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

Yannick Forster, Dominik Kirst, Gert Smolka

CPP 2019
January 15
How to formalise decidability?

Classical 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:

- Work in a 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 formalise undecidability?

Problem of the synthetic approach:
- Constructive type theories like MLTT or CIC are consistent with the assumption that every problem is 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
- Deduce that the undecidability of the base problem implies the undecidability of the considered problem

(Again similar for other notions of computability theory)
Related Work

Andrej Bauer.
First steps in synthetic computability theory.

Yannick Forster, Edith Heiter, and Gert Smolka.
Verification of PCP-Related Computational Reductions in Coq.

Zohar Manna.
*Mathematical Theory of Computation*.

Alan M. Turing.
On computable numbers, with an application to the entscheidungsproblem.

Emil L. Post.
A variant of a recursively unsolvable problem.
Elementary Synthetic Computability Theory
Decidability and Enumerability

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

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

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

\[
\forall x. \ p x \iff \exists n. \ f n = \downarrow x \uparrow.
\]

**Fact**

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

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

Computability theory is usually developed for computational domains.

A type $X$ is called
- enumerable if $\lambda x. \top$ is enumerable,
- discrete if $\lambda xy. x = y$ is decidable, and
- data type if it is both enumerable and discrete.

Fact

Decidable predicates on data types are enumerable and co-enumerable.

Proof.

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

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

defines an enumerator for $p$. 

Y. Forster, D. Kirst, G. Smolka
Synthetic Undecidability in Coq
CPP 2019 – January 15
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 \( p \). \( \square \)
Post’s Theorem and Markov’s Principle

Post: Bi-enumerable predicates on data types are decidable.

Not directly provable, needs extra assumption:

Lemma

Post holds for logically decidable predicates.

Proof.

By (guarded) unbounded linear search and parallel enumeration.

Markov: \( \forall f : \mathbb{N} \to \mathbb{B}. \neg \neg (\exists n. f n = \text{tt}) \to \exists n. f n = \text{tt} \)

Theorem

Markov is equivalent to the logical decidability of bi-enumerable predicates on discrete types. So in particular, Markov is equivalent to Post.
The Post Correspondence Problem

Recap: given a **stack** \( S \) of cards \( s/t \), find a derivable match. This (undecidable) problem can be expressed by an inductive predicate:

\[
\begin{align*}
S \upharpoonright s/t & \quad \text{if } s/t \in S \\
S \upharpoonright u/v \quad s/t \in S & \quad \text{if } S \upharpoonright u/v \quad S \upharpoonright s/s \\
S \upharpoonright su/tv & \quad \text{if } S \in L \quad (s/t) \in S \\
\end{align*}
\]

**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

\[
\begin{align*}
L_0 & := [] \\
L(S n) & := L n + [(S, (s, t)) \mid S \in L \quad (s, t) \in S] \\
& \quad + [(S, (su, tv)) \mid (S, (u, v)) \in L n \quad (s, t) \in S]
\end{align*}
\]

defines a list enumerator for \( \lambda Sst. S \upharpoonright s/t \).
Undecidability of First-Order Logic
Syntax and Tarski Semantics

Terms and formulas are defined for a fixed signature:

\[ \tau : \mathcal{T} := x \mid a \mid e \mid g_{tt} \tau \mid g_{ff} \tau \quad x, a : \mathbb{N} \]

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

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

\[ \rho \models_{\mathcal{I}} \bot := \bot \]

\[ \rho \models_{\mathcal{I}} Q := Q^\mathcal{I} \]

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

\[ \rho \models_{\mathcal{I}} \varphi \rightarrow \psi := \rho \models_{\mathcal{I}} \varphi \rightarrow \rho \models_{\mathcal{I}} \psi \]

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

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

Strings can be encoded as terms, e.g. $ttff ff tt = g_{tt} (g_{ff} (g_{ff} (g_{tt} e)))$.

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

$$
e^B := []$$
$$g^B_b s := b :: s$$

$$Q^B := \text{PCP } S$$
$$P^B s t := S \triangleright s / t.$$ 

Lemma

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

Fact

Markov $\rightarrow \neg\neg \rho \models^B \varphi \rightarrow \rho \models^B \varphi$
Undecidability of Validity

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

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

**Theorem**

PCP $S$ iff $\varphi_S$ is valid.

**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 $I \models \varphi_S$. \qed
Undecidability of Minimal Provability

We define a minimal natural deduction system inductively:

\[
\begin{align*}
\varphi \in A & \Rightarrow A \vdash \varphi \\
\varphi :: A \vdash \psi & \Rightarrow I \frac{A \vdash \varphi \rightarrow \psi \quad A \vdash \varphi}{A \vdash \psi} \\
A \vdash \varphi^x_a & \quad a \notin \mathcal{P}(\varphi) \cup \mathcal{P}(A) \Rightarrow AI \frac{A \vdash \varphi^x_a \quad \forall x. \varphi \quad \mathcal{V}(\tau) = \emptyset}{A \vdash \forall x. \varphi} \\
\end{align*}
\]

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

Fact (Soundness)

\[A \vdash \varphi \text{ implies } A \models \varphi, \text{ so provable formulas are valid.}\]

Theorem

- PCP \( S \) iff \( \varphi_S \) is provable.  
  (as before using soundness)
- Provability is enumerable.  
  (by giving a list enumerator)
Undecidability of Classical Provability

We extend the deduction system by a classical rule for falsity:

\[
\frac{A \vdash C \vdash \neg \varphi}{A \vdash C \varphi} \quad \text{DN}
\]

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.

**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 \( \vdash \varphi_S^Q \). Then by soundness \( \mathcal{B} \models \varphi_S^Q \) which implies \( \mathcal{B} \models Q \) and PCP \( S \) as before. \( \square \)
Wrap-Up
Coq Formalisation

Structure:
- Elementary synthetic computability (900 loc)
- Metatheory of first-order logic (750 loc)
- Undecidability of first-order logic (550 loc)

Features:
- Tagged inductive types and predicates representing syntax and deduction systems to avoid code duplication
- Convenient method to define enumerators for inductive types and predicates via cumulative lists
- Type class inference for automated decidability proofs

www.ps.uni-saarland.de/extras/fol-undec
Future Work

- Constructive metatheory of first-order logic (completeness, syntax representations, ...)

- Realisability model of the calculus of inductive constructions witnessing (the propositional version) of excluded middle

- Automated translation of Coq function definitions into a concrete model of computation (e.g. call-by-value lambda calculus)

- Library of formalised undecidability results
  - Particular axiomatic theories (PA, ZF, ...)
  - Diophantine equations (Hilbert’s 10th problem)
  - Higher-order unification
  - ...

www.github.com/uds-psl/coq-library-undecidability
Backup
## Development Details

<table>
<thead>
<tr>
<th>File</th>
<th>Spec</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prelim.v</td>
<td>402</td>
<td>456</td>
</tr>
<tr>
<td>DecidableEnumerable.v</td>
<td>124</td>
<td>257</td>
</tr>
<tr>
<td>Reductions.v</td>
<td>31</td>
<td>38</td>
</tr>
<tr>
<td>MarkovPost.v</td>
<td>38</td>
<td>82</td>
</tr>
<tr>
<td>PCP.v</td>
<td>42</td>
<td>50</td>
</tr>
<tr>
<td>Infinite.v</td>
<td>103</td>
<td>134</td>
</tr>
<tr>
<td>FOL.v</td>
<td>49</td>
<td>78</td>
</tr>
<tr>
<td>Semantics.v</td>
<td>107</td>
<td>21</td>
</tr>
<tr>
<td>Deduction.v</td>
<td>124</td>
<td>165</td>
</tr>
<tr>
<td>Kripke.v</td>
<td>89</td>
<td>142</td>
</tr>
<tr>
<td>Weakening.v</td>
<td>61</td>
<td>83</td>
</tr>
<tr>
<td>BPCP_FOL.v</td>
<td>151</td>
<td>163</td>
</tr>
<tr>
<td>BPCP_IFOL.v</td>
<td>58</td>
<td>16</td>
</tr>
<tr>
<td>BPCP_CND.v</td>
<td>65</td>
<td>93</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1461</td>
<td>1788</td>
</tr>
</tbody>
</table>
Undecidability of Satisfiability

**Theorem**

\( \neg (\text{PCP } S) \) iff \( \neg \varphi_S \) is satisfiable.

**Proof.**

Suppose that \( \neg (\text{PCP } S) \). We show that \( B \models \neg \varphi_S \), so let \( B \models \varphi_S \). As before this implies that PCP \( S \), contradiction.

Now suppose that \( I \models \neg \varphi_S \) and that PCP \( S \). The latter implies that \( \varphi_S \) is valid, contradicting the former.
Kripke Semantics

Definition

Kripke models $\mathcal{M}$ consists of a domain $D$, an assignment $\eta : \mathbb{N} \rightarrow D$, and

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

Definition

We define the forcing relation $\rho, w \models_\mathcal{M} \varphi$ on Kripke models by

\[
\begin{align*}
\rho, w & \models_\mathcal{M} \bot := \bot \\
\rho, w & \models_\mathcal{M} Q := Q^w
\end{align*}
\]

\[
\begin{align*}
\rho, w & \models_\mathcal{M} P \tau_1 \tau_2 := P^w (\hat{\rho} \tau_1) (\hat{\rho} \tau_2) \\
\rho, w & \models_\mathcal{M} \varphi \rightarrow \psi := \forall w'. w \leq w' \rightarrow \rho, w' \models_\mathcal{M} \varphi \rightarrow \rho, w' \models_\mathcal{M} \psi \\
\rho, w & \models_\mathcal{M} \forall x. \varphi := \forall w'. w \leq w' \rightarrow \forall d. \rho[x := d], w' \models_\mathcal{M} \varphi
\end{align*}
\]