Undecidability and Incompleteness of Second-Order Logic

Mark Koch

April 12, 2021

We give a proof of the incompleteness of second-order logic under Markov's principle as well as proofs for the undecidability of validity, satisfiability and second-order Peano arithmetic. We first show the categoricity of PA_2 and conclude undecidability and incompleteness using a reduction from the solvability of Diophantine equations (Hilbert's 10th problem). All results are formalized with the Coq proof assistant.

1 Syntax and Semantics of Second-Order Logic

For the syntax and semantics we largely follow the existing formalization of first-order logic developed in [1] and [2]. We represent the terms and formulas of second-order logic as inductive types, using function symbols $\mathsf{f} : \Sigma_F$ and predicate symbols $\mathsf{P} : \Sigma_P$ with arities $|\mathsf{f}|$ and $|\mathsf{P}|$ from a fixed signature $\Sigma = (\Sigma_F, \Sigma_P)$. We use a de Bruijn encoding, where a bound variable is represented by the number of quantifiers shadowing its binder.

Definition 1 (Syntax) We define functions $f : \mathcal{F}_n$ and predicates $P : \mathcal{P}_n$ with arity $n : \mathbb{N}$, as well as terms $t : \mathbb{T}$ and formulas $\varphi : \mathbb{F}$ by

$$f ::= \mathsf{fvar}_x \mid \mathsf{f}, \qquad P ::= \mathsf{pvar}_x \mid \mathsf{P}, \qquad t ::= \mathsf{ivar}_x \mid f\ \vec{t},$$

$$\varphi, \psi ::= \dot{\bot} \mid P\ \vec{t} \mid \varphi \dot{\to} \psi \mid \varphi \dot{\land} \psi \mid \varphi \dot{\lor} \psi \mid \dot{\forall}\ \varphi \mid \dot{\exists}_i\ \varphi \mid \dot{\forall}^n_f\ \varphi \mid \dot{\exists}^n_f\ \varphi \mid \dot{\forall}^n_p\ \varphi \mid \dot{\exists}^n_p\ \varphi$$

where f and P have the expected arity, $\vec{t}$ is a vector of the expected arity and $x : \mathbb{N}$. We call a formula *first-order*, if it neither contains function or predicate quantifiers, nor function or predicate variables. A formula is *closed*, if it does not contain any free variables. Finally, we write $\dot{\neg}\varphi$ for $\varphi \dot{\to} \dot{\bot}$.

Note that instead of defining quantifiers and variables for individuals, we could also represent them as nullary functions. The benefit of having an explicit representation of individuals is that we can easier define the fragment of first-order formulas this way. As we will make heavy use of this later, we chose this approach.

We also want to point out, that we annotate function and predicate quantifiers with the arity of the object they quantify over. Alternatively, quantifiers could quantify over over functions and predicates of all arities and the right one is picked based each usage. This however would allow unintuitive formulas like $\forall P.\, P(x) \to P(a, b)$, where P is unary and binary at the same time. In our approach, the quantifier would instead specify the arity of P. For example, if P is unary, the binary occurrence should no longer be bound to the quantifier, but be free instead. This of course will make the semantics slightly more complicated, as the scoping of variables now depends on the arity.

Definition 2 (Tarski semantics) A *model* $\mathcal{M}$ over some signature Σ consists of a domain D as well as interpretation functions for function and predicate symbols:

$$i_f : \forall \mathsf{f} : \Sigma_f.\, D^{|\mathsf{f}|} \to D \qquad\qquad i_p : \forall \mathsf{P} : \Sigma_p.\, D^{|\mathsf{P}|} \to \mathbb{P}$$

Environments $\rho = \langle \rho_i, \rho_f, \rho_p \rangle$ consist of assignments

$$\rho_i : \mathbb{N} \to D \qquad\qquad \rho_f : \mathbb{N} \to \forall n.\, D^n \to D \qquad\qquad \rho_p : \mathbb{N} \to \forall n.\, D^n \to \mathbb{P}$$

for individual, function and predicate variables. Those are extended to term evaluations $[\![\cdot]\!]_\rho : \mathbb{T} \to D$ and formula satisfiability $\rho \vDash \varphi$ with

$$[\![\mathsf{ivar}_x]\!]_\rho := \rho_i\, x \qquad [\![\mathsf{f}\, \vec{t}\,]\!]_\rho := i_f\, \mathsf{f}\, [\![\vec{t}\,]\!]_\rho \qquad [\![(\mathsf{fvar}_x : \mathcal{F}_n)\, \vec{t}\,]\!]_\rho := \rho_f\, x\, n\, [\![\vec{t}\,]\!]_\rho$$

$$\begin{aligned} \rho \vDash \mathsf{P}\, \vec{t} &:= i_p\, \mathsf{P}\, [\![\vec{t}\,]\!]_\rho & \rho \vDash \dot{\forall}\varphi &:= \forall d : D.\, \langle d :: \rho_i, \rho_f, \rho_p \rangle \vDash \varphi \\ \rho \vDash (\mathsf{fvar}_\mathsf{x} : \mathcal{P}_p)\, \vec{t} &:= \rho_p\, x\, n\, [\![\vec{t}\,]\!]_\rho & \rho \vDash \dot{\forall}^n_f \varphi &:= \forall f : D^n \to D.\, \langle \rho_i, f :: \rho_f, \rho_p \rangle \vDash \varphi \\ \rho \vDash \varphi \mathrel{\dot{\to}} \psi &:= \rho \vDash \varphi \to \rho \vDash \psi & \rho \vDash \dot{\forall}^n_p \varphi &:= \forall P : D^n \to \mathbb{P}.\, \langle \rho_i, \rho_f, P :: \rho_p \rangle \vDash \varphi. \end{aligned}$$

The remaining cases map to their meta-level counterpart in a similar way. The modified environment $d :: \rho_i$ in the individual quantifier maps 0 to d and Sx to $\rho_i\, x$. The function and predicate quantifiers only replace the function with the corresponding arity, i.e.

$$(f :: \rho_f)\, 0\, n := \begin{cases} f & \text{if } f \text{ has arity } n \\ \rho_f\, 0\, n & \text{otherwise} \end{cases}.$$

We write $\mathcal{M} \vDash \varphi$ if $\rho \vDash \varphi$ for all ρ, and for a theory $\mathcal{T} : \mathbb{F} \to \mathbb{P}$, we write $\mathcal{M} \vDash \mathcal{T}$ if $\mathcal{T}\psi$ implies $\mathcal{M} \vDash \psi$ for all ψ. Finally, we write $\mathcal{T} \vDash \varphi$ if $\mathcal{M} \vDash \varphi$ for every model $\mathcal{M}$ with $\mathcal{M} \vDash \mathcal{T}$.

In the definition above, functions are simply interpreted as type theoretic functions. Alternatively, we could also use total, functional relations instead. As a shorthand, we will write $X \hookrightarrow Y$ for the type describing a total functional relation from X to Y:

$$X \hookrightarrow Y := \Sigma F : X \to Y \to \mathbb{P}.\, \mathsf{functional}\, F \wedge \mathsf{total}\, F$$

We also use function application notation to represent applications of the first projection of this dependent pair.

In our setting those two representations of functions would be indeed different, as functions in our type theory are always computable (in the absence of axioms), which is not the case for relations. We will see later, that this also makes a difference when looking at the categoricity of PA_2. Therefore, we now also define this alternative Tarski semantics.

Definition 3 (Relational Tarski semantics) The function interpretation and variable assignment in a *relational Model* $\hat{\mathcal{M}}$ have types $i_f : \forall \mathsf{f}.\, D^{|\mathsf{f}|} \hookrightarrow D$ and $\rho_f : \mathbb{N} \to \forall n.\, D^n \hookrightarrow D$. Term evaluation turns into a relation $[\![\cdot]\!]'_\rho : \mathbb{T} \to D \to \mathbb{P}$ and we get formula satisfiability $\rho \vDash' \varphi$ with

$$[\![\mathsf{ivar}_x]\!]'_\rho\, d := \rho_i\, x = d \qquad [\![(\mathsf{fvar}_x : \mathcal{F}_n)\, \vec{t}\,]\!]'_\rho\, d := \forall \vec{v}.\, [\![\vec{t}\,]\!]'_\rho\, \vec{v} \to \rho_f\, x\, n\, d$$

$$[\![\mathsf{f}\, \vec{t}\,]\!]'_\rho\, d := \forall \vec{v}.\, [\![\vec{t}\,]\!]'_\rho\, \vec{v} \to i_f\, \mathsf{f}\, v\, d \qquad \rho \vDash' \mathsf{P}\, \vec{t} := \forall \vec{v}.\, [\![\vec{t}\,]\!]'_\rho\, \vec{v} \to i_p\, \mathsf{P}\, v$$

$$\rho \vDash' \dot{\forall}^n_f \varphi := \forall F : D^n \hookrightarrow D.\, \langle \rho_i, F :: \rho_f, \rho_p \rangle \vDash \varphi$$

Fact 4 Under unique choice, we can translate between the two representations of environments and both semantics agree, i.e.

$$\mathsf{UC} \to (\rho \vDash \varphi \;\leftrightarrow\; \mathsf{toRelEnv}\; \rho \vDash' \varphi), \qquad \mathsf{UC} \to (\mathsf{toFunEnv}\; \rho \vDash \varphi \;\leftrightarrow\; \rho \vDash' \varphi)$$

where $\mathsf{UC} := \forall R : X \to Y \to \mathbb{P}.\mathsf{functional}\; R \to \mathsf{total}\; R \to \forall x. \Sigma y. R\; x\; y.$

Fact 5 Both semantics agree for first-order formulas.

Fact 6 If φ is closed, then $\rho \vDash \varphi \leftrightarrow \sigma \vDash \varphi$ for all environments ρ, σ.

2 Categoricity of Peano Arithmetic

An important area where second-order logic differs from its first-order counterpart is the notion of so-called *categoricity*. A theory $\mathcal{T}$ is called *categorical*, if all models that satisfy $\mathcal{T}$ are equal up to isomorphism. Thus, $\mathcal{T}$ uniquely determines its domain and the interpretation of its function and predicate symbols. In other words $\mathcal{T}$ is strong enough to uniquely characterize its own intended structure.

One such theory we will now look at is the theory PA_2 of second-order Peano arithmetic. This will highlight an important difference to first-order logic, where Peano arithmetic is *not* categorical. In fact, because of the Löwenheim-Skolem theorems, first-order logic allows non-standard models of any theory with an infinite model. Therefore it also has models of arithmetic, that are different from the natural numbers $\mathbb{N}$. We will see, that this is not the case for second-order logic and we will later use this fact to conclude undecidability and incompleteness.

The signature we will now work in contains symbols for the constant zero, the sucessor, addition and multiplication functions, as well as an equality predicate:

$$(O, S\_\,, \_\oplus\_\,, \_\otimes\_\,; \_\equiv\_)$$

Next, we give the axioms that make up the theory PA_2. Note, that for better legibility we write formulas with named binders instead of the de Bruijn representation.

$$\begin{array}{llll}
\oplus\text{-zero}: & \dot\forall x.\, O \oplus x \equiv x & \oplus\text{-rec}: & \dot\forall xy.\, (Sx) \oplus y \equiv S(x \oplus y) \\
\otimes\text{-zero}: & \dot\forall x.\, O \otimes x \equiv O & \otimes\text{-rec}: & \dot\forall xy.\, (Sx) \otimes y \equiv y \oplus x \otimes y \\
\text{zero-succ}: & \dot\forall x.\, O \equiv Sx \,\dot\to\, \dot\bot & \text{succ-inj}: & \dot\forall xy.\, Sx \equiv Sy \,\dot\to\, x \equiv y \\
\equiv\text{-refl}: & \dot\forall x.\, x \equiv x & \equiv\text{-symm}: & \dot\forall xy.\, x \equiv y \,\dot\to\, y \equiv x
\end{array}$$

$$\text{induction}: \quad \dot\forall^1_p P.\, P\ O \,\dot\to\, (\dot\forall x.\, P\ x \,\dot\to\, P\ (Sx)) \,\dot\to\, \dot\forall x.\, P\ x$$

The most notable axiom is the induction axiom, as it is the only one that fully exploits the power of second-order logic and cannot be expressed in first-order logic. There, one cannot quantify over predicates and must instead work with an axiom scheme over formulas with a free variable. This is of course much weaker and is also the reason why the categoricity proof will not work for first-order PA.

Also note that equality is actually definable in second order logic by encoding the Leibniz characterization, but we chose to add an equality symbol with axioms instead. Otherwise, formulas using equality could no longer be first-order, which would break our reduction later on. Nonetheless we can show, that the predicate coincides with equality:

Fact 7 Let $\equiv^i$ be the interpretation of the $\equiv$-symbol in some model $\mathcal{M}$ with $\mathcal{M} \vDash \mathsf{PA}_2$. Then $x \equiv^i y \leftrightarrow x = y$.

Proof. By induction on x and case analysis on y using the induction axiom. The different cases follow with zero-succ, succ-inj, $\equiv$-refl and $\equiv$-symm.[1] ■

[1]Notice that we do not require an axiom for transitivity. It turns out that those four axioms together with induction are already strong enough to imply transitivity.

Finally, we want to prove that PA_2 is categorical: Suppose we have two models $\mathcal{M}_1$, $\mathcal{M}_2$ with corresponding domains D_1, D_2 that satisfy PA_2 (i.e. $\mathcal{M}_1 \vDash \mathsf{PA}_2$ and $\mathcal{M}_2 \vDash \mathsf{PA}_2$). We will write S_1^i, S_2^i, $\oplus_1^i$, $\oplus_2^i$ etc. for the interpretation of function and predicate symbols in the two models. Our goal now is to give an isomorphism between $\mathcal{M}_1$ and $\mathcal{M}_2$.

Definition 8 We define the relation $\approx \, : D_1 \to D_2 \to \mathbb{P}$ inductively by $O_1^i \approx O_2^i$ and $S_1^i x \approx S_2^i y$ if $x \approx y$.

Fact 9 The relation $\approx$ is total, functional, surjective and injective. It also respects the structure of the models:

1. $O_1^i \approx O_2^i$
2. $x \approx y \to S_1^i x \approx S_2^i y$
3. $x \approx y \to x' \approx y' \to x \oplus_1^i x' \approx y \oplus_2^i y'$
4. $x \approx y \to x' \approx y' \to x \otimes_1^i x' \approx y \otimes_2^i y'$
5. $x \approx y \to x' \approx y' \to (x \equiv_1^i x' \leftrightarrow y \equiv_2^i y')$

Proof. Totality etc. follow by induction using the induction axiom. 1 and 2 hold by definition. 3 and 4 follow by induction on x using the axioms for $\oplus$ and $\otimes$. 5 follows by induction on x using Fact 7. ■

We can conclude that $\approx$ is the isomorphism we were looking for and that PA_2 is indeed categorical. However, we need to point out that the isomorphism is in fact not computational. For example, although knowing that it is total, given an $x : D_1$, we actually cannot compute a y with $x \approx y$. The reason is that our only means of finding out the structure of x is by using the induction axiom. But as this axiom is completely restricted to propositions, we will not be able to extract computational information from it. This has some important consequences: For example, one interesting property that we would expect to hold, is that satisfiability of formulas agrees in all models of PA_2:

$$\rho \approx \sigma \to (\mathcal{M}_1, \rho \vDash \varphi \leftrightarrow \mathcal{M}_2, \sigma \vDash \varphi)$$

where $\rho \approx \sigma$ extends the isomorphism to environments in the obvious way. But unfortunately we cannot prove this using the semantics from Definition 2. Imagine there exists some function f_1 in $\mathcal{M}_1$ that has some property. We would need to translate this function to $\mathcal{M}_2$ in order to show that the property also holds there, but this translation would require a computational isomorphism, so the statement above cannot be proven.

That is the reason, why having a relational semantics as introduced in Definition 3 is useful. There, functions are propositional and thus no computational isomorphism is required in order to translate from one domain to the other.

Lemma 10 For all relational Models $\hat{\mathcal{M}}_1$, $\hat{\mathcal{M}}_2$ and environments ρ, σ with $\rho \approx \sigma$ it holds that $\hat{\mathcal{M}}_1, \rho \vDash' \varphi \leftrightarrow \hat{\mathcal{M}}_2, \sigma \vDash' \varphi$ for all formulas φ.

Of course, if the formula does not contain any function quantifiers that would need to be translated, we can also get the result for the original function semantics.

Lemma 11 For all Models $\mathcal{M}_1$, $\mathcal{M}_2$ and environments ρ, σ with $\rho \approx \sigma$ it holds that $\mathcal{M}_1, \rho \vDash \varphi \leftrightarrow \mathcal{M}_2, \sigma \vDash \varphi$ for all formulas φ without function quantifiers.

Lemma 12 If $\rho \vDash \varphi$ for a closed first-order formula φ in some model $\mathcal{M}$ of PA_2, then φ is true in any model of PA_2, i.e. $\mathsf{PA}_2 \vDash \varphi$.

Lemma 12 is the main property that will later allow us to conclude the incompleteness of second-order logic. Since the reduction we will use is entirely based on first-order formulas, the argument would work in both semantics (see Fact 5). But since the function semantics is easier to work with, we will continue with it.

3 Undecidability and Incompleteness of Second-Order Logic

We now want to prove that no sound, complete and enumerable deduction system for second-order logic can exist. Our approach will be to show that the existence of such a deduction system would imply the decidability of the solvability of Diophantine equations (Hilbert's 10th problem H_{10}), which is known to be undecidable [3]. This has already been verified up to the halting problem for Turing machines as part of the library of undecidability proofs [4]. We rely on a synthetic approach of undecidability [5], based on the computability of all functions in Coq's type theory, following the reduction in [6].

H_{10} is about deciding whether a Diophantine equation $p = q$ has a solution in the natural numbers, where p and q are polynomials given by

$$p, q ::= \mathsf{num}\ n \mid \mathsf{var}\ x \mid \mathsf{add}\ p\ q \mid \mathsf{mul}\ p\ q \qquad (n, x : \mathbb{N})$$

The evaluation $\langle\!\langle \cdot \rangle\!\rangle_\alpha$ of polynomials under a variable assignment $\alpha : \mathbb{N} \to \mathbb{N}$ is given by

$$\begin{aligned} \langle\!\langle \mathsf{num}\ n \rangle\!\rangle_\alpha &:= n & \qquad \langle\!\langle \mathsf{add}\ p\ q \rangle\!\rangle_\alpha &:= \langle\!\langle p \rangle\!\rangle_\alpha + \langle\!\langle q \rangle\!\rangle_\alpha \\ \langle\!\langle \mathsf{var}\ x \rangle\!\rangle_\alpha &:= \alpha\ x & \qquad \langle\!\langle \mathsf{mul}\ p\ q \rangle\!\rangle_\alpha &:= \langle\!\langle p \rangle\!\rangle_\alpha \cdot \langle\!\langle q \rangle\!\rangle_\alpha \end{aligned}$$

and a Diophantine equation $p = q$ has a solution, if there is a variable assignment α with $\langle\!\langle p \rangle\!\rangle_\alpha = \langle\!\langle q \rangle\!\rangle_\alpha$. We now want to encode such an equation in PA_2. Because of Lemma 12, the choice of which PA_2 model to work in does not matter. To make our lives easier, we will work with the standard model $\mathbb{N}$, as this coincides with the domain of H_{10}. We start with encoding polynomials as terms by defining a translation function η:

$$\eta\ (\mathsf{num}\ n) := \nu(n) \qquad \eta\ (\mathsf{add}\ p\ q) := \eta\ p + \eta\ q$$
$$\eta\ (\mathsf{var}\ x) := \mathsf{ivar}_x \qquad \eta\ (\mathsf{mul}\ p\ q) := \eta\ p \cdot \eta\ q$$

where ν is recursively defined by $\nu(0) = O$ and $\nu(n+1) = S(\nu(n))$.

Lemma 13 The evaluation of the encoded term in the standard model is equal to the evaluation of the polynomial: $[\![\eta\ p]\!]^{\mathbb{N}}_{\langle\alpha,\rho_f,\rho_p\rangle} = \langle\!\langle p\rangle\!\rangle_\alpha$

Now we can encode a Diophantine equation $p = q$ as a formula $\varphi_{p,q}$ by binding all variables with existential quantifiers. Let n be the largest variable occurring in p or q, then set

$$\varphi_{p,q} := \dot{\exists}_i x_1 ... x_n.\, \eta\ p \equiv \eta\ q$$

Lemma 14 $p = q$ has a solution iff $\mathbb{N} \vDash \varphi_{p,q}$.

Corollary 15 $p = q$ has a solution iff $\mathsf{PA}_2 \vDash \varphi_{p,q}$.

Proof. Follows with Lemma 14 and Lemma 12 since $\varphi_{p,q}$ is closed and first-order.■

Since the theory PA_2 is finite, we can encode it as a conjunction in a formula φ_{PA_2}. This way we can reduce to arbitrary models, which will give us some other interesting undecidability results unrelated to incompleteness:

Corollary 16

1. $p = q$ has a solution iff $\forall \mathcal{M}.\, \mathcal{M} \vDash \varphi_{\mathsf{PA}_2} \dot{\rightarrow} \varphi_{p,q}$.
2. $p = q$ has a solution iff $\exists \mathcal{M}.\, \mathcal{M} \vDash \varphi_{\mathsf{PA}_2} \dot{\wedge} \varphi_{p,q}$.

Theorem 17 (Undecidability) The decidability of the following problems over the PA_2 signature implies the decidability of H_{10}:

1. Validity in second-order logic: $\lambda\varphi.\, \forall \mathcal{M}.\, \mathcal{M} \vDash \varphi$
2. Satisfiability in second-order logic: $\lambda\varphi.\, \exists \mathcal{M}.\, \mathcal{M} \vDash \varphi$
3. Validity in second-order PA_2: $\lambda\varphi.\, \mathsf{PA}_2 \vDash \varphi$

Now, we can finally use this reduction to conclude incompleteness. Suppose we had a deduction system $\vdash\ : \mathsf{list}\,\mathbb{F} \rightarrow \mathbb{F} \rightarrow \mathbb{P}$ which is

1. Sound: $\mathcal{L} \vdash \varphi \rightarrow (\lambda\psi.\, \psi \in \mathcal{L}) \vDash \varphi$
2. Complete: $(\lambda\psi.\, \psi \in \mathcal{L}) \vDash \varphi \rightarrow \mathcal{L} \vdash \varphi$
3. Enumerable: $\mathsf{enumerable}\ (\lambda\varphi.\, \mathcal{L} \vdash \varphi)$.

Note that the deduction system works with finite theories represented as lists. As PA_2 is finite, we will now also write PA_2 for the list representation of the theory.

Corollary 18 $p = q$ has a solution iff $\mathsf{PA}_2 \vdash \varphi_{p,q}$.

Therefore, to show that H_{10} is decidable, it suffices to show that $\lambda(p, q).\, \mathsf{PA}_2 \vdash \varphi_{p,q}$ is decidable. We achieve this by using Post's theorem [5, 7], which states that under Markov's principle[2] enumerability and co-enumerability of a problem on a discrete type implies decidability. As polynomials are discrete and enumerability follows from the enumerability of our deduction system, the only remaining obligation is to show co-enumerability, which will follow from this lemma:

Lemma 19 For all closed, first-order formulas φ it holds that $\mathcal{T} \vdash \dot\neg\varphi \leftrightarrow \neg\mathcal{T} \vdash \varphi$.

Proof. $\rightarrow$ follows by soundness. $\leftarrow$ follows by completeness and Lemma 12. ■

Note that this is the key point in the incompleteness proof where categoricity is required. The previous use in Corollary 15 is merely for convenience to allow us to work in the standard model. With some more effort one can get Corollary 18 without using categoricity [6]. Lemma 19 however only works because of categoricity and does not hold in first-order logic. There, we cannot conclude that all PA_2 models satisfy $\dot\neg\varphi$ just from knowing that not all PA_2 models satisfy φ. This requires categoricity.

Using Lemma 19, we get co-enumerability and thus incompleteness.

Lemma 20 $\mathsf{enumerable}\ (\lambda(p, q).\, \neg\mathsf{PA}_2 \vdash \varphi_{p,q})$

Proof. By Lemma 19, it suffices to show $\mathsf{enumerable}\ (\lambda(p, q).\, \mathsf{PA}_2 \vdash \dot\neg\varphi)$. This follows from the enumerability of the deduction system. ■

Lemma 21 $\mathsf{MP} \rightarrow \mathsf{decidable}\ (\lambda(p, q).\, \mathsf{PA}_2 \vdash \varphi_{p,q})$

Proof. By Post's theorem [5, 7]. ■

Theorem 22 (Incompleteness) The existence of a sound, complete and enumerable deduction system for second-order logic implies the decidability of H_{10} under MP.

[2]Markov's principle is a weak and constructively widely accepted consequence of excluded middle. It is defined as $\mathsf{MP} := \forall f : \mathbb{N} \rightarrow \mathbb{B}. \neg\neg(\exists n. fn = \mathsf{true}) \rightarrow \exists n. fn = \mathsf{true}$. MP is consistent with Church's thesis [8]. Therefore we can still work with the assumption that all functions in Coq are computable and our reduction stays computable.

References

[1] Yannick Forster, Dominik Kirst, and Dominik Wehr. Completeness theorems for first-order logic analysed in constructive type theory. In *International Symposium on Logical Foundations of Computer Science*, pages 47–74. Springer, 2020.

[2] Dominik Kirst and Dominique Larchey-Wendling. Trakhtenbrot's theorem in coq. In *International Joint Conference on Automated Reasoning*, pages 79–96. Springer, 2020.

[3] Yuri Matiyasevich. *Martin Davis and Hilbert's Tenth Problem*, pages 35–54. Springer International Publishing, Cham, 2016.

[4] Dominique Larchey-Wendling and Yannick Forster. Hilbert's tenth problem in coq. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019*, volume 131, pages 27–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[5] Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, page 38–51, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Dominik Kirst and Marc Hermes. Synthetic undecidability and incompleteness of first-order axiom systems in coq. 2021.

[7] Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

[8] Andrew W Swan and Taichi Uemura. On church's thesis in cubical assemblies. 2019.