On Synthetic Undecidability in Coq, with an Application to the Entscheidungsproblem

Yannick Forster
Saarland University
Saarbrücken, Germany
forster@ps.uni-saarland.de

Dominik Kirst
Saarland University
Saarbrücken, Germany
kirst@ps.uni-saarland.de

Gert Smolka
Saarland University
Saarbrücken, Germany
smolka@ps.uni-saarland.de

Abstract

We formalise the computational undecidability of validity, satisfiability, and provability of first-order formulas following a synthetic approach based on the computation native to Coq's constructive type theory. Concretely, we consider Tarski and Kripke semantics as well as classical and intuitionistic natural deduction systems and provide compact many-one reductions from the Post correspondence problem (PCP). Moreover, developing a basic framework for synthetic computability theory in Coq, we formalise standard results concerning decidability, enumerability, and reducibility without reference to a concrete model of computation. For instance, we prove the equivalence of Post's theorem with Markov's principle and provide a convenient technique for establishing the enumerability of inductive predicates such as the considered proof systems and PCP.

CCS Concepts • **Theory of computation → Logic**;

Keywords synthetic undecidability, Entscheidungsproblem, Coq, Post's theorem, Markov's principle, first-order logic

ACM Reference Format:
Yannick Forster, Dominik Kirst, and Gert Smolka. 2019. 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 '19), January 14–15, 2019, Cascais, Portugal.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3293880.3294091

1 Introduction

Every function definable in constructive type theory is computable. Thus, standard notions from computability theory like decidability, enumerability, and reductions are available without reference to a concrete model of computation such as Turing machines, general recursive functions, or the λ-calculus. For instance, representing a given decision problem by a predicate p on a type X, a function $f : X \to \mathbb{B}$ with $\forall x.\, p\,x \leftrightarrow f\,x = \mathrm{tt}$ is a decision procedure, a function $g : \mathbb{N} \to X$ with $\forall x.\, p\,x \leftrightarrow (\exists n.\, g\,n = x)$ is an enumeration, and a function $h : X \to Y$ with $\forall x.\, p\,x \leftrightarrow q\,(h\,x)$ for a predicate q on a type Y is a many-one reduction from p to q. Working formally with concrete models instead is cumbersome, given that every defined procedure needs to be shown representable by a concrete entity of the model. To avoid this tedium, many presentations resort to informal arguments regarding the algorithmic properties at the core of their constructions.

Enabling the outlined *synthetic* approach to computability as explored by Richman [36] and Bauer [2, 3], constructive type thoery is well-suited for formalising positive statements about decision problems. Turning to negative statements like undecidability and non-enumerability, however, the situation becomes more intricate. Typical formulations of constructive type theory such as Martin-Löf type theory (MLTT) or Coq's underlying calculus of inductive constructions (CIC) are consistent with the assumption that every predicate is decidable, so proving a concrete decision problem undecidable is not outright possible. A potential way out would be to roll back to a concrete model and establish negative results w.r.t. the modelled computation – again producing unwelcome technical overhead. The preferable solution, employed in this work, is to verify a synthetic reduction from a problem informally known to be undecidable – establishing negative results relative to the chosen base in a transparent way.

The base we choose is the *Post correspondence problem* (PCP), an easy to formulate combinatorial problem concerning matching sequences of strings. Proven undecidable by Post in 1946 [34], PCP became a powerful tool in computability theory, often admitting elegant reductions. In previous work [12], we formalised a reduction from the halting problem of Turing machines to PCP in Coq, hence providing reliable evidence that the assumption that PCP is undecidable in the synthetic sense can be safely added to constructive type theory. Concretely, (locally) assuming that PCP is not co-enumerable implies that every problem it reduces to is not co-enumerable and hence undecidable.

CPP '19, January 14–15, 2019, Cascais, Portugal

ACM ISBN 978-1-4503-6222-1/19/01…$15.00
https://doi.org/10.1145/3293880.3294091

In this work, we use the outlined strategy to address another seminal decision problem in computability theory, namely the *Entscheidungsproblem*. Asking for a decision procedure for the validity of first-order formulas, it was programmatically posed by Hilbert and Ackermann in 1928 [11] and famously answered to the negative by Turing [44] and Church [8]. We verify a simple reduction following Floyd and Manna [31] from PCP to the Entscheidungsproblem for formulas in the universal-implicative fragment of first-order logic over a small signature. Furthermore, we give similar reductions to related problems such as intuitionistic and classical variants of natural deduction systems and satisfiability of formulas including negation. To this end, we develop the metatheory of first-order logic to an extent necessary to verify the reductions, including soundness and a generalised double negation translation. We refrain from a comprehensive account covering further standard results like completeness since they do not yield shorter undecidability proofs and (mostly) rely on additional assumptions.

As we formalise our results using the Coq proof assistant based on CIC, which comes with an impredicative universe of propositions, we deviate in some aspects from a development using alternative provers such as Agda based on pure MLTT. Most importantly, CIC allows for a propositional version of existence that hides witnesses from elimination into the computational fragment. Thus, an operation definable in Coq[1] we call *guarded minimisation* computing minimal witnesses for satisfiable decidable predicates on enumerable types is crucial for our results surrounding Post's theorem.

Contributions Our main contribution is a compact Coq formalisation[2] of elementary synthetic computability (900 lines of code), various decision problems in first-order logic (750 loc), and their respective undecidability (550 loc). All definitions and propositions in the PDF version of this paper are hyperlinked with the corresponding statements in the browsable HTML version of the formalisation. We elaborate on some formalisation tricks like tagged inductive types for fragments of syntax and deduction systems and an enumerability technique using cumulative lists in the discussion.

As mathematical results, we examine synthetic renderings of standard notions in computability theory such as decidability, enumerability, and many-one reductions and prove basic results culminating in the observed equivalence of Post's theorem and Markov's principle. Moreover, we give short reductions from PCP to every problem under consideration and verify their correctness purely constructively.

Finally, the text on its own functions as a surveying account of synthetic computability and undecidability in first-order logic, providing a compact presentation of the chosen reductions and an extensive discussion of related work.

[1] Variants of this operation can be found in common Coq libraries.
[2] https://www.ps.uni-saarland.de/extras/fol-undec

Outline The technical part of this paper is organised in two sections. First, in Section 2, we develop the basics of synthetic computability theory addressing decidability and enumerability, many-one reductions, Post's theorem and Markov's principle, the Post correspondence problem, and infinite data types. Secondly, in Section 3, we apply the synthetic framework to proving the undecidability of validity, provability, satisfiability, their intuitionistic counterparts, and classical provability of first-order formulas. We conclude in Section 4 with general comments as well as some remarks concerning the Coq formalisation, related work, and future work.

2 Elementary Synthetic Computability

We work in a constructive type theory providing a hierarchy of predicative universes $\mathbb{T}$ of computational types, an impredicative universe $\mathbb{P}$ of propositions, and a general scheme for inductive types subject to pattern matching and structural recursion. We employ the standard notation for (dependent) products, (dependent) sums, and logical connectives placed in $\mathbb{P}$. The inductive type of *booleans* is defined as $\mathbb{B} := \mathrm{tt} \mid \mathrm{ff}$, the *unit type* is defined as $\mathbb{1} := *$, and the *natural numbers* are defined as $\mathbb{N} := 0 \mid \mathrm{S}\, n$ for $n : \mathbb{N}$.

We define the *option type* over a type X as $O(X) := \emptyset \mid \ulcorner x \urcorner$ for $x : X$. Moreover, the type of *lists* over X is defined as $\mathcal{L}(X) := [] \mid x :: A$ for $x : X$ and $A : \mathcal{L}(X)$. We write $A + B$ for *concatenation*, $x \in A$ for *membership*, and $A \subseteq B$ for *inclusion*. Finally, we write $[f\, x \mid x \in A]$ for *map*, $[x \in A \mid p\, x]$ for *filter* and $A[n] : O(X)$ for *element access*.

2.1 Decidability and Enumerability

We begin by defining the standard notions of decidability and enumerability in the synthetic setting.

Definition 2.1. *Given a type X, a predicate $p : X \to \mathbb{P}$ is*

- decidable *if there is a decider $f : X \to \mathbb{B}$ such that $\forall x.\, p\, x \leftrightarrow f\, x = \mathrm{tt}$.*
- enumerable *if there is an enumerator $f : \mathbb{N} \to O(X)$ such that $\forall x.\, p\, x \leftrightarrow \exists n.\, f\, n = \ulcorner x \urcorner$.*

The complement *$\overline{p}$ is defined as $\overline{p} := \lambda x.\, \neg p x$ and we say that p is* co-enumerable *if $\overline{p}$ is enumerable and that p is* co-decidable *if $\overline{p}$ is decidable. These definitions generalise to predicates with more than one argument as one would expect.*

Since every function definable in a constructive type theory is inherently computable, there is no need to demand additional representations of deciders and enumerators in some explicit model of computation.

Fact 2.1 (Informative deciders). *From a decider for $p : X \to \mathbb{P}$ one can obtain a function of type $\forall x.\, p\, x + \neg p\, x$ and vice versa.*

Proof. Given a decider $f : X \to \mathbb{B}$ we easily establish a decision $d\, x : p\, x + \neg p\, x$ via case distinction on $f\, x$. Conversely, given a decision $d\, x : p\, x + \neg p\, x$ the function defined by $f\, x :=$ if $d\, x$ then tt else ff is a decider for p. □

Fact 2.2. *Decidable predicates are closed under conjunction, disjunction, and complement.*

Proof. Routine by deciding the logical connectives using their boolean equivalents. □

Note that the converse directions do not hold in general. For instance, the notions of decidability and co-decidability only coincide using classical assumptions.

Definition 2.2. *A type X is*

- enumerable *if there is an enumerator $f : \mathbb{N} \to O(X)$ such that $\forall x.\, \exists n.\, f\, n = \ulcorner x \urcorner$.*
- discrete *if there is an equality decider for $\lambda xy : X.\, x = y$.*
- *a* data type *if it is both enumerable and discrete.*

Fact 2.3. *Decidable predicates on enumerable types are enumerable and co-enumerable.*

Proof. Let $f_p : X \to \mathbb{B}$ be a decider for a predicate $p : X \to \mathbb{P}$ on a type X with enumerator $f_X : \mathbb{N} \to O(X)$. Then

$$f\, n := \text{match } f_X\, n \text{ with } \ulcorner x \urcorner \Rightarrow \text{if } f_p\, x \text{ then } \ulcorner x \urcorner \text{ else } \emptyset \mid \emptyset \Rightarrow \emptyset$$

is an enumerator for p. That decidable predicates are also co-enumerable follows with Fact 2.2. □

Fact 2.4. *$\mathbb{B}$, $\mathbb{N}$, and $\mathbb{N} \times \mathbb{N}$ are data types.*

Proof. Establishing $\mathbb{B}$ and $\mathbb{N}$ as data types as well as the discreteness of $\mathbb{N}\times\mathbb{N}$ are routine. Concerning the enumerability of $\mathbb{N} \times \mathbb{N}$, one first defines a function $L : \mathbb{N} \to \mathcal{L}(\mathbb{N} \times \mathbb{N})$ by

$$L\, 0 := [] \qquad L\, (\mathrm{S}\, n) := L\, n +\!\!+ [(k, l) \mid k, l \in [0, \ldots, n]].$$

Then $f\, n := (L\, n)[n]$ enumerates $\mathbb{N} \times \mathbb{N}$. □

Note that the function L in the previous proof is *cumulative* in the sense that $L\, n$ is a prefix of $L\, (\mathrm{S}\, n)$ for all n. This enumeration technique can be generalised as follows.

Fact 2.5 (List enumerators)**.** *Given a type X and a function $L : \mathbb{N} \to \mathcal{L}(X)$, one can obtain a function $f : \mathbb{N} \to O(X)$ such that $\forall x.\, (\exists n.\, f\, n = \ulcorner x \urcorner) \leftrightarrow (\exists n.\, x \in L\, n)$ and vice versa.*

Proof. Given an enumerator g for $\mathbb{N} \times \mathbb{N}$ and $L : \mathbb{N} \to \mathcal{L}(X)$

$$f\, n := \text{match } g\, n \text{ with } \ulcorner (k, l) \urcorner \Rightarrow (L\, k)[l] \mid \emptyset \Rightarrow \emptyset$$

defines f as claimed. Conversely, given $f : \mathbb{N} \to O(X)$

$$L\, n := \text{match } f\, n \text{ with } \ulcorner x \urcorner \Rightarrow [x] \mid \emptyset \Rightarrow []$$

defines L as claimed. □

Note that every function $L : \mathbb{N} \to \mathcal{L}(X)$ can trivially be made cumulative, so in particular we can assume every list enumerator to be cumulative. Employing list enumerators yields a convenient technique for establishing closure properties for enumerability regarding types and predicates.

Lemma 2.6. *If L is cumulative, then $L\, n \subseteq L\, m$ if $n \leq m$.*

Proof. By straightforward induction on $n \leq m$. □

Fact 2.7. *Enumerable types, discrete types, and data types are all closed under products, sums, options, and lists.*

Proof. The closure properties for discreteness are routine. Concerning enumerability, let L_X be a list enumerator for X. We exemplarily define a list enumerator for $\mathcal{L}(X)$:

$$L_{\mathcal{L}(X)}\, 0 := [[]]$$
$$L_{\mathcal{L}(X)}\, (\mathrm{S}\, n) := L_{\mathcal{L}(X)}\, n +\!\!+ [x :: A \mid x \in L_X\, n, A \in L_{\mathcal{L}(X)}\, n]$$

The proof that $\forall A : \mathcal{L}(X).\, \exists n.\, A \in L_{\mathcal{L}(X)}\, n$ is by induction on A. The case $A = []$ is trivial. Thus consider $x :: A$. We know that $x \in L_X\, n_1$ and by IH that $A \in L_{\mathcal{L}(X)}\, n_2$ for some n_1, n_2. By Lemma 2.6 we know that $x \in L_X\, (1 + n_1 + n_2)$ and that $A \in L_{\mathcal{L}(X)}\, (1+n_1+n_2)$ and thus $x :: A \in L_{\mathcal{L}(X)}\, (1+n_1+n_2)$. □

Fact 2.8. *Enumerable predicates*

(1) are closed under disjunction.
(2) on discrete types are closed under conjunction.

Proof.

(1) Let L_p and L_q be list enumerators for p and q, respectively. Then $L\, n := [x \mid x \in L_p\, n] +\!\!+ [x \mid x \in L_q\, n]$ is a list enumerator for $\lambda x.\, p\, x \vee q\, x$.
(2) Let L_p and L_q be list enumerators for p and q, respectively.

$$L\, 0 := [] \qquad L\, (\mathrm{S}\, n) := L\, n +\!\!+ [x \mid x \in L_p\, n, x \in L_q\, n]$$

is a list enumerator for $\lambda x.\, p\, x \wedge q\, x$. The discreteness of the domain of p and q is employed in the second equation to filter all $x \in L_p\, n$ that are also in $L_q\, n$. □

The last fact in this section states that domain and range of enumerable binary relations are enumerable.

Fact 2.9 (Projection)**.** *Let $p : X \to Y \to \mathbb{P}$ be an enumerable predicate. Then $\lambda x.\, \exists y.\, p\, x\, y$ and $\lambda y.\, \exists x.\, p\, x\, y$ are enumerable.*

Proof. Assume an enumerator for pairs (x, y) such that $p\, x\, y$. Projecting out x and y yields enumerators for $\lambda x.\, \exists y.\, p\, x\, y$ and for $\lambda y.\, \exists x.\, p\, x\, y$, respectively □

Note that most of the previous and upcoming statements can be simplified when only considering data types. For instance, enumerable predicates on data types are closed under disjunction and conjunction. We give the stronger statements for the sake of generality but remark that usual decision problems such as the ones in first-order logic addressed in this work are defined on enumerable and discrete domains. In Section 2.5, we show that these domains can in fact be encoded using the standard data type $\mathbb{N}$.

2.2 Many-One Reductions

Definition 2.3. *Given predicates $p : X \to \mathbb{P}$ and $q : Y \to \mathbb{P}$ we call a function $f : X \to Y$ a* (many-one) reduction *from p to q if $\forall x.\, p\, x \leftrightarrow q\, (f\, x)$. We say that p* reduces *to q and write $p \preccurlyeq q$ if there is a reduction from p to q.*

Again, this synthetic definition of reductions does not refer to a concrete model of computation but instead relies on the computability of definable functions in constructive type theory.

The next two facts establish that reducibility is a pre-order on predicates and that decidability and co-decidability transport along reductions.

Fact 2.10. *Reducibility is reflexive and transitive.*

Proof. Given a predicate $p : X \to \mathbb{P}$, the identity function $\mathsf{id}_X := \lambda x.\, x$ witnesses $p \preccurlyeq p$. Given a reduction f from p to q and a reduction g from q to r, composition $g \circ f := \lambda x.\, g\,(f\,x)$ witnesses $p \preccurlyeq r$. □

Fact 2.11. *Let $p \preccurlyeq q$. Then $\overline{p} \preccurlyeq \overline{q}$ and p is decidable if q is.*

Proof. The same function witnessing $p \preccurlyeq q$ witnesses $\overline{p} \preccurlyeq \overline{q}$. Moreover, given a reduction f from p to q and a decider g for q the function $g \circ f$ is a decider for p. □

Enumerability of predicates does not in general transport along reductions. We show a proposition stating the necessary conditions and remark that the statement again simplifies when only considering data types.

Lemma 2.12. *Let $p \preccurlyeq q$ such that the domain of p is enumerable and the domain of q is discrete. Then p is enumerable if q is enumerable.*

Proof. Let $f : X \to Y$ be a reduction from p to q such that $\forall x.\, p\,x \leftrightarrow q\,(f\,x)$ and $g : \mathbb{N} \to O(Y)$ be an enumerator for q with $\forall y.\, q\,y \leftrightarrow \exists n.\, g\,n = \ulcorner y \urcorner$. So $\forall x.\, p\,x \leftrightarrow \exists n.\, g\,n = \ulcorner f\,x \urcorner$ and since X is enumerable and Y is discrete we can apply Fact 2.9 to derive that $\lambda x.\, \exists n.\, g\,n = \ulcorner f\,x \urcorner$ and hence p is enumerable. □

Note that Fact 2.11 and Lemma 2.12 imply that p is also co-enumerable if q is co-enumerable under the stated conditions.

The previous facts are summarised by the following reduction theorem providing the standard strategy to establish the undecidability of decision problems used in Section 3.

Theorem 2.13 (Reduction). *Let $p \preccurlyeq q$ such that the domain of p is enumerable and p is not co-enumerable. Then:*

(1) q is neither decidable nor co-decidable.
(2) q is not co-enumerable if the domain of q is discrete.

Proof. (1) follows with Facts 2.11, and 2.3 and (2) follows with Facts 2.11 and 2.12. □

2.3 Post's Theorem and Markov's Principle

Recall that Fact 2.3 establishes that decidable predicates on data types are enumerable and co-enumerable. The converse direction, stating that predicates on data types are decidable if they are enumerable and co-enumerable, i.e. *Post's theorem*, is not provable constructively (cf. Section 4.5.3 in Troelstra and van Dalen [43] and [10]).

To precisely analyse the classical content of Post's theorem, we first establish a constructively provable but logically weaker statement. Its justification already contains the main algorithmic insight based on guarded linear search and parallel enumeration.

In this section, we call a predicate $p : X \to \mathbb{P}$ *logically decidable* if $\forall x.\, p\,x \vee \neg p\,x$ and *bi-enumerable* if both p and $\overline{p}$ are enumerable.

Lemma 2.14 (Guarded minimisation). *Let $p : \mathbb{N} \to \mathbb{P}$ be decidable. Then there is a function $\mu : (\exists n.\, p\,n) \to \mathbb{N}$ such that $p\,(\mu\,h)$ for all proofs h of $\exists n.\, p\,n$. In particular, μ can be constructed such that it yields the least witness for p.*

Proof. We outline a proof variant for the first claim from the Coq standard library[3]. Let $f : \mathbb{N} \to \mathbb{B}$ be a decider for p. We define an inductive predicate $A : \mathbb{N} \to \mathbb{P}$ with a proof constructor of type $\forall n.\, (f\,n = \mathrm{ff} \to A\,(\mathrm{S}\,n)) \to A\,n$. One first shows that $\forall n.\, p\,(n + k) \to A\,k$ by induction on k and then constructs a function $\mu' : \forall n.\, A\,n \to \Sigma m.\, p\,m$ via recursion on the proof of $A\,n$, exploiting that A is subject to singleton elimination. Then μ is obtained by composing the parts. Concerning the second claim, we show that μ' yields the least witness m above n using the full induction lemma for A with dependency on the proof of $A\,n$. □

Lemma 2.15. *Logically decidable bi-enumerable predicates on discrete types are decidable.*

Proof. Let X be discrete, $p : X \to \mathbb{P}$ be logically decidable, f be an enumerator for p, and g be an enumerator for $\overline{p}$. We construct a function $\forall x.\, p\,x + \neg p\,x$ (suffices by Lemma 2.1).

Fix x. From the assumptions we obtain $\exists n.\, f\,n = \ulcorner x \urcorner \vee g\,n = \ulcorner x \urcorner$. Since X is discrete, the predicate $\lambda n.\, f\,n = \ulcorner x \urcorner \vee g\,n = \ulcorner x \urcorner$ is decidable, and thus Lemma 2.14 gives us a number n such that $H : f\,n = \ulcorner x \urcorner \vee g\,n = \ulcorner x \urcorner$. We now distinguish five cases using the equality decider for X.

- $f\,n = \emptyset$ and $g\,n = \emptyset$. Contradiction with H.
- $f\,n = \ulcorner x \urcorner$. Then $p\,x$ since f is an enumerator for p.
- $f\,n = \ulcorner y \urcorner$ and $x \neq y$. Then $\neg p\,x$ by H since g is an enumerator for $\overline{p}$.
- $g\,n = \ulcorner x \urcorner$. Then $\neg p\,x$ since g is an enumerator for $\overline{p}$.
- $g\,n = \ulcorner y \urcorner$ and $x \neq y$. Then $p\,x$ by H since f is an enumerator for p. □

We now show that the additionally assumed logical decidability is exactly equivalent to *Markov's principle*. We call propositions P and predicates $p : X \to \mathbb{P}$ *stable* if $\neg\neg P \to P$ and $\forall x.\, \neg\neg p\,x \to p\,x$, respectively. Note that stability is transported along reductions:

Fact 2.16. *If $p \preccurlyeq q$ and q is stable, then p is stable.*

Proof. Let f be a reduction from p to q. Then for every x, the claim $\neg\neg p\,x \to p\,x$ is equivalent to $\neg\neg q\,(f\,x) \to q\,(f\,x)$, which holds since q is stable. □

[3] https://coq.inria.fr/library/Coq.Logic.ConstructiveEpsilon.html

We state Markov's principle as stability of the satisfiability of boolean sequences:

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

Note that excluded middle ($\mathsf{EM} := \forall P.\ P \vee \neg P$) implies that every proposition is stable, so it particularly implies MP. Moreover, it is well-known that MP is strictly weaker than EM (e.g. [20]). We first consider some alternative characterisations of MP.

Fact 2.17. *The following propositions are equivalent:*

(1) MP.
(2) Satisfiability of decidable predicates over $\mathbb{N}$ *is stable.*
(3) Satisfiability of enumerable predicates is stable.

Proof. (1) → (2) is straightforward.

(2) → (3). We need to show that $\exists x.\ \exists n.\ f\ n = \ulcorner x \urcorner$ is stable. Holds since $\exists n.\ \exists x.\ f\ n = \ulcorner x \urcorner$ is stable by (2).

(3) → (1). Let $f : \mathbb{N} \to \mathbb{B}$. So $\lambda n.\ f\ n = \mathrm{tt}$ is decidable and hence enumerable by Fact 2.3. Thus $\exists n.\ f\ n = \mathrm{tt}$ is stable. □

Next, we establish the following consequence of MP:

Fact 2.18. *Assuming* MP*, enumerable predicates on discrete types are stable.*

Proof. We need to show that $\exists n.\ f\ n = \ulcorner x \urcorner$ is stable. Follows with (2) of Fact 2.17 since $\lambda n.\ f\ n = \ulcorner x \urcorner$ is decidable. □

Finally, we show that MP is exactly the assumption necessary to establish Post's theorem.

Fact 2.19. MP *is equivalent to the logical decidability of bi-enumerable predicates on discrete types.*

Proof. (→). Let f be an enumerator for p and g be an enumerator for $\overline{p}$. It suffices to show $\forall x.\ \exists n.\ f\ n = \ulcorner x \urcorner \vee g\ n = \ulcorner x \urcorner$. So fix x and note that $\lambda n.\ f\ n = \ulcorner x \urcorner \vee g\ n = \ulcorner x \urcorner$ is decidable as the domain of p is discrete. By MP and (2) of Fact 2.17 we just show $\neg\neg\exists n.\ fn = \ulcorner x \urcorner \vee gn = \ulcorner x \urcorner$. Hence we assume $H : \neg\exists n.\ fn = \ulcorner x \urcorner \vee gn = \ulcorner x \urcorner$ and derive a contradiction. Since $\neg\neg(p\,x \vee \neg p\,x)$ is provable, we can assume $p\,x \vee \neg p\,x$. But then the enumerators f and g yield $(\exists n.\ f\ n = \ulcorner x \urcorner) \vee (\exists n.\ g\ n = \ulcorner x \urcorner)$, which contradicts H.

(←). Applying (2) of Fact 2.17, we show that every decidable predicate on numbers is stable. So let $p : \mathbb{N} \to \mathbb{P}$ and $H : \neg\neg\exists n.\ p\ n$. We show $\exists n.\ p\ n$, for which it suffices to show that $\lambda m.\ \exists n.\ p\ n$ is logically decidable. Now given (2) one just needs to justify that $\lambda m.\ \exists n.\ p\ n$ is bi-enumerable. The enumerability of $\lambda m.\ \exists n.\ p\ n = \lambda m.\ \exists n.\ (\lambda mn.\ p\ n)\ m\ n$ follows with Fact 2.9. The co-enumerability of $\lambda m.\ \exists n.\ p\ n$ follows from $\neg(\exists n.\ p\ n) \leftrightarrow \bot$, which follows with H. □

Theorem 2.20. *Post's theorem is equivalent to* MP.

Proof. Assuming Post's theorem, we know that every bi-enumerable predicate on a discrete type is decidable. In particular, such predicates are logically decidable and thus MP follows from Fact 2.19. Conversely, Fact 2.19 and Lemma 2.15 together yield that MP implies Post's theorem. □

2.4 The Post Correspondence Problem

We now define the Post correspondence problem, the starting point for the reductions in the present work. In this context, a *string* s is a list over $\mathbb{B}$, a *card* s/t is a pair of strings, and a *stack* S is a list of cards.

Definition 2.4. *We define the derivability* $S \triangleright s/t$ *of cards* s/t *from a stack* S *and the Post correspondence problem* PCP *on the the type* $\mathbb{S} := \mathcal{L}(\mathcal{L}(\mathbb{B}) \times \mathcal{L}(\mathbb{B}))$ *of stacks inductively:*

$$\frac{s/t \in S}{S \triangleright s/t} \qquad \frac{S \triangleright u/v \quad s/t \in S}{S \triangleright s +\!\!+ u/t +\!\!+ v} \qquad \frac{S \triangleright s/s}{\mathrm{PCP}\ S}$$

Informally, the argument stack S fixes a set of cards for which any solution constitutes an arrangement of arbitrarily many uses of cards such that the upper and lower trace agree.

The main result of a previous paper [12] is a formal reduction from the halting problem for Turing machines to PCP, with a more traditional definition of PCP by

$$\lambda S.\ \exists A \subseteq S.\ A \neq [] \wedge A^1 = A^2$$

where A^1 is the upper and A^2 the lower trace of a stack defined recursively by

$$[s_1/t_1, \ldots, s_k/t_k]^1 := s_1 +\!\!+ \ldots +\!\!+ s_k,$$
$$[s_1/t_1, \ldots, s_k/t_k]^2 := t_1 +\!\!+ \ldots +\!\!+ t_k.$$

The more traditional definition is equivalent to our inductive characterisation:

Lemma 2.21. $S \triangleright s/t \to \exists A \subseteq S.\ A \neq [] \wedge A^1 = s \wedge A^2 = t$

Proof. By induction on the derivation $S \triangleright s/t$. □

Fact 2.22. $(\exists A \subseteq S.\ A \neq [] \wedge A^1 = A^2) \leftrightarrow \mathrm{PCP}\ S$

Proof. Suppose S has a solution $A \subseteq S$ with $A \neq []$ and $A^1 = A^2$. By induction on A we can reconstruct the derivation $S \triangleright A^1/A^2$. The other direction follows from Lemma 2.21. □

We now examine PCP from the computational perspective developed in the previous sections. On the positive side, PCP ranges over a computational domain and is enumerable.

Fact 2.23. $\mathbb{S}$ *is a data type.*

Proof. $\mathbb{S} = \mathcal{L}(\mathcal{L}(\mathbb{B}) \times \mathcal{L}(\mathbb{B}))$ is a data type since $\mathbb{B}$ is a data type (Fact 2.4) and since data types are closed under lists and products (Fact 2.7). □

Lemma 2.24. $\lambda S(s,t).\ S \triangleright s/t$ *is enumerable.*

Proof. Let $L_{\mathbb{S}}$ be the list enumerator for $\mathbb{S}$ from the previous fact. The wanted list enumerator is then defined as follows:

$$L\,0 := []$$
$$L\,(\mathrm{S}\,n) := L\,n +\!\!+ [(S,(x,y)) \mid S \in L_{\mathbb{S}}\,n, (x,y) \in S] +\!\!+ [(S,(x +\!\!+ u, y +\!\!+ v)) \mid (S,(u,v)) \in L\,n, (x,y) \in S]$$

□

Fact 2.25. PCP *is enumerable.*

Proof. $\lambda S(s,t).\ S \triangleright s/t$ is enumerable by the previous lemma. Moreover, $\lambda S(s,t).\ s = t$ is decidable, has enumerable domain, and is thus enumerable by Fact 2.3. By closure of enumerability under conjunction (Fact 2.8 (2)) and projection (Fact 2.9) it follows that PCP is enumerable. □

On the negative side, a classical standard result states that PCP is undecidable. However, given that constructive type theory is consistent with the assumption that every predicate is decidable in the sense of Definition 2.1, this undecidability result is not provable in our setting. A way to still get a meaningful technique for establishing negative results is to postulate the undecidability of a concrete problem. Using such a base, further problems can be shown undecidable utilising the usual means of computable reductions.

For instance, we believe that it is consistent to assume that PCP is not co-enumerable, even in the context of MP or EM. We denote this undecidability assumption by UA and state the strategic consequences as follows:

Lemma 2.26. *Assuming* UA, PCP *and every predicate* PCP *reduces to are undecidable.*

Proof. Follows with Fact 2.3 and Theorem 2.13. □

2.5 Infinite Data Types

Explicit models of computation usually operate on a fixed domain such as natural numbers (general recursive functions), strings (Turing machines), or syntactic encodings (λ-calculus). In the implicit approach chosen for this paper, however, the considered problems range over several infinite data types like stacks (in the case of PCP) or logical formulas (in the case of FOL). The purpose of this section is to illustrate that all such infinite data types can generally be reduced to the concrete case of natural numbers.

A type X is *infinite* if there is an injection $F : \mathbb{N} \to X$ and *generating* if for every list $A : \mathcal{L}(X)$ there is some x with $x \notin A$. We first establish that both notions are equivalent over data types.

Fact 2.27. *Every data type is infinite iff it is generating.*

Proof. Let X be a data type and let $F : \mathbb{N} \to X$ be an injection. Then for every list A the list $[F\,n \mid n \in [0, \ldots, |A|]]$ contains an element not in A, so X is generating.

Conversely, suppose that X is generating. First note that we can assume the enumerator f_X for X to have type $\mathbb{N} \to X$ given that X is clearly inhabited. Using guarded minimisation (Lemma 2.14), we obtain a function $g : \mathcal{L}(X) \to X$ such that $g\,A \notin A$ for all A. We then set

$$L\,0 := [] \qquad L\,(\mathrm{S}\,n) := L\,n +\!\!+ [g\,(L\,n)] \qquad F\,n := g\,(L\,n)$$

and show that F is an injection. So let $F\,n = F\,n'$ and suppose w.l.o.g. $n < n'$. Then $F\,n \in L\,n'$ and hence $F\,n' \in L\,n'$, the latter contradicting the specification of g. Thus $n = n'$ is the only possible case. □

In fact, the injection $F : \mathbb{N} \to X$ constructed in the previous proof is already surjective.

Theorem 2.28. *Every infinite data type is in bijection to* $\mathbb{N}$.

Proof. Let X be infinite and x be in X, we show that there is a number n with $F\,n = x$ for the injection $F : \mathbb{N} \to X$ defined as in Fact 2.27. One can show that $f_X\,n \in L\,(\mathrm{S}\,n)$ by complete induction for n using the fact that g was defined by an operation yielding least witnesses. So since there is n' with $f_X\,n' = x$, we know that $x \in L\,(\mathrm{S}\,n')$ and can hence find some $n \leq n'$ with $F\,n = x$ as desired.

Note that using guarded minimisation again, the sujectivity of F can be turned into an actual inverse $F^{-1} : X \to \mathbb{N}$ with $F\,(F^{-1}\,x) = x$ and $F^{-1}\,(F\,n) = n$. □

We exemplarily apply this method to the type $\mathbb{S}$ of stacks.

Fact 2.29. $\mathbb{S}$ *is infinite.*

Proof. An injection from $\mathbb{N}$ to $\mathbb{S}$ can be defined by mapping a number n to the stack consisting of n-times the card $\mathtt{tt}/\mathtt{tt}$. □

Corollary 2.30. $\mathbb{S}$ *is in bijection to* $\mathbb{N}$.

This concludes our development of elementary synthetic computability and we move on to its applications in the metatheory of first-order logic.

3 Undecidability of First-Order Logic

We consider first-order formulas over a fixed signature containing a constant individual symbol e, two unary function symbols $f_{\mathtt{tt}}$ and $f_{\mathtt{ff}}$, a constant proposition symbol Q, and a binary relation symbol P. Formally, we accommodate *terms* as an inductive type $\mathcal{T}$ by

$$\tau : \mathcal{T} := x \mid a \mid e \mid f_{\mathtt{tt}}\,\tau \mid f_{\mathtt{ff}}\,\tau$$

where variables x, y, z and parameters a, b, c range over $\mathbb{N}$. Concerning the *formulas*, we first consider an inductive type $\mathcal{F}$ expressing the universal-implicative fragment:

$$\varphi, \psi : \mathcal{F} := Q \mid P\,\tau_1\,\tau_2 \mid \varphi \dot{\to} \psi \mid \dot{\forall} x.\,\varphi$$

Fact 3.1. $\mathcal{T}$ *and* $\mathcal{F}$ *are infinite data types.*

Proof. Discreteness of both types is straightforward and enumerability of both types can be shown using a standard construction via list enumerators. $\mathcal{T}$ is infinite because variables provide an injection from $\mathbb{N}$ and for $\mathcal{F}$ the function $\lambda n.\,P\,n\,n$ is a witnessing injection. □

3.1 Validity

We begin by formalising the undecidability of the Entscheidungsproblem whether or not a given formula is valid. Therefore we establish the standard (Tarski) semantics of formulas with the next three definitions:

Definition 3.1. *An* interpretation $\mathcal{I}$ *consists of a domain* $D : \mathbb{T}$ *together with a parameter assignment* $\eta : \mathbb{N} \to D$ *and symbol interpretations*

$$e^{\mathcal{I}} : D \qquad Q^{\mathcal{I}} : \mathbb{P}$$
$$f_{\mathrm{tt}}^{\mathcal{I}}, f_{\mathrm{ff}}^{\mathcal{I}} : D \to D \qquad P^{\mathcal{I}} : D \to D \to \mathbb{P}.$$

Definition 3.2. *Given an interpretation* $\mathcal{I}$*, we extend* environments $\rho : \mathbb{N} \to D$ *to* term evaluations $\hat{\rho} : \mathcal{T} \to D$ *by*

$$\hat{\rho}\, x := \rho\, x \qquad \hat{\rho}\, e := e^{\mathcal{I}}$$
$$\hat{\rho}\, a := \eta\, a \qquad \hat{\rho}\,(f_b\, \tau) := f_b^{\mathcal{I}}\,(\hat{\rho}\, \tau).$$

Definition 3.3. *Given an interpretation* $\mathcal{I}$ *together with an environment* ρ*, we define the* satisfaction relation $\rho \vDash_{\mathcal{I}} \varphi$ *by*

$$\rho \vDash_{\mathcal{I}} Q := Q^{\mathcal{I}} \qquad \rho \vDash_{\mathcal{I}} \varphi \dot{\to} \psi := \rho \vDash_{\mathcal{I}} \varphi \to \rho \vDash_{\mathcal{I}} \psi$$
$$\rho \vDash_{\mathcal{I}} P\, \tau_1\, \tau_2 := P^{\mathcal{I}}\,(\hat{\rho}\, \tau_1)(\hat{\rho}\, \tau_2) \qquad \rho \vDash_{\mathcal{I}} \dot{\forall} x.\, \varphi := \forall d : D.\, \rho[x := d] \vDash_{\mathcal{I}} \varphi$$

where $\rho[x := d]$ *denotes the environment mapping* x *to* d *and all other variables in accordance to* ρ*. We say that* $\mathcal{I}$ satisfies φ *and write* $\vDash_{\mathcal{I}} \varphi$ *if* $\rho \vDash_{\mathcal{I}} \varphi$ *for all environments* ρ*. A formula* φ *is* valid *if* $\vDash_{\mathcal{I}} \varphi$ *for all interpretations* $\mathcal{I}$*.*

The following simplifies managing multiple assumptions:

Definition 3.4. *Prepending a list* A *of formulas is defined by*

$$[]\dot{\to}\varphi := \varphi \qquad (\psi :: A)\dot{\to}\varphi := \psi\dot{\to}(A\dot{\to}\varphi).$$

Fact 3.2. $\rho \vDash_{\mathcal{I}} A\dot{\to}\varphi \leftrightarrow ((\forall \psi \in A.\, \rho \vDash_{\mathcal{I}} \psi) \to \rho \vDash_{\mathcal{I}} \varphi)$

Proof. By induction on A. □

We say that an environment ρ over an interpretation $\mathcal{I}$ satisfies a list A and write $\rho \vDash_{\mathcal{I}} A$ whenever $\rho \vDash_{\mathcal{I}} \varphi$ for every $\varphi \in A$. We use the analogous notation $\vDash_{\mathcal{I}} A$ for interpretations.

We now show that validity of the fragment $\mathcal{F}$ of first-order logic is undecidable by constructing a reduction from PCP. We follow the proof from the textbook of Manna [31], who attributes the original proof idea to Floyd. The key idea is to encode strings and card derivations into first-order language using the non-logical symbols of the signature. We define the term encoding $\overline{s}$ of a string s by:

$$[] \mathbin{+\!\!+} \tau := \tau \qquad (b :: s) \mathbin{+\!\!+} \tau := f_b\,(s \mathbin{+\!\!+} \tau) \qquad \overline{s} := s \mathbin{+\!\!+} e$$

So for instance we have $\overline{\mathrm{tt}\,\mathrm{ff}\,\mathrm{ff}\,\mathrm{tt}} = f_{\mathrm{tt}}\,(f_{\mathrm{ff}}\,(f_{\mathrm{ff}}\,(f_{\mathrm{tt}}\, e)))$.

We now fix a stack S for the remainder of this text. The cards derivable from S give rise to a standard interpretation over boolean strings.

Definition 3.5. *We define the standard interpretation* $\mathcal{B}$ *with domain* $\mathcal{L}(\mathbb{B})$*, parameter assignment* $\eta\, a := []$*, and*

$$e^{\mathcal{B}} := [] \qquad Q^{\mathcal{B}} := \mathrm{PCP}\, S$$
$$f_b^{\mathcal{B}}\, s := b :: s \qquad P^{\mathcal{B}}\, s\, t := S \triangleright s/t.$$

The following lemma lists some expected properties of $\mathcal{B}$.

Lemma 3.3. *Let* ρ *be any variable environment in* $\mathcal{B}$*. Then* $\hat{\rho}\,(s \mathbin{+\!\!+} \tau) = s \mathbin{+\!\!+} \hat{\rho}\, \tau$*,* $\hat{\rho}\, \overline{s} = s$*, and* $\rho \vDash_{\mathcal{B}} P\, \tau_1\, \tau_2 \leftrightarrow S \triangleright \hat{\rho}\, \tau_1 / \hat{\rho}\, \tau_2$*.*

Proof. Immediate by construction. □

Fact 3.4. $\mathrm{MP} \to \neg\neg\rho \vDash_{\mathcal{B}} \varphi \to \rho \vDash_{\mathcal{B}} \varphi$

Proof. It suffices to show that the satisfiability of the atomic formulas Q and $P\, \tau_1\, \tau_2$ is stable, which follows from Fact 2.18 and the enumerability of PCP and $S \triangleright s/t$. □

Next, we construct a formula φ_S such that PCP S iff φ_S is valid as follows:

$$\varphi_1 := [P\, \overline{s}\, \overline{t} \mid s/t \in S]$$
$$\varphi_2 := [\dot{\forall} xy.\, P\, x\, y \dot{\to} P\,(s \mathbin{+\!\!+} x)\,(t \mathbin{+\!\!+} y) \mid s/t \in S]$$
$$\varphi_3 := \dot{\forall} x.\, P\, x\, x \dot{\to} Q$$
$$\varphi_S := \varphi_1 \dot{\to} \varphi_2 \dot{\to} \varphi_3 \dot{\to} Q$$

Note that φ_1 contains formulas representing the first constructor of the derivability relation $S \triangleright s/t$. Moreover, φ_2 represents the second constructor and φ_3 represents the single constructor of PCP. So an interpretation satisfies φ_S if it deems S to admit a solution. Since $\mathcal{B}$ correctly interprets the predicate symbols P and Q as derivability and solvability, respectively, it satisfies the constructor representations.

Fact 3.5. $\vDash_{\mathcal{B}} \varphi_1$*,* $\vDash_{\mathcal{B}} \varphi_2$*, and* $\vDash_{\mathcal{B}} \varphi_3$*.*

Proof. Let $P\, \overline{s}\, \overline{t} \in \varphi_1$ for a card $s/t \in S$, then Lemma 3.3 yields $\vDash_{\mathcal{B}} P\, \overline{s}\, \overline{t}$. Similarly, let $\dot{\forall} xy.\, P\, x\, y \dot{\to} P\,(s \mathbin{+\!\!+} x)\,(t \mathbin{+\!\!+} y) \in \varphi_2$ for a card $s/t \in S$. Applying Lemma 3.3, we have to show that $S \triangleright (s \mathbin{+\!\!+} u)/(t \mathbin{+\!\!+} v)$ for all strings u and v with $S \triangleright u/v$ which is exactly the second rule for derivability. Finally, we have to show that $\forall s.\, P^{\mathcal{B}}\, s\, s \to Q^{\mathcal{B}}$. This is a triviality given the definitions of $P^{\mathcal{B}}$ and $Q^{\mathcal{B}}$. □

It follows that S admits a solution if $\mathcal{B}$ satisfies φ_S.

Lemma 3.6. $\rho \vDash_{\mathcal{B}} \varphi_S \to \mathrm{PCP}\, S$

Proof. Assuming $\rho \vDash_{\mathcal{B}} \varphi_S$ and since $\mathcal{B}$ satisfies φ_1, φ_2, and φ_3 by Fact 3.5, we know that $\rho \vDash_{\mathcal{B}} Q$. Hence PCP S holds by definition of $\rho \vDash_{\mathcal{B}} Q$. □

Conversely, we show that φ_1 and φ_2 correctly axiomatise derivations in arbitrary interpretations.

Lemma 3.7. $S \triangleright s/t \to \rho \vDash_{\mathcal{I}} \varphi_1 \dot{\to} \varphi_2 \dot{\to} P\, \overline{s}\, \overline{t}$

Proof. By induction on $S \triangleright s/t$ using Fact 3.2. □

It follows that φ_S exactly expresses that S has a solution.

Theorem 3.8. PCP S *iff* φ_S *is valid.*

Proof. Suppose there is s with $S \triangleright s/s$. Then by Lemma 3.7 $\rho \vDash_{\mathcal{I}} \varphi_1 \dot{\to} \varphi_2 \dot{\to} P\, \overline{s}\, \overline{s}$ holds for all environments ρ in all interpretations $\mathcal{I}$. It follows that $\rho \vDash_{\mathcal{I}} \varphi_1 \dot{\to} \varphi_2 \dot{\to} \varphi_3 \dot{\to} Q$ and hence that φ_S is valid.

Now suppose that φ_S is valid. Then in particular $\mathcal{B}$ together with the trivial environment $\rho\, x := []$ satisfies φ_S. Thus PCP S by Lemma 3.6. □

Corollary 3.9. PCP *reduces to validity. Thus, assuming* UA*, validity is undecidable and not co-enumerable.*

3.2 Minimal Provability

It is well-known that validity and provability (in a reasonable deduction system) of first-order formulas coincide (in a classical metatheory). So classically, the previously established undecidability of validity directly implies the undecidability of provability. However, since we work in constructive type theory, this strategy is not available and we instead show that the same reduction as in the previous section directly yields the undecidability of a minimal (and in fact incomplete) natural deduction system.

We denote the list of *parameters* occuring in a term τ, a formula φ, or a list A of formulas by $\mathcal{P}(\tau)$, $\mathcal{P}(\varphi)$, and $\mathcal{P}(A)$, respectively. Moreover, we denote the list of *variables* in a term τ by $\mathcal{V}(\tau)$. We interpret variable lists as finite sets and admit set-theoretic notation such as $\mathcal{P}(\varphi) \cup \mathcal{P}(A)$ and $\mathcal{V}(\tau) = \emptyset$. We define the standard capturing *substitution* φ^x_τ of a variable x by a term τ recursively:

$$\begin{aligned} x^x_\tau &:= \tau & Q^x_\tau &:= Q \\ y^x_\tau &:= y & (P\,\tau_1\,\tau_2)^x_\tau &:= P\,(\tau_1)^x_\tau\,(\tau_2)^x_\tau \\ a^x_\tau &:= a & (\varphi \dot{\to} \psi)^x_\tau &:= \varphi^x_\tau \dot{\to} \psi^x_\tau \\ e^x_\tau &:= e & (\dot\forall x.\,\varphi)^x_\tau &:= \dot\forall x.\,\varphi \\ (f_b\,\tau')^x_\tau &:= f_b\,(\tau')^x_\tau & (\dot\forall y.\,\varphi)^x_\tau &:= \dot\forall y.\,\varphi^x_\tau \end{aligned}$$

Definition 3.6. *We define a* minimal natural deduction *system $A \vdash \varphi$ for formulas in $\mathcal{F}$ inductively by*

$$\frac{\varphi \in A}{A \vdash \varphi}\ A \qquad \frac{\varphi :: A \vdash \psi}{A \vdash \varphi \dot{\to} \psi}\ II \qquad \frac{A \vdash \varphi \dot{\to} \psi \quad A \vdash \varphi}{A \vdash \psi}\ IE$$

$$\frac{A \vdash \varphi^x_a \quad a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A)}{A \vdash \dot\forall x.\,\varphi}\ AI \qquad \frac{A \vdash \dot\forall x.\,\varphi \quad \mathcal{V}(\tau) = \emptyset}{A \vdash \varphi^x_\tau}\ AE$$

and say that a formula φ is provable *if $\vdash \varphi$ (short for $[] \vdash \varphi$).*

Note that minimal deduction cannot be complete for Tarski semantics as it does not contain a classical rule such as Peirce's law. A deduction system that could be shown complete (classically) will be defined in Section 3.5.

A key property of any deduction system is soundness w.r.t. to a chosen semantics. In fact, establishing soundness first allows for a simple undecidability reduction for provability. To prepare this strategy, we establish some properties of the satisfaction relation explaining the side conditions in (AI) and (AE). In the following, let $\mathcal{I}[a := d]$ be the interpretation obtained by replacing the parameter assignment η of an interpretation $\mathcal{I}$ by $\eta[a := d]$.

Lemma 3.10. *Let φ be a formula and $\mathcal{I}, \mathcal{I}'$ be interpretations over the same domain only differing in the interpretation of parameters not in $\mathcal{P}(\varphi)$. Further suppose ρ and ρ' are pointwise equal environments. Then $\rho \models_\mathcal{I} \varphi$ iff $\rho' \models_{\mathcal{I}'} \varphi$.*

Proof. By induction on φ using an analogous fact for term evaluations. □

Lemma 3.11. $\rho \models_\mathcal{I} A \leftrightarrow \rho \models_{\mathcal{I}[a:=d]} A$ *if* $a \notin \mathcal{P}(A)$.

Proof. Follows with Lemma 3.10. □

Lemma 3.12. $\rho[x := \hat\rho\,\tau] \models_\mathcal{I} \varphi \leftrightarrow \rho \models_\mathcal{I} \varphi^x_\tau$ *if* $\mathcal{V}(\tau) = \emptyset$.

Proof. By induction on φ with ρ generalised. All cases but universal quantification are straightforward. Concerning formulas $\dot\forall y.\,\varphi$, we distinguish two cases. If $x = y$, it suffices to show for all d that

$$\rho[x := \hat\rho\,\tau][x := d] \models_\mathcal{I} \varphi \leftrightarrow \rho[x := d] \models_\mathcal{I} \varphi$$

which follows with Lemma 3.10. If $x \neq y$, it suffices to show

$$\rho[x := \hat\rho\,\tau][y := d] \models_\mathcal{I} \varphi \leftrightarrow \rho[y := d] \models_\mathcal{I} \varphi^x_\tau$$

where we replace the right-hand side by $\rho'[x := \hat\rho'\,\tau] \models_\mathcal{I} \varphi$ using the inductive hypothesis for $\rho' := \rho[y := d]$. Now since $\mathcal{V}(\tau) = \emptyset$ we know that $\hat\rho'\,\tau = \hat\rho\,\tau$ and thus conclude the equivalence to the left-hand side with Lemma 3.10. □

Lemma 3.13. $\rho[x := d] \models_\mathcal{I} \varphi \leftrightarrow \rho \models_{\mathcal{I}[a:=d]} \varphi^x_a$ *if* $a \notin \mathcal{P}(\varphi)$.

Proof. Assume $a \notin \mathcal{P}(\varphi)$. By Lemma 3.12 it suffices to show $\rho[x := d] \models_\mathcal{I} \varphi \leftrightarrow \rho[x := \hat\rho\,a] \models_{\mathcal{I}[a:=d]} \varphi$. This follows immediately from Lemma 3.10 given that $a \notin \mathcal{P}(\varphi)$ and that $\hat\rho\,a = d$ in the interpretation $\mathcal{I}[a := d]$. □

The soundness of the minimal natural deduction system now follows easily.

Fact 3.14. *If $A \vdash \varphi$ then $A \dot{\to} \varphi$ is valid.*

Proof. By induction on the derivation of $A \vdash \varphi$. The cases (A), (II), and (IE) are straightforward. Consider a derivation by (AI), so we have that $A \dot{\to} \varphi^x_a$ is valid for some $a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A)$ and want to show that $A \dot{\to} \dot\forall x.\,\varphi$ is valid. By Fact 3.2 this means to show $\rho[x := d] \models_\mathcal{I} \varphi$ for all ρ with $\rho \models_\mathcal{I} A$. Now let $\mathcal{I}' := \mathcal{I}[a := d]$, then by Lemma 3.11 we also have $\rho \models_{\mathcal{I}'} A$ given that $a \notin \mathcal{P}(A)$. Hence, since $A \dot{\to} \varphi^x_a$ is valid we can derive $\rho \models_{\mathcal{I}'} \varphi^x_a$ and thus conclude $\rho[x := d] \models_\mathcal{I} \varphi$ with Lemma 3.13 given that $a \notin \mathcal{P}(\varphi)$.

Now consider a derivation by (AE) where we have that $A \dot{\to} \dot\forall x.\,\varphi$ is valid and want to show that $A \dot{\to} \varphi^x_\tau$ is valid for all terms τ with $\mathcal{V}(\tau) = \emptyset$. So for any environment ρ satisfying A we have to show $\rho \models_\mathcal{I} \varphi^x_\tau$. Since $A \dot{\to} \dot\forall x.\,\varphi$ is valid, in particular $\rho[x := \hat\rho\,\tau] \models_\mathcal{I} \varphi$. Then $\rho \models_\mathcal{I} \varphi^x_\tau$ follows from Lemma 3.12 given that $\mathcal{V}(\tau) = \emptyset$. □

Corollary 3.15 (Soundness)**.** *Every provable formula is valid.*

Now consider the formula φ_S as defined in Section 3.1 and define the context $A_S := \varphi_3 :: \varphi_2 :: \varphi_1$ to be the list containing all premises of φ_S. Then every encoded card derivation is provable from A_S.

Lemma 3.16. $S \triangleright s/t \to A_S \vdash P\,\overline{s}\,\overline{t}$

Proof. By induction on the derivation $S \triangleright s/t$. In the base case we have $s/t \in S$ and $P\,\overline{s}\,\overline{t}$ is among the assumptions in φ_1 and hence provable from A_S by (A). In the inductive step we have $A_S \vdash P\,\overline{u}\,\overline{v}$ as inductive hypothesis and want to prove $A_S \vdash P\,(\overline{s +\!\!+ u})\,(\overline{t +\!\!+ v})$ for a given card $s/t \in S$.

From the corresponding assumption for s/t in φ_2 we get that $A_S \vdash \dot\forall xy.\, P\,x\,y \dot\to P\,(s \mathbin{+\!\!+} x)\,(t \mathbin{+\!\!+} y)$ by (A). Now we can use (AE) twice for $x := \overline{u}$ and $y := \overline{v}$ and (IE) for the inductive hypothesis to establish the goal. □

Theorem 3.17. PCP S *iff* φ_S *is provable.*

Proof. Let PCP S, so there is s with $S \triangleright s/s$. After applying (II) multiple times we have to show $A_S \vdash Q$. By (A) we have $A_S \vdash \dot\forall x.\, P\,x\,x \dot\to Q$. Now Lemma 3.16 yields $A_S \vdash P\,\overline{s}\,\overline{s}$, so we just have to use (AE) and (IE) to conclude the proof.

Now suppose that φ_S is provable. By soundness (Corollary 3.15) we know that φ_S is valid and conclude PCP S by Theorem 3.8. □

Corollary 3.18. PCP *reduces to provability. Thus, assuming* UA, *provability is undecidable and not co-enumerable.*

On the positive side, provable formulas can be enumerated.

Fact 3.19. *Provability is enumerable.*

Proof. We show that for all A, the predicate $\lambda\varphi.\, A \vdash \varphi$ is enumerable by defining a list enumerator L_A as follows:

$$\begin{aligned} L_A\,0 &:= [\varphi \mid \varphi \in A] \\ L_A\,(\mathrm{S}\,n) &:= L\,n \mathbin{+\!\!+} \left[\varphi \dot\to \psi \mid \varphi \in L_{\mathcal{F}}\,n, \psi \in L_{\varphi::A}\,n\right] \\ &\quad \mathbin{+\!\!+} \left[\psi \mid \varphi \in L_A\,n, \varphi \dot\to \psi \in L_A\,n\right] \\ &\quad \mathbin{+\!\!+} [\dot\forall x.\, \varphi \mid \varphi \in L_{\mathcal{F}}\,n, x \in L_{\mathbb{N}}\,n, a \in L_{\mathbb{N}}\,n, \\ &\qquad\qquad a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A) \wedge \varphi_a^x \in L_A\,n] \\ &\quad \mathbin{+\!\!+} [\varphi_\tau^x \mid x \in L_{\mathbb{N}}\,n, \tau \in L_{\mathcal{T}}\,n, \varphi \in L_{\mathcal{F}}\,n, \\ &\qquad\qquad \mathcal{V}(\tau) = \emptyset \wedge \dot\forall x.\, \varphi \in L_A\,n] \end{aligned}$$

$A \vdash \varphi \to \exists n.\, \varphi \in L_A\,n$ follows by induction on $A \vdash \varphi$. For the other direction, we prove $\forall A.\, \varphi \in L_A\,n \to A \vdash \varphi$ by induction on n. Thus, $L_{[]}$ enumerates provable formulas. □

3.3 Satisfiability

A formula φ is *satisfiable* if there is an environment ρ in an interpretation $\mathcal{I}$ such that $\rho \models_{\mathcal{I}} \varphi$. In the negation-free fragment $\mathcal{F}$ of first-order logic we have considered so far, every formula is actually satisfied by a trivial interpretation.

Definition 3.7. *We define the trivial interpretation* $\mathcal{U}$ *with domain* $\mathbb{1}$, *parameter assignment* $\eta\,a := *$, *and*

$$e^{\mathcal{U}} := * \qquad Q^{\mathcal{U}} := \top$$
$$f_b^{\mathcal{U}}\,s := * \qquad P^{\mathcal{U}}\,s\,t := \top.$$

Fact 3.20. *Let* φ *be a formula of the fragment* $\mathcal{F}$. *Then* $\models_{\mathcal{U}} \varphi$.

Proof. By induction on φ. □

So satisfiability is trivially decidable in the absence of negation. We thus switch to an extended representation $\mathcal{F}_\bot$ of first-order formulas containing a symbol $\dot\bot$ for falsity

$$\varphi, \psi : \mathcal{F}_\bot := \dot\bot \mid Q \mid P\,\tau_1\,\tau_2 \mid \varphi \dot\to \psi \mid \dot\forall x.\, \varphi$$

and introduce negation $\dot\neg\varphi$ as a notation for $\varphi \dot\to \dot\bot$.

Fact 3.21. $\mathcal{F}_\bot$ *is an infinite data type.*

Proof. Analogous to Fact 3.1. □

Definition 3.8. *We extend the relation* $\rho \models_{\mathcal{I}} \varphi$ *from Definition 3.3 to* $\mathcal{F}_\bot$ *by setting* $\rho \models_{\mathcal{I}} \dot\bot := \bot$.

Note that $\rho \models_{\mathcal{I}} \dot\neg\varphi$ is equivalent to $\rho \not\models_{\mathcal{I}} \varphi$ as expected. Hence a connection between validity and satisfiability can be established as follows:

Lemma 3.22. *If* φ *is valid, then* $\dot\neg\varphi$ *is unsatisfiable.*

Proof. Let φ be valid and suppose $\rho \models_{\mathcal{I}} \dot\neg\varphi$. So $\rho \not\models_{\mathcal{I}} \varphi$ but since φ is valid also $\rho \models_{\mathcal{I}} \varphi$ holds, contradiction. □

Note that the converse direction relies on classical assumptions, so Lemma 3.22 does not yield an immediate reduction from validity to unsatisfiability. However, given the information encoded in the standard interpretation $\mathcal{B}$, we can verify a reduction from $\overline{\mathrm{PCP}}$ to satisfiability.

Theorem 3.23. $\overline{\mathrm{PCP}}\,S$ *iff* $\dot\neg\varphi_S$ *is satisfiable.*

Proof. Suppose $\neg$PCP S, we show that $\rho \models_{\mathcal{B}} \dot\neg\varphi_S$ where we pick $\rho\,x := []$. Hence assume $\rho \models_{\mathcal{B}} \varphi_S$, then Lemma 3.6 yields PCP S, contradiction.

Conversely, suppose that $\dot\neg\varphi_S$ is satisfiable and that PCP S. Then φ_S is valid by Theorem 3.8 but then Lemma 3.22 implies that $\dot\neg\varphi_S$ is unsatisfiable. □

Corollary 3.24. $\overline{\mathrm{PCP}}$ *reduces to satisfiability. Thus, assuming* UA, *satisfiability is undecidable and not enumerable.*

We conclude this section with an observation regarding the limitations of axiomatising the Post correspondence problem in first-order logic. To this end, we define an interpretation that adds a further element to $\mathcal{B}$ which is misinterpreted as a solution for S while φ_1 and φ_2 are maintained.

Definition 3.9. *We define a non-standard interpretation* $\mathcal{B}_\bot$ *with domain* $\mathcal{O}(\mathcal{L}(\mathbb{B}))$, *parameter assignment* $\eta\,a := \ulcorner[]\urcorner$, *and*

$$e^{\mathcal{B}_\bot} := \ulcorner[]\urcorner \qquad Q^{\mathcal{B}_\bot} := \bot$$
$$f_b^{\mathcal{B}_\bot}\,\ulcorner s\urcorner := \ulcorner b :: s\urcorner \qquad P^{\mathcal{B}_\bot}\,\ulcorner s\urcorner\,\ulcorner t\urcorner := S \triangleright s/t$$

where $f_b^{\mathcal{B}_\bot}$ *and* $P^{\mathcal{B}_\bot}$, *respectively, return* $\emptyset$ *and* $\top$ *on input* $\emptyset$.

At first, $\mathcal{B}_\bot$ appears to be a meaningful interpretation similar to $\mathcal{B}$:

Fact 3.25. *Let* ρ *be any environment in* $\mathcal{B}_\bot$. *Then:*

(1) $\hat\rho\,\overline{s} = \ulcorner s\urcorner$ *(3)* $\rho \models_{\mathcal{B}_\bot} \varphi_1$
(2) $\rho \models_{\mathcal{B}_\bot} P\,\overline{s}\,\overline{t} \leftrightarrow S \triangleright s/t$ *(4)* $\rho \models_{\mathcal{B}_\bot} \varphi_2$

Proof. Analogous to Lemma 3.3 and Fact 3.5. □

However, $\mathcal{B}_\bot$ deems S to admit a trivial solution even though S might be an unsolvable instance of the Post correspondence problem.

Fact 3.26. $\models_{\mathcal{B}_\bot} \dot\neg\dot\forall x.\, \dot\neg P\,x\,x$ *and in particular* $\models_{\mathcal{B}_\bot} \varphi_S$.

Proof. Suppose there were a variable environment ρ with $\rho \models_{\mathcal{B}_\bot} \dot\forall x.\, \dot\neg P\, x\, x$. Then $\rho[x := \emptyset] \models_{\mathcal{B}_\bot} \dot\neg P\, x\, x$, contradicting $\rho[x := \emptyset] \models_{\mathcal{B}_\bot} P\, x\, x$ holding trivially by definition. The claim $\models_{\mathcal{B}_\bot} \varphi_S$ follows similarly. □

3.4 Intuitionistic First-Order Logic

Having extended the logical fragment $\mathcal{F}$ to $\mathcal{F}_\bot$ permits two options for adding an elimination rule for $\dot\bot$ to a deduction system. In this section, we show that the intuitionistic variant is undecidable via the same reduction as in Section 3.2.

Definition 3.10. *We define the* intuitionistic natural deduction *system $A \vdash_I \varphi$ for formulas in $\mathcal{F}_\bot$ as an extension of the minimal deduction system $A \vdash \varphi$ by the explosion rule:*

$$\frac{A \vdash_I \dot\bot}{A \vdash_I \varphi}\ E$$

We say that φ is intuitionistically provable *if $\vdash_I \varphi$.*

Fact 3.27. *Intuitionistic provability is enumerable.*

Proof. By adding $[\varphi \mid \varphi \in L_{\mathcal{F}_\bot}\, n, \dot\bot \in L_A\, n]$ to the list enumerator given in Fact 3.19. □

Fact 3.28. *If $A \vdash \varphi$ then $A \vdash_I \varphi$ for formulas $\varphi : \mathcal{F}$.*

Proof. By definition of $A \vdash_I \varphi$. □

Note that $A \dot\to \varphi$ is valid if $A \dot\to \dot\bot$ is valid, so $A \vdash_I \varphi$ is sound for the Tarski semantics defined in Section 3.1. For the same reasons as mentioned for the minimal deduction system, however, $A \vdash_I \varphi$ cannot be complete for Tarski semantics. Although completeness proofs are not in the scope of this work, we next consider Kripke semantics (for which $A \vdash_I \varphi$ could be shown complete) since they give rise to two further meaningful decision problems.

Definition 3.11. *Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations sharing the same domain and parameter assignment. We say that $\mathcal{I}$* embeds into *$\mathcal{I}'$ and write $\mathcal{I} \hookrightarrow \mathcal{I}'$ if:*

$$e^{\mathcal{I}} = e^{\mathcal{I}'} \qquad Q^{\mathcal{I}} \to Q^{\mathcal{I}'}$$
$$f_b^{\mathcal{I}}\, d = f_b^{\mathcal{I}'}\, d \qquad P^{\mathcal{I}}\, d\, d' \to P^{\mathcal{I}'}\, d\, d'$$

Definition 3.12. *A* Kripke model *$\mathcal{M}$ consists of a domain D, a parameter assignment $\eta : \mathbb{N} \to D$ and the following data:*

- *A preorder $(W, \leq)$ called* accessibility relation.
- *A function $\mathcal{W}$ mapping nodes $w : W$ to interpretations over D and η with $\mathcal{W}\, w \hookrightarrow \mathcal{W}\, w'$ whenever $w \leq w'$.*

Definition 3.13. *Given a Kripke model $\mathcal{M}$ together with an environment ρ, we define the* forcing relation *$\rho, w \Vdash_{\mathcal{M}} \varphi$ by*

$$\rho, w \Vdash_{\mathcal{M}} \dot\bot := \bot$$
$$\rho, w \Vdash_{\mathcal{M}} Q := Q^w$$
$$\rho, w \Vdash_{\mathcal{M}} P\, \tau_1\, \tau_2 := P^w\, (\hat\rho\, \tau_1)\, (\hat\rho\, \tau_2)$$
$$\rho, w \Vdash_{\mathcal{M}} \varphi \dot\to \psi := \forall w'.\, w \leq w' \to \rho, w' \Vdash_{\mathcal{M}} \varphi \to \rho, w' \Vdash_{\mathcal{M}} \psi$$
$$\rho, w \Vdash_{\mathcal{M}} \dot\forall x.\varphi := \forall w'.\, w \leq w' \to \forall d.\, \rho[x := d], w' \Vdash_{\mathcal{M}} \varphi$$

where Q^w is a shorthand for $Q^{\mathcal{W}\, w}$ and analogously for P. A formula φ is intuitionistically satisfiable *if there are $\mathcal{M}$, w and ρ with $\rho, w \Vdash_{\mathcal{M}} \varphi$ and* intuitionistically valid *if $\rho, w \Vdash_{\mathcal{M}} \varphi$ holds for all $\mathcal{M}$, w and ρ. We denote the latter by $\Vdash \varphi$.*

As it was the case with Tarski interpretations, the two new semantic notions are connected in the following sense:

Lemma 3.29. *If φ is int. valid then $\dot\neg\varphi$ is int. unsatisfiable.*

Proof. Analogous as in Lemma 3.22. □

Note that one can turn every interpretation $\mathcal{I}$ into an equivalent Kripke model $\mathcal{M}_{\mathcal{I}}$ by setting $W := \mathbb{1}, * \leq * := \top$, and $\mathcal{W}\, * := \mathcal{I}$. So Kripke semantics generalise Tarski semantics, which simplifies proving the former undecidable.

Lemma 3.30. $\rho \models_{\mathcal{I}} \varphi \leftrightarrow \rho, * \Vdash_{\mathcal{M}_{\mathcal{I}}} \varphi$

Proof. By straightforward induction on φ. □

Corollary 3.31. *Int. validity implies validity.*

Corollary 3.32. *Satisfiability implies int. satisfiability.*

Soundness w.r.t. the chosen semantics is again a key property to verify the reduction from PCP. The soundness proof for intutiotionistic provability follows the same structure as the one outlined in Section 3.2, so we omit most of the details here. In fact, the lemmas needed to establish Corollary 3.15 have direct counterparts in Kripke semantics with analogous proofs. The only interesting difference is the frequent use of the characteristic monotonicity of the forcing relation regarding accessibility:

Lemma 3.33. $\rho, w \Vdash_{\mathcal{M}} \varphi \to w \leq w' \to \rho, w' \Vdash_{\mathcal{M}} \varphi$

Proof. By induction on φ with ρ generalised, we exemplarily discuss the case $P\, \tau_1\, \tau_2$. Suppose $\rho, w \Vdash_{\mathcal{M}} P\, \tau_1\, \tau_2$, so $P^w\, (\hat\rho\, \tau_1)\, (\hat\rho\, \tau_1)$. Since $\mathcal{W}\, w \hookrightarrow \mathcal{W}\, w'$ we know that evaluation in $\mathcal{W}\, w$ is equal to evaluation in $\mathcal{W}\, w'$ and that P^w is included in $P^{w'}$. Thus also $P^{w'}\, (\hat\rho\, \tau_1)\, (\hat\rho\, \tau_1)$. □

Fact 3.34. *Every int. provable formula is int. valid.*

Proof. Analogous to Corollary 3.15. □

After this preparation, we can now easily show that intuitionistic provability as well as the two semantic notions concerning Kripke models are undecidable.

Theorem 3.35.

(1) PCP *S iff φ_S is int. provable.*
(2) PCP *S iff φ_S is int. valid.*
(3) $\overline{\text{PCP}}$ *S iff $\dot\neg\varphi_S$ is int. satisfiable.*

Proof.

(1) The first direction follows from 3.17 and 3.28 and the converse follows from 3.34, 3.31, and 3.8.
(2) The first direction follows from (1) and 3.34 and the converse follows from 3.31 and 3.8.

(3) The first direction follows from 3.23 and 3.32 and the converse follows from (2) and 3.29. □

Corollary 3.36. *Assuming* UA, *int. provability, int. validity, and int. satisfiability are undecidable. More precisely, int. provability and int. validity are not co-enumerable and int. satisfiability is not enumerable.*

3.5 Classical Provability

We end by examining the second alternative of accommodating $\dot\bot$ in a deduction system by using the double negation rule instead of explosion:

Definition 3.14. *We define the* classical natural deduction *system $A \vdash_C \varphi$ for formulas in $\mathcal{F}_\bot$ as extension of the minimal deduction system $A \vdash \varphi$ by the double negation rule:*

$$\frac{A \vdash_C \dot\neg\dot\neg\varphi}{A \vdash_C \varphi}\ DN$$

We say that φ is classically provable *if $\vdash_C \varphi$.*

Fact 3.37. *Classical provability is enumerable.*

Proof. By adding $\left[\varphi \mid \varphi \in L_{\mathcal{F}_\bot}\, n, \dot\neg\dot\neg\varphi \in L_A\, n\right]$ to the enumerator given in Fact 3.19. □

We first establish the connection to the other presented natural deduction systems.

Fact 3.38. *If $A \vdash \varphi$ then $A \vdash_C \varphi$ for formulas $\varphi : \mathcal{F}$.*

Proof. By definition of $A \vdash_C \varphi$. □

Proving the explosion rule (E) admissible for $A \vdash_C \varphi$ and therefore concluding that classical provability subsumes intuitionistic provability relies on the *weakening* property ($B \vdash \varphi$ whenever $A \vdash \varphi$ for $A \subseteq B$) satisfied by all given systems. Establishing weakening is an intricate matter because of the (AI) rule, as the extension B might block more parameters than A. We could not find a rigorous and directly formalisable proof in one of the standard text books and thus sketch a strategy given by Herbelin and Lee [24] via generalised weakening using parallel parameter renamings.

We express parallel parameter renamings as functions $\theta : \mathbb{N} \to \mathbb{N}$ and write $t[\theta]$, $\varphi[\theta]$, and $A[\theta]$ for the application of θ to all parameters in the term t, the formula φ and the context A, respectively. We use the notation $\theta[a := b]$ to update a renaming and $[a := b]$ for a renaming changing a to b and leaving every other parameter untouched.

Fact 3.39. $(\varphi_t^x)[\theta] = (\varphi[\theta])^x_{t[\theta]}$

Proof. By induction on φ using a similar fact for terms. □

We can now prove generalised weakening for $A \vdash_C \varphi$ economically. Note that in the Coq development, the proof is simultaneous for all deduction systems.

Theorem 3.40. $A \vdash_C \varphi \to A \subseteq B \to B[\theta] \vdash_C \varphi[\theta]$

Proof. By induction on $A \vdash_C \varphi$ with θ and B generalised. Every case is easy, apart from (AE) and (AI). (AE) follows directly from Fact 3.39.

For (AI), we have $A \vdash_C \varphi_a^x$ for $a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A)$, $A \subseteq B$, and the inductive hypothesis $\forall B\, \theta.\, A \subseteq B \to B[\theta] \vdash_C (\varphi_a^x)[\theta]$. We have to prove $B[\theta] \vdash_C (\varphi[\theta])_b^x$ for a fresh parameter b.

Since $B[\theta] = B'[\theta']$ and $(\varphi[\theta])_b^x = (\varphi_a^x)[\theta']$ where we set $B' := B[a := b]$ and $\theta' := \theta[b := \theta\, a][a := b]$ we can apply the inductive hypothesis for B' and θ'. It remains to prove $A \subseteq B'$, which follows from $A \subseteq B$ and $a \notin \mathcal{P}(A)$. □

Corollary 3.41 (Weakening). $A \vdash_C \varphi \to A \subseteq B \to B \vdash_C \varphi$

Fact 3.42. *If $A \vdash_I \varphi$ then $A \vdash_C \varphi$ for formulas $\varphi : \mathcal{F}_\bot$.*

Proof. By induction on $A \vdash_I \varphi$. All cases are easy, apart from (E),where we know $A \vdash_C \dot\bot$ and have to prove $A \vdash_C \varphi$. By (DN) and (II) it suffices to prove $\dot\neg\varphi :: A \vdash_C \dot\bot$, which follows with weakening. □

We now prove the undecidability of classical natural deduction using the formula φ_S from before. One direction of the reduction follows trivially from previous results:

Lemma 3.43. *If* PCP S *then* $\vdash_C \varphi_S$.

Proof. $\vdash \varphi_S$ by Lemma 3.16 and thus $\vdash_C \varphi_S$ by Fact 3.38. □

For minimal and intuitionistic provability, soundness w.r.t. Tarski and Kripke semantics was key to prove the other direction of the reduction. For classical natural deduction, soundness is not constructively provable, because it implies double negation for all propositions (and hence EM):

Lemma 3.44. *$\neg\neg P \to P$ for all propositions P if we assume that classical provability implies validity.*

Proof. $\vdash_C \neg\neg Q \dot\to Q$ is easily provably. Thus, by assumption, $\neg\neg Q \dot\to Q$ is valid. Hence every interpretation assigning P to Q yields $\neg\neg P \to P$. □

To circumvent this issue, we define a translation φ^Q of formulas by a combination of the Gödel-Gentzen double negation translation [17, 19] and Friedman's A-translation [16] (with Q in the place of A). The key property of the translation is that $\vdash_C \varphi \to \vdash_I \varphi^Q$. This will allow us to conclude $\vdash_I \varphi_S^Q$ from $\vdash_C \varphi_S$, enabling the use of soundness for $\vdash_I$.

Definition 3.15. *We define the translation φ^Q as follows:*

$$\dot\bot^Q := Q \qquad Q^Q := Q$$
$$(P\,\tau_1\,\tau_2)^Q := ((P\,\tau_1\,\tau_2) \dot\to Q) \dot\to Q \qquad (\varphi_1 \dot\to \varphi_2)^Q := \varphi_1{}^Q \dot\to \varphi_2{}^Q$$
$$(\dot\forall x.\varphi)^Q := \dot\forall x.\varphi^Q$$

Lemma 3.45. $A \vdash_I (\dot\neg\dot\neg\varphi)^Q \dot\to \varphi^Q$

Proof. By induction on the size of φ with A generalised. In the case of universal quantification one uses $|\varphi_a^x| = |\varphi|$. □

This suffices for the key property of the translation:

Fact 3.46. *If $A \vdash_C \varphi$ then $A^Q \vdash_I \varphi^Q$.*

Proof. By induction on $A \vdash_C \varphi$. The only interesting case is (DN) where we have to prove $A^Q \vdash_I \varphi^Q$ given $A^Q \vdash_I (\dot\neg\dot\neg\varphi)^Q$. This follows by using (IE) for Lemma 3.45. □

Lemma 3.47. *If $\vdash_C \varphi_S$ then* PCP *S.*

Proof. Assume $\vdash_C \varphi_S$. By Fact 3.46 and Fact 3.34 we know that $\mathcal{B} \models \varphi_S^Q$. Note that $\varphi_S^Q = \varphi_1^Q \dot\to \varphi_2^Q \dot\to \varphi_3^Q \dot\to Q$. From the fact that $\mathcal{B}$ satisfies φ_1, φ_2, and φ_3 (Lemma 3.3), we obtain that $\mathcal{B}$ also satisfies φ_1^Q, φ_2^Q, and φ_3^Q by simple calculation. Thus, we obtain $\mathcal{B} \models Q$, which is equivalent to PCP S by definition. □

We finally conclude the full reduction theorem:

Theorem 3.48. PCP *S iff φ_S is classically provable.*

Proof. Follows with Lemma 3.43 and Lemma 3.47. □

Corollary 3.49. PCP *reduces to classical provablity. Thus, assuming* UA*, classical provability is undecidable and not enumerable.*

4 Discussion

The reduction by Floyd and Manna [31], which we use as basis for all presented undecidability proofs, uses a signature without the propositional constant Q but a logical fragment including existential quantification. We circumvent using the larger fragment by adding Q based on the observation that $(\exists x.\, P\, x\, x)^Q$ would be $((\forall x.(P\, x\, x \dot\to Q) \dot\to Q) \dot\to Q) \dot\to Q$, yielding a slightly sharper result. Note that in principle Q and e could be eliminated as well via exploiting free variables.

We do not consider completeness proofs in this work since the direct reductions we gave yield the shorter undecidability proofs. For intuitionistic natural deduction, completeness is equivalent to Markov's principle [28]. Veldman [46] shows that if Kripke models are extended with so-called exploding nodes, completeness becomes provable constructively. We remark that without completeness, we are not able to easily show that valid formulas are enumerable.

For classical logic, the situation is even more intricate. Already soundness fails to be provable in a constructive metalogic, witnessed by Lemma 3.44. One could follow Herbelin and Ilik [21] in restricting to interpretations validating all instances of double negation. However, our central standard model $\mathcal{B}$ of boolean strings does not satisfy $\dot\neg\dot\neg Q \dot\to Q$ in general as this expresses the stability of PCP. Therefore, $\mathcal{B}$ is in the scope of the restricted semantics as soon as one assumes Markov's principle (Fact 3.4). So only accepting additional assumptions would simplify maintaining soundness and at the same time allow for the standard completeness proofs for classical deduction w.r.t Tarksi semantics [18, 38]. Alternatively, for minimal classical natural deduction, a fully constructive completeness proof w.r.t. a further adapted notion of Tarksi semantics can be given [4, 29].

4.1 Coq Formalisation

Our formalisation has about 2200 lines of code, with around 40% of specification. The development of elementary synthetic computability theory and PCP needs 800 loc. The definitions of first-order syntax, Tarski and Kripke semantics, and the deduction systems sum up to 800 loc. The reductions have 460 loc and the weakening proof has 140 loc.

On paper, it is easy to extend an inductive syntax by more constructors, as we did for $\mathcal{F}$ and $\mathcal{F}_\bot$. In type theory, extending by one constructor usually means defining an entirely new inductive type and a recursive embedding function. Similarly, extending minimal deduction represented by an inductive predicate to intuitionistic and classical deduction is straightforward on paper. In Coq, e.g. proving weakening for all three systems would require a lot of code duplication.

We circumvent this issue by employing a tagged type of formulas $\mathtt{form}_{(b:\mathbb{B})}$ where we force $\dot\bot : \mathtt{form}_{\mathtt{tt}}$. Similarly, we define a unified inductive predicate $\vdash_{b,b'}$ where b signals the inclusion of falsity and b' signals whether the explosion rule or the double negation rule is used. We then define minimal provability as $\vdash_{\mathtt{ff},\mathtt{ff}}$, intuitionistic as $\vdash_{\mathtt{tt},\mathtt{ff}}$, and classical as $\vdash_{\mathtt{tt},\mathtt{tt}}$. A parametrised proof of weakening and other statements is then possible, avoiding any code duplication. In fact, instead of tagging syntax and deductions systems with booleans, we use type classes to let Coq automatically infer the correct tags following a defined priority.

We define interpretations $\mathcal{I} : \mathtt{interp}_{(D:\mathbb{T}),(\eta:\mathbb{N}\to\mathbb{N})}$ as type classes, allowing to write $\rho \models \varphi$ with $\mathcal{I}$ implicit. We use a standard approach [41] and make the domain type D and the parameter assignment η in the definition of interpretations and Kripke models an argument instead of a field. Hence stating the extensionality of the satisfaction and forcing relations is convenient and the functor of type $W \to \mathtt{interp}_{D,\eta}$ in Kripke models can be defined without projections.

We are formalising binders using names, faithful to the paper presentation. Using explicit parameters as constants, capturing becomes a non-issue, which considerably simplifies the deduction systems and nearly all proofs concerning the two semantics. Interestingly, the scalable standard solution of binding via de Bruijn presentation with parallel renamings as suggested in [33] was not needed in our case.

We found the dependently-typed version of decidability allowing the usage of proof scripts to be considerably easier to work with than the definition via boolean deciders. Moreover, defining a type class $\mathtt{dec}(P : \mathbb{P})$ allows for writing $\mathtt{Dec}\ P$ for the decider of P, making the inference of the concrete implementation automatic. We would wish for the inclusion of such a type class in the upcoming new Coq standard library.

Using the new Coq notation system allowing patterns, we employed common abbreviations for list functions such as

```
Notation "[ s | p ∈ A ',' P ]" :=
(map (fun p => s) (filter (fun p => Dec P) A)) (p pattern),
```

making the definition of list enumerators pleasant.

4.2 Related Work

Undecidability of FOL The reductions in this paper are based on Floyd's idea in Manna [31]. A detailed analysis of undecidable fragments of first-order logic is given in [6]. More recently, Kontchakov et al. [27] prove the positive fragment of intuitionistic logic with only two variables, a binary predicate, and infinitely many unary predicates undecidable.

Formalised FOL O'Connor [33] formalises first-order logic and Gödel's incompleteness theorem in Coq. He considers a classical natural deduction system, but no Tarski semantics.

Herbelin, Kim and Lee [22–24], formalise a cut-free sequent calculus for intuionistic first-order logic and prove weakening, soundness, and completeness. They use name tracing and consider representations with variables only as well as with both variables and parameters.

Constructive from classical proofs In Section 3.5 we employed a generalised double negation translation to obtain $\vdash_I \varphi_S^Q$ from $\vdash_C \varphi_S$. Berger et al. [5] show that this technique can be used for a general class of formulas subsuming our observation concerning φ_S. Schwichtenberg and Senjak [39] use explicit proof transformations to eliminate classical rules from natural deduction derivations for a similar class.

Synthetic computability Richman [36] relies on the computability of all functions in Bishop style constructive mathematics and adds an axiom stating the enumerability of all enumerable sets. He then gives a purely synthetic proof of the undecidability of the halting problem.

Bauer [2] works in Hyland's effective topos [25], where countable choice, Markov's principle and Richman's enumeration axiom are valid. He proves that Markov's principle implies Post's theorem and reconstructs further standard results like Rice's theorem. In [3], he extends this exploration, amongst others, to the Kleene-Rogers recursion theorem.

Formalised undecidability Forster and Smolka [15] develop constructive computability in Coq based on the call-by-value λ-calculus. They prove the equivalence of Markov's principle with Post's theorem for their model. Ciaffaglione [9] gives a coinductive definition of Turing machines in Coq and proves the halting problem undecidable. Forster, Heiter, and Smolka [12] give a Coq formalisation of a reduction from the halting problem for Turing machines to PCP, string rewriting, and various problems regarding context-free grammars. Forster and Larchey-Wendling [14] reduce PCP to the halting problems of binary stack machines and Minsky machines as well as to intuitionistic linear logic in Coq.

Norrish [32] develops computability theory in HOL, based on the full λ-caculus. Xu, Zhang, and Urban [47] implement Turing machines in Isabelle, construct a universal Turing machine, and prove the halting problem to be undecidable. Ramos et al. [35] formalise the undecidability of the halting problem of a simple functional language in PVS.

4.3 Future Work

Since all functions in Coq are computable, we believe that UA is consistent. Adding strong excluded middle, i.e. that $\forall P.\ \{P\} + \{\neg P\}$, is incompatible with UA, because all predicates, including PCP, become decidable. On the other hand, we conjecture that the computability of functions on data types is untouched if only the propositional form of excluded middle is assumed. To the best of our knowledge, there is no model for the calculus of inductive construction witnessing this conjecture. A model validating EM while maintaining the computability of functions on data types would necessarily invalidate strong excluded middle. In usual computable functions models, the interpretation of logical and computational sums coincides. However, we would need a model where the interpretation of propositions is changed in a way such that EM holds, but strong excluded middle does not.

Furthermore, we plan to consider completeness proofs for minimal, classical minimal, intuitionistc and classical natural deduction, developing a fully-formalised metatheory of first-order logics. In this context, it would also be interesting to study proof terms and the formalisation of cut-free completeness proofs, yielding a cut-elimination theorem.

For most inductive types, higher-order abstract syntax (HOAS) cannot be used in Coq, because of the strict positivity constraint, and one has to resort to parametric HOAS [7], de Bruijn, named, or locally nameless representations. For first-order logic, a HOAS representation is definable in principle, because there is no quantification over formulas. We want to analyse a HOAS implementation to investigate the connection to the present and to a de Bruijn presentation, for instance using the Autosubst tool [37].

We further want to study the application of synthetic computability theory to proving axiomatic first-order theories undecidable, e.g. following Treinen [42]. For instance, PCP is easily encoded in ZF set theory, so for instance deductive and semantic consequence of an higher-order axiomatisation of set theory as studied in [26] are undecidable.

Another direction would be to formalise the undecidability of provability for higher-order logics. One possibility could be to prove inhabitation in System F undecidable, initially proven by Löb [30] and subsequently simplified by Arts and Dekkers [1] and Urzyczyn [45]. For full second-order propositional logic with all quantifiers, there is a reduction starting at first-order logic by Sørensen and Urzyczyn [40].

Finally, using the techniques from Forster and Kunze [13], we could prove the computability of all reductions in this paper in the call-by-value λ-calculus, hence substantiating that the synthetic approach to undecidability is faithful to the classical strategy based on a concrete model.

Acknowledgments

We thank Steven Schäfer and Dominique Larchey-Wendling for fruitful discussions and the anonymous reviewers for their helpful comments.

References

[1] Thomas Arts and W Dekkers. 1992. Embedding first order predicate logic in second order propositional logic. *Master's thesis, University of Njimegen* (1992).

[2] Andrej Bauer. 2006. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science* 155 (2006), 5–31.

[3] Andrej Bauer. 2017. On fixed-point theorems in synthetic computability. *Tbilisi Mathematical Journal* 10, 3 (2017), 167–181.

[4] Stefano Berardi and Silvio Valentini. 2004. Krivine's intuitionistic proof of classical completeness (for countable languages). *Annals of Pure and Applied Logic* 129, 1-3 (2004), 93–106.

[5] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. 2002. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic* 114, 1-3 (2002), 3–25.

[6] E. Boerger, E. Grädel, and Y. Gurevich. 1997. *The classical decision problem.* Springer.

[7] Adam Chlipala. 2008. Parametric higher-order abstract syntax for mechanized semantics. In *ACM Sigplan Notices*, Vol. 43. ACM, 143–156.

[8] Alonzo Church. 1936. A note on the Entscheidungsproblem. *The journal of symbolic logic* 1, 1 (1936), 40–41.

[9] Alberto Ciaffaglione. 2016. Towards Turing computability via coinduction. *Science of Computer Programming* 126 (2016), 31–51.

[10] Thierry Coquand and Bassel Mannaa. 2016. The independence of Markov's principle in type theory. *arXiv preprint arXiv:1602.04530* (2016).

[11] Hilbert David and Ackermann Wilhelm. 1928. Grundzüge der theoretischen Logik. (1928).

[12] Yannick Forster, Edith Heiter, and Gert Smolka. 2018. Verification of PCP-Related Computational Reductions in Coq. In *ITP 2018 (Lecture Notes in Computer Science)*, Jeremy Avigad and Assia Mahboubi (Eds.). Springer, 253–269.

[13] Yannick Forster and Fabian Kunze. 2016. Verified Extraction from Coq to a Lambda-Calculus. *Coq Workshop 2016* (2016).

[14] Yannick Forster and Dominique Larchey-Wendling. 2019. 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*.

[15] Yannick Forster and Gert Smolka. 2017. Weak call-by-value lambda calculus as a model of computation in Coq. In *ITP 2017*. Springer, 189–206.

[16] Harvey Friedman. 1978. Classically and intuitionistically provably recursive functions. In *Higher set theory*. Springer, 21–27.

[17] Gerhard Gentzen. 1936. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Math. Ann.* 112, 1 (1936), 493–565.

[18] Kurt Gödel. 1930. Die vollständigkeit der axiome des logischen funktionenkalküls. *Monatshefte für Mathematik und Physik* 37, 1 (1930), 349–360.

[19] Kurt Gödel. 1933. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums* 4, 1933 (1933), 34–38.

[20] Hugo Herbelin. 2010. An Intuitionistic Logic that Proves Markov's Principle. In *Proceedings - Symposium on Logic in Computer Science*. 50–56.

[21] Hugo Herbelin and Danko Ilik. 2016. An analysis of the constructive content of Henkin's proof of Gödel's completeness theorem. (2016).

[22] Hugo Herbelin, SunYoung Kim, and Gyesik Lee. 2017. Formalizing the meta-theory of first-order predicate logic. *Journal of the korean Mathematical society* 54, 5 (2017), 1521–1536.

[23] Hugo Herbelin and Gyesik Lee. 2009. Forcing-based cut-elimination for gentzen-style intuitionistic sequent calculus. In *International Workshop on Logic, Language, Information, and Computation*. Springer, 209–217.

[24] Hugo Herbelin and Gyesik Lee. 2014. Formalizing Logical Meta-theory – Semantical Cut-Elimination using Kripke Models for first-order Predicate Logic. (2014).

[25] J Martin E Hyland. 1982. The effective topos. In *The LEJ Brouwer centenary symposium*, Vol. 110. 165–216.

[26] Dominik Kirst and Gert Smolka. 2018. Categoricity Results and Large Model Constructions for Second-Order ZF in Dependent Type Theory. *Journal of Automated Reasoning* (11 Oct 2018).

[27] Roman Kontchakov, Agi Kurucz, and Michael Zakharyaschev. 2005. Undecidability of first-order intuitionistic and modal logics with two variables. *Bulletin of Symbolic Logic* 11, 3 (2005), 428–438.

[28] Georg Kreisel. 1958. Elementary completeness properties of intuitionistic logic with a note on negations of prenex formulae. *The Journal of Symbolic Logic* 23, 3 (1958), 317–330.

[29] Jean-Louis Krivine. 1996. Une preuve formelle et intuitionniste du théorème de complétude de la logique classique. *Bulletin of Symbolic Logic* 2, 4 (1996), 405–421.

[30] Martin H Löb. 1976. Embedding first order predicate logic in fragments of intuitionistic logic. *The Journal of Symbolic Logic* 41, 4 (1976), 705–718.

[31] Zohar Manna. 2003. *Mathematical theory of computation.* Dover Publications, Incorporated.

[32] Michael Norrish. 2011. Mechanised computability theory. In *International Conference on Interactive Theorem Proving*. Springer, 297–311.

[33] Russell SS O'Connor. 2009. Incompleteness & completeness: formalizing logic and analysis in type theory. *PhD thesis, Radboud University of Njimegen* (2009).

[34] Emil L Post. 1946. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.* 52, 4 (1946), 264–268.

[35] Thiago Mendonça Ferreira Ramos, César Muñoz, Mauricio Ayala-Rincón, Mariano Moscato, Aaron Dutle, and Anthony Narkawicz. 2018. Formalization of the Undecidability of the Halting Problem for a Functional Language. In *International Workshop on Logic, Language, Information, and Computation*. Springer, 196–209.

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

[37] Steven Schäfer, Tobias Tebbi, and Gert Smolka. 2015. Autosubst: Reasoning with de Bruijn terms and parallel substitutions. In *International Conference on Interactive Theorem Proving*. Springer, 359–374.

[38] George F Schumm. 1975. A Henkin-style completeness proof for the pure implicational calculus. *Notre Dame Journal of Formal Logic* 16, 3 (1975), 402–404.

[39] Helmut Schwichtenberg and Christoph Senjak. 2013. Minimal from classical proofs. *Ann. Pure Appl. Logic* 164, 6 (2013), 740–748.

[40] Morten H Sørensen and Paweł Urzyczyn. 2010. A syntactic embedding of predicate logic into second-order propositional logic. *Notre Dame Journal of Formal Logic* 51, 4 (2010), 457–473.

[41] Bas Spitters and Eelis Van der Weegen. 2011. Type classes for mathematics in type theory. *Mathematical Structures in Computer Science* 21, 4 (2011), 795–825.

[42] Ralf Treinen. 1992. A new method for undecidability proofs of first order theories. *Journal of Symbolic Computation* 14, 5 (1992), 437–457.

[43] A.S. Troelstra and D. Van Dalen. 1988. *Constructivism in Mathematics.* North-Holland, Amsterdam.

[44] Alan M Turing. 1937. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society* 2, 1 (1937), 230–265.

[45] Pawel Urzyczyn. 1997. Inhabitation in typed lambda-calculi (a syntactic approach). In *International Conference on Typed Lambda Calculi and Applications*. Springer, 373–389.

[46] Wim Veldman. 1976. An intuitionistic completeness theorem for intuitionistic predicate logic. *The Journal of Symbolic Logic* 41, 1 (1976), 159–166.

[47] Jian Xu, Xingyuan Zhang, and Christian Urban. 2013. Mechanising turing machines and computability theory in Isabelle/HOL. In *International Conference on Interactive Theorem Proving*. Springer, 147–162.