lift of $\equiv_B$, $\equiv_{A \to \mathbb{P}}$ denotes extensional equivalence, and $\equiv_{\mathsf{ran}}$ denotes range equivalence.

Assuming the existence of surjections $A \to (A \to B)$ may or may not be consistent, depending on the particular equivalence relation. We introduce the notion of *surjection w.r.t.* $\equiv_B$ as $\forall b : B.\ \exists a : A. fa \equiv_B b$. We call a function $f : A \to B$ an *injection w.r.t.* $\equiv_A$ *and* $\equiv_B$ if $\forall a_1 a_2.\ fa_1 \equiv_B fa_2 \to a_1 \equiv_A a_2$ and a *bijection* if it is an injection and surjection.

One formulation of Cantor's theorem is that there is no surjection $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ w.r.t. $=$. However, the same proof can be used for the following strengthening of Cantor's theorem:

Fact 2 (Cantor)**.** *There is no surjection* $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ *w.r.t.* $\equiv_{\mathbb{N} \to \mathbb{N}}$.

2.3 Decidability, Semi-decidability, Enumerability, Reducibility

We define decidability, (co-)semi-decidability, and enumerability for predicates $p : X \to \mathbb{P}$:

$$\mathcal{D}p := \exists f : X \to \mathbb{B}.\quad \forall x.\ px \leftrightarrow fx = \mathsf{true} \qquad (\text{“}p\text{ is decidable”})$$
$$\mathcal{S}p := \exists f : X \to \mathbb{N} \to \mathbb{B}.\quad \forall x.\ px \leftrightarrow \exists n. fxn = \mathsf{true} \qquad (\text{“}p\text{ is semi-decidable”})$$
$$\overline{\mathcal{S}}p := \exists f : X \to \mathbb{N} \to \mathbb{B}.\quad \forall x.\ px \leftrightarrow \forall n. fxn = \mathsf{false} \qquad (\text{“}p\text{ is co-semi-decidable”})$$
$$\mathcal{E}p := \exists f : \mathbb{N} \to \mathbb{O}X.\quad \forall x.\ px \leftrightarrow \exists n. fn = \mathsf{Some}\,x \qquad (\text{“}p\text{ is enumerable”})$$

Although all notions are defined on unary predicates, we use them on n-ary relations via (implicit) uncurrying. We write $\overline{p}$ for the complement $\lambda x.\ \neg px$ of p. We call a type X *discrete* if its equality relation $=_X$ is decidable and *enumerable* if the predicate $\lambda x.\top$ is enumerable.

Traditionally, propositions P s.t. $P \leftrightarrow (\exists n.\ fn = \mathsf{true})$ for some f are often called Σ^0_1 or "simply existential", and P s.t. $P \leftrightarrow (\forall n.\ fn = \mathsf{false})$ are called Π^0_1 or "simply universal". Semi-decidable predicates are pointwise Σ^0_1, and co-semi-decidable predicates are pointwise Π^0_1. Note that neither $\overline{\mathcal{S}}p \to \mathcal{S}\overline{p}$ nor the converse is provable, only the following connections:

Lemma 3. *The following hold:*
1. *Decidable predicates are semi-decidable and co-semi-decidable.*
2. *Semi-decidable predicates on enumerable types are enumerable.*
3. *Enumerable predicates on discrete types are semi-decidable.*
4. *The complement of semi-decidable predicates is co-semi-decidable.*

Lemma 4. *Decidable predicates are closed under complementation. Decidable, enumerable, and semi-decidable predicates are closed under (pointwise) conjunction and disjunction.*

3 Church's thesis in type theory

Church's thesis for total functions (CT) states that every function of type $\mathbb{N} \to \mathbb{N}$ is algorithmic. Thus CT is a relativisation of the function space $\mathbb{N} \to \mathbb{N}$ w.r.t. a given (Turing-complete) model of computation, reminiscent of the axiom $V = L$ in set theory [29].

We first define CT by abstracting away from a concrete model of computation and work with an *abstract model of computation*, consisting of an *abstract computation function* $Tcxn$ (with $T : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{O}\mathbb{N}$), assigning to a code c (to be interpreted as the code of a partial recursive function in a model of computation), an input number x, and a step index n an output number y if the code terminates in n steps on x with value y. The function Tcx is assumed to be monotonic, i.e. increasing the step index does not change the potential value:

$$Tcxn_1 = \mathsf{Some}\, y \to \forall n_2 \geq n_1.\ Tcxn_2 = \mathsf{Some}\, y.$$

Based on T we define a computability relation between $c : \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$:

$$c \sim f := \forall x. \exists n.\ Tcxn = \mathsf{Some}\,(fx).$$

Since T is monotonic, $\sim$ is extensional, i.e. $n \sim f_1 \to n \sim f_2 \to \forall x.\ f_1 x = f_2 x$. We define Church's thesis for total functions relative to an abstract computation function T:

$$\mathsf{CT}_T := \forall f : \mathbb{N} \to \mathbb{N}. \exists n : \mathbb{N}.\ n \sim f$$

Note that CT_T is clearly not consistent for every choice of T. If we write CT without index, we mean T to be the step-indexed evaluation function of a concrete, Turing-complete model of computation. For the mechanisation we could for instance pick the equivalent models of Turing machines [17], λ-calculus [21], μ-recursive functions [30], or register machines [18, 31]. It seems likely that the consistency proof of CT in [43] can be adapted to Coq.

Since specific properties of the model of computation are not needed, we develop and mechanise all results of this paper parameterised in an arbitrary T. Thus, we could also state all results in terms of a fully synthetic Church's thesis axiom $\Sigma T.\mathsf{CT}_T$.

▶ **Fact 5.** $\mathsf{CT} \to \Sigma T.\mathsf{CT}_T$

Note that the implication is strict: An abstract computation function does not rule out oracles for e.g. the halting problem of Turing machines, whereas CT – with T defined in terms of a standard, Turing-complete model of computation – proves the undecidability of the Turing machine halting problem.

3.1 Bauer's enumerability axiom EA

In proofs of theorems with CT_T as assumption, T can be used as replacement for a *universal machine*. Bauer [1] develops computability theory synthetically using the axiom "the set of enumerable sets of natural numbers is enumerable", which is equivalent to $\Sigma T.\mathsf{CT}_T$ and thus strictly weaker than CT, but can also be used in place of a universal machine. We introduce Bauer's axiom in our setting as EA' and immediately introduce a strengthening EA s.t. $(\Sigma T.\mathsf{CT}_T) \leftrightarrow \mathsf{EA}$ and $\mathsf{EA} \to \mathsf{EA}'$:

$$\mathsf{EA}' := \Sigma \mathcal{W} : \mathbb{N} \to (\mathbb{N} \to \mathbb{P}).\forall p : \mathbb{N} \to \mathbb{P}.\ \mathcal{E}p \leftrightarrow \exists c.\ \mathcal{W}c \equiv_{\mathbb{N}\to\mathbb{P}} p$$

That is, EA' states that there is an enumerator $\mathcal{W}$ of all enumerable predicates, up to extensionality. In contrast, EA poses the existence of an enumerator of all possible enumerators, up to range equivalence:

$$\mathsf{EA} := \Sigma \varphi : \mathbb{N} \to (\mathbb{N} \to \mathbb{O}\mathbb{N}).\forall f : \mathbb{N} \to \mathbb{O}\mathbb{N}.\exists c.\ \varphi c \equiv_{\mathsf{ran}} f$$

That is, φ is a surjection w.r.t. range equivalence $f \equiv_{\mathsf{ran}} g$, where $\varphi c \equiv_{\mathsf{ran}} f \leftrightarrow \forall x.(\exists n.\varphi cn = \mathsf{Some}\,x) \leftrightarrow (\exists n.fn = \mathsf{Some}\,x)$.

Note the two different roles of natural numbers in the two axioms: If we would consider predicates over a general type X we would have $\mathcal{W} : \mathbb{N} \to (X \to \mathbb{P})$ and $\varphi : \mathbb{N} \to (\mathbb{N} \to \mathbb{O}X)$, i.e. $\mathcal{W}c$ would be an enumerable predicate and φc an enumerator of a predicate $X \to \mathbb{P}$.

We start by proving $\mathsf{CT}_T \to \mathsf{EA}$ by constructing φ from an arbitrary T:

$$\varphi c\langle n, m\rangle := \textbf{if } Tcnm \textbf{ is } \mathsf{Some}\,x \textbf{ then } \mathsf{S}x \textbf{ else } 0$$

Lemma 6. *If* CT_T *then* $\forall f : \mathbb{N} \to \mathbb{O}\mathbb{N}.\exists c.\ \varphi c \equiv_{\mathsf{ran}} f$.

Proof. The direction from left to right to establish $\equiv_{\mathsf{ran}}$ is based on the fact that if $Tcxn_1 = \mathsf{Some}\,y_1$ and $Tcxn_2 = \mathsf{Some}\,y_2$ then $y_1 = y_2$. The other direction is straightforward. ◀

Theorem 7. $\forall T.\ \mathsf{CT}_T \to \mathsf{EA}$

We now prove $\mathsf{EA} \to \mathsf{EA}'$ by constructing $\mathcal{W}$ from φ: $\mathcal{W}cx := \exists n.\varphi cn = \mathsf{Some}\,x$.

Lemma 8. *If* EA *then* $\forall p : \mathbb{N} \to \mathbb{P}.\ \mathcal{E}p \leftrightarrow \exists c.\ \mathcal{W}c \equiv_{\mathbb{N}\to\mathbb{P}} p$.

Proof.
$$\begin{aligned} \mathcal{E}p &\leftrightarrow \exists f : \mathbb{N} \to \mathbb{O}\mathbb{N}.\forall x.\ px \leftrightarrow \exists n.\ fn = \mathsf{Some}\,x && (\text{def. } \mathcal{E}) \\ &\leftrightarrow \exists c.\forall x.\ px \leftrightarrow \exists n.\ \varphi cn = \mathsf{Some}\,x && (\mathsf{EA}) \\ &\leftrightarrow \exists c.\ \mathcal{W}c \equiv_{\mathbb{N}\to\mathbb{P}} p && (\text{def. } \equiv_{\mathbb{N}\to\mathbb{P}}) \end{aligned}$$
◀

Theorem 9. $\mathsf{EA} \to \mathsf{EA}'$

3.2 Richman's Enumerability of Partial Functions EPF

Richman [37] introduces a different purely synthetic axiom as replacement for a universal machine and assumes that "partial functions are countable", which is equivalent to EA.

$$\mathsf{EPF} := \Sigma e : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N}).\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\exists n.\ en \equiv_{\mathbb{N}\rightharpoonup\mathbb{N}} f$$

Theorem 10. $\mathsf{EPF} \to \mathsf{EA}$

Proof. Let e be given. $\varphi c\langle n, m\rangle := \mathsf{seval}\ (ecn)\ m$ is the wanted enumerator. ◀

Theorem 11. $\mathsf{EA} \to \mathsf{EPF}$

Proof. Let φ be given. Then

$$ecx := (\mu\,(\lambda n.\ \textbf{if}\ \varphi cn\ \textbf{is}\ \mathsf{Some}\,\langle x', y'\rangle\ \textbf{then}\ x =_{\mathbb{B}} x'\ \textbf{else}\ \mathsf{false})) \gg\!= \\ \lambda n.\ \textbf{if}\ \varphi cn\ \textbf{is}\ \mathsf{Some}\,\langle x', y'\rangle\ \textbf{then}\ \mathsf{ret}\,y'\ \textbf{else}\ \mathsf{undef}$$

is the wanted enumerator. ◀

EPF implies the fully synthetic version of CT:

Lemma 12. $\mathsf{EPF} \to \Sigma T.\ \mathsf{CT}_T$

Proof. Assume $e : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ surjective w.r.t. $\equiv_{\mathbb{N}\rightharpoonup\mathbb{N}}$. Define $Tcxn := \mathsf{seval}\ (ecx)\ n$. It is straightforward to prove that T is monotonic and that CT holds. ◀

The axiom EPF can be weakened to cover just boolean functions:

$$\mathsf{EPF}_{\mathbb{B}} := \Sigma e : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{B}).\forall f : \mathbb{N} \rightharpoonup \mathbb{B}.\exists n.\ en \equiv_{\mathbb{N}\rightharpoonup\mathbb{B}} f$$

Lemma 13. $\mathsf{EPF} \to \mathsf{EPF}_{\mathbb{B}}$

The reverse direction seems not to be provable.

4 Halting Problems

For this section we assume EA, i.e. $\varphi : \mathbb{N} \to (\mathbb{N} \to \mathbb{O}\mathbb{N})$ s.t. $\forall f : \mathbb{N} \to \mathbb{O}\mathbb{N}.\exists c.\ \varphi c \equiv_{\mathsf{ran}} f$. Recall Lemma 8 stating that $\forall p : \mathbb{N} \to \mathbb{P}.\ \mathcal{E}p \leftrightarrow \exists c.\ \mathcal{W}c \equiv_{\mathbb{N}\to\mathbb{P}} p$.

We define $\mathsf{K}_0 n := \mathcal{W}nn$ and prove our first negative result:

Lemma 14. $\neg\mathcal{E}\overline{\mathsf{K}_0}$

Proof. Assume $\mathcal{E}(\lambda n.\neg\mathcal{W}nn)$. By specification of $\mathcal{W}$ there is c s.t. $\forall n.\mathcal{W}cn \leftrightarrow \neg\mathcal{W}nn$. In particular, $\mathcal{W}cc \leftrightarrow \neg\mathcal{W}cc$, which is contradictory. ◀

Corollary 15. *$\neg\mathcal{D}\mathsf{K}_0$, $\neg\mathcal{D}\overline{\mathsf{K}_0}$, $\neg\mathcal{D}\mathcal{W}$ and $\neg\mathcal{D}\overline{\mathcal{W}}$.*

Intuitively, K_0 can be seen as analogous to the self-halting problem: $\mathsf{K}_0 n$ states that n considered as an enumerator outputs itself in its range (rather than halting on itself).

It is also easy to show that $\mathcal{W}$ and thus K_0 are enumerable:

Lemma 16. $\mathcal{E}\mathcal{W}$

Proof. Via $f\langle n, m\rangle := \textbf{if}\ \varphi nm\ \textbf{is}\ \mathsf{Some}\,k\ \textbf{then}\ \mathsf{Some}\,(n,k)\ \textbf{else}\ \mathsf{None}$. ◀

Corollary 17. $\mathcal{E}\mathsf{K}_0$

Since Bauer [1] bases his development on EA', he needs the axiom of countable choice to prove that $\mathcal{W}$ is enumerable, whereas EA allows an axiom-free proof of this fact.

Another well-known traditional result is that a problem is enumerable if and only if it many-one reduces to the halting problem K, which can be proved without reference to EA.

$$p \preceq_m q := \exists f : X \to Y.\forall x.\ px \leftrightarrow q(fx) \qquad \mathsf{K}(f : \mathbb{N} \to \mathbb{B}) := \exists n.\ fn = \mathsf{true}$$

Fact 18. *For all $p : X \to \mathbb{P}$, $p \preceq_m \mathsf{K} \leftrightarrow \mathcal{S}p$.*

Corollary 19. $\mathcal{S}\mathsf{K}$

Corollary 20. *For all* $p : \mathbb{N} \to \mathbb{P}$, $p \preceq_m \mathsf{K} \leftrightarrow \mathcal{E}p$.

Using the non-enumerability of $\overline{\mathsf{K}_0}$ we can now prove our first negative result by reduction:

Corollary 21. $\mathsf{K}_0 \preceq_m \mathsf{K}$*, and thus* $\neg\mathcal{E}\overline{\mathsf{K}}$, $\neg\mathcal{D}\mathsf{K}$*, and* $\neg\mathcal{D}\overline{\mathsf{K}}$.

We can also define $\mathsf{K}_\mathbb{N} := \lambda f : \mathbb{N} \to \mathbb{N}.\ \exists n.\ fn \neq 0$:

Fact 22. $\mathsf{K} \preceq_m \mathsf{K}_\mathbb{N}$, $\mathsf{K}_\mathbb{N} \preceq_m \mathsf{K}$, $\overline{\mathsf{K}_\mathbb{N}} \equiv_{(\mathbb{N}\to\mathbb{N})\to\mathbb{P}} \lambda f.\ \forall n.\ fn = 0$*, and thus* $\neg\mathcal{D}(\lambda f.\ \forall n.\ fn = 0)$.

5 Kleene Trees

In a lecture in 1953 Kleene [28] gave an example how the axioms of Brouwer's intuitionism fail if all functions are considered computable by constructing an infinite decidable binary tree with no computable infinite path. The existence of such a Kleene tree (KT) is in contradiction to Brouwer's fan theorem, which we will discuss later. We prove that $\mathsf{EPF}_\mathbb{B}$ implies KT.

For this purpose, we call a predicate $\tau : \mathbb{L}\mathbb{B} \to \mathbb{P}$ a (decidable) *binary tree* if

(a) τ is decidable: $\exists f. \forall u. \tau u \leftrightarrow fu = \mathsf{true}$

(b) τ is non-empty: $\exists u. \tau u$

(c) τ is prefix-closed: If τu_2 and $u_1 \sqsubseteq u_2$ then τu_1 (where $u_1 \sqsubseteq u_2 := \exists u'.\ u_2 = u_1 \mathbin{+\!\!+} u'$).

We will just speak of trees instead of decidable binary trees in the following.

Fact 23. *For every tree* τ, $\tau[]$ *holds.*

Furthermore, a decidable binary tree τ ...

- ...is *bounded* if $\exists n. \forall u. |u| \geq n \to \neg\tau u$
- ...is *well-founded* if $\forall f. \exists n. \neg\tau[f0, \dots, fn]$
- ...has an *infinite path* if $\exists f. \forall n. \tau[f0, \dots, fn]$

Fact 24. *A tree is not bounded if and only if it is* infinite*, defined as* $\forall n. \exists u.\ |u| \geq n \wedge \tau u$.

Fact 25. *Every bounded tree is well-founded and every tree with an infinite path is infinite.*

Note that both implications are strict: In our setting we cannot prove boundedness from well-foundedness nor obtain an infinite path from infiniteness, as can be seen from a Kleene tree:

KT := *There exists an infinite, well-founded, decidable binary tree.*

We follow Bauer [2] to construct a Kleene tree.

Lemma 26. *Given* $\mathsf{EPF}_\mathbb{B}$ *one can construct* $d : \mathbb{N} \rightharpoonup \mathbb{B}$ *s.t.* $\forall f : \mathbb{N} \to \mathbb{B}. \exists nb.\ dn \overset{!}{=} b \wedge fn \neq b$.

Proof. Define $dn := enn \ggg \lambda b.\ \mathsf{ret}\,(\neg_\mathbb{B} b)$. ◀

We define $\tau_K u := \forall n < |u|. \forall x.\ \mathsf{seval}\ (dn)\ |u| = \mathsf{Some}\,x \to u[n] = \mathsf{Some}\,x$. Intuitively, τ_K contains all paths $u = [b_0, b_1, \dots, b_n]$ which might be prefixes of d given n as step index, i.e. where n does not suffice to verify that d is no prefix of d. An infinite path through τ_K would be a totalisation of d.

Theorem 27. $\mathsf{EPF}_\mathbb{B} \to \mathsf{KT}$

Proof. We show that τ_K is a Kleene tree. That τ_K is a decidable tree is immediate. To show that τ_K is infinite let k be given. We define $f0 := []$ and $f(Sn) := fn \mathbin{+\!\!+} [\mathbf{if}\ Dkn\ \mathbf{is}\ \mathsf{Some}\,x\ \mathbf{then}\ x\ \mathbf{else}\ \mathsf{false}]$. We have $|fn| = n$. In particular, $|fk| \geq k$ and $\tau_K(fk)$.

For well-foundedness let $f : \mathbb{N} \to \mathbb{B}$ be given. There is n s.t. $dn \overset{!}{=} b$ and $fn \neq b$. Thus there is k s.t. $\mathsf{seval}\ (dn)\ k = \mathsf{Some}\,b$. Now $\neg\tau_K u$ for $u := [f0, \dots, f(n+k)]$. ◀

6 Extensionality Axioms

Coq's type theory is intensional, i.e. $f \equiv_{A \to B} g$ and $f = g$ do not coincide. Extensionality properties can however be consistently assumed as axioms. In this section we briefly discuss the relationship between CT and functional extensionality Fext, propositional extensionality Pext and proof irrelevance PI, defined as follows:

$$\begin{aligned}
\mathsf{Fext} &:= \forall AB. \forall fg : A \to B.\ (\forall a. fa = ga) \to f = g \\
\mathsf{Pext} &:= \forall PQ : \mathbb{P}.\ (P \leftrightarrow Q) \to P = Q \\
\mathsf{PI} &:= \forall P : \mathbb{P}. \forall (x_1 x_2 : P).\ x_1 = x_2
\end{aligned}$$

Fact 28. $\mathsf{Pext} \to \mathsf{PI}$

Swan and Uemura [43] prove that intensional predicative Martin-Löf type theory remains consistent if CT, the axiom of univalence, and propositional truncation are added. Since functional extensionality and propositional extensionality are a consequence of univalence, and propositions are semantically defined as exactly the irrelevant types, Fext, Pext, and PI hold in this extension of type theory. It seems likely that the consistency result can then be adapted to Coq's type theory, yielding a consistency proof for CT with Fext, Pext, and PI.

It is however crucial to formulate CT using $\exists$ instead of Σ. The formulation as $\mathsf{CT}_\Sigma := \forall f.\ \Sigma n.\ n \sim f$ is inconsistent with functional extensionality Fext, as already observed in [46].

Lemma 29. $\mathsf{CT}_\Sigma \to \mathsf{Fext} \to \bot$

Proof. Since CT_Σ implies EA, it suffices to prove that $\lambda f. \forall n.\ fn = 0$ is decidable by Fact 22. Assume $G : \forall f.\ \Sigma c.\ c \sim f$ and let $Ff :=$ **if** $\pi_1(Gf) = \pi_1(G(\lambda x.0))$ **then** true **else** false.

If $Ff = \mathsf{true}$, then $\pi_1(Gf) = \pi_1(G(\lambda x.0))$ and by extensionality of $\sim$, $fn = (\lambda x.0)n = 0$.

If $\forall n.\ fn = 0$, then $f = \lambda x.\ 0$ by Fext, thus $\pi_1(Gf) = \pi_1(G(\lambda x.\ 0))$ and $Ff = \mathsf{true}$. ◀

7 Classical Logical Axioms

In this section we consider consequences of the law of excluded middle LEM. Precisely, besides LEM, we consider the weak law of excluded middle WLEM, the Gödel-Dummett-Principle DGP[2] and the principle of independence of premises IP, together with their respective restriction of propositions to the satisfiability of boolean functions, resulting in the limited principle of omniscience LPO, the weak limited principle of omniscience WLPO, and the lesser limited principle of omniscience LLPO.

$$\begin{aligned}
\mathsf{LEM} &:= \forall P : \mathbb{P}.\ P \lor \neg P & \mathsf{LPO} &:= \forall f : \mathbb{N} \to \mathbb{B}.\ (\exists n.\ fn = \mathsf{true}) \lor \neg(\exists n.\ fn = \mathsf{true}) \\
\mathsf{WLEM} &:= \forall P : \mathbb{P}.\ \neg\neg P \lor \neg P & \mathsf{WLPO} &:= \forall f : \mathbb{N} \to \mathbb{B}.\ \neg\neg(\exists n.\ fn = \mathsf{true}) \lor \neg(\exists n.\ fn = \mathsf{true}) \\
\mathsf{DGP} &:= \forall PQ : \mathbb{P}. (P \to Q) \lor (Q \to P) & \mathsf{LLPO} &:= \forall fg : \mathbb{N} \to \mathbb{B}.\ ((\exists n.\ fn = \mathsf{true}) \to (\exists n.\ gn = \mathsf{true})) \\
& & &\qquad \lor ((\exists n.\ gn = \mathsf{true}) \to (\exists n.\ fn = \mathsf{true})) \\
\mathsf{IP} &:= \forall P : \mathbb{P}. \forall q : \mathbb{N} \to \mathbb{P}.\ (P \to \exists n. qn) \to \exists n.\ P \to qn
\end{aligned}$$

Fact 30. $\mathsf{LEM} \to \mathsf{DGP}$, $\mathsf{DGP} \to \mathsf{WLEM}$, $\mathsf{LEM} \to \mathsf{IP}$.

The converses are likely not provable: Diener constructs a topological model where DGP holds but not LEM, and one where WLEM holds but not DGP [11, Proposition 8.5.3]. Pédrot and Tabareau [36] construct a syntactic model where IP holds, but LEM does not.

[2] We follow Diener [11] in using the abbreviation DGP instead of GDP.

Fact 31. LPO $\to$ WLPO *and* WLPO $\to$ LLPO.

The converses are likely not provable: Both implications are strict over IZF with dependent choice [23, Theorem 5.1].

LPO is Σ^0_1-LEM and WLPO is simultaneously Σ^0_1-WLEM and Π^0_1-LEM, due to the following:

Fact 32. $(\forall n. fn = \mathsf{false}) \leftrightarrow \neg(\exists n. fn = \mathsf{true})$

Both can also be formulated for predicates:

Fact 33. *The following equivalences hold:*

1. LPO $\leftrightarrow \forall X.\forall(p : X \to \mathbb{P}).\ \mathcal{S}p \to \forall x.\ \ px \lor \neg px$
2. WLPO $\leftrightarrow \forall X.\forall(p : X \to \mathbb{P}).\ \mathcal{S}p \to \forall x. \neg px \lor \neg\neg px$
3. WLPO $\leftrightarrow \forall X.\forall(p : X \to \mathbb{P}).\ \overline{\mathcal{S}}p \to \forall x.\ \ px \lor \neg px$

In our formulation, LLPO is the Gödel-Dummet rule for Σ^0_1 propositions. It can also be formulated as Σ^0_1 or $\mathcal{S}$ De Morgan rule (**2**, **3** in the following Lemma), $\mathcal{S}$-DGP (**4**), or as a double negation elimination principle on $\overline{\mathcal{S}}$ relations into booleans (**5**):

Lemma 34. *The following are equivalent:*

1. LLPO
2. $\forall fg : \mathbb{N} \to \mathbb{B}.\ \neg((\exists n. fn = \mathsf{true}) \land (\exists n. gn = \mathsf{true})) \to \neg(\exists n. fn = \mathsf{true}) \lor \neg(\exists n. gn = \mathsf{true})$
3. $\forall X.\forall(p\ q : X \to \mathbb{P}).\ \mathcal{S}p \to \mathcal{S}q \to \forall x.\ \neg(px \land qx) \to \neg px \lor \neg qx$
4. $\forall X.\forall(p : X \to \mathbb{P}).\ \mathcal{S}p \to \forall xy.\ (px \to py) \lor (py \to px)$
5. $\forall X.\forall(R : X \to \mathbb{B} \to \mathbb{P}).\ \overline{\mathcal{S}}R \to \forall x.\ \neg\neg(\exists b.\ Rxb) \to \exists b.\ Rxb$
6. $\forall f.\ (\forall nm. fn = \mathsf{true} \to fm = \mathsf{true} \to n = m) \to (\forall n. f(2n) = \mathsf{false}) \lor (\forall n. f(2n+1) = \mathsf{false})$

We define the *principle of finite possibility* as PFP $:= \forall f. \exists g.\ (\forall n.\ fn = \mathsf{false}) \leftrightarrow (\exists n.\ gn = \mathsf{true})$. PFP unifies WLPO and LLPO:

Fact 35. WLPO $\leftrightarrow$ LLPO $\land$ PFP

A principle unifying the classical axioms with their counterparts for Σ^0_1 is *Kripke's schema* KS $:= \forall P : \mathbb{P}. \exists f : \mathbb{N} \to \mathbb{B}.\ P \leftrightarrow \exists n.\ fn = \mathsf{true}$:

Fact 36. LEM $\to$ KS

Fact 37. *Given* KS *we have* LPO $\to$ LEM*,* WLPO $\to$ WLEM*, and* LLPO $\to$ DGP.

KS could be strengthened to state that every predicate is semi-decidable (to which KS is equivalent using $\mathsf{AC}_{\mathbb{N},\mathbb{N}\to\mathbb{N}}$). The strengthening would be incompatible with CT.

In general, the compatibility of classical logical axioms (without assuming choice principles) with CT seems open. We conjecture that Coq's restriction preventing large elimination principles for non-sub-singleton propositions makes LEM and CT consistent in Coq.

8 Axioms of Russian Constructivism

The Russian school of constructivism morally identifies functions with computable functions, sometimes assuming CT explicitly. Another axiom considered valid is Markov's principle:

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

Markov's principle is consistent with CT [43] and follows from LPO:

Fact 38. LPO $\leftrightarrow$ WLPO $\land$ MP

Corollary 39. LPO $\to$ MP.

It seems likely that the converse is not provable: There is a logic where MP holds, but not LPO [24]. As observed by Herbelin [24] and Pedrót and Tabareau [36], IP $\land$ MP yields LPO:

Lemma 40. MP $\to$ IP $\to$ LPO

Proof. Given $f : \mathbb{N} \to \mathbb{B}$ there is $n_0 : \mathbb{N}$ s.t. $\forall k.\ fk = \mathsf{true} \to fn_0 = \mathsf{true}$ using MP and IP: By MP, $\neg\neg(\exists k.\ fk = \mathsf{true}) \to \exists n.\ fn = \mathsf{true}$ and by IP, $\exists n. \neg\neg(\exists k. fk = \mathsf{true}) \to fn = \mathsf{true}$, which suffices. Now $fn_0 = \mathsf{true} \leftrightarrow \exists n.\ fn = \mathsf{true}$ and LPO follows. ◀

A nicer factorisation would be to prove IP $\to$ WLPO, but the implication seems unlikely.

Lemma 41. *The following are equivalent:*

1. MP
2. $\forall X. \forall p : X \to \mathbb{P}.\ \mathcal{S}p \to \forall x.\ \neg\neg px \to px$
3. $\forall X. \forall p : X \to \mathbb{P}.\ \mathcal{S}p \to \mathcal{S}\overline{p} \to \forall x.\ px \lor \neg px$
4. $\forall X. \forall p : X \to \mathbb{P}.\ \mathcal{S}p \to \mathcal{S}\overline{p} \to \mathcal{D}p$
5. $\forall X. \forall (R : X \to \mathbb{B} \to \mathbb{P}).\ \mathcal{S}R \to \forall x.\ \neg\neg(\exists b.\ Rxb) \to \exists b.\ Rxb$

Proof.
- **1** $\to$ **2** is immediate.
- **2** $\to$ **3**: Since $\mathcal{S}$ is closed under disjunctions and since $\neg\neg(px \lor \neg px)$ is a tautology.
- **3** $\to$ **4** is immediate by Lemma 49 with $Rxb := (px \land b = \mathsf{true}) \lor (\neg px \land b = \mathsf{false})$.
- **4** $\to$ **1**: Let $\neg\neg(\exists n. fn = \mathsf{true})$. Let $p(x : \mathbb{N}) := \exists n. fn = \mathsf{true}$. Now p is semi-decided by $\lambda x.f$, $\overline{p}$ by $\lambda xn.\mathsf{false}$, and $p0 \lor \neg p0$ by **4**. One case is easy, the other contradictory. ◀

Note that **4** is often called "Post's theorem". **1** $\leftrightarrow$ **3** $\leftrightarrow$ **4** is already discussed in [14]. **5** is dual to Lemma 34 (**5**). Replacing $\mathcal{S}p$ with $\overline{\mathcal{S}}p$ in **2** does however not result in an equivalent of LLPO, but turns **2** into an assumption-free fact. While in general $\mathcal{S}\overline{p} \leftrightarrow \overline{\mathcal{S}}p$ does not hold it seems possible that they can be exchanged in **3** and **4**, but we are not aware of a proof.

9 Choice Axioms

We consider the axioms of choice AC, unique choice AUC, dependent choice ADC, and countable choice ACC. $\mathsf{AC}_{\mathbb{N},\mathbb{N}}$ and $\mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}}$ are often called $\mathsf{AC}_{0,0}$ and $\mathsf{AC}_{1,0}$ in the literature.

$$\mathsf{AC}_{X,Y} := \forall R : X \to Y \to \mathbb{P}. (\forall x. \exists y. Rxy) \to \exists f : X \to Y. \forall x.\ Rx(fx)$$
$$\mathsf{AUC}_{X,Y} := \forall R : X \to Y \to \mathbb{P}. (\forall x. \exists! y. Rxy) \to \exists f : X \to Y. \forall x.\ Rx(fx)$$
$$\mathsf{ADC}_{X} := \forall R : X \to X \to \mathbb{P}. (\forall x. \exists x'. Rxx') \to \forall x_0. \exists f : \mathbb{N} \to X. f0 = x_0 \land \forall n.\ R(fn)(f(n+1)))$$
$$\mathsf{AC} := \forall XY : \mathbb{T}.\ \mathsf{AC}_{X,Y} \qquad \mathsf{AUC} := \forall XY.\ \mathsf{AUC}_{X,Y} \qquad \mathsf{ADC} := \forall X : \mathbb{T}.\ \mathsf{ADC}_X \qquad \mathsf{ACC} := \forall X : \mathbb{T}.\ \mathsf{AC}_{\mathbb{N},X}$$

Fact 42. $\mathsf{AC}_{X,X} \to \mathsf{ADC}_X$, $\mathsf{AC}_{X,Y} \to \mathsf{AUC}_{X,Y}$, ADC $\to$ ACC, ACC $\to$ $\mathsf{AC}_{\mathbb{N},\mathbb{N}}$, *and* $\mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}} \to \mathsf{AC}_{\mathbb{N},\mathbb{N}}$.

The following well-known fact is due to Diaconescu [10] and Myhill and Goodman [22]:

Fact 43. AC $\to$ Fext $\to$ Pext $\to$ LEM

Given that $\mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}}$ turns CT into CT_Σ, and that EA $\leftrightarrow \Sigma T.\mathsf{CT}_T$ we have:

Fact 44. $\mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}} \to$ Fext $\to$ EA $\to \bot$

We will later see that LLPO $\land$ $\mathsf{AC}_{\mathbb{N},\mathbb{N}}$ implies weak Kőnig's lemma, which is incompatible with KT. Already now we can prove that WLPO $\land$ $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$ is incompatible with EA:

Fact 45. $\mathsf{AUC}_{\mathbb{N},\mathbb{B}} \to (\forall n : \mathbb{N}.\ pn \lor \neg pn) \to \mathcal{D}p$

Lemma 46. WLPO $\to \mathsf{AUC}_{\mathbb{N},\mathbb{B}} \to$ EA $\to \mathcal{D}\overline{\mathsf{K}_0}$

Proof. WLPO implies $\forall n. \neg\mathsf{K}_0 n \lor \neg\neg\mathsf{K}_0 n$. By $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$ and the last lemma $\overline{\mathsf{K}_0}$ is decidable. ◀

Corollary 47. WLPO $\to \mathsf{AUC}_{\mathbb{N},\mathbb{B}} \to$ EA $\to \bot$

9.1 Provable choice axioms

In contrast to predicative Martin-Löf type theory, Coq's type theory does not prove the axiom of choice, nor the axioms of dependent and countable choice. This is due to the fact that arbitrary large eliminations are not allowed. However, recall that a large elimination principle for the accessibility predicate is provable, resulting in Lemma 1. Using Lemma 1 we can then prove $\mathcal{D}\text{-}\mathsf{AC}_{X,\mathbb{N}}$ for all X, i.e. choice for decidable relations into natural numbers:

Lemma 48. $\forall X.\forall R : X \to \mathbb{N} \to \mathbb{P}.\ \mathcal{D}R \to (\forall x.\exists n.Rxn) \to \exists f : X \to \mathbb{N}.\forall x.\ Rx(fx)$.

As a consequence and with no further reference to Lemma 1 we can then prove choice principles for semi-decidable and enumerable relations, i.e. $\mathcal{S}\text{-}\mathsf{AC}_{X,\mathbb{N}}$ and $\mathcal{E}\text{-}\mathsf{AC}_{\mathbb{N},X}$ for all X:

Lemma 49. *The following two choice principles are provable*[3]*:*

1. $\forall X.\forall R : X \to \mathbb{N} \to \mathbb{P}.\ \mathcal{S}R \to (\forall x.\exists n.\ Rxn) \to \exists f : X \to \mathbb{N}.\forall x.\ Rx(fx)$
2. $\forall X.\forall R : \mathbb{N} \to X \to \mathbb{P}.\ \mathcal{E}R \to (\forall n.\exists x.\ Rnx) \to \exists f : \mathbb{N} \to X.\forall n.\ Rn(fn)$

Principle **2** can be relaxed to arbitrary discrete types instead of $\mathbb{N}$, and in particular $\mathcal{S}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ follows from **1**. In Appendix A we discuss consequences of the here mentioned principles with regards to CT for oracles and in the next section $\overline{\mathcal{S}}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ will be central.

10 Axioms on Trees

We have already introduced (decidable) binary trees and Kleene trees in Section 5. We now give a broader overview and give formulations of LPO, WLPO, LLPO, and MP in terms of decidable binary trees, following Berger et al. [5].

Fact 50. *Let τ be a tree. Then $\tau_u v := \tau(u +\!\!+\, v)$ is a tree if and only if τu.*

If τu holds we call τ_u a *subtree* of τ and $\tau_{[b]}$ a *direct subtree* of τ.

Lemma 51. *The following equivalences hold:*

1. LPO $\leftrightarrow$ *every tree is bounded or infinite.*
2. WLPO $\leftrightarrow$ *every tree is infinite or not infinite.*
3. LLPO $\leftrightarrow$ *every infinite tree has a direct infinite subtree.*
4. MP $\leftrightarrow$ *if a tree is not infinite it is bounded.*
5. MP $\leftrightarrow$ *if a tree has no infinite path it is well-founded.*

Recall Fact 25 stating that every bounded tree is well-founded and that every tree with an infinite path is infinite. The respective converse implications are known as Brouwer's *fan theorem* FAN and *weak Kőnig's lemma* WKL respectively:

FAN := *Every well-founded decidable binary tree is bounded.*

WKL := *Every infinite decidable binary tree has an infinite path.*

Fact 52. KT $\to \neg$FAN *and* KT $\to \neg$WKL.

Note that FAN is called FAN'_Δ in [26] and FAN_Δ in [11], and WKL is called $\mathsf{WKL}_\mathcal{D}$ in [15]. Ishihara [26] shows how to deduce FAN from WKL constructively:

[3] A formulation of (**1**) for disjunctions (equivalently: $R : X \to \mathbb{B} \to \mathbb{P}$) is due to Andrej Dudenhefner and was received in private communication. (**2**) was anticipated by Larchey-Wendling [30], who formulated it for μ-recursively enumerable instead of synthetically enumerable predicates.

Fact 53. *Bounded trees τ have a longest element, i.e. $\exists u.\ \tau u \land \forall v.\ \tau v \to |v| \le |u|$.*

Lemma 54. *For every tree τ there is an infinite tree τ' s.t. for any infinite path f of τ' $\forall u.\ \tau u \to \tau[f0, \dots, f|u|]$.*

Theorem 55. $\mathsf{WKL} \to \mathsf{FAN}$

Proof. Let τ be well-founded. By Lemma 54 and WKL, there is f s.t. $\forall a.\ \tau u \to \tau[f0, \dots, f|u|]$. Since τ is well-founded there is n s.t. $\neg\tau[f0, \dots, fn]$. Then n is a bound for τ: For u with $|u| > n$ and τu we have $\tau[f0, \dots, fn, \dots, f|u|]$. But then $\tau[f0, \dots, fn]$, contradiction. ◀

Corollary 56. $\mathsf{KT} \to \neg\mathsf{WKL}$.

Berger and Ishihara [4] show that $\mathsf{FAN} \leftrightarrow \mathsf{WKL!}$, a restriction of WKL stating that every infinite decidable binary tree with *at most one* infinite path has an infinite path. Schwichtenberg [40] gives a more direct construction and mechanises the proof in Minlog.

Berger, Ishihara, and Schuster [5] characterise WKL as the combination of the logical principle LLPO and the function existence principle $\overline{\mathcal{S}}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ (called $\Pi^0_1\text{-}\mathsf{ACC}^\vee$ in [5]). We observe that WKL can also be characterised as one particular choice or dependent choice principle. The proofs are essentially rearrangements of [5, Theorem 27 and Corollary 5].

Theorem 57. *The following are equivalent:*

1. WKL
2. $\mathsf{LLPO} \land \overline{\mathcal{S}}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$
3. $\forall R : \mathbb{N} \to \mathbb{B} \to \mathbb{P}.\ \overline{\mathcal{S}}R \to (\forall n.\neg\neg\exists b.Rnb) \to \exists f : \mathbb{N} \to \mathbb{B}.\forall n.\ R\ n\ (fn)$
4. $\forall R : \mathbb{LB} \to \mathbb{B} \to \mathbb{P}.\ \overline{\mathcal{S}}R \to (\forall u.\neg\neg\exists b.Rub) \to \exists f : \mathbb{N} \to \mathbb{B}.\forall n.\ R\ [f0, \dots, f(n-1)]\ (fn)$

Proof. For $\mathsf{WKL} \to \mathsf{LLPO}$ we use the characterisation **3** of LLPO from Lemma 51. Let τ be an infinite tree. By WKL there is an infinite path f. Then $\tau_{[f0]}$ is a direct infinite subtree.

For $\mathsf{WKL} \to \overline{\mathcal{S}}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ let R be total and f s.t. $\forall nb.\ Rnb \leftrightarrow \forall m.fnbm = \mathsf{false}$. Define the tree $\tau u := \forall i < |u|.\forall m < |u|.\ fi(u[i])m = \mathsf{false}$. Infinity of τ follows from $\forall n.\exists u.|u| = n \land \forall i < n.Ri(u[i])$, proved by induction on n using totality of R. If g is an infinite path of τ, $Rn(gn)$ follows from $\forall m.\tau[g0, \dots, g(n+m+1)]$.

2 $\to$ **3** is immediate using characterisation **3** of LLPO from Lemma 34.

For **3** $\to$ **4** let $F : \mathbb{N} \to \mathbb{LB}$ and $G : \mathbb{LB} \to \mathbb{N}$ invert each other.[4] Let $R : \mathbb{LB} \to \mathbb{B} \to \mathbb{P}$ and f be the choice function obtained from **3** for $\lambda nb.R(Fn)b$. Then $\lambda n.f(G(gn))$ where $g0 := []$ and $g(\mathsf{S}n) := gn +\!\!+ [f(G(gn))]$ is a choice function for R as wanted.

For **4** $\to$ **1** let τ be an infinite tree and let $d_u m := \exists v.|v| = m \land \tau_u v$, i.e. $d_u m$ if τ_u has depth at least m and in particular τ_u is infinite iff $\forall m.d_u m$. Define $Rub := \forall m.d_{u +\!\!+ [b]} m \lor \neg d_{u +\!\!+ [\neg_{\mathbb{B}} b]}$. R is co-semi-decidable (since d is decidable), and $\neg R\,u\,\mathsf{true} \land \neg R\,u\,\mathsf{false}$ is contradictory. Thus **4** yields a choice function f which fulfils $\tau[f0, \dots, fn]$ by induction on n. ◀

11 Continuity: Baire Space, Cantor Space, and Brouwer's Intuitionism

The total function space $\mathbb{N} \to \mathbb{N}$ is often called *Baire space*, whereas $\mathbb{N} \to \mathbb{B}$ is called *Cantor space*. We will from now on write $\mathbb{N}^\mathbb{N}$ and $\mathbb{B}^\mathbb{N}$ for the spaces.

Constructively, one cannot prove that $\mathbb{N}^\mathbb{N}$ and $\mathbb{B}^\mathbb{N}$ are in bijection. However, KT is equivalent to the existence of a continuous bijection $\mathbb{B}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ with a continuous modulus of continuity, i.e. a modulus function which is continuous (in the point) itself [11]. Furthermore, KT yields a continuous bijection $\mathbb{N}^\mathbb{N} \to \mathbb{B}^\mathbb{N}$ [3].

[4] These so called coding functions is easy to construct even formally using e.g. techniques from [14].

We call a function $F : A^{\mathbb{N}} \to B^{\mathbb{N}}$ *continuous* if $\forall f : A^{\mathbb{N}}.\forall n : \mathbb{N}.\exists L : \mathbb{L}\mathbb{N}.\forall g : A^{\mathbb{N}}.\ (\mathsf{map}\ f\ L = \mathsf{map}\ g\ L) \to Ffn = Fgn$. A function $M : A^{\mathbb{N}} \to \mathbb{N} \to \mathbb{L}\mathbb{N}$ is called the *modulus of continuity* for F if $\forall n : \mathbb{N}.\forall fg : A^{\mathbb{N}}.\ \mathsf{map}\ f\ (Mfn) = \mathsf{map}\ g\ (Mfn) \to Ffn = Fgn$. We define:

$$\mathsf{Homeo}(A^{\mathbb{N}}, B^{\mathbb{N}}) := \exists F : A^{\mathbb{N}} \to B^{\mathbb{N}}.\exists M.\ M \textit{ is a continuous modulus of continuity for } F$$

We start by proving that $\mathsf{KT} \leftrightarrow \mathsf{Homeo}(\mathbb{B}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}})$. To do so, we say that $u +\!\!+ [b]$ is a *leaf of a Kleene tree* τ_K if $\tau_K u$, but $\neg\tau_K(u +\!\!+ [b])$.

Fact 58. *For every τ_K, there is an injective enumeration $\ell : \mathbb{N} \to \mathbb{L}\mathbb{B}$ of the leaves of τ_K.*

We define $F(f : \mathbb{N} \to \mathbb{N})n := (\ell(f0) +\!\!+ \cdots +\!\!+ \ell(f(n+1)))[n]$. Since leaves cannot be empty, the length of the accessed list is always larger than n and F is well-defined.

Lemma 59. *F is injective w.r.t. $\equiv_{\mathbb{N}^{\mathbb{B}}}$ and $\equiv_{\mathbb{N}^{\mathbb{N}}}$.*

Lemma 60. *F is continuous with continuous modulus of continuity.*

Lemma 61. *The following hold for a Kleene tree τ_K:*
1. *There is a function $\ell^{-1} : \mathbb{L}\mathbb{B} \to \mathbb{N}$ s.t. for all leafs l, $\ell(\ell^{-1}l) = l$.*
2. *For all l s.t. $\neg\tau_K l$ there exists $l' \sqsubseteq l$ s.t. l' is a leaf of τ_K.*
3. *There is* $\mathsf{pref} : (\mathbb{N} \to \mathbb{B}) \to \mathbb{L}\mathbb{B}$ *s.t.* $\mathsf{pref}\ g$ *is a leaf of τ_K and $\exists n.$* $\mathsf{pref}\ g = \mathsf{map}\ g\ [0, \ldots, n]$.

We can now define the inverse as $G\ g\ n := \ell^{-1}(\mathsf{pref}\ (\mathsf{nxt}^n g))$ where $\mathsf{nxt}\ g\ n := g(n + |\mathsf{pref}\ g|)$.

Lemma 62. $F\ (G\ g) \equiv_{\mathbb{N}\to\mathbb{B}} g$

Lemma 63. *G is continuous with continuous modulus of continuity.*

The following proof is due to Diener [11, Proposition 5.3.2].

Lemma 64. $\mathsf{Homeo}(\mathbb{B}^{\mathbb{N}}, \mathbb{N}^{\mathbb{B}}) \to \mathsf{KT}$

Proof. Let F be a bijection with continuous modulus of continuity M. Then $\tau u := \forall 0 < i \leq |u|.\exists k < i.k \in M(\lambda n.\textbf{if } l[n] \textbf{ is } \mathsf{Some}\, b \textbf{ then } b \textbf{ else } \mathsf{false})\ 0$ is a Kleene tree. ◀

Theorem 65. $\mathsf{KT} \leftrightarrow \mathsf{Homeo}(\mathbb{B}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}})$ *and* $\mathsf{KT} \to \mathsf{Homeo}(\mathbb{N}^{\mathbb{N}}, \mathbb{B}^{\mathbb{N}})$.

Deiser [9] proves in a classical setting that $\mathsf{Homeo}(\mathbb{N}^{\mathbb{N}}, \mathbb{B}^{\mathbb{N}})$ holds. It would be interesting to see whether the proof can be adapted to a constructive proof $\mathsf{WKL} \to \mathsf{Homeo}(\mathbb{N}^{\mathbb{N}}, \mathbb{B}^{\mathbb{N}})$.

We have already seen that CT is inconsistent with FAN. Besides FAN, in Brouwer's intuitionism the continuity of functionals $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is routinely assumed:

$$\mathsf{Cont} := \forall F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \forall f : \mathbb{N} \to A.\exists L : \mathbb{L}\mathbb{N}.\forall g : \mathbb{N} \to A.\ (\mathsf{map}\ f\ L = \mathsf{map}\ g\ L) \to Ff \equiv_B Fg$$

Since every computable function is continuous, we believe Cont to be consistent with CT. Combining Cont with $\mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}}$ yields *Brouwer's continuity principle*[5] called WC-N in [46]:

$$\mathsf{WC\text{-}N} := \forall R : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{P}.(\forall f.\exists n.Rfn) \to \forall f.\exists Ln.\forall g.\ \mathsf{map}\ f\ L = \mathsf{map}\ g\ L \to Rgn$$

[5] But note that $\mathsf{Cont} \to \mathsf{AC}_{\mathbb{N}\to\mathbb{N},\mathbb{N}} \to \bot$, since the resulting modulus of continuity function allows for the construction of a non-continuous function [13].

Theorem 66. $\mathsf{WC\text{-}N} \to \mathsf{Cont}$

$\mathsf{WC\text{-}N}$ is inconsistent with CT, since the computability relation $\sim$ is not continuous:

Theorem 67. $\mathsf{WC\text{-}N} \to \mathsf{CT} \to \bot$

Proof. Recall that if two functions have the same code they are extensionally equal. By CT, $\lambda fc.c \sim f$ is a total relation. Using $\mathsf{WC\text{-}N}$ for this relation and $\lambda x.\ 0$ yields a list L and a code c s.t. $\forall g.\ \mathsf{map}\ g\ L = [0, \dots, 0] \to c \sim g$.

The functions $\lambda x.\ 0$ and $\lambda x.\ \textbf{if}\ x \in L\ \textbf{then}\ 0\ \textbf{else}\ 1$ both fulfil the hypothesis and thus have the same code – a contradiction since they are not extensionally equal. ◀

12 Conclusion

In this paper we surveyed the known connections of axioms in Coq's type theory, a constructive type theory with a separate, impredicative universe of propositions, with a special focus on Church's thesis CT and formulations of axioms in terms of notions of synthetic computability. Furthermore, all results are mechanised in the Coq proof assistant.

In constructive mathematics, countable choice is often silently assumed, as critised e.g. by Richman [38, 39]. In contrast, constructive type theory with a universe of propositions seems to be a suitable base system for matters of constructive (reverse) mathematics sensitive to applications of countable choice. Due to the separate universe of propositions, such a constructive type theory neither proves countable nor dependent choice, allowing equivalences like the one in Theorem 57 to be stated sensitively to choice. We conjecture that Lemma 49 deducing $\mathcal{S}\text{-}\mathsf{AC}_{X,\mathbb{N}}$ and $\mathcal{E}\text{-}\mathsf{AC}_{\mathbb{N},X}$ directly from $\mathcal{D}\text{-}\mathsf{AC}_{X,\mathbb{N}}$ cannot be significantly strengthened. The proof of $\mathcal{D}\text{-}\mathsf{AC}_{X,\mathbb{N}}$ in turn crucially relies on a large elimination principle for $\exists n.\ fn = \mathsf{true}$ (Lemma 1). The theory of [5] proves $\mathcal{D}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ and thus likely also $\mathcal{S}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$.

Predicative Martin-Löf type theory proves AC and type theories with propositional truncation and a semantic notion of (homotopy) propositions prove $\mathsf{AUC}_{\mathbb{N},\mathbb{B}}$, thus LEM suffices to disprove CT for both these flavours of type theory. Based on the current state of knowledge in the literature it seems likely that $\mathcal{S}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{B}}$ and LEM together do not suffice to disprove CT, which seems to require at least classical logic of the strength of LLPO and a choice axiom for *co*-semi-decidable predicates. Thus we conjecture that a consistency proof of e.g. $\mathsf{LEM} \wedge \mathsf{CT}$ might be possible for Coq's type theory.

Another advantage of basing constructive investigations on constructive type theory is that implementations of type theory in proof assistants already exist. For this paper, mechanising the results in Coq was tremendously helpful in keeping track of all details. For example, many of the presented proofs are very sensitive to small changes in formulations, and Coq actually helped in understanding the proofs and getting them right.

Besides consistency, another interesting property of axioms is admissibility. For instance, Pédrot and Tabareau [36] prove MP admissible in constructive type theory. CT seems to be admissible in constructive type theory in the sense that for every defined function $f : \mathbb{N} \to \mathbb{N}$ one can define a program in a model of computation with the same input output behaviour, as witnessed by the certifying extraction for a fragment of Coq to the λ-calculus [16]. An admissibility proof of CT could then serve as a theoretical underpinning of the Coq library of undecidability proofs [19]. However, any formal admissibility proof would have to deal with the intricacies of Coq's type theory. It would be interesting to investigate whether Letouzey's semantic proof for the correctness of type and proof erasure [33] can be connected with the mechanisation of meta-theoretical properties of Coq's type theory [41] in the MetaCoq project [42], yielding a mechanised admissibility proof for CT in Coq's type theory.

References

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

2 Andrej Bauer. König's lemma and Kleene tree. *unpublished notes*, 2006.

3 Michael J. Beeson. *Foundations of constructive mathematics: Metamathematical studies*, volume 6. Springer Verlag, 1987.

4 Josef Berger and Hajime Ishihara. Brouwer's fan theorem and unique existence in constructive analysis. *Mathematical Logic Quarterly*, 51(4):360–364, 2005.

5 Josef Berger, Hajime Ishihara, and Peter Schuster. The weak König lemma, Brouwer's fan theorem, De Morgan's law, and dependent choice. *Reports on Mathematical Logic*, (47):63, 2012.

6 Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

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

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

9 Oliver Deiser. A simple continuous bijection from natural sequences to dyadic sequences. *The American Mathematical Monthly*, 116(7):643–646, 2009.

10 Radu Diaconescu. Axiom of choice and complementation. *Proceedings of the American Mathematical Society*, 51(1):176–178, 1975.

11 Hannes Diener. Constructive Reverse Mathematics. *arXiv:1804.05495 [math]*, April 2020. `arXiv:1804.05495`.

12 Martín H. Escardó and Cory M. Knapp. Partial Elements and Recursion via Dominances in Univalent Type Theory. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.CSL.2017.21`.

13 Martín Hötzel Escardó and Chuangjie Xu. The inconsistency of a Brouwerian continuity principle with the Curry–Howard interpretation. In *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

14 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.

15 Yannick Forster, Dominik Kirst, and Dominik Wehr. Completeness theorems for first-order logic analysed in constructive type theory (extended version). *arXiv preprint arXiv:2006.04399*, 2020.

16 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. `doi:10.4230/LIPIcs.ITP.2019.17`.

17 Yannick Forster, Fabian Kunze, and Maximilian Wuttke. Verified programming of Turing machines in Coq. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 114–128, 2020.

18 Yannick Forster and Dominique Larchey-Wendling. Certified undecidability of intuitionistic linear logic via binary stack machines and minsky machines. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 104–117, 2019.

19 Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq library of undecidable problems. In *The Sixth International Workshop on Coq for Programming Languages (CoqPL 2020).*, 2020. URL: `https://github.com/uds-psl/coq-library-undecidability`.

20 Yannick Forster and Gert Smolka. Weak call-by-value lambda calculus as a model of computation in coq. In *International Conference on Interactive Theorem Proving*, pages 189–206. Springer, 2017.

21 Yannick Forster and Gert Smolka. Call-by-value lambda calculus as a model of computation in Coq. *Journal of Automated Reasoning*, 63(2):393–413, 2019.

22 Noah Goodman and John Myhill. Choice implies excluded middle. *Mathematical Logic Quarterly*, 24(25-30):461–461, 1978. `doi:10.1002/malq.19780242514`.

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

24 Hugo Herbelin. An intuitionistic logic that proves Markov's principle. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 50–56. IEEE, 2010.

25 Hajime Ishihara. Reverse mathematics in Bishop's constructive mathematics. *Philosophia Scientiæ. Travaux d'histoire et de philosophie des sciences*, (CS 6):43–59, 2006.

26 Hajime Ishihara. Weak König's lemma implies Brouwer's fan theorem: a direct proof. *Notre Dame Journal of Formal Logic*, 47(2):249–252, 2006.

27 Stephen Cole Kleene. On the interpretation of intuitionistic number theory. *The journal of symbolic logic*, 10(4):109–124, 1945.

28 Stephen Cole Kleene. Recursive functions and intuitionistic mathematics, 1953.

29 Georg Kreisel. Church's thesis: a kind of reducibility axiom for constructive mathematics. In *Studies in Logic and the Foundations of Mathematics*, volume 60, pages 121–150. 1970.

30 Dominique Larchey-Wendling. Typing total recursive functions in Coq. In *International Conference on Interactive Theorem Proving*, pages 371–388. Springer, 2017.

31 Dominique Larchey-Wendling and Yannick Forster. Hilbert's tenth problem in Coq. *arXiv preprint arXiv:2003.04604*, 2020.

32 Dominique Larchey-Wendling and Jean-François Monin. The Braga method: Extracting certified algorithms from complex recursive schemes in Coq. In Klaus Mainzer, Peter Schuster, and Helmut Schwichtenberg, editors, *Proof and Computation: From Proof Theory and Univalent Mathematics to Program Extraction and Verification.* World Scientific Singapore, 2021.

33 Pierre Letouzey. *Programmation fonctionnelle certifiée: l'extraction de programmes dans l'assistant Coq.* PhD thesis, L'Université de Paris-Sud, July 2004. URL: `http://www.pps.jussieu.fr/~letouzey/download/these_letouzey.pdf`.

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

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

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

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

38 Fred Richman. The fundamental theorem of algebra: a constructive development without choice. *Pacific Journal of Mathematics*, 196(1):213–230, 2000.

39 Fred Richman. Constructive Mathematics without Choice. In Peter Schuster, Ulrich Berger, and Horst Osswald, editors, *Reuniting the Antipodes — Constructive and Nonstandard Views of the Continuum*, pages 199–205. Springer Netherlands, Dordrecht, 2001. `doi:10.1007/978-94-015-9757-9_17`.

40 Helmut Schwichtenberg. A direct proof of the equivalence between Brouwer's fan theorem and König's lemma with a uniqueness hypothesis. *J. UCS*, 11(12):2086–2095, 2005.

41 Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. The MetaCoq Project. *Journal of Automated Reasoning*, February 2020. doi:10.1007/s10817-019-09540-0.

42 Matthieu Sozeau, Simon Boulier, Yannick Forster, Nicolas Tabareau, and Théo Winterhalter. Coq Coq correct! verification of type checking and erasure for Coq, in Coq. *Proceedings of the ACM on Programming Languages*, 4(POPL):1–28, 2019.

43 Andrew Swan and Taichi Uemura. On Church's thesis in cubical assemblies. *arXiv preprint arXiv:1905.03014*, 2019.

44 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.

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

46 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.

47 Benjamin Werner. Sets in types, types in sets. In *International Symposium on Theoretical Aspects of Computer Software*, pages 530–546. Springer, 1997.

A Modesty and Oracles

Using $\mathcal{D}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{N}}$ from Lemma 48 allows proving a choice axiom w.r.t. models of computation, observed by Larchey-Wendling [30] and called "modesty" by Forster and Smolka [20].

▶ **Lemma 68.** *Let T be an abstract computation function. We have*

$$\forall c.(\forall n.\exists mk.\ T c n k = \mathsf{Some}\, m) \to \exists f : \mathbb{N} \to \mathbb{N}.\forall n.\exists k.\ T c n k = \mathsf{Some}\,(f n)$$

That is, if c is the code of a function inside the model of computation which is provably total, the total function can be computed outside of the model. This modesty principle simplifies the mechanisation of computability theory in type theory as e.g. in [21]. For instance, it allows to prove that defining decidability as "a total function in the model of computation deciding the predicate" and as "a meta-level function deciding the predicate which is computable in the model of computation" is equivalent.

However, the modesty principle prevents synthetic treatments of computability theory based on oracles. Traditionally, computability theory based on oracles is formulated using a computability function T_p, s.t. for $p : \mathbb{N} \to \mathbb{P}$ there exists a code c_p representing a total function s.t. $\forall n.(\exists k.T c_p n k = \mathsf{Some}\, 0) \leftrightarrow p n$.

Synthetically, we would now like to assume an abstract computability function for every p as "Church's thesis with oracles". "Church's thesis with oracles" implies CT, and we know that under CT the predicate K_0 is not decidable. However, under the presence of $\mathcal{D}\text{-}\mathsf{AC}_{\mathbb{N},\mathbb{N}}$ we can use T_{K_0} and obtain c_{K_0} which can be turned into a decider $f : \mathbb{N} \to \mathbb{B}$ for K_0 using the choice principle above – a contradiction.

B Coq mechanisation

The Coq mechanisation of the paper comprises 4250 lines of code, with 3300 lines of proofs and 950 lines of statements and definitions, i.e. 77% proofs. The mechanisation is based on the Coq-std++ library [45], plus around 1500 additional lines of code with custom extensions to Coq's standard library which are shared with the Coq library of undecidability proofs [19].

The 4250 lines of the main development are distributed as follows: The basics of synthetic computability (decidablility, semi-decidability, enumerability, many-one reductions) need 1150 lines of code. The mechanisation of Section 3, covering CT, EA, and EPF, comprises 400 lines of code. 120 lines of codes are needed for the undecidability results of Section 4. Section 5 and Section 10, covering trees and in particular Kleene trees, need 1000 lines of code. Section 11 on continuity is mechanised in 800 lines. The rest, i.e. Sections 6 to 9, needs 750 lines of code.

No advanced mechanisation techniques were needed. Discreteness and enumerability proofs for types were eased using type classes to assemble proofs for compound types such as $\mathbb{LB} \times \mathbb{ON}$, as already done in [14]. Defining the notions of $\equiv_{A \to B}$, $\equiv_{A \to \mathbb{P}}$, and so on was made possible by using type classes as well.

The technically most challenging mechanised proofs correspond to Lemmas 59 - 63, i.e. prove $\mathsf{KT} \to \mathsf{Homeo}(\mathbb{B}^{\mathbb{N}}, \mathbb{N}^{\mathbb{N}}) \land \mathsf{Homeo}(\mathbb{N}^{\mathbb{N}}, \mathbb{B}^{\mathbb{N}})$. For these proofs, lots of manipulation of prefixes of lists was needed, and while the functions `firstn` and `dropn` are defined in Coq's standard library, the very useful lemmas of Coq-std++ where needed to make the proofs feasible.

In the development of this paper, the Coq proof assistant, while also acting as proof checker, was truly used as an assistant: Lots of proofs were developed and understood directly while working in Coq rather than on paper, allowing to identify for instance the equivalent characterisations of LLPO, MP, and WKL as in Lemma 34 (**5**), Lemma 41 (**5**), and Theorem 57 (**3**,**4**), which are hard to observe on paper because lots of bookkeeping for side-conditions would have to be done manually then.