Formalising Metamathematics in Constructive Type Theory

Synthetic Undecidability and Incompleteness

Dominik Kirst

Proof and Computation
September 14th, 2021
What will this talk be about?

Metamathematics (of first-order logic):
- Mostly negative results: undecidability and incompleteness*
- Sketch positive results: completeness and (relative) consistency†

Constructive type theory:
- Basic concepts of the calculus of inductive constructions (CIC)‡
- Implementation in the Coq proof assistant§
- Synthetic computability¶

---

*Tarski (1953); Gödel (1931)
†Gödel (1930); Werner (1997)
‡Coquand and Huet (1986); Paulin-Mohring (1993)
§The Coq Development Team (2021)
¶Richman (1983); Bauer (2006)
Outline

- Framework: Synthetic Undecidability
- Example 1: The Entscheidungsproblem
- Example 2: Trakhtenbrot’s Theorem
- Example 3: First-Order Axiom Systems
- Conclusion
Framework: Synthetic Undecidability*

*Yannick Forster, K., and Gert Smolka at CPP’19.
How to mechanise decidability?

Conventional approach:

- Pick a concrete model of computation
  (Turing machines, $\mu$-recursive functions, untyped $\lambda$-calculus, etc.)
- Invent a decision procedure for the given problem
- Explicitly code the algorithm in the chosen model!

Synthetic approach (Richman (1983); Bauer (2006)):

- Work in a constructive foundation, e.g. constructive type theory
- Define a decision procedure e.g. as a Boolean function
- Definable functions are computable, so that’s it!

(Similar for other notions like enumerability and reducibility)
How to mechanise undecidability?

Problem of the synthetic approach:

- Constructive type theories like CIC are consistent with classical assumptions, rendering every problem decidable
- Proving a given problem undecidable is not outright possible

Possible solutions:

- Resort to a concrete model of computation
- Verify a synthetic reduction from an undecidable problem
  ▶ Computability axioms could be used to obtain expected results

(Again similar for other negative notions of computability theory)
Coq’s Type Theory

Main features of Coq’s underlying CIC:

- Standard type formers: $X \rightarrow Y$, $X \times Y$, $X + Y$, $\forall x. F x$, $\Sigma x. F x$
- Inductive types: $\mathbb{B}$, $\mathbb{N}$, lists $L(X)$, options $O(X)$, vectors $X^n$, ...
- Propositional universe $\mathbb{P}$ with logical connectives: $\rightarrow$, $\land$, $\lor$, $\forall$, $\exists$
- $\mathbb{P}$ is impredicative and separate from computational types

All definable functions $\mathbb{N} \rightarrow \mathbb{N}$ are computable!
Decidability and Enumerability

A problem interpreted as a predicate \( p : X \rightarrow \mathbb{P} \) on a type \( X \) is **decidable** if there is a function \( f : X \rightarrow \mathbb{B} \) with

\[
\forall x. p x \iff f x = \text{tt},
\]

**enumerable** if there is a function \( f : \mathbb{N} \rightarrow \mathcal{O}(X) \) with

\[
\forall x. p x \iff \exists n. f n = \llbracket x \rrbracket.
\]

**Fact**

Let \( p : X \rightarrow \mathbb{P} \) be a predicate, then \( p \) is

- **decidable** iff \( \forall x. p x + \neg p x \) is inhabited and
- **enumerable** iff there is \( L : \mathbb{N} \rightarrow \mathcal{L}(X) \) s.t. \( \forall x. p x \iff \exists n. x \in L n \).
Data Types

Computability theory is usually developed on computational domains.

A type $X$ is called

- **enumerable** if $\lambda x. T$ is enumerable,
- **discrete** if $\lambda xy. x = y$ is decidable, and
- **data type** if it is both enumerable and discrete.

**Fact**

*Decidable predicates $p$ on data types $X$ are enumerable and co-enumerable.*

**Proof.**

Let $f_X : \mathbb{N} \rightarrow \mathcal{O}(X)$ enumerate $X$ and $f_p : X \rightarrow \mathbb{B}$ decide $p$. Then

$$f n := \text{match } f_X n \text{ with } \begin{cases} \exists x \Rightarrow \text{ if } f_p x \text{ then } \exists x \text{ else } \emptyset \mid \emptyset \Rightarrow \emptyset \end{cases}$$

defines an enumerator for $p$. 

Dominik Kirst

Formalising Metamathematics

September 14th, 2021
Post’s Theorem

**Theorem**

Let $p$ on a data type $X$ be enumerable and co-enumerable. If $p$ is also logically decidable, i.e. $\forall x. p \ x \lor \neg p \ x$, then it is decidable.

**Proof.**

- Let $f$ enumerate $p$ and $g$ enumerate its complement $\overline{p}$.
- $\forall x. \exists n. f \ n = \check{\neg x} \lor g \ n = \check{x}$ by logical decidability.
- For given $x$, corresponding $n$ can be computed by linear search.
- Disjunction $f \ n = \check{\neg x} \lor g \ n = \check{x}$ lacks computational information.
- Use discreteness to computably compare $\check{\neg x}$ with $f \ n$ and $g \ n$.
- Obtain decision whether $p \ x$ or $\neg p \ x$. 

Dominik Kirst
Formalising Metamathematics
September 14th, 2021
Many-One Reductions

Given predicates $p : X \rightarrow \mathbb{P}$ and $q : Y \rightarrow \mathbb{P}$ we call a function $f : X \rightarrow Y$ a (many-one) reduction from $p$ to $q$ if

$$\forall x. \ p x \leftrightarrow q (f x).$$

We write $p \preceq q$ if a reduction from $p$ to $q$ exists.

**Theorem (Reduction)**

Let $p$ and $q$ be predicates on data types with $p \preceq q$.

- If $q$ is decidable/enumerable/co-enumerable, then so is $p$.
- If $p$ is not co-enumerable, then $q$ is not co-enumerable.

**Proof.**

If $f$ witnesses $p \preceq q$ and $g$ decides $q$, then $g \circ f$ decides $p$. 

Dominik Kirst
Formalising Metamathematics
September 14th, 2021
11
The Post Correspondence Problem

Intuition: given a stack $S$ of cards $s/t$, find a derivable match.

This (undecidable) problem can be expressed by an inductive predicate:

$$
\frac{s/t \in S}{S \triangleright s/t} \quad \frac{S \triangleright u/v \quad s/t \in S}{S \triangleright su/tv} \quad \frac{S \triangleright s/s}{\text{PCP } S}
$$

Fact

The type $S$ of stacks is a data type and PCP is enumerable.

Proof.

The former follows from closure properties and for the latter

$$
L 0 := [] \\
L (S \cdot n) := L n + [(S, (s, t)) | S \in L S n, (s, t) \in S] \\
+ [(S, (su, tv)) | (S, (u, v)) \in L n, (s, t) \in S]
$$

defines a list enumerator for $\lambda Sst. S \triangleright s/t$. 

Coq Library of Undecidability Proofs*

- Merge of a few initial Coq developments:
  - Computability theory using a cbv. lambda calculus
  - Synthetic computability
  - Initial undecidability proofs

- Extended with further undecidability reductions over past 3 years

- Unified framework to ease external contribution

- 11+ contributors and more than 100k lines of code

- 14+ related publications (ITP, CPP, IJCAR, FSCD, etc.)

- Currently roughly 13 (groups of) undecidable problems

*https://github.com/uds-psl/coq-library-undecidability
Classification in seed problems and target problems

This talk: mostly the PCP $\rightarrow$ FOL edge, a bit of H10 $\rightarrow$ FOL
Example 1:
The Entscheidungsproblem*

*Yannick Forster, K., and Gert Smolka at CPP’19.
General Idea

Given a FOL formula $\varphi$, is $\varphi$ valid in all models?

Conventional outline following Turing:
- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature
- Verify that $M$ halts if and only if $\varphi_M$ holds in all models
  (Argue why the used signature could have been minimised)

Our outline:
- Follow the simpler proof given in Manna (2003) using PCP
- Also don’t bother with signature minimisation yet...
Terms and formulas are defined for a fixed signature:

\[ \tau : \text{Term} := x \mid a \mid e \mid f_{tt} \tau \mid f_{ff} \tau \quad x, a : \mathbb{N} \]

\[ \varphi, \psi : \text{Form} := \bot \mid Q \mid P \tau_1 \tau_2 \mid \varphi \rightarrow \psi \mid \forall x. \varphi \]

Formulas are interpreted in models \( I = (D, \eta, e^I, f^I_{tt}, f^I_{ff}, Q^I, P^I) \) given a variable environment \( \rho : \mathbb{N} \rightarrow D \):

\[ I \models_\rho \bot := \bot \]

\[ I \models_\rho Q := Q^I \]

\[ I \models_\rho P \tau_1 \tau_2 := P^I (\hat{\rho} \tau_1) (\hat{\rho} \tau_2) \]

\[ I \models_\rho \varphi \rightarrow \psi := I \models_\rho \varphi \rightarrow I \models_\rho \psi \]

\[ I \models_\rho \forall x. \varphi := \forall d : D. I \models_\rho [x:=d] \varphi \]

A formula \( \varphi \) is valid if \( I \models_\rho \varphi \) for all \( I \) and \( \rho \).
A Standard Model

Strings can be encoded as terms, e.g. \$tt \text{ff} \text{ff} \text{tt} = f_{tt} \left( f_{ff} \left( f_{ff} \left( f_{tt} \left(e\right)\right)\right)\right)\$.

The standard model \( \mathcal{B} \) over the type \( \mathcal{L}(\mathcal{B}) \) of Boolean strings captures exactly the cards derivable from a fixed stack \( S \):

\[
\begin{align*}
e^B & := [] \\
f^B_b s & := b :: s \\
Q^B & := \text{PCP } S \\
P^B s t & := S \triangleright s/t.
\end{align*}
\]

Lemma

Let \( \rho : \mathbb{N} \rightarrow \mathcal{L}(\mathcal{B}) \) be an environment for the standard model \( \mathcal{B} \).
Then \( \hat{\rho} \bar{s} = s \) and \( \mathcal{B} \models_\rho P \tau_1 \tau_2 \iff S \triangleright \hat{\rho} \tau_1/\hat{\rho} \tau_2 \).
Undecidability of Validity

We express the constructors of $S \triangleright s/t$ and PCP as formulas:

$$\varphi_1 := [P \overline{s} \overline{t} \mid s/t \in S]$$
$$\varphi_2 := [\forall xy. P \times y \Rightarrow P(\overline{s} x)(\overline{t} y) \mid s/t \in S]$$
$$\varphi_3 := \forall x. P \times x \Rightarrow Q$$
$$\varphi_S := \varphi_1 \Rightarrow \varphi_2 \Rightarrow \varphi_3 \Rightarrow Q$$

**Theorem**

PCP $S$ iff $\varphi_S$ is valid, hence PCP reduces to validity.

**Proof.**

Let $\varphi_S$ be valid, so in particular $B \models \varphi_S$. Since $B$ satisfies all of $\varphi_1$, $\varphi_2$, and $\varphi_3$ it follows that $B \models Q$ and thus PCP $S$.

Now suppose that $S \triangleright s/s$ for some $s$ and that some model $I$ satisfies all of $\varphi_1$, $\varphi_2$, and $\varphi_3$. Then $I \models P \overline{s} \overline{s}$ by $\varphi_1$ and $\varphi_2$, hence $I \models Q$ by $\varphi_3$, and thus $I \models \varphi_S$. 

Dominik Kirst
Formalising Metamathematics
September 14th, 2021
Undecidability of Satisfiability

Disclaimer: validity does not directly reduce to (co-)satisfiability!
- If \( \varphi \) is valid, then certainly \( \neg \varphi \) is unsatisfiable
- However, the converse does not hold constructively

Fortunately, we can give a direct reduction from the complement of PCP:

**Theorem**
\[ \neg \text{PCP} \iff \neg \varphi_S \text{ is satisfiable, hence co-PCP reduces to satisfiability.} \]

**Proof.**
If \( \neg \text{PCP} \), then \( B \models \neg \varphi_S \) since \( B \models \varphi_S \) would yield PCP \( S \) as before.
Now suppose there are \( I \) and \( \rho \) with \( I \models \rho \neg \varphi_S \). Then assuming PCP \( S \) yields the contradiction that \( \varphi_S \) is valid. \( \square \)
Interlude: Completeness Theorems for FOL

Completeness of deduction systems for FOL relies on Markov’s principle:

\[ \text{MP} := \forall f : \mathbb{N} \rightarrow \mathbb{B}. \neg\neg(\exists n. f n = \text{tt}) \rightarrow \exists n. f n = \text{tt} \]

MP is independent but admissible in Coq’s type theory*

Theorem (cf. Yannick Forster, K., and Dominik Wehr at LFCS’20.):

- \( \mathcal{T} \vdash \varphi \) implies \( \neg\neg(\mathcal{T} \vdash_{c} \varphi) \) for all \( \mathcal{T} : \text{Form} \rightarrow \mathbb{P} \) and \( \varphi : \text{Form} \)
- If \( \mathcal{T} \) is enumerable, then MP is equivalent to the stability of \( \mathcal{T} \vdash_{c} \varphi \)

\( \Rightarrow \) Completeness for enumerable \( \mathcal{T} \) is equivalent to MP and admissible

Possible strategies:

a) Verify a weak reduction from PCP integrating the double negation
b) Obtain a standard reduction by proving \( A \vdash_{c} \varphi_S \) by hand (done so far)

*Coquand/Mannaa ’17, Pédrot/Tabareau ’18
Undecidability of Minimal Provability

We define a minimal natural deduction system inductively:

\[
\frac{\varphi \in A}{A \vdash \varphi} \quad \frac{\varphi :: A \vdash \psi}{A \vdash \varphi \rightarrow \psi} \quad \frac{A \vdash \varphi \rightarrow \psi \quad A \vdash \varphi}{A \vdash \psi} \quad \frac{A \vdash \varphi^x_a \quad a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A)}{A \vdash \forall x. \varphi} \quad \frac{A \vdash \forall x. \varphi \quad \forall(\tau) = \emptyset}{A \vdash \varphi^x_\tau}
\]

A formula \( \varphi \) is provable if \( \vdash \varphi \).

**Fact (Soundness)**

\( A \vdash \varphi \) implies \( A \models \varphi \), so provable formulas are valid.

**Theorem**

- PCP S iff \( \varphi_S \) is provable.  
  (proving \( \vdash \varphi_S \) by hand)
- Provability is enumerable.  
  (by giving a list enumerator)
Undecidability of Classical Provability

We extend the deduction system by classical double negation elimination:

\[
A \vdash_c \neg\neg \varphi \quad \Rightarrow \\
A \vdash_c \varphi
\]

Unfortunately, this rule is not sound constructively!

As a remedy, we define a Gödel-Gentzen-Friedman translation \( \varphi^Q \) of formulas \( \varphi \) such that \( A \vdash_c \varphi \) implies \( A^Q \vdash \varphi^Q \).

**Theorem**

PCP \( S \) iff \( \varphi_S \) is classically provable, hence PCP reduces to classical ND.

**Proof.**

If PCP \( S \) then \( \vdash \varphi_S \) by the previous theorem and hence \( \vdash_c \varphi_S \). Conversely, let \( \vdash_c \varphi_S \) and hence \( \vdash \varphi^Q_S \). Then by soundness \( B \vdash \varphi^Q_S \) which still implies \( B \vdash Q \) and PCP \( S \) as before.
Example 2: Trakhtenbrot’s Theorem*
Given a FOL formula $\varphi$, is $\varphi$ finitely satisfiable?

Textbook proofs by dual reduction from the halting problem:*  
- Encode Turing machine $M$ as formula $\varphi_M$ over custom signature  
- Verify that the models of $\varphi_M$ correspond to the runs of $M$  
- Conclude that $M$ halts if and only if $\varphi_M$ has a finite model

Our mechanisation:  
- Illustrates that one can still use PCP for a simpler reduction  
- Signature minimisations are constructive for finite models

*e.g. Libkin (2010); Börger et al. (1997)
First-Order Satisfiability over Signatures

Given a signature $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$, we represent terms and formulas by:

$$
\begin{align*}
t : \text{Term}_\Sigma & ::= x \mid f \bar{t} \\
\varphi, \psi : \text{Form}_\Sigma & ::= \bot \mid P \bar{t} \mid \varphi \Box \psi \mid \nabla \varphi
\end{align*}
$$

$(x : \mathbb{N}, f : \mathcal{F}_\Sigma, \bar{t} : \text{Term}_\Sigma^{\mid f \mid})$

$(P : \mathcal{P}_\Sigma, \bar{t} : \text{Term}_\Sigma^{\mid P \mid})$

A model $\mathcal{M}$ over a domain $D$ is a pair of interpretation functions:

$$
\begin{align*}
\mathcal{M}^- : \forall f : \mathcal{F}_\Sigma. D^{\mid f \mid} & \rightarrow D \\
\mathcal{M}^- : \forall P : \mathcal{P}_\Sigma. D^{\mid P \mid} & \rightarrow \mathbb{P}
\end{align*}
$$

For assignments $\rho : \mathbb{N} \rightarrow D$ define evaluation $\hat{\rho} t$ and satisfaction $\mathcal{M} \models_\rho \varphi$:

$$
\begin{align*}
\hat{\rho} x & ::= \rho x \\
\hat{\rho} (f \bar{t}) & ::= f^{\mathcal{M}} (\hat{\rho} \bar{t}) \\
\mathcal{M} \models_\rho \bot & ::= \bot \\
\mathcal{M} \models_\rho \varphi \Box \psi & ::= \mathcal{M} \models_\rho \varphi \Box \mathcal{M} \models_\rho \psi \\
\mathcal{M} \models_\rho P \bar{t} & ::= P^{\mathcal{M}} (\hat{\rho} \bar{t}) \\
\mathcal{M} \models_\rho \nabla \varphi & ::= \nabla a : D. \mathcal{M} \models_{a,\rho} \varphi
\end{align*}
$$

$$\text{SAT}(\Sigma) \varphi ::= \text{there are } \mathcal{M} \text{ and } \rho \text{ such that } \mathcal{M} \models_\rho \varphi$$
Finiteness in Constructive Type Theory

Definition

A type $X$ is finite if there exists a list $l_X$ with $x \in l_X$ for all $x : X$.

This seems to be a good compromise:

- Easy to establish and work with
- Does not enforce discreteness
- Enough to get expected properties:
  - Every strict order on a finite type is well-founded
  - Every finite decidable equivalence relation admits a quotient on $\mathbb{F}_n$

\[
\text{FSAT}(\Sigma) \varphi \text{ if additionally } D \text{ is finite and all } P^M \text{ are decidable}
\]

\[
\text{FSATEQ}(\Sigma; \equiv) \varphi \text{ if } x \equiv^M y \iff x = y \text{ for all } x, y : D \text{ (hence discrete)}
\]
Encoding the Post Correspondence Problem

We use the signature $\Sigma_{BPCP} := (\{\star^0, e^0, f_{tt}^1, f_{ff}^1\}; \{P^2, \prec^2, \equiv^2\})$:

- Chains like $f_{ff}(f_{tt}(e))$ represent strings while $\star$ signals overflow
- $P$ concerns only defined values and $\prec$ is a strict ordering:

$$\begin{align*}
\varphi_P & := \forall xy. P x y \rightarrow x \not\equiv \star \land y \not\equiv \star \\
\varphi_{\prec} & := (\forall x. x \not\prec x) \land (\forall xyz. x \prec y \rightarrow y \prec z \rightarrow x \prec z)
\end{align*}$$

- Sanity checks on $f$ regarding overflow, disjointness, and injectivity:

$$\begin{align*}
\varphi_f & := \left( f_{tt} \star \equiv \star \land f_{ff} \star \equiv \star \right) \land \left( \forall xy. f_{tt} x \not\equiv \star \rightarrow f_{tt} x \equiv f_{tt} y \rightarrow x \equiv y \right) \\
& \land \left( \forall xy. f_{ff} x \not\equiv \star \rightarrow f_{ff} x \equiv f_{ff} y \rightarrow x \equiv y \right) \\
& \land \left( \forall x. f_{tt} x \not\equiv e \right) \land \left( \forall x. f_{ff} x \not\equiv e \right)
\end{align*}$$
Trakhtenbrot’s Theorem

Given an instance $R$ of PCP, we construct a formula $\varphi_R$ by:

$$
\varphi_R := \varphi_P \land \varphi_\prec \land \varphi_f \land \varphi_\triangleright \land \exists x. P x x
$$

Crucially, we enforce that $P$ satisfies the inversion principle of $R \triangleright (s, t)$:

$$
\varphi_\triangleright := \forall xy. P x y \rightarrow \bigvee_{(s,t) \in R} \exists uv. P u v \land x \equiv \bar{s} u \land y \equiv \bar{t} v \land u/v \prec x/y
$$

Theorem

PCP $R$ iff $\text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv)\varphi_R$, hence $\text{PCP} \preceq \text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv)$.

Proof.

If $R$ has a solution of length $n$, then $\varphi_R$ is satisfied by the model of strings of length bounded by $n$. Conversely, if $\mathcal{M} \models \varphi_R$ we can extract a solution of $R$ from $\varphi_\triangleright$ by well-founded induction on $\prec^\mathcal{M}$ (which is applicable since $\mathcal{M}$ is finite).
Signature Transformations

Given a finite and discrete signature $\Sigma$ with arities bounded by $n$, we have:

$$\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma) \preceq \text{FSAT}(\emptyset; P^{n+2}) \preceq \text{FSAT}(\emptyset; \in^2)$$

First reduction: axiomatise that $\equiv$ is a congruence for the symbols in $\Sigma$

Second reduction:

- Encode $k$-ary functions as $(k + 1)$-ary relations
- Align the relation arities to be constantly $n + 1$
- Merge relations into a single $(n + 2)$-ary relation indexed by constants
- Interpret constants with fresh variables

Caveat: intermediate reductions may rely on discrete models...
Discrete Models

FSAT'($\Sigma$) $\varphi$ if FSAT($\Sigma$) $\varphi$ on a discrete model

Can every finite model $\mathcal{M}$ be transformed to a discrete finite model $\mathcal{M}'$?

Idea: first-order indistinguishability $x \equiv y := \forall \varphi \rho. \mathcal{M} \vDash x \cdot \rho \varphi \iff \mathcal{M} \vDash y \cdot \rho \varphi$

Lemma

The relation $x \equiv y$ is a decidable congruence for the symbols in $\Sigma$.

Fact

FSAT'($\Sigma$) $\varphi$ iff FSAT($\Sigma$) $\varphi$, hence in particular FSAT'($\Sigma$) $\varphi \preceq$ FSAT($\Sigma$) $\varphi$.

Proof.

If $\mathcal{M} \vDash \rho \varphi$ pick $\mathcal{M}'$ to be the quotient of $\mathcal{M}$ under $x \equiv y$. 

$\square$
Compressing Relations: $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$

Intuition: encode $P \times_1 \ldots \times_n$ as $(x_1, \ldots, x_n) \in p$ for a set $p$ representing $P$

So let’s play set theory! For a set $d$ representing the domain we define $\varphi'_{\in}$:

$$(P \times_1 \ldots \times_n)'_{\in} := "(x_1, \ldots, x_n) \in p"$$

$$(\forall z. \varphi)'_{\in} := \forall z.z \in d \rightarrow (\varphi)'_{\in}$$

$$(\varphi \square \psi)'_{\in} := (\varphi)'_{\in} \square (\psi)'_{\in}$$

$$(\exists z. \varphi)'_{\in} := \exists z.z \in d \land (\varphi)'_{\in}$$

Then $\varphi_{\in}$ is $\varphi'_{\in}$ plus asserting $\in$ to be extensional and $d$ to be non-empty.

**Fact**

$\text{FSAT}(\emptyset; P^n) \varphi$ iff $\text{FSAT}(\emptyset; \in^2) \varphi_{\in}$, hence $\text{FSAT}(\emptyset; P^n) \preceq \text{FSAT}(\emptyset; \in^2)$.

**Proof.**

The hard direction is to construct a model of $\varphi_{\in}$ given a model $\mathcal{M}$ of $\varphi$. We employ a segment of the model of hereditarily finite sets by Smolka and Stark (2016) large enough to accommodate $\mathcal{M}$. 

□
Full Signature Classification

Composing all signature transformations verified we obtain:

**Theorem**

*If* $\Sigma$ *contains either an at least binary relation or a unary relation together with an at least binary function, then PCP reduces to FSAT$(\Sigma)$.***

On the other hand, FSAT for monadic signatures remains decidable:

**Theorem**

*If* $\Sigma$ *is discrete and has all arities bounded by 1 or if all relation symbols have arity 0, then FSAT$(\Sigma)$ is decidable.*

In any case, since one can enumerate all finite models up to extensionality:

**Fact**

*If* $\Sigma$ *is discrete and enumerable, then FSAT$(\Sigma)$ is enumerable.*
Example 3: First-Order Axiom Systems*

*K. and Marc Hermes at ITP’21.
General Idea

Is a formula $\varphi$ entailed by an axiomatisation $A$?

Strategy if $A$ is strong enough to capture computation:

- Encode Turing machine $M$ as formula $\varphi_M$
- Verify that $M$ halts iff $A \models \varphi_M$
- Verify that $M$ halts iff $A \vdash \varphi_M$ (→ direction by hand)
- Instead of TM use problems suitable to encode in $A$

As hard as consistency and incompleteness:

- Reducing a non-trivial problem $P$ to $A \vdash \varphi$ shows $A$ consistent
- Undecidability implies incompleteness for enumerable axiomatisations
Connections to Consistency and Incompleteness

Fact (Consistency)

If \( p \preceq A \vdash \) and there is \( x \) with \( \neg p \times \) then \( A \not\vdash \bot \).

Proof.

Let \( f \) witness \( p \preceq A \vdash \). Then \( A \not\vdash f \times \) by \( \neg p \times \) and thus \( A \not\vdash \bot \).

Fact (Synthetic Incompleteness)

If \( A \) is complete (\( \forall \varphi. A \vdash \varphi \lor A \vdash \neg \varphi \)) and consistent, then \( A \vdash \) is decidable.

Proof.

By application of Post’s theorem. The premises are enumerability of \( A \vdash \) (immediate), enumerability of its complement (as \( A \not\vdash \varphi \) iff \( A \vdash \neg \varphi \)), and logical decidability of \( A \vdash \) (as \( A \vdash \varphi \lor A \vdash \neg \varphi \) implies \( A \vdash \varphi \lor A \not\vdash \varphi \)).
Sketch for Peano Arithmetic

Use axiomatisation PA over standard signature \((0, S, +, \cdot; \equiv)\).

Diophantine constraints (cf. Larchey-Wendling and Forster (2019)):
- Instances are lists \(L\) of constraints \(x_i = 1 \mid x_i + x_j = x_k \mid x_i \cdot x_j = x_k\)
- \(L\) is solvable if there is an evaluation \(\eta : \mathbb{N} \rightarrow \mathbb{N}\) solving all constraints

**Theorem**

\[L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } \text{PA} \models \exists^n c_1 \land \cdots \land c_k.\]

**Proof.**

If \(L\) has solution \(\eta\) instantiate the existential quantifiers with numerals \(\eta_1, \ldots, \eta_n\). Then the axioms of PA entail the constraints.

If \(\text{PA} \models \exists^n c_1 \land \cdots \land c_k\) use the standard model \(\mathbb{N}\) to extract solution \(\eta\).

**Fact**

\[L = [c_1, \ldots, c_k] \text{ with maximal index } x_n \text{ is solvable iff } \text{PA} \vdash \exists^n c_1 \land \cdots \land c_k.\]
Interlude: Models of ZF

Sets-as-trees interpretation (Aczel (1978)):

- Type $\mathcal{T}$ of well-founded trees with constructor $\tau : \forall X. (X \to \mathcal{T}) \to \mathcal{T}$
- Equality of trees $s, t$ given by isomorphism $s \approx t$
- Membership defined by $s \in \tau X f := \exists x. s \approx f x$
- Set operations implemented by tree operations:
  - $\emptyset := \tau \bot \text{ elim}_\bot$
  - $\{ s, t \} := \tau \mathbb{B} (\lambda b. \text{ if } b \text{ then } s \text{ else } t)$
  - $\omega := \tau \mathbb{N} (\lambda n. \overline{n})$ where $\overline{0} := \emptyset$ and $\overline{S} n := \overline{n} \cup \{\overline{n}\}$
  - ...

Axioms needed in Coq:

- EM to really interpret ZF instead of IZF
- Replacement needs a type-theoretical choice axiom (Werner (1997))
- Strong quotient axiom for $(\mathcal{T}, \approx)$ suffices (Kirst and Smolka (2019))
- This yields a well-behaved model $S$: quotiented, standard numbers
Sketch for ZF Set Theory

Use axiomatisation ZF over explicit signature \((\emptyset, \{\_,\_\}, \cup, P, \omega; \equiv, \in)\).

Reduction from PCP:

- Boolean encoding: \(\text{tt} = \{\emptyset\}\) and \(\text{ff} = \emptyset\)
- String encoding: \(\text{tt ff ff tt} = (\text{tt}, (\text{ff}, (\text{tt}, (\text{ff}, \emptyset))))\)
- Stack encoding: \(\overline{S} = \{(\overline{s_1}, \overline{t_1}), \ldots, (\overline{s_k}, \overline{t_k})\}\)
- Combination encoding: \(S \leftrightarrow B := \bigcup_{s/t \in S} \{(\overline{s}x, \overline{ty}) \mid (x, y) \in B\}\)
- \(f \triangleright n := (\emptyset, \overline{S}) \in f \land \forall (k, B) \in f. k \in n \rightarrow (k + 1, S \leftrightarrow B) \in f\)

\[\varphi_S := \exists f, n, B, x. n \in \omega \land f \triangleright n \land (n, B) \in f \land (x, x) \in B\]

**Theorem**

\(\text{PCP } S \iff \text{ZF} \models \varphi_S \) and \(\text{PCP } S \iff \text{ZF} \vdash \varphi_S\).

**Proof.**

Direction \(\rightarrow\) by proofs in ZF and \(\leftarrow\) relies on standard model \(S\).
Conclusion
Ongoing and Future Work

- Undecidability and incompleteness of finitary set theories
- Minimalistic undecidability proof for the binary signature
- Undecidability and incompleteness of second-order logic
- Constructive analysis of Tennenbaum’s theorem
- Engineering: tool support, connect Coq developments
Take-Home Messages

- Synthetic computability: elegant formalism, feasible to mechanise
- Metamathematics: rewarding to revisit in constructive type theory
- Coq mechanisation: implements constructive proofs as algorithms
- If you work on undecidability proofs in Coq:
  Our library could help you and is open for contributions

Thank You!
Bibliography I


The Coq Development Team (2021). The coq proof assistant.