Undecidability, Incompleteness, and Completeness of Second-Order Logic in Coq

Mark Koch
Saarland University
Saarland Informatics Campus, Saarbrücken, Germany

Dominik Kirst
Saarland University
Saarland Informatics Campus, Saarbrücken, Germany
kirst@cs.uni-saarland.de

Abstract

We mechanise central metatheoretic results about second-order logic (SOL) using the Coq proof assistant. Concretely, we consider undecidability via many-one reduction from Diophantine equations (Hilbert's tenth problem), incompleteness regarding full semantics via categoricity of second-order Peano arithmetic, and completeness regarding Henkin semantics via translation to mono-sorted first-order logic (FOL). Moreover, this translation is used to transport further characteristic properties of FOL to SOL, namely the compactness and Löwenheim-Skolem theorems.

CCS Concepts: • **Theory of computation** → **Type theory**; *Constructive mathematics*; Higher order logic.

Keywords: second order logic, undecidability, completeness

ACM Reference Format:
Mark Koch and Dominik Kirst. 2022. Undecidability, Incompleteness, and Completeness of Second-Order Logic in Coq. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '22), January 17–18, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3497775.3503684

1 Introduction

Second-order logic (SOL) extends the standard formalism of first-order logic (FOL) with quantifiers for relations and, depending on the presentation, functions over individuals. As a consequence, SOL can be seen as a more suitable general purpose language for mathematics than FOL (for instance argued by Shapiro [33]), capturing the intuitive treatment of relations and functions as first-class objects in common mathematical practice. For instance, the induction principle in first-order Peano arithmetic is typically stated as a scheme

$$\mathsf{Ind}_\varphi \;:=\; \varphi(0) \rightarrow (\forall n.\, \varphi(n) \rightarrow \varphi(n+1)) \rightarrow \forall n.\, \varphi(n)$$

approximating unary relations by first-order formulas φ, while it can be formulated by the single axiom

$$\mathsf{Ind}_2 \;:=\; \forall P.\, P(0) \rightarrow (\forall n.\, P(n) \rightarrow P(n+1)) \rightarrow \forall n.\, P(n)$$

in second order Peano arithmetic (PA_2), quantifying over all unary predicates P.

It is well-known that the metatheoretical properties of SOL differ heavily depending on how the second-order quantifiers are interpreted (for instance contrasted by Väänänen [38]). In the rather natural *full semantics*, the induction principle Ind_2 ranges over the full relation space over natural numbers – admitting just a single model of PA_2 and consequently ruling out completeness for instance by Gödel's first incompleteness theorem [12]. On the other hand, in the more versatile *Henkin semantics* generalising to variable relation spaces, Ind_2 behaves similarly like the scheme Ind_φ – allowing enough models to maintain completeness as in the case of FOL [15].

In this paper, we investigate these and related metatheoretical properties of SOL using the Coq proof assistant [37] and driven by computational aspects. Using a proof assistant to formally develop the metatheory of SOL is beneficial since the extension over FOL considerably increases the complexity of notions like substitution, that are typically kept informal on paper and make metatheoretic proofs prone to error. For instance, the folklore translation of SOL to FOL (straightforward targetting multi-sorted FOL but intricate for the more common mono-sorted case) is only sketched in Van Dalen's textbook [39] and in fact found inadequate by Nour and Raffalli [29]. We mechanise Nour and Raffalli's proposed correction and derive transportation theorems of the form "if FOL has property P, then so does SOL", orthogonalising from the proofs that FOL satisfies P itself.

The particular choice of Coq is well-suited to express the computational aspects involved. Implementing a constructive logic, Coq's type theory allows for a synthetic treatment of computability [1, 32], based on the fact that all functions definable in axiom-free Coq are computable. Therefore sparing tedious encodings in a concrete model of computation such as Turing machines, the mechanisation of positive properties like enumerability of deduction systems and negative properties like undecidability of validity becomes feasible. In previous work, this approach has already been employed for synthetic undecidability [8, 21] and synthetic incompleteness [20] of FOL, and with this paper, we illustrate that the same methods scale to the case of SOL.

To the best of our knowledge, we provide the first mechanisation[1] of general SOL, including the following main results:

- Soundness and enumerability of the standard natural deduction system with full comprehension.
- Categoricity of second-order Peano arithmetic PA_2.
- Undecidability and incompleteness by synthetic many-one reduction from Diophantine equations, contributed to the Coq Library of Undecidability Proofs [10].
- Translation of SOL in Henkin semantics to mono-sorted FOL, transporting completeness, compactness, and Löwenheim-Skolem theorems from FOL to SOL.

These main results are covered in Sections 3 through 8, respectively, where the translation from SOL to FOL is organised in the latter three sections. The technical material is enclosed by preliminaries on constructive type theory and synthetic computability (Section 2) as well as a discussion of the Coq development and related and future work (Section 9).

2 Preliminaries

2.1 Constructive Type Theory

We work in the calculus of inductive constructions [4, 30] as implemented in the Coq proof assistant [37]. It features a cumulative hierarchy of predicative universes $\mathbb{T}_i$ (we omit the universe level $i : \mathbb{N}$ throughout the paper) and an impredicative universe $\mathbb{P} \subseteq \mathbb{T}$ of propositions. On the level of $\mathbb{T}$ we have function spaces $X \to Y$, products $X \times Y$, sums $X + Y$, dependent products $\forall x : X.\, F\,x$, and dependent sums $\Sigma x : X.\, F\,x$. The corresponding propositional versions are denoted by the usual logical notation, i.e. $\to$, $\land$, $\lor$, $\forall$, and $\exists$, along with $\top$ and $\bot$ denoting truth and falsity. Elimination of $\lor$ and $\exists$ into $\mathbb{T}$ is prohibited, whereas sums and dependent sums can be eliminated.

We use inductive types for Booleans ($\mathbb{B} ::= \mathsf{true} \mid \mathsf{false}$), Peano natural numbers ($\mathbb{N} ::= 0 \mid n{+}1$), option types ($\mathcal{O}(X) ::= {}^\circ x \mid \emptyset$), and lists ($\mathcal{L}(X) ::= \mathsf{nil} \mid x :: A$). We write $A \mathbin{+\!\!+} B$ for concatenation, $x \in A$ for membership, $|A|$ for length and $A \subseteq B$ for inclusion. We overload the function application notation such that $f\,A$ denotes the point-wise application of a function $f : X \to Y$ to a list $A : \mathcal{L}(X)$. We use the type X^n constructed by $\mathsf{nil} : X^0$ and $(x :: \boldsymbol{v}) : X^{n+1}$ for $\boldsymbol{v} : X^n$ to denote vectors $\boldsymbol{v}$ of length n and reuse the definitions and notations introduced for lists.

Furthermore, we make use of a bijective Cantor pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, embedding pairs of natur´al numbers into the natural numbers.

[1]Systematically hyperlinked with the PDF version of this paper and accessible via the project page www.ps.uni-saarland.de/extras/cpp22-sol/.

2.2 Synthetic Computability Theory

The standard notions of computability theory can be synthesised in constructive type theory since every constructively definable function is computable. Instead of an explicit model of computation such as Turing machines or the lambda calculus, the synthetic approach relies on this intrinsic notion of computation [1, 8]:

Definition 2.1 (Decidability). *A predicate $p : X \to \mathbb{P}$ is decidable if there is a Boolean decider $f : X \to \mathbb{B}$ such that $\forall x.\, p\,x \leftrightarrow f\,x = \mathsf{true}$. This notion generalises to predicates with multiple arguments by implicit uncurrying.*

Definition 2.2 (Discreteness). *A type X is called discrete if there is an equality decider for $\lambda xy : X.\, x = y$.*

Similarly, (recursive) enumerability is witnessed by an enumerating function:

Definition 2.3 (Enumerability).

1. *A predicate $p : X \to \mathbb{P}$ is enumerable, if there is an enumerator $f : \mathbb{N} \to \mathcal{O}(X)$ with $\forall x.\, px \leftrightarrow \exists n.\, f\,n = {}^\circ x$. We say that p is co-enumerable if its complement $\overline{p}$, defined as $\overline{p} = \lambda x.\, \neg p\,x$, is enumerable. We call p bi-enumerable if p is both enumerable and co-enumerable. These notions generalise to predicates with multiple arguments by implicit uncurrying.*
2. *A type X is called enumerable, if there is an enumerator $f : \mathbb{N} \to \mathcal{O}(X)$, such that $\forall x.\, \exists n.\, f\,n = {}^\circ x$.*

A standard result from computability theory known as Post's theorem states that bi-enumerable predicates are decidable. Establishing this fact in constructive type theory requires the synthetic version of Markov's principle [1, 8]:

Definition 2.4. *Markov's principle* MP *is the following proposition:* $\forall f : \mathbb{N} \to \mathbb{B}.\, \neg\neg(\exists n.\, f\,n = \mathsf{true}) \to \exists n.\, f\,n = \mathsf{true}$

MP is independent in Coq's type theory [5] and a consequence of the law of excluded middle $\mathsf{LEM} := \forall P : \mathbb{P}.\, P \lor \neg P$. MP is sufficient and necessary to obtain Post's theorem:

Fact 2.5 (Post's Theorem). MP *is equivalent to the decidability of all bi-enumerable predicates on discrete types.*

Proof. Theorem 2.20 in [8]. □

Note that negative notions like undecidability cannot simply be defined as the negation of the positive notions as constructive type theory is consistent with the assumption that every predicate is decidable (in the synthetic sense). Instead, synthetic undecidability can be characterised by chains of reductions from a problem that is known to be undecidable, for example the halting problem on Turing machines.

Definition 2.6 (Many-One Reductions). *Given predicates $p : X \to \mathbb{P}$ and $q : Y \to \mathbb{P}$, a function $f : X \to Y$ is a (many-one) reduction if $\forall x.\, P\,x \leftrightarrow Q\,(f\,x)$. We say p reduces to q and write $p \preccurlyeq q$ if such a function exists.*

Fact 2.7. *If $p \preccurlyeq q$ and q is decidable, then so is p.*

3 Second-Order Syntax, Standard Semantics, and Natural Deduction

In this section, we introduce the syntax, standard Tarski semantics, and natural deduction system for FOL and SOL, following [33].

3.1 Syntax

We parametrise the syntax over a signature specifying the non-logical symbols:

Definition 3.1 (Signature). *A signature $\Sigma = (\Sigma_\mathcal{F}, \Sigma_\mathcal{P})$ consists of a type $\Sigma_\mathcal{F}$ of function symbols and a type $\Sigma_\mathcal{P}$ of predicate symbols, where all symbols come with a specific arity. We write $\mathcal{F} : \Sigma_\mathcal{F}$ and $\mathcal{P} : \Sigma_\mathcal{P}$ for members of those types. The arity of the symbols is denoted by $|\mathcal{F}| : \mathbb{N}$ and $|\mathcal{P}| : \mathbb{N}$. Constants can be represented by nullary function symbols.*

Fixing a signature Σ, we give the syntax of first-order logic featuring individual quantifiers and the usual logical connectives:

Definition 3.2 (First-Order Syntax). *The type of terms $\mathfrak{T}$ and first-order formulas $\mathfrak{F}_1$ is defined by*

$$t : \mathfrak{T} ::= \mathsf{x}_i \mid \mathcal{F}\, \boldsymbol{t}$$

$$\varphi, \psi : \mathfrak{F}_1 ::= \bot \mid \mathcal{P}\, \boldsymbol{t} \mid \varphi \dot{\rightarrow} \psi \mid \varphi \dot{\wedge} \psi \mid \varphi \dot{\vee} \psi \mid \dot{\forall}\varphi \mid \dot{\exists}\varphi$$

where $i : \mathbb{N}$, $\mathcal{F} : \Sigma_\mathcal{F}$, $\mathcal{P} : \Sigma_\mathcal{P}$, and the vectors $\boldsymbol{t}$ are of the expected length $|\mathcal{F}|$ and $|\mathcal{P}|$. We write $\dot{\neg}\varphi$ for $\varphi \dot{\rightarrow} \bot$ and $\varphi \dot{\leftrightarrow} \psi$ for $(\varphi \dot{\rightarrow} \psi) \dot{\wedge} (\psi \dot{\rightarrow} \varphi)$. Furthermore, we use $\dot{\Box}$ as a placeholder for any of the connectives $\dot{\rightarrow}$, $\dot{\wedge}$, and $\dot{\vee}$, and $\dot{\nabla}$ for $\dot{\forall}$ and $\dot{\exists}$. We write $\mathfrak{T}(\Sigma)$ and $\mathfrak{F}_1(\Sigma)$ if we want to emphasise the dependency on a concrete signature Σ.

We employ a de Bruijn representation of variables where a bound variable is encoded as the number of quantifiers shadowing its binder [6]. For example, the formula $\exists a.\, P(a) \wedge \forall b.\, Q(a, b)$ is represented by $\dot{\exists}\, P(\mathsf{x}_0) \dot{\wedge} \dot{\forall}\, Q(\mathsf{x}_1, \mathsf{x}_0)$. For further explanation of de Bruijn syntax we refer to [35]. For the sake of legibility, we fall back to named binders when giving examples or stating concrete formulas, and keep in mind that the formalisation uses the de Bruijn paradigm behind the scenes.

The terms of SOL are exactly the terms $\mathfrak{T}$ of FOL. The type $\mathfrak{F}_2$ of second-order formulas is obtained by extending $\mathfrak{F}_1$ with two new constructs: Besides the individual quantifiers $\dot{\forall}$ and $\dot{\exists}$ we add n-ary predicate quantifiers $\dot{\forall}^n_\mathsf{p}$ and $\dot{\exists}^n_\mathsf{p}$, along with n-ary predicate variables p^n_i for all $n : \mathbb{N}$:

Definition 3.3 (Second-Order Syntax). *The type $\mathfrak{F}_2$ of second-order formulas is defined by extending Definition 3.2 with*

$$\varphi, \psi : \mathfrak{F}_2 ::= \ldots \mid \mathsf{p}^n_i\, \boldsymbol{t} \mid \dot{\forall}^n_\mathsf{p}\, \varphi \mid \dot{\exists}^n_\mathsf{p}\, \varphi \qquad (i, n : \mathbb{N}, \boldsymbol{t} : \mathfrak{T}^n)$$

The type $\mathfrak{P}_n$ denotes all n-ary predicates, that is variables p^n_i and predicate symbols $\mathcal{P} : \Sigma_\mathsf{p}$ with $|\mathcal{P}| = n$. The $|\cdot|$ notation is also extended to predicate variables.

Note, that quantifiers are annotated with the arity of the predicate they quantify over. Otherwise, unintuitive formulas like $\forall P.\, P(x) \rightarrow P(a, b)$ where P is unary and binary at the same time would need to be ruled out by further well-formedness constraints. In our approach, the quantifier instead specifies the arity of P. For example, if P is unary then the binary occurrence would be considered free. This of course requires predicate variables to count de Bruijn indices independently of individual variables and of predicate variables with other arities. For example, the formula $\forall a.\, \exists P.\, P(a) \rightarrow \exists Q.\, \forall b.\, \exists R.\, P(a) \wedge Q$ is represented by $\dot{\forall}\, \dot{\exists}^1_\mathsf{p}\, \mathsf{p}^1_0(\mathsf{x}_0) \dot{\rightarrow} \dot{\exists}^0_\mathsf{p}\, \dot{\forall}\, \dot{\exists}^1_\mathsf{p}\, \mathsf{p}^1_1(\mathsf{x}_1) \wedge \mathsf{p}^0_0$.

When writing concrete formulas on paper, we usually drop the subscript p annotation of second-order quantifiers and distinguish the different kinds of variables by names, using lower case letters for individuals and upper case letters for predicates. We also leave out the aritiy annotations if arities can be derived from the context.

Besides predicate quantifiers, some authors also consider function quantifiers in SOL (for example [33, 42]). We discuss this in Appendix A.

Definition 3.4 (Substitutions). *Capture avoiding individual instantiation $\varphi[\sigma]$ with a parallel substitution $\sigma : \mathbb{N} \rightarrow \mathfrak{T}$ is defined by*

$$\begin{aligned} \mathsf{x}_i[\sigma] &:= \sigma\, i & \bot[\sigma] &:= \bot \\ (\mathcal{F}\, \boldsymbol{t})[\sigma] &:= \mathcal{F}\, (\boldsymbol{t}[\sigma]) & (\varphi \dot{\Box} \psi)[\sigma] &:= \varphi[\sigma] \dot{\Box} \psi[\sigma] \\ (\mathcal{P}\, \boldsymbol{t})[\sigma] &:= \mathcal{P}\, (\boldsymbol{t}[\sigma]) & (\dot{\nabla}\, \varphi)[\sigma] &:= \dot{\nabla}\, \varphi[\mathsf{x}_0 \cdot {\uparrow}\sigma] \\ (\mathsf{p}^n_i\, \boldsymbol{t})[\sigma] &:= \mathsf{p}^n_i\, (\boldsymbol{t}[\sigma]) & (\dot{\nabla}^n_p\, \varphi)[\sigma] &:= \dot{\nabla}^n_p\, \varphi[\sigma] \end{aligned}$$

where $\mathsf{x}_0 \cdot {\uparrow}\sigma$ maps 0 to x_0 and $i + 1$ to $(\sigma\, i)[\lambda i.\, \mathsf{x}_{i+1}]$. We in general write $\varphi[{\uparrow}]$ for the shift substitution $\varphi[\lambda i.\, \mathsf{x}_{i+1}]$ and $\varphi[t]$ as a shorthand for $\varphi[t \cdot \lambda i.\, \mathsf{x}_i]$, instantiating the index 0 with t and reducing the other indices by one.

Predicate instantiation $\varphi[\sigma]^n_\mathsf{p}$ with a parallel substitution $\sigma : \mathbb{N} \rightarrow \mathfrak{P}_n$ for arity n is defined in a similar way. In particular:

$$(\mathsf{p}^m_i\, \boldsymbol{t})[\sigma]^n_\mathsf{p} := \begin{cases} (\sigma\, i)\, \boldsymbol{t} & \text{if } m = n \\ \mathsf{p}^m_i\, \boldsymbol{t} & \text{otherwise} \end{cases}$$

$$(\dot{\nabla}^m_p\, \varphi)[\sigma]^n_\mathsf{p} := \begin{cases} \dot{\nabla}^m_p\, \varphi[\mathsf{p}^n_0 \cdot {\uparrow}\sigma]^n_\mathsf{p} & \text{if } m = n \\ \dot{\nabla}^m_p\, \varphi & \text{otherwise} \end{cases}$$

Note that only variables of matching arities are substituted. As with individual substitution, we use the shorthand $\varphi[P]^n_\mathsf{p}$ to denote the substitution $\varphi[P \cdot \lambda i.\, \mathsf{p}^n_i]^n_\mathsf{p}$.

Finally, we observe that discreteness and enumerability transport from the signature to the syntax.

Fact 3.5 (Discreteness and Enumerability).

1. *If Σ is discrete, then so are $\mathfrak{F}_1$ and $\mathfrak{F}_2$.*
2. *If Σ is enumerable, then so are $\mathfrak{F}_1$ and $\mathfrak{F}_2$.*

3.2 Standard Semantics

Next, we define the standard Tarski semantics for SOL, determining the validity of formulas:

Definition 3.6 (Standard Semantics)**.** *A model $\mathcal{M}$ consists of a domain $D : \mathbb{T}$ and an interpretation $\mathcal{I}$ for function and predicate symbols given by $\mathcal{F}^{\mathcal{I}} : D^{|\mathcal{F}|} \to D$ and $\mathcal{P}^{\mathcal{I}} : D^{|\mathcal{P}|} \to \mathbb{P}$ for each $\mathcal{F}$ and $\mathcal{P}$. Environments $\rho = (\rho_{\mathrm{i}}, \rho_{\mathrm{p}})$ consist of assignments $\rho_{\mathrm{i}} : \mathbb{N} \to D$ and $\rho_{\mathrm{p}}^n : \mathbb{N} \to D^n \to \mathbb{P}$ for individual and predicate variables of each arity $n : \mathbb{N}$, respectively. Term evaluation $[\![\cdot]\!]_\rho : \mathfrak{T} \to D$ and formula satisfaction $\rho \vDash \cdot : \mathfrak{F}_2 \to \mathbb{P}$ are defined by*

$$\begin{aligned}
[\![\mathrm{x}_i]\!]_\rho &:= \rho_{\mathrm{i}}\, i & \rho \vDash \dot\bot &:= \bot \\
[\![\mathcal{F}\, \boldsymbol{t}]\!]_\rho &:= \mathcal{F}^{\mathcal{I}}\ [\![\boldsymbol{t}]\!]_\rho & \rho \vDash \varphi \mathbin{\dot\Box} \psi &:= (\rho \vDash \varphi) \mathbin{\Box} (\rho \vDash \psi) \\
\rho \vDash \mathcal{P}\, \boldsymbol{t} &:= \mathcal{P}^{\mathcal{I}}\ [\![\boldsymbol{t}]\!]_\rho & \rho \vDash \dot\nabla\, \varphi &:= \nabla d : D.\, d \cdot \rho \vDash \varphi \\
\rho \vDash \mathrm{p}_i^n\, \boldsymbol{t} &:= \rho_{\mathrm{p}}^n\, i\, [\![\boldsymbol{t}]\!]_\rho & \rho \vDash \dot\nabla_p^n\, \varphi &:= \nabla P : D^n \to \mathbb{P}.\, P \cdot \rho \vDash \varphi
\end{aligned}$$

where $\Box$ and ∇ are the meta-logical counterparts of the logical symbols $\dot\Box$ and $\dot\nabla$, respectively. The modified environment $d \cdot \rho_{\mathrm{i}}$ maps 0 to d and $x + 1$ to $\rho_{\mathrm{i}}\, x$. The predicate quantifier only extends the predicates environment for the corresponding arity:

$$(P \cdot \rho_{\mathrm{p}}^{|P|})\, 0 := P \qquad (P \cdot \rho_{\mathrm{p}}^{|P|})\, (i+1) := \rho_{\mathrm{p}}^{|P|}\, i$$
$$(P \cdot \rho_{\mathrm{p}}^{n})\, i := \rho_{\mathrm{p}}^{n}\, i \qquad \text{if } n \neq |P|$$

We write $\mathcal{M}, \rho \vDash \varphi$ and $[\![t]\!]_\rho^{\mathcal{M}}$ when we want to make the used model explicit. We also extend satisfiability to theories $\mathcal{T} : \mathfrak{F}_2 \to \mathbb{P}$ and write $\mathcal{M}, \rho \vDash \mathcal{T}$ if $\mathcal{M}, \rho \vDash \varphi$ for all $\varphi \in \mathcal{T}$. Furthermore, we write $\mathcal{M} \vDash \varphi$ if $\mathcal{M}, \rho \vDash \varphi$ for all ρ. Similarly, we say that $\mathcal{M}$ is a model of $\mathcal{T}$, written $\mathcal{M} \vDash \mathcal{T}$, if $\mathcal{M}, \rho \vDash \mathcal{T}$ for all ρ. Finally, we say that φ is valid under $\mathcal{T}$, written $\mathcal{T} \vDash \varphi$, if $\mathcal{M}, \rho \vDash \varphi$ for all $\mathcal{M}$ and ρ with $\mathcal{M}, \rho \vDash \mathcal{T}$.

3.3 Natural Deduction

We represent the natural deduction (ND) system as an inductive predicate of type $\mathcal{L}(\mathfrak{F}_2) \to \mathfrak{F}_2 \to \mathbb{P}$. We use two versions of the deduction system denoted by $\vdash_c$ and $\vdash_i$ distinguishing between classical and intuitionistic provability. The following (Peirce) axiom is only available in the classical system $\vdash_c$ and may not be used in $\vdash_i$:

$$\textsc{Peirce}\ \frac{}{\Gamma \vdash_c ((\varphi \mathbin{\dot\to} \psi) \mathbin{\dot\to} \varphi) \mathbin{\dot\to} \varphi}$$

It allows to enforce classicality without relying on $\dot\bot$. We write $\vdash$ if the choice between $\vdash_c$ and $\vdash_i$ does not matter. The remaining rules characterising the two ND systems are listed in Appendix B. Since most of them are standard, we only highlight the rules specific to SOL:

$$\textsc{AllI}_{\mathrm{p}}\ \frac{\Gamma[\uparrow]_{\mathrm{p}}^n \vdash \varphi}{\Gamma \vdash \dot\forall_{\mathrm{p}}^n\, \varphi} \qquad \textsc{AllE}_{\mathrm{p}}\ \frac{\Gamma \vdash \dot\forall_{\mathrm{p}}^n\, \varphi}{\Gamma \vdash \varphi[P]_{\mathrm{p}}^n} \qquad \textsc{ExI}_{\mathrm{p}}\ \frac{\Gamma \vdash \varphi[P]_{\mathrm{p}}^n}{\Gamma \vdash \dot\exists_{\mathrm{p}}^n\, \varphi}$$

$$\textsc{ExE}_{\mathrm{p}}\ \frac{\Gamma \vdash \dot\exists_{\mathrm{p}}^n\, \psi \qquad \Gamma[\uparrow]_{\mathrm{p}}^n, \psi \vdash_2 \varphi}{\Gamma \vdash \varphi}$$

$$\textsc{Compr}\ \frac{P \text{ not free in } \varphi}{\Gamma \vdash \dot\exists P.\, \dot\forall x_1 \ldots x_n.\, P(x_1, \ldots, x_n) \mathbin{\dot\leftrightarrow} \varphi}$$

The quantifier rules are a trivial extension of their first-order counterparts. Note that the n-ary predicate variable 0 becomes free in the shifted context $\Gamma[\uparrow]_{\mathrm{p}}^n$, thus taking the role as an arbitrary but fixed predicate. Hence, we can conclude $\Gamma \vdash \dot\forall_{\mathrm{p}}^n\, \varphi$ from $\Gamma[\uparrow]_{\mathrm{p}}^n \vdash \varphi$ as expressed by ($\textsc{AllI}_{\mathrm{p}}$). The ($\textsc{ExE}_{\mathrm{p}}$) makes similar use of shifts. This approach allows for an easier mechanisation of abstract properties like weakening. However, the more traditional approach of replacing the index 0 with a fresh variable can also be shown admissible (see Lemma 3.9).

Finally, $\vdash$ also includes the so called comprehension axiom (Compr). It states that for each formula φ there is a predicate P that extensionally behaves the same as the n-ary property expressed by φ. Importantly, P may not occur freely in φ, otherwise the axiom would be inconsistent. We write Compr_φ^n for a concrete instance of the comprehension scheme for the formula φ and arity n.

While the context of $\vdash$ is represented by finite lists Γ, the notion of provability naturally extends to potentially infinite theories $\mathcal{T} : \mathfrak{F}_2 \to \mathbb{P}$:

Definition 3.7 (Deduction under Theories)**.** *A formula φ is provable under a theory $\mathcal{T} : \mathfrak{F}_2 \to \mathbb{P}$, written $\mathcal{T} \vdash \varphi$, if there is a list $\Gamma \subseteq \mathcal{T}$, such that $\Gamma \vdash \varphi$. We also extend the classicality annotations $\vdash_c$ and $\vdash_i$ to provability under theories.*

The usual weakening properties allow for reformulations of the ($\textsc{AllI}_{\mathrm{p}}$) and ($\textsc{ExE}_{\mathrm{p}}$) rules that are helpful when deriving concrete statements:

Lemma 3.8 (Weakening)**.** *The following rules hold for $\vdash$:*

$$\textit{Weak}\ \frac{\Gamma' \vdash \varphi \qquad \Gamma' \subseteq \Gamma}{\Gamma \vdash \varphi} \qquad \textit{Subst}\ \frac{\Gamma \vdash \varphi}{\Gamma[\sigma] \vdash \varphi[\sigma]}$$

$$\textit{Subst}_{\mathrm{p}}\ \frac{\Gamma \vdash \varphi}{\Gamma[\sigma]_{\mathrm{p}}^n \vdash \varphi[\sigma]_{\mathrm{p}}^n}$$

Lemma 3.9 (Named Quantifier Rules)**.** *The following rules hold for $\vdash$:*

$$\textit{AllI}'_{\mathrm{p}}\ \frac{\Gamma \vdash \varphi[\mathrm{p}_i]_{\mathrm{p}}^n \qquad \mathrm{p}_i^n \notin \Gamma, \dot\forall_{\mathrm{p}}^n \varphi}{\Gamma \vdash \dot\forall_{\mathrm{p}}^n \varphi}$$

$$\textit{ExE}'_{\mathrm{p}}\ \frac{\Gamma \vdash \dot\exists_{\mathrm{p}}^n \psi \qquad \Gamma, \varphi[\mathrm{p}_i]_{\mathrm{p}}^n \vdash \psi \qquad \mathrm{p}_i^n \notin \Gamma, \dot\forall_{\mathrm{p}}^n \varphi}{\Gamma \vdash \varphi}$$

where $\mathrm{p}_i^n \notin \Gamma$ denotes that the variable does not occur in Γ. Similar rules ($\textit{AllI}'$) and ($\textit{ExE}'$) also hold for the first-order quantifiers (cf. Lemma 4 in [8]).

Finally, we establish enumerability and soundness of $\vdash$:

Fact 3.10. *Let Σ be discrete and enumerable. Then $\lambda\varphi.\, \Gamma \vdash \varphi$ is enumerable for all Γ.*

Proof. Via the techniques discussed in [8]. □

Theorem 3.11 (Soundness)**.** *Let $\mathcal{T}$ and φ be given, then:*

1. *$\mathcal{T} \vdash_i \varphi \to \mathcal{T} \vDash \varphi$*
2. *$\mathcal{T} \vdash_c \varphi \to \mathcal{T} \vDash \varphi$ iff* LEM *holds*

Proof. We discuss both cases:

1. It suffices to show soundness for contexts $\Gamma \subseteq \mathcal{T}$ which follows by induction on the derivation.

2. Similar to (2) requiring LEM to verify the (Peirce) axiom since we work in constructive logic. Classical soundness in turn also entails LEM since $\vdash_c P \dot{\vee} \dot{\neg} P$ for all nullary predicate variables P. □

4 Second-Order Peano Arithmetic

An important property where SOL differs from FOL is the notion of categoricity [31]. A theory $\mathcal{T}$ is called *categorical*, if all models of $\mathcal{T}$ are equal up to isomorphism. Thus, $\mathcal{T}$ is strong enough to uniquely characterise its own intended structure.

FOL is not able to characterise any infinite structures in a categorical way as a consequence of the Löwenheim-Skolem theorem. SOL on the other hand allows for categorical axiomatisations of many infinite structures. We demonstrate this in this section using the example of second-order Peano arithmetic (PA_2), characterising the natural numbers.

The signature Σ_{PA} of PA_2 contains symbols for the constant zero, the successor, addition and multiplication functions, as well as an equality predicate:

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

The addition of an equality symbol might seem puzzling at first glance, since equality is actually second-order definable via the Leibniz characterisation $x = y \sim \dot{\forall} P.\, P(x) \dot{\to} P(y)$. We settled on the more uncommon symbol approach to stay in closer correspondence to first-order Peano arithmetic and allow for easier arguments in the FOL-fragment later on.

PA_2 is finite and contains the following formulas:

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

$$\mathsf{Ind}_2 : \dot{\forall} P.\, P(O) \dot{\to} (\dot{\forall} x.\, P(x) \dot{\to} P(Sx)) \dot{\to} \dot{\forall} x.\, P(x)$$

Instead of $\equiv$-refl and $\equiv$-sym, one could also axiomatise $\equiv$ via the Leibniz characterisation mentioned above. But interestingly, the first-order properties reflexivity and symmetry already suffice, with neither transitivity, nor congruence axioms for S, $\oplus$, and $\otimes$ being necessary, because of the strong Ind_2 axiom:

Fact 4.1 (Equality). *Let $\mathcal{M}$ be a model of PA_2 with domain D and interpretation $\mathcal{I}$. Then $x \equiv^{\mathcal{I}} y$ iff $x = y$ for all $x, y : D$.*

Proof. By induction on x and case analysis on y using the Ind_2 axiom. The different cases follow with O-succ, S-inj, $\equiv$-refl and $\equiv$-sym. □

Note that this fact immediately implies the usual transitivity and congruence axioms.

In first-order Peano arithmetic (PA_1) the induction axiom is usually replaced by an axiom scheme:

$$\mathsf{Ind}_\varphi : \varphi[O] \dot{\to} (\dot{\forall} x.\, \varphi[x] \dot{\to} \varphi[Sx]) \dot{\to} \dot{\forall} x.\, \varphi[x]$$

Thus, PA_1 only allows to induce on first-order definable properties, which is of course weaker than the induction principle obtained in PA_2. For example, Fact 4.1 cannot be established for PA_1.[2]

The natural choice for the standard model of PA_2 is the type $\mathbb{N}$ of natural numbers:

Fact 4.2 (Standard Model). *The standard model of PA_2 consists of the domain $\mathbb{N}$ and interprets the signature with the usual operations on $\mathbb{N}$. We also denote this model by $\mathbb{N}$.*

4.1 Categoricity

The categoricity of PA_2 is originally due to Dedekind [7], but we follow the more contemporary proof by Shapiro [33]. We fix two models $\mathcal{M}_1$ and $\mathcal{M}_2$ of PA_2 and write D_1 and D_2 for the domains, as well as $\mathcal{I}_1$ and $\mathcal{I}_2$ for the interpretations.

Next, we construct an isomorphism between $\mathcal{M}_1$ and $\mathcal{M}_2$. There is not much hope for being able to define this isomorphism as a computable function $D_1 \to D_2$. The only means of determining the structure of $x : D_1$ is via the induction axiom which is inherently propositional and cannot be used to compute a corresponding $y : D_2$. Thus, we need to give the isomorphism as a relation $D_1 \to D_2 \to \mathbb{P}$:

Definition 4.3. *We define the relation $\cong : D_1 \to D_2 \to \mathbb{P}$ inductively by $O^{\mathcal{I}_1} \cong O^{\mathcal{I}_2}$ and $S^{\mathcal{I}_1}\, x \cong S^{\mathcal{I}_2}\, y$ if $x \cong y$. We also extend $\cong$ to vectors, functions, predicates and environments in the pointwise way.*

Fact 4.4. *$\cong$ is an isomorphism, i.e. it is total, surjective, functional, injective, and a congruence relation. Totality and surjectivity also extended to vectors and predicates.*

Proof. All properties follow by induction using the Ind_2 axiom. The congruence properties use the axioms for $\oplus$ and $\otimes$, as well as Fact 4.1. □

This shows that PA_2 is indeed categorical. As a direct consequence, satisfiability of formulas agrees in both models:

Lemma 4.5 (Agreement). *Let ρ_1 and ρ_2 be environments with $\rho_1 \cong \rho_2$. Then $\mathcal{M}_1, \rho_1 \vDash \varphi$ iff $\mathcal{M}_2, \rho_2 \vDash \varphi$ for all $\varphi : \mathfrak{F}_2$.*

Proof. By induction on φ with ρ_1 and ρ_2 generalised. In the case of atomic formulas, we verify $[\![t]\!]_{\rho_1}^{\mathcal{M}_1} \cong [\![t]\!]_{\rho_2}^{\mathcal{M}_2}$ for all $t : \mathfrak{T}$ by induction using the congruence properties of $\cong$. For first-order quantifiers, we use totality and surjectivity to obtain $d_2 : D_2$ from $d_1 : D_1$ with $d_1 \cong d_2$ and vice versa, and use the inductive hypothesis with $d_1 \cdot \rho_1 \cong d_2 \cdot \rho_2$. Second-order quantifiers are handled in the same way. □

Finally, we remark that $\cong$ becomes computational if a recursion principle is available for the domains:

[2]Consider the intensional model $\mathbb{N} \times \mathbb{B}$ with $(n_1, \_) \equiv^{\mathcal{I}} (n_2, \_) := n_1 = n_2$, which satisfies the axioms of PA_1 but violates Ind_2.

Fact 4.6. *If D_1 satisfies the computational recursion principle*

$$\forall P : D_1 \to \mathbb{T}.\, P\, O^{\mathcal{I}_1} \to (\forall x.\, P\, x \to P\, (S^{\mathcal{I}_1}\, x)) \to \forall x.\, P\, x$$

we have $\forall x.\, \Sigma y.\, x \cong y$. The same holds symmetrically for D_2.

4.2 Consequences of Categoricity

Since all models of PA_2 are isomorphic to the standard model $\mathbb{N}$, the upward Löwenheim-Skolem theorem fails for SOL:

Fact 4.7 (Failure of Löwenheim-Skolem)**.** *There is a countable second-order theory with only countably infinite models.*

Proof. PA_2 is finite and thus countable. Let $\mathcal{M}$ be a model of PA_2 with domain D. D is in bijection with $\mathbb{N}$ via $\cong$ (Fact 4.4) and therefore countably infinite. □

Similarly, the downwards Löwenheim-Skolem theorem can be refuted by showing the categoricity of second-order real analysis [33] or set theory[3] [44]. Thus, SOL is expressive enough to distinguish different infinite cardinalities. Furthermore, the categoricity result can be used to refute yet another of the main properties of FOL:

Fact 4.8 (Failure of Compactness)**.** *There is a second-order theory whose finite subsets all have a model, but the theory itself does not.*

Proof. Consider the infinite theory

$$\mathcal{T}_{\neq} := \mathsf{PA}_2, \mathsf{x}_0 \neq O, \mathsf{x}_0 \neq S\, O, \mathsf{x}_0 \neq S\, (S\, O), \ldots$$

Every finite subset $\Gamma \subseteq \mathcal{T}_{\neq}$ has a model. Take for example the standard model $\mathbb{N}$ and choose a number for x_0 that is not ruled out by the constraints in Γ. However, suppose that $\mathcal{T}_{\neq}$ itself also has a model. By categoricity this model would be isomorphic to the standard model $\mathbb{N}$. But $\mathbb{N}$ cannot be a model of $\mathcal{T}_{\neq}$ since there is no numeral to assign to x_0. Thus, $\mathcal{T}_{\neq}$ cannot have a model. □

We can use the proof above to show another major point of disagreement between FOL and SOL. There is no sound deduction system $\vdash\, : \mathcal{L}(\mathfrak{F}_2) \to \mathfrak{F}_2 \to \mathbb{P}$ that is strongly complete when lifted to theories as described in Definition 3.7. The argument is surprisingly simple (see for example [36]):

Theorem 4.9 (Failure of Strong Completeness)**.** *Every sound second-order deduction system $\vdash\, : \mathcal{L}(\mathfrak{F}_2) \to \mathfrak{F}_2 \to \mathbb{P}$ is not strongly complete (i.e. $\mathcal{T} \vDash \varphi$ does not imply $\mathcal{T} \vdash \varphi$ in general). In fact, strong completeness already fails for decidable $\mathcal{T}$.*

Proof. Let $\vdash$ be sound and strongly complete. In the proof of Fact 4.8 we showed that $\mathcal{T}_{\neq}$ does not have a model. Thus, $\mathcal{T}_{\neq} \vDash \bot$ and by strong completeness $\mathcal{T}_{\neq} \vdash \bot$. Hence, there is a finite context $\Gamma \subseteq \mathcal{T}_{\neq}$ with $\Gamma \vdash \bot$. This contradicts the fact that every finite $\Gamma \subseteq \mathcal{T}_{\neq}$ has a model (cf. proof of Fact 4.8). Moreover, observe that $\mathcal{T}_{\neq}$ is decidable. Therefore, strong completeness for decidable theories already suffices to arrive at the contradiction, emphasising that we do not exploit the computational power of the theories. □

[3]Second-order ZF is categorical for equipotent models.

Remarkably, no computability properties like enumerability of the proof system are required to rule out strong completeness. The only assumption we have is that the system is sound and that it is naturally lifted to theories.

Corollary 4.10 (Incompleteness of $\vdash$)**.**

1. $\vdash_i$ *is not strongly complete.*
2. $\vdash_c$ *is not strongly complete under* LEM.

Proof. Follows by Theorem 4.9 since $\vdash$ is sound. Recall that classical soundness requires LEM. □

However, the proof above does not rule out (weak) completeness. For example, the "system" $\Gamma \vdash \varphi := \Gamma \vDash \varphi$ is sound and weakly complete, but not strongly complete by Theorem 4.9. It suffices to require enumerability of deduction systems in order to rule out the one above and show that no complete system can exist. Establishing this stronger incompleteness result requires a more intricate argument involving computability and Gödel's first incompleteness theorem, which is investigated in the next section.

5 Undecidability and Incompleteness

In this section we derive the undecidability of PA_2 and SOL by reduction from Diophantine equations. By the same reduction, we also establish the incompleteness of SOL in a more general way than already recorded in Theorem 4.9.

5.1 Reduction from Diophantine Equations

Hilbert's tenth problem (H_{10}) is about deciding whether Diophantine equations $s = t$ (where s and t are polynomials with natural coefficients) have a solution in the natural numbers. This problem is undecidable [28] which was mechanised by reduction from the halting problem for Turing machines as part of the Coq Library of Undecidability Proofs [10, 26].

Definition 5.1 (Diophantine Equations)**.** *Polynomials consist of constants, variables, addition and multiplication:*

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

Evaluation $\langle\!\langle s \rangle\!\rangle_\alpha$ of polynomials under a variable assignment $\alpha : \mathbb{N} \to \mathbb{N}$ is defined by

$$\langle\!\langle \mathsf{num}\ n \rangle\!\rangle_\alpha := n \qquad \langle\!\langle \mathsf{add}\ s\ t \rangle\!\rangle_\alpha := \langle\!\langle s \rangle\!\rangle_\alpha + \langle\!\langle t \rangle\!\rangle_\alpha$$
$$\langle\!\langle \mathsf{var}\ x \rangle\!\rangle_\alpha := \alpha\ x \qquad \langle\!\langle \mathsf{mul}\ s\ t \rangle\!\rangle_\alpha := \langle\!\langle s \rangle\!\rangle_\alpha \cdot \langle\!\langle t \rangle\!\rangle_\alpha$$

A Diophantine equation $s = t$ has a solution, written $\mathsf{H}_{10}(s, t)$, if there is an assignment α such that $\langle\!\langle s \rangle\!\rangle_\alpha = \langle\!\langle t \rangle\!\rangle_\alpha$.

H_{10} is well suited to reduce into PA_2 since the solvability condition can easily be expressed as an arithmetic formula. We start with encoding polynomials as terms by defining a translation function η, following Kirst and Hermes [20]:

Definition 5.2. *We define $\eta : \mathsf{poly} \to \mathfrak{T}(\Sigma_{\mathsf{PA}})$ by*

$$\eta\ (\mathsf{num}\ n) := \overline{n} \qquad \eta\ (\mathsf{add}\ s\ t) := \eta\ s \oplus \eta\ t$$
$$\eta\ (\mathsf{var}\ i) := \mathsf{x}_i \qquad \eta\ (\mathsf{mul}\ s\ t) := \eta\ s \otimes \eta\ t$$

with $\overline{0} = O$ *and* $\overline{n+1} = S\,\overline{n}$.

A Diophantine equation can now be encoded by binding all variables with existential quantifiers:

Definition 5.3. *We define* $\varphi_{s,t} := \overline{\exists}\ \eta\, s \equiv \eta\, t$ *where* $\overline{\exists}$ *adds existential quantifiers to the formula until all variables are bound, thus closing the formula.*

The following fact fully characterises the behaviour of the existential closure operation $\overline{\exists}$:

Fact 5.4. *Let* $\mathcal{M}$ *be a model and* φ *a formula whose free variables are all first-order. Then* $\mathcal{M} \vDash \overline{\exists}\varphi$ *iff there is an environment* ρ *with* $\mathcal{M}, \rho \vDash \varphi$.

Proof. We only discuss the general intuition for the proof. Let n be the number of quantifiers required to close φ.

- $\rightarrow$ If $\mathcal{M} \vDash \overline{\exists}\varphi$, then in particular $\mathcal{M}, \rho \vDash \overline{\exists}\varphi$ for some ρ (e.g. $\rho = (\lambda x.\, O^{\mathcal{I}}, \lambda xnv.\, \top)$). Hence, there exist $d_1, ..., d_n$ such that $\mathcal{M}, d_1 \cdot ... \cdot d_n \cdot \rho \vDash \varphi$.
- $\leftarrow$ If there is ρ with $\mathcal{M}, \rho \vDash \varphi$, then the existential quantifiers can be satisfied by $\rho_\mathrm{i}\ 0, ..., \rho_\mathrm{i}\ (n-1)$ and the remaining environment is $\rho' = (\lambda x.\, \rho_\mathrm{i}\ (x+n), \rho_\mathrm{p})$, such that $\mathcal{M}, \rho' \vDash \dot{\exists}^n\varphi$. Since $\overline{\exists}\varphi$ is closed, we can switch to any other environment and have $\mathcal{M} \vDash \overline{\exists}\varphi$. □

H_{10} reduces to validity in the standard model:

Lemma 5.5. *Let* s *be a polynomial,* α *an assignment and* ρ_p *a predicate environment. Then* $[\![\eta\ s]\!]^{\mathbb{N}}_{(\alpha,\rho_\mathrm{p})} = \langle\!\langle s \rangle\!\rangle_\alpha$.

Lemma 5.6 (Undecidability of $\mathbb{N}$). $\mathsf{H}_{10}(s,t)$ *iff* $\mathbb{N} \vDash \varphi_{s,t}$.

Proof. We show both directions separately:

- $\rightarrow$ Let α be a solution to $s = t$. By Fact 5.4 it suffices to give an environment ρ with $\mathbb{N}, \rho \vDash \eta\, s \equiv \eta\, t$ which reduces to $[\![\eta\ s]\!]^{\mathbb{N}}_\rho = [\![\eta\ t]\!]^{\mathbb{N}}_\rho$. By Lemma 5.5 this holds for the environment $(\alpha, \rho_\mathrm{p})$ for arbitrary ρ_p.
- $\leftarrow$ If $\mathbb{N} \vDash \varphi_{s,t}$, we have ρ with $\mathbb{N}, \rho \vDash \eta\, s \equiv \eta\, t$ by Fact 5.4. Thus, ρ_i is a solution to $s = t$ by Lemma 5.5. □

Because of categoricity, we can replace $\mathbb{N}$ with any abstract model of PA_2:

Corollary 5.7. *Let* $\mathcal{M} \vDash \mathsf{PA}_2$. *Then* $\mathsf{H}_{10}(s,t)$ *iff* $\mathcal{M} \vDash \varphi_{s,t}$.

Proof. By Lemma 5.6, Lemma 4.5, and the fact that $\varphi_{s,t}$ is closed. □

This yields undecidability of PA_2 as follows:

Corollary 5.8 (Undecidability of PA_2).

1. $\mathsf{H}_{10}(s,t)$ *iff* $\varphi_{s,t}$ *is valid in* PA_2.
2. $\mathsf{H}_{10}(s,t)$ *iff* $\varphi_{s,t}$ *is satisfiable in* PA_2.

More precisely, those problems are also not enumerable:

Corollary 5.9 (Non-Enumerability of PA_2).

1. *Enumerability of validity in* PA_2 *implies co-enumerability of* H_{10} .
2. *Enumerability of satisfiability in* PA_2 *implies co-enumerability of* H_{10}.

Proof. We only discuss (1): Co-enumerability of H_{10} is equivalent to enumerability of $\lambda st.\, \neg(\mathsf{PA}_2 \vDash \varphi_{s,t})$ by Corollary 5.8. By categoricity, this is in turn equivalent to enumerability of $\lambda st.\, \mathsf{PA}_2 \vDash \dot{\neg}\varphi_{s,t}$ which follows from the assumption. □

Since $\overline{\mathsf{H}_{10}} \preccurlyeq \overline{\mathsf{Halt}}$ [26, 28] and the fact that the halting problem is not co-enumerable it follows that validity and satisfiability in PA_2 are not enumerable. Alternatively, one can assume MP to show that enumerablity of those problems would imply decidability of the halting problem via Post's theorem (Fact 2.5) since H_{10} is enumerable.

5.2 Incompleteness

The results from the previous section directly yield incompleteness of SOL, because if SOL were complete (i.e. $\Gamma \vDash \phi$ would imply $\Gamma \vdash \phi$), then we could enumerate all closed formulas that are valid in PA_2:

Lemma 5.10. *A sound, complete, and enumerable second-order deduction system for the signature* Σ_{PA} *enumerates all valid formulas in* PA_2.

Thus, SOL cannot be complete:

Theorem 5.11 (Incompleteness). *The existence of a sound, complete, and enumerable deduction system in* Σ_{PA} *implies bi-enumerability of* H_{10}.

Proof. Immediate by Corollary 5.9 and Lemma 5.10. □

Corollary 5.12 (Incompleteness of $\vdash$).

1. *Completeness of* $\vdash_i$ *implies co-enumerability of* H_{10}.
2. *Completeness of* $\vdash_c$ *implies co-enumerability of* H_{10} *under* LEM.

Proof. Follows from Theorem 5.11 since $\vdash$ is sound and enumerable. Soundness for $\vdash_c$ requires LEM. □

5.3 Extension to Arbitrary Signatures

The previous results can be extended to arbitrary signatures since SOL allows us to encode finite signatures and axiomatisations into formulas themselves. Function quantifiers are very useful for this task as they can directly embed the function symbols from the signature. A formal introduction of the function quantifier fragment $\mathfrak{F}_2^F$ is contained in Appendix A.

Lemma 5.13 (Signature Embedding in $\mathfrak{F}_2^F$). *Let* $\varphi : \mathfrak{F}_2^F(\Sigma_{\mathsf{PA}_2})$ *be a formula and* Σ' *an arbitrary signature. Then there is a formula* $\varphi' : \mathfrak{F}_2^F(\Sigma')$ *and theory* PA_2' *in* Σ' *such that*

1. φ *is valid in* PA_2 *iff* $\vDash \dot{\forall} f_O\, f_S\, f_\oplus\, f_\otimes.\, \dot{\forall} P_\equiv.\, \mathsf{PA}_2' \dot{\rightarrow} \varphi'$,
2. φ *is satisfiable in* PA_2 *iff* $\dot{\exists} f_O\, f_S\, f_\oplus\, f_\otimes.\, \dot{\exists} P_\equiv.\, \mathsf{PA}_2' \dot{\wedge} \varphi'$ *is satisfiable.*

Proof. We obtain φ' and PA_2' by replacing function and predicate symbols with variables referencing the corresponding quantifier at the beginning. We only discuss (1), as (2) is proven in a similar way.

- $\rightarrow$ We have to show $\mathcal{M}', f_O \cdot f_S \cdot f_\oplus \cdot f_\otimes \cdot P_\equiv \cdot \rho \vDash \mathsf{PA}_2' \dot\rightarrow \varphi'$ for all $\mathcal{M}'$, f_O, f_S, $f_\oplus$, $f_\otimes$, $P_\equiv$, and ρ. We construct a model $\mathcal{M}$ of PA_2 by using the domain of $\mathcal{M}'$ and interpreting O using f_O, S using f_S, and so on. We have $\mathcal{M}', f_O \cdot f_S \cdot f_\oplus \cdot f_\otimes \cdot P_\equiv \cdot \rho \vDash \varphi'$ since φ is valid in every model of PA_2.
- $\leftarrow$ We get a model $\mathcal{M}$ of PA_2 and construct a model $\mathcal{M}'$ in Σ' by choosing an arbitrary interpretation. Then, we instantiate f_O, f_S, etc. in the assumption with the symbol interpretations of $\mathcal{M}$, yielding $\mathcal{M}', O^\mathcal{I} \cdot S^\mathcal{I} \cdot \oplus^\mathcal{I} \cdot \otimes^\mathcal{I} \cdot \equiv^\mathcal{I} \cdot \rho \vDash \mathsf{PA}_2' \dot\rightarrow \varphi'$. This suffices, since the interpretations satisfy PA_2. □

Hence, we get undecidability, non-enumerability, and incompleteness of SOL in particular in the empty signature:

Corollary 5.14 (Undecidability of SOL).

1. *$s = t$ has a solution iff $\dot\forall f_O\, f_S\, f_\oplus\, f_\otimes.\, \dot\forall P_\equiv.\, \mathsf{PA}_2' \dot\rightarrow \varphi'_{s,t}$ is valid in the empty signature.*
2. *$s = t$ has a solution iff $\dot\exists f_O\, f_S\, f_\oplus\, f_\otimes.\, \dot\exists P_\equiv.\, \mathsf{PA}_2' \dot\wedge \varphi'_{s,t}$ is satisfiable in the empty signature.*

Corollary 5.15 (Non-Enumerability of SOL).

1. *Enumerability of validity in the empty signature implies bi-enumerability of* H_{10}.
2. *Enumerability of satisfiability in the empty signature implies bi-enumerability of* H_{10}.

Corollary 5.16 (Incompleteness). *The existence of a sound, weakly complete, and enumerable deduction system for $\mathfrak{F}_2^F$ in the empty signature implies co-enumerability of* H_{10}.

6 Henkin Semantics

In 1950, Henkin introduced an alternative semantics for SOL and simple type theory with the aim of recovering completeness [15]. In Henkin semantics, second-order quantifiers no longer range over all predicates of a given domain, but only over some subset $\mathbb{U}$ of them that is provided by the model. This predicate universe $\mathbb{U}$ should at least contain all second-order definable properties, so that the comprehension rule of the ND-system is sound.

Definition 6.1 (Henkin Semantics). *A Henkin model $\mathcal{H}$ consists of a domain $D : \mathbb{T}$, an interpretation function $\mathcal{I}$ for function and predicate symbols and a family of relations $\mathbb{U}_n : (D^n \to \mathbb{P}) \to \mathbb{P}$ for each $n : \mathbb{N}$. The predicate interpretations should be included in $\mathbb{U}$, in other words $\mathbb{U}_n\, \mathcal{P}_n^\mathcal{I}$ for all n-ary predicate symbols $\mathcal{P}_n : \Sigma_\mathcal{P}$.*

The model must satisfy $\mathcal{H} \vDash_H \mathsf{Compr}_\varphi^n$ for all φ and n, where $\vDash_H$ is defined by

$$\rho \vDash_H \dot\forall_\mathrm{p}^n\, \varphi \;:=\; \forall P : D^n \to \mathbb{P}.\, \mathbb{U}_n\, P \to P \cdot \rho \vDash_H \varphi$$
$$\rho \vDash_H \dot\exists_\mathrm{p}^n\, \varphi \;:=\; \exists P : D^n \to \mathbb{P}.\, \mathbb{U}_n\, P \wedge P \cdot \rho \vDash_H \varphi$$

The remaining cases are defined in the same way as for standard semantics. We say a formula φ is valid in $\mathcal{H}$, written $\mathcal{H} \vDash_H \varphi$, if $\mathcal{H}, \rho \vDash_H \varphi$ for all environments ρ with $\mathbb{U}_n\, (\rho_\mathrm{p}^n\, x)$ for all $x, n : \mathbb{N}$. Those environments are called Henkin environments.

In the remainder of this paper we will mostly work with Henkin semantics. Therefore, we write $\vDash$ instead of $\vDash_H$ if it is clear from the context that we are in the Henkin setting.

Henkin semantics agree with the standard semantics if $\mathbb{U}_n\, P$ for all $P : D^n \to \mathbb{P}$:[4]

Fact 6.2. *Let $\mathcal{M}$ be a model in standard semantics. Then we can also interpret M as a Henkin model with $\mathbb{U}_n\, P := \top$. It holds that $\mathcal{M}, \rho \vDash \varphi$ iff $\mathcal{M}, \rho \vDash_H \varphi$ for all ρ and φ.*

Therefore, we have $\mathcal{T} \vDash_H \varphi \to \mathcal{T} \vDash \varphi$. The converse does not hold, since there are many more Henkin models than there are models for standard semantics.

Finally, we remark that the ND-system is also sound for Henkin semantics:

Theorem 6.3 (Soundness for Henkin Semantics). *Let T be a theory and φ a formula. Then:*

1. $\mathcal{T} \vdash_i \varphi \to \mathcal{T} \vDash_H \varphi$
2. $\mathcal{T} \vdash_c \varphi \to \mathcal{T} \vDash_H \varphi$ *iff* LEM

Proof. Analogous to the proof of Theorem 3.11. The comprehension axiom is sound since every Henkin model $\mathcal{H}$ satisfies $\mathcal{H} \vDash \mathsf{Compr}_\varphi^n$ by definition. □

7 Translation to First-Order Logic

While the usual Henkin-style completeness proof used for FOL can also be adapted to second-order Henkin semantics [33], we follow a different strategy in this paper to obtain completeness. Interestingly, the switch to Henkin semantics not only yields completeness, but also brings many of the other (in)famous properties of FOL like compactness or the Löwenheim-Skolem theorems with it [33]. SOL interpreted in Henkin semantics actually reduces to FOL [27, 29, 39], meaning that given a second-order formula φ, one can construct a first-order formula $\varphi^\star$, that preserves validity and provability. In particular, this allows us to transport completeness from FOL to SOL:

$$\begin{tikzcd}
\mathcal{T} \vDash_2 \varphi \arrow[d, "(a)"'] & & \mathcal{T} \vdash_2 \varphi \\
\mathcal{T}^\star \vDash_1 \varphi^\star \arrow[rr, "\text{FOL Completeness}"] & & \mathcal{T}^\star \vdash_1 \varphi^\star \arrow[u, "(b)"]
\end{tikzcd}$$

The Completeness theorem for FOL is a well-known result due to Gödel [11] and has already been mechanised in Coq [9]. To keep the formalisation compact, we factor out this dependency and only verify that completeness of FOL implies completeness of SOL using properties (a) and (b).

[4] Comprehension is trivially satisfied by such a model.

We call (a) *semantic reduction* (Theorem 7.17) and (b) *deductive reduction* (Theorem 7.31). Note, that we write $\vdash_2$ and $\vDash_2$ for the classical second-order natural deduction system and Henkin validity to make the distinction between first- and second-order derivations and the different semantics clearer.

It is easy to see that Henkin semantics collapses SOL to many-sorted FOL [27], where quantifiers can range over different sorts unlike the mono-sorted case. For example, the second-order formula

$$\varphi := \dot{\forall} P.\, \dot{\exists} Q.\, \dot{\forall} x.\, P(x) \dot{\leftrightarrow} \dot{\neg} Q(x)$$

corresponds to many-sorted first-order formula

$$\varphi^\star := \dot{\forall} p^{\mathcal{P}_1}.\, \dot{\exists} q^{\mathcal{P}_1}.\, \dot{\forall} x^{\mathcal{I}}.\, \mathsf{App}_1(p, x) \dot{\leftrightarrow} \dot{\neg}\mathsf{App}_1(q, x)$$

with sorts $\mathcal{I}$ for individuals and $\mathcal{P}_1$ for unary predicates where the custom symbol App_1 replaces the unary predicate application.

However, we want to focus on the mono-sorted case in this paper since it is the more common formalism and also compatible with the FOL development in the Coq Library of Undecidability Proofs [10]. A common approach, for example taken by Van Dalen [39], simulates the sorts by guarding the quantifiers with custom predicates:

$$\varphi^\star := \dot{\forall} p.\, \mathsf{isPred}_1(p) \dot{\rightarrow} \dot{\exists} q.\, \mathsf{isPred}_1(q) \dot{\wedge} \dot{\forall} x.\, \mathsf{isIndi}(x)$$
$$\dot{\rightarrow} (\mathsf{App}_1(p, x) \dot{\leftrightarrow} \dot{\neg}\mathsf{App}_1(q, x))$$

While proving the semantic reduction property for this translation is straightforward, showing the deductive part is challenging. Van Dalen mentions that there is a "tedious but routine proof" [39], but he only gives a very brief sketch. Nour and Raffalli investigated this claim and "were not able to end his proof" [29]. They point out the problem, that this translation is not surjective, meaning that the formulas occurring in a proof of $\varphi^\star$ do not necessarily have the shape $\psi^\star$. Thus, it is not obvious how to translate such a proof in the first-order system to the second-order one (at least we are not aware of any proposed solution to this problem and also were not able to come up with one on our own).

Instead, Nour and Raffalli suggest a slightly simpler reduction that avoids this issue by staying closer to the original structure of the second-order formula [29]:

$$\varphi^\star := \dot{\forall} p.\, \dot{\exists} q.\, \dot{\forall} x.\, (\mathsf{App}_1(p, x) \dot{\leftrightarrow} \dot{\neg}\mathsf{App}_1(q, x))$$

They get rid of the sorts entirely and have p and q represent individuals and predicates of all all arities at the same time. The semantics of the App symbol then interprets them differently based on their position in the arguments.

7.1 Translation Function

We first fix a discrete and enumerable signature Σ in which we will work for the remainder of this section. As illustrated in the previous examples, the translation requires us to replace applications of predicate variables with corresponding first-order primitives. Therefore, we extend the signature with custom symbols that represent this operation:

Definition 7.1 (Extended Signature)**.** *The extended signature Σ_+ of Σ is obtained by adding $(n+1)$-ary predicate symbols App_n for all $n : \mathbb{N}$, representing the application of n-ary predicate variables.*

To implement the translation in the de Bruijn encoding, we need to map the independent scopes for individual and predicate variables of different arities into the single variable space provided by mono-sorted FOL. We use functions

$$\pi_{\mathsf{i}} : \mathbb{N} \to \mathbb{N} \qquad \pi_{\mathsf{p}} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

to keep track of which first-order de Bruijn index a given second-order variable corresponds to. An individual variable x_i gets turned into the first-order variable $\pi_{\mathsf{i}}\, i$ and a predicate variable p_i^n gets turned into $\pi_{\mathsf{p}}\, i\, n$.

We carry those functions throughout the whole translation process and update them on each quantifier. For example, when turning an individual quantifier into a first-order one, the individual variable 0 should still bind to this quantifier, while all other variables need to be shifted to accommodate the new variable. We write this transformation as $\uparrow_{\mathsf{i}} \pi$ and analogously $\uparrow_{\mathsf{p}}^n \pi$ for predicate quantifiers:

Definition 7.2. *The $\uparrow_{\mathsf{i}} \pi$ and $\uparrow_{\mathsf{p}}^n \pi$ operations are defined by*

$$(\uparrow_{\mathsf{i}} \pi_{\mathsf{i}})\, 0 := 0 \qquad (\uparrow_{\mathsf{p}}^n \pi_{\mathsf{i}})\, x := x + 1$$
$$(\uparrow_{\mathsf{i}} \pi_{\mathsf{i}})\, (x+1) := (\pi_{\mathsf{i}}\, x) + 1 \qquad (\uparrow_{\mathsf{p}}^n \pi_{\mathsf{p}})\, 0\, n := 0$$
$$(\uparrow_{\mathsf{i}} \pi_{\mathsf{p}})\, x\, n := x + 1 \qquad (\uparrow_{\mathsf{p}}^n \pi_{\mathsf{p}})\, (x+1)\, n := (\pi_{\mathsf{p}}\, x\, n) + 1$$
$$(\uparrow_{\mathsf{p}}^n \pi_{\mathsf{p}})\, x\, m := m, \quad \textit{if } m \neq n$$

This allows us to define the translation function:

Definition 7.3 (Translation)**.** *The translation function $\_^\star_\pi : \mathfrak{F}_2(\Sigma) \to \mathfrak{F}_1(\Sigma_+)$ and $\_^\star_\pi : \mathfrak{T}(\Sigma) \to \mathfrak{F}(\Sigma_+)$ is defined by:*

$$(\mathsf{x}_i)^\star_\pi := x_{\pi_{\mathsf{i}}\, i} \qquad \bot^\star_\pi := \bot$$
$$(\mathcal{F}\, \boldsymbol{t})^\star_\pi := \mathcal{F}\, \boldsymbol{t}^\star_\pi \qquad (\varphi \mathbin{\dot{\Box}} \psi)^\star_\pi := \varphi^\star_\pi \mathbin{\dot{\Box}} \psi^\star_\pi$$
$$(\mathcal{P}\, \boldsymbol{t})^\star_\pi := \mathcal{P}\, \boldsymbol{t}^\star_\pi \qquad (\dot{\nabla}\, \varphi)^\star_\pi := \dot{\nabla}\, \varphi^\star_{\uparrow_{\mathsf{i}} \pi}$$
$$(\mathsf{p}_i^n\, \boldsymbol{t})^\star_\pi := \mathsf{App}_n\, (x_{\pi_{\mathsf{p}}\, i\, n} :: \boldsymbol{t}^\star_\pi) \qquad (\dot{\nabla}_{\mathsf{p}}^n\, \varphi)^\star_\pi := \dot{\nabla}\, \varphi^\star_{\uparrow_{\mathsf{p}}^n \pi}$$

Initially, each free individual and predicate variable should be mapped to a unique first-order variable:

Definition 7.4. *We define the variable translation functions $\pi_{\mathsf{i}}^0\, x := \langle 0, x\rangle$ and $\pi_{\mathsf{p}}^0\, x\, n := \langle 1, \langle x, n\rangle\rangle$ and write $\varphi^\star$ for $\varphi^\star_{\pi^0}$.*

7.2 Semantic Reduction

We begin by verifying the semantic part of the reduction. That is, we need to translate first-order models $\mathcal{M}$ into Henkin models $\mathcal{M}^\diamond$ that preserve validity in relation to $\_^\star$. However, we extend Nour and Raffalli's approach and also verify the converse direction, allowing us to transport further meta-logical properties besides completeness. We begin with this latter direction, translating Henkin models $\mathcal{H}$ into first-order models $\mathcal{H}^\star$.

Let $\mathcal{H}$ be a Henkin model that consists of a domain D, a predicate universe $\mathbb{U}$ and a symbol interpretation $\mathcal{I}$. We also assume that $\mathcal{H}$ is not empty, i.e. there is some $d_0 : D$. By comprehension, we also get $P_0^n : D^n \to \mathbb{P}$ with $\mathbb{U}_n P_0^n$ for each n. Since first-order variables represent individuals and predicates at the same time, the translated first-order domain $D^\star$ should contain both the individuals and the predicates of $\mathcal{H}$:

Definition 7.5. *We set $D^\star := D + \Sigma n P.\, \mathbb{U}_n P$ and use injections* $\mathsf{fromIndi} : D \to D^\star$ *and* $\mathsf{fromPred}_n : \forall P.\, \mathbb{U}_n P \to D^\star$. *We just write* $\mathsf{fromPred}_n\, P$ *if $\mathbb{U}_n P$ is clear from the context.*

We interpret every object in $D^\star$ as an individual and predicate of every arity, using the dummy values d_0 and P_0^n if the types do not match up:

Definition 7.6. *We define projections* $\mathsf{toIndi} : D^\star \to D$ *and* $\mathsf{toPred}_n : D^\star \to (D^n \to \mathbb{P})$ *by*

$$\begin{aligned} \mathsf{toIndi}\,(\mathsf{fromIndi}\, d) &:= d \\ \mathsf{toIndi}\,(\mathsf{fromPred}_n\, \_) &:= d_0 \end{aligned} \qquad \begin{aligned} \mathsf{toPred}_n\,(\mathsf{fromIndi}\, \_) &:= P_0^n \\ \mathsf{toPred}_n\,(\mathsf{fromPred}_n\, P) &:= P \\ \mathsf{toPred}_n\,(\mathsf{fromPred}_m\, \_) &:= P_0^n, \ \ \textit{if } n \neq m \end{aligned}$$

Definition 7.7. *The first-order interpretation $\mathcal{I}^\star$ is given by*

$$\begin{aligned} \mathcal{F}^{\mathcal{I}^\star}\, \boldsymbol{v} &:= \mathsf{fromIndi}\,(\mathcal{F}^{\mathcal{I}}\,(\mathsf{toIndi}\, \boldsymbol{v})) \\ \mathcal{P}^{\mathcal{I}^\star}\, \boldsymbol{v} &:= \mathcal{P}^{\mathcal{I}}\,(\mathsf{toIndi}\, \boldsymbol{v}) \\ \mathsf{App}_n^{\mathcal{I}^\star}\,(d :: \boldsymbol{v}) &:= \mathsf{toPred}_n\, d\,(\mathsf{toIndi}\, \boldsymbol{v}) \end{aligned}$$

The last step is to translate Henkin environments ρ into first-order environments $\rho^\star$. Here, we need to make sure that $\rho^\star$ maps the free variables in a translated formula $\varphi^\star$ to the same values as ρ does in φ. Therefore, $\rho^\star$ needs to reverse π_i^0 and π_p^0 as defined in Definition 7.3:

Definition 7.8. *Let ρ be a Henkin environment. We set*

$$\rho^\star\, n := \begin{cases} \mathsf{fromIndi}\,(\rho_\mathrm{i}\, x) & \textit{if } n = \langle 0, x\rangle \\ \mathsf{fromPred}_m\,(\rho_\mathrm{p}^m\, x) & \textit{if } n = \langle a+1, \langle x, m\rangle\rangle \end{cases}$$

Fact 7.9. *For all Henkin environments ρ it holds that*

1. $\forall x.\, \mathsf{toIndi}\,(\rho^\star\,(\pi_\mathrm{i}^0\, x)) = \rho_\mathrm{i}\, x$
2. $\forall x n.\, \mathsf{toPred}_n(\rho^\star\,(\pi_\mathrm{p}^0\, x\, n)) = \rho_\mathrm{p}^n\, x$

Lemma 7.10 (Correctness of $\mathcal{H}^\star$). *Let ρ be a Henkin environment and $\varphi : \mathfrak{F}_2$. Then $\mathcal{H}, \rho \vDash_2 \varphi$ iff $\mathcal{H}^\star, \rho^\star \vDash_1 \varphi^\star$.*

Proof. We show the stronger claim

$$\begin{aligned} &\forall \rho_1 \rho_2 \pi.\, (\forall x.\, \mathsf{toIndi}\,(\rho_1\,(\pi_\mathrm{i}\, x)) = \rho_{2,\mathrm{i}}\, x) \\ &\quad \to (\forall x n.\, \mathsf{toPred}_n\,(\rho_1\,(\pi_\mathrm{p}\, x\, n)) = \rho_{2,\mathrm{p}}^n\, x) \\ &\quad \to (\mathcal{H}, \rho_2 \vDash_2 \varphi \leftrightarrow \mathcal{H}^\star, \rho_1 \vDash_1 \varphi_\pi^\star) \end{aligned}$$

by induction on φ, which suffices according to Fact 7.9. In the case of atomic formulas, we verify $[\![t]\!]_{\rho_2}^{\mathcal{H}} = \mathsf{toIndi}\, [\![t_\pi^\star]\!]_{\rho_1}^{\mathcal{H}^\star}$ for all $t : \mathfrak{T}$ by induction on t. For universal first-order quantifiers we have to show

$$(\forall d : D.\, \mathcal{H}, d \cdot \rho_2 \vDash_2 \varphi) \leftrightarrow (\forall a : D^\star.\, \mathcal{H}^\star, a \cdot \rho_1 \vDash_1 \varphi_{\uparrow_\mathrm{i}\pi}^\star).$$

- $\to$ Let $a : D^\star$. If $a = \mathsf{fromIndi}\, d$, apply the IH with $d \cdot \rho_2$. The preconditions of the IH hold since $\uparrow_\mathrm{i} \pi$ matches the updated environments $(\mathsf{fromIndi}\, d) \cdot \rho_1$ and $d \cdot \rho_2$. If $a = \mathsf{fromPred}_n\, P$ then use $d_0 \cdot \rho_2$ instead.
- $\leftarrow$ Let $d : D$. Instantiate the assumption with $(\mathsf{fromIndi}\, d)$, yielding $\mathcal{H}^\star, (\mathsf{fromIndi}\, d) \cdot \rho_1 \vDash_1 \psi_{\uparrow_\mathrm{i}\pi}^\star$ which is equivalent to $\mathcal{H}, d \cdot \rho_2 \vDash_2 \psi$ by the IH.

The remaining cases are trivial or handled similarly. □

For the second part of the semantic reduction, we have to translate first-order models $\mathcal{M}$ into Henkin models $\mathcal{M}^\diamond$. Let $\mathcal{M}$ be a first-order model that consists of a domain D and interpretation $\mathcal{I}$. Since each object in D represents an individual, we set $D^\diamond := D$ and $\mathcal{I}^\diamond := \mathcal{I}$ (excluding the App symbol). The predicate universe $\mathbb{U}^\diamond$ contains the set of predicates induced by the interpretation of the App symbol:

Definition 7.11. $\mathbb{U}_n^\diamond\, P := \exists d : D.\, \forall \boldsymbol{v}.\, P\, \boldsymbol{v} \leftrightarrow \mathsf{App}_n^{\mathcal{I}}\,(d :: \boldsymbol{v})$.

Definition 7.12. *Let ρ be a first-order environment. We set*

$$\rho_\mathrm{i}^\diamond\, x := \rho\,(\pi_\mathrm{i}^0\, x) \qquad (\rho_\mathrm{p}^n)^\diamond\, x := \lambda \boldsymbol{v}.\, \mathsf{App}_n^{\mathcal{I}}\,(\rho\,(\pi_\mathrm{p}^0\, x\, n) :: \boldsymbol{v})$$

Fact 7.13. *$\rho^\diamond$ is a well-formed Henkin environment, that is $\mathbb{U}_n^\diamond\,((\rho_\mathrm{p}^n)^\diamond\, x)$ for all x and n.*

Lemma 7.14 (Correctness of $\mathcal{M}^\diamond$). *Let ρ be a first-order environment and $\varphi : \mathfrak{F}_2$. Then $\mathcal{M}, \rho \vDash_1 \varphi^\star$ iff $\mathcal{M}^\diamond, \rho^\diamond \vDash_2 \varphi$.*

Proof. Similar to the proof of Lemma 7.10. □

However, $\mathcal{M}^\diamond$ does not necessarily satisfy comprehension. By the definition of $\mathbb{U}^\diamond$, this depends on how $\mathcal{M}$ interprets the App symbol. We introduce a special theory to enforce comprehension:

Definition 7.15. *Let C be the second-order theory containing the formulas $\overline{\forall}\, \mathsf{Compr}_\varphi^n$ for all $n : \mathbb{N}$ and $\varphi : \mathfrak{F}_2$, where the $\overline{\forall}$ operation closes the formulas with universal quantifiers.*

Fact 7.16. *If $\mathcal{M}, \rho_1 \vDash_1 C^\star$ for some first-order environment ρ_1, then $\mathcal{M}^\diamond$ satisfies comprehension, i.e. $\mathcal{M}^\diamond \vDash_2 \mathsf{Compr}_\varphi^n$ for all n and φ.*

Proof. We have $\mathcal{M}^\diamond, \rho_1^\diamond \vDash_2 \overline{\forall}\, \mathsf{Compr}_\varphi^n$ by Lemma 7.14. Given an arbitrary Henkin environment ρ_2, we can instantiate the quantifiers with the first values from ρ_2 and replace $\rho_1^\diamond$ with the remaining part of ρ_2, yielding $\mathcal{M}^\diamond, \rho_2 \vDash_2 \mathsf{Compr}_\varphi^n$. □

Theorem 7.17 (Semantic Reduction). *For all second-order theories $\mathcal{T}$ and second-order formulas φ it holds that*

$$\mathcal{T} \vDash_2 \varphi \ \leftrightarrow\ (\mathcal{T}, C)^\star \vDash_1 \varphi^\star.$$

Proof. The first direction follows from Lemma 7.14. The other direction uses Lemma 7.10 and the fact that $\mathcal{H}, \rho \vDash_2 C$ for all Henkin models $\mathcal{H}$ by definition. The Henkin models are also non-empty since we are given an environment, justifying the existence of the dummy individuals used in Definition 7.7. □

7.3 Deductive Reduction

Next, we verify the reduction on the deductive level. To obtain completeness, it suffices to show that $(\mathcal{T}, C)^\star \vdash_1 \varphi^\star$ implies$\mathcal{T} \vdash_2 \varphi$. Note that the converse direction easily follows from first-order completeness and the previous results:

Fact 7.18. *Assuming that FOL is complete, $\mathcal{T} \vdash_2 \varphi$ implies $(\mathcal{T}, C)^\star \vdash_1 \varphi^\star$ under* LEM*.*

Proof. We have $\mathcal{T} \vDash_2 \varphi$ by soundness using LEM, hence $(\mathcal{T}, C)^\star \vDash_1 \varphi^\star$ by Theorem 7.17, and finally $(\mathcal{T}, C)^\star \vdash_1 \varphi^\star$ by first-order completeness. □

Proving the opposite direction is more challenging. Nour and Raffalli define a backwards translation function $\_^\diamond : \mathfrak{F}_1(\Sigma_+) \to \mathfrak{F}_2(\Sigma)$ that turns arbitrary first-order formulas in the extended signature back into second-order formulas of the original signature, fulfilling:

1. $\Gamma \vdash_1 \varphi \to \Gamma^\diamond \vdash_2 \varphi^\diamond$
2. $\vdash_2 \varphi^{\star\diamond} \dot\leftrightarrow \varphi$

Using (1) one can turn $(\mathcal{T}, C)^\star \vdash_1 \varphi^\star$ into $(\mathcal{T}, C)^{\star\diamond} \vdash_1 \varphi^{\star\diamond}$ which is equivalent to $\mathcal{T}, C \vdash_2 \varphi$ according to (2), yielding $\mathcal{T} \vdash_2 \varphi$ as $\vdash_2$ proves comprehension. This avoids a direct induction on $(\mathcal{T}, C)^\star \vdash_1 \varphi^\star$ where intermediate formulas might not have the shape $\psi^\star$.

To motivate the definition of $\_^\diamond$, consider the formula

$$\varphi := \dot\forall x.\, \mathsf{App}_0(x) \dot\wedge \mathsf{App}_1(x, x)$$

where x is used as an individual as well as a nullary and unary predicate. We split the different usages of x into an individual variable x and predicate variables X^0 and X^1 that each get bound by their own quantifier, and replace the App symbols with the actual second-order atomic formulas:

$$\varphi^\diamond := \dot\forall x.\, \dot\forall X^1 X^0.\, X^0 \dot\wedge X^1(x)$$

Implementing this in the de Bruijn encoding is straightforward, since the variables X^0, X^1, and x still correspond to the index 0 as they live in their own scopes. In general, $\mathsf{App}_n\ (\mathsf{x}_i :: \boldsymbol{t})$ is turned into $\mathsf{p}_i^n\ \boldsymbol{t}$ and x_i stays the same, leaving terms unchanged by the backwards translation.

To ease the mechanisation, we over-approximate the number of second-order quantifiers needed to bind the different variables, which does not alter the meaning of the formula. To this end, we introduce the notion of arity bounds:

Definition 7.19 (Arity Bounds)**.** *b is a bound on the arities occuring in φ, if all predicate variables in φ have a smaller arity than b. We write $|\varphi|$ for the lowest arity bound on ϕ.*

Definition 7.20 (Predicate Closing Operation)**.** *We define $\dot\nabla_\mathsf{p}^{<n}\ \varphi$ recursively by $\dot\nabla_\mathsf{p}^{<0}\ \varphi := \varphi$ and $\dot\nabla_\mathsf{p}^{<n+1}\ \varphi := \dot\nabla_\mathsf{p}^{n}\ \dot\nabla_\mathsf{p}^{<n}\ \varphi$.*

Finally, one needs to handle the case that the first argument of App is not a variable, but a function application $\mathcal{F}\ \boldsymbol{t}$. Since this cannot be turned into a second-order variable, it is treated as an error case. While Nour and Raffalli directly return $\bot$, we introduce a special falsity symbol $\bot_n$ for every arity, which eases the mechanisation later on (see Footnote 5).

Definition 7.21. *The extended signature $\Sigma_\bot$ is obtained by adding n-ary predicate symbols $\bot_n$ for each $n : \mathbb{N}$ to Σ.*

Definition 7.22 (Backwards Translation Function)**.** *The backwards translation function $\_^\diamond : \mathfrak{F}_1(\Sigma_+) \to \mathfrak{F}_2(\Sigma_\bot)$ is recursively defined by*

$$\begin{aligned} \bot^\diamond &:= \bot & (\mathcal{P}\ \boldsymbol{t})^\diamond &:= \mathcal{P}\ \boldsymbol{t} \\ (\mathsf{App}_n\ (\mathsf{x}_i :: \boldsymbol{t}))^\diamond &:= \mathsf{p}_i^n\ \boldsymbol{t} & (\varphi \mathbin{\dot\Box} \psi)^\diamond &:= \varphi^\diamond \mathbin{\dot\Box} \psi^\diamond \\ (\mathsf{App}_n\ (\mathcal{F}\ \boldsymbol{v} :: \boldsymbol{t}))^\diamond &:= \bot_n\ \boldsymbol{t} & (\dot\nabla \varphi)^\diamond &:= \dot\nabla\ \dot\nabla_\mathsf{p}^{<|\varphi|^\diamond}\ \varphi^\diamond \end{aligned}$$

Next, we verify property (1):

Theorem 7.23. *$\Gamma \vdash_1 \varphi$ implies $\Gamma^\diamond \vdash_2 \varphi^\diamond$ for all Γ and φ.*

Proof. By induction on the first-order derivation $\Gamma \vdash_1 \varphi$. We only discuss the cases of all-introduction and -elimination:

(AllI): We get the inductive hypothesis $(\Gamma[\uparrow])^\diamond \vdash_2 \varphi^\diamond$ and have to show $\Gamma^\diamond \vdash_2 \dot\forall_\mathsf{p}^{<|\varphi^\diamond|}\ \dot\forall\, \varphi^\diamond$. We introduce the quantifiers with (AllI′) and (AllI′$_\mathsf{p}$) and have to show

$$\Gamma^\diamond \vdash_2 (\varphi^\diamond)[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p}^{<|\varphi^\diamond|}$$

where $\psi[\mathsf{p}_i]_\mathsf{p}^{<n}$ is notation for $\psi[\mathsf{p}_i^0]_\mathsf{p}^0 \dots [\mathsf{p}_i^{n-1}]_\mathsf{p}^{n-1}$ and i is an unused index. The shift in the IH can be written as $(\Gamma[\uparrow])^\diamond = (\Gamma^\diamond)[\uparrow][\uparrow]_\mathsf{p}$, where $[\uparrow]_\mathsf{p}$ shifts variables of every arity. We "undo" the shift using (Subst), a variant of (Subst$_\mathsf{p}$), and the fact that $(\Gamma^\diamond)[\uparrow][\uparrow]_\mathsf{p}[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p} = \Gamma^\diamond$, yielding

$$\Gamma^\diamond \vdash_2 (\varphi^\diamond)[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p}.$$

This suffices, since all variables in $\varphi^\diamond$ have arities smaller than $|\varphi^\diamond|$ and hence $(\varphi^\diamond)[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p} = (\varphi^\diamond)[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p}^{<|\varphi^\diamond|}$.

(AllE): In the case of all elimination we get the inductive hypothesis $\Gamma^\diamond \vdash_2 \dot\forall_\mathsf{p}^{<|\varphi^\diamond|}\ \dot\forall\, \varphi^\diamond$ and have to show $\Gamma^\diamond \vdash_2 (\varphi[t])^\diamond$ for some $t : \mathfrak{T}$. Our next steps depend on whether t is a variable or not:

- If $t = \mathsf{x}_i$, we have $(\varphi[t])^\diamond = (\varphi^\diamond)[\mathsf{x}_i][\mathsf{p}_i]_\mathsf{p}$. Thus, we conclude by using (AllE) and (AllE$_\mathsf{p}$) on the IH.
- If $t = \mathcal{F}\ \boldsymbol{t}$, then the error case is triggered whenever the substitution occurs at the first argument of App. Thus, we have $(\varphi[t])^\diamond = (\varphi^\diamond)[\mathcal{F}\ \boldsymbol{t}][\bot]_\mathsf{p}$.[5] Thus, we once again use (AllE) and (AllE$_\mathsf{p}$) on the IH.

Existential quantifiers are handled similarly and the remaining cases are trivial. □

We remove the error symbols from $\Gamma^\diamond \vdash_2 \varphi^\diamond$ by replacing them with $\bot$ using comprehension:

[5] $[\bot]_\mathsf{p}$ refers to substituting the falsity symbol $\bot_n$ from Definition 7.21 for every arity. If we had set $(\mathsf{App}_n\ (\mathcal{F}\ \boldsymbol{v} :: \boldsymbol{t}))^\diamond := \bot$ like Nour and Raffalli, we would need to use (Compr) to obtain predicates $P_\bot^0, \dots, P_\bot^{n-1}$ that are extensionally equivalent to $\bot$ and verify that $\Gamma^\diamond \vdash_2 (\varphi[\mathcal{F}\ \boldsymbol{t}])^\diamond$ is equivalent to $\Gamma^\diamond \vdash_2 (\varphi^\diamond)[\mathcal{F}\ \boldsymbol{t}][P_\bot^0]_\mathsf{p}^0 \dots [P_\bot^{n-1}]_\mathsf{p}^{n-1}$. Instead, we use the custom symbols $\bot_n$ here and remove them later in an extra step, easing the mechanisation.

Definition 7.24 (Falsity Symbol Removal). *We recursively define the function* $\_^{\perp} : \mathfrak{F}_2(\Sigma_{\perp}) \to \mathfrak{F}_2(\Sigma)$ *by*

$$\begin{aligned} \dot{\bot}^{\perp} &:= \dot{\bot} & (\varphi \mathbin{\dot{\Box}} \psi)^{\perp} &:= \varphi^{\perp} \mathbin{\dot{\Box}} \psi^{\perp} \\ (\dot{\bot}_n\, \boldsymbol{t})^{\perp} &:= \dot{\bot} & (\dot{\nabla}\, \varphi)^{\perp} &:= \dot{\nabla}\, \varphi^{\perp} \\ (\mathcal{P}\, \boldsymbol{t})^{\perp} &:= \mathcal{P}\, \boldsymbol{t} & (\dot{\nabla}^n_{\mathrm{p}}\, \varphi)^{\perp} &:= \dot{\nabla}^n_{\mathrm{p}}\, \varphi^{\perp} \end{aligned}$$

Lemma 7.25 (Falsity Symbol Derivations). *For all* $\varphi : \mathfrak{F}_2(\Sigma_{\perp})$ *and contexts* Γ *it holds that* $\Gamma \vdash_2 \varphi$ *implies* $\Gamma^{\perp} \vdash_2 \varphi^{\perp}$.

Proof. By induction on the derivation $\Gamma \vdash_2^{\perp} \varphi$. We only discuss the case of eliminating a predicate quantifier with $\dot{\bot}_n$, where we have to show $\Gamma^{\perp} \vdash_2 (\varphi[\dot{\bot}_n]^n_{\mathrm{p}})^{\perp}$ using the IH $\Gamma^{\perp} \vdash_2 \dot{\forall}^n_{\mathrm{p}}\, \varphi^{\perp}$. We obtain a variable p^n_f with

$$\Gamma^{\perp} \vdash_2 \dot{\forall} x_1, ..., x_n.\, \mathrm{p}^n_f\, (x_1, ..., x_n) \dot{\leftrightarrow} \dot{\bot}$$

using (Compr) and (ExE′$_{\mathrm{p}}$). We can show $\vdash_2 (\varphi[\dot{\bot}_n]^n_{\mathrm{p}})^{\perp} \dot{\leftrightarrow} (\varphi^{\perp})[\mathrm{p}^n_f]^n_{\mathrm{p}}$ such that it suffices to prove $\Gamma^{\perp} \vdash_2 (\varphi^{\perp})[\mathrm{p}^n_f]^n_{\mathrm{p}}$ which follows from the IH using (AllE$_{\mathrm{p}}$). □

Next, we show that $\varphi^{\star\diamond\perp}$ is equivalent to φ (property (2)). Again, we begin with an example:

$$\begin{aligned} \varphi &= \dot{\forall} x.\, Q(x,x) \dot{\to} \dot{\exists} P.\, P(x) \\ \varphi^{\star} &= \dot{\forall} x.\, \mathrm{App}_2(q,x,x) \dot{\to} \dot{\exists} p.\, \mathrm{App}_1(p,x) \\ \varphi^{\star\diamond} &= \dot{\forall} x.\, \dot{\forall} X_2 X_1 X_0.\, Q(x,x) \dot{\to} \dot{\exists} p.\, \dot{\exists} P_1 P_0.\, P_1(x) \end{aligned}$$

Notice that the backwards translation did not trigger the error case since $\_^{\star}$ does not introduce subformulas of form $\mathrm{App}_n\, (\mathcal{F}\, \boldsymbol{t} :: \_)$. Hence, $\varphi^{\star\diamond\perp}$ is identical to $\varphi^{\star\diamond}$ in general. Also notice that the variable x got split up into multiple second-order variables but only the original variable x is bound in $\varphi^{\star\diamond}$. Similarly, P got split up, but only the unary predicate variable P_1 is bound, corresponding to the original variable P in φ. Thus, translating to FOL and back only adds some unused quantifiers and should therefore be equivalent to the initial formula.[6]

However, the translation of φ with named binders above is not entirely accurate, as we translated the unbound variable Q into the first-order variable q and then conveniently back into Q. In the de Bruijn encoding, Q is represented by some de Bruijn index i. The first-order translation uses its π^0 functions to replace i with $\pi^0_{\mathrm{p}}\, i\, 2$. Since the backwards translation preserves the indices, what we end up with no longer represents the same variable. To resolve this issue, we need to modify the backwards translation such that it turns $\pi^0_{\mathrm{p}}\, i\, 2$ back into i. The key is that the choice of π can be simulated via a substitution:

[6]One might hope to show $\varphi^{\star\diamond\perp} = \varphi$ by more carefully defining $\_^{\diamond}$ such that the number of quantifiers is not over-approximated. However, this would at least require a pre-processing step that removes unused formulas: Consider $\psi := \dot{\forall} R.\, \dot{\bot}$ with $\psi^{\star} := \dot{\forall} r.\, \dot{\bot}$. It is not possible to deduce from $\psi^{\star}$ whether r was an individual or a predicate in ψ, nor can we deduce its arity in the latter case. Thus, without additional pre-processing, the backwards translation is not able to restore the initial formula in general.

Lemma 7.26 (π-Substitutions). *For all* φ *and* π *it holds that* $(\varphi^{\star}_{\pi})^{\diamond} = (\varphi^{\star}_{\mathrm{id}})^{\diamond}[\lambda i.\, \mathrm{x}_{\pi_{\mathrm{i}}\, i}][\lambda in.\, \mathrm{p}^n_{\pi_{\mathrm{p}}\, i\, n}]_{\mathrm{p}}$, *with* $\mathrm{id} = (\lambda x.x, \lambda xn.x)$.

Thus, one can "undo" the effect of π^0 by substituting the inverse functions:

Definition 7.27 (Inverse π^0 Functions). *We define* $(\pi_{\mathrm{i}})^{-1}$ *by* $(\pi^0_{\mathrm{i}})^{-1}\, \langle \_, x \rangle := x$ *and* $(\pi^0_{\mathrm{p}})^{-1}\, \langle \_, \langle x, \_ \rangle \rangle := x$

Definition 7.28 (Fixed Backwards Translation). *We define* $\varphi^{\blacklozenge} := (\varphi^{\diamond})[\lambda xn.\, \mathrm{p}^n_{(\pi^0_{\mathrm{p}})^{-1}\, x}]_{\mathrm{p}}\, [\lambda x.\, \mathrm{x}_{(\pi^0_{\mathrm{i}})^{-1}\, x}]$.

With the fixed translation, we can derive equivalence:

Lemma 7.29. $\vdash_2 \varphi^{\star\blacklozenge\perp} \dot{\leftrightarrow} \varphi$ *for all* φ.

Proof. By Lemma 7.26 it suffices to prove $\vdash_2 (\varphi^{\star}_{\mathrm{id}})^{\diamond\perp} \dot{\leftrightarrow} \varphi$ by induction on φ. We only give a brief intuition for the case of universal first-order quantification since the proof is fairly technical. We get the IH $\vdash_2 (\varphi^{\star}_{\mathrm{id}})^{\diamond\perp} \dot{\leftrightarrow} \varphi$ and have to show

$$\vdash_2 \left(\dot{\forall}_{\mathrm{p}}^{< |(\varphi^{\star}_{\uparrow_{\mathrm{i}} \mathrm{id}})^{\diamond}|}\, \dot{\forall}\, (\varphi^{\star}_{\uparrow_{\mathrm{i}} \mathrm{id}})^{\diamond} \right)^{\perp} \dot{\leftrightarrow} \dot{\forall}\, \varphi.$$

Recall from the motivating example that the added predicate quantifiers are all unused. This is the case because the operation $\uparrow_{\mathrm{i}}$ id shifts all predicate variables and hence makes the index 0 unbound, whereas first-order variables remain unchanged (follows from Lemma 7.26).

→ It suffices to instantiate the unused predicate quantifiers with arbitrary values, which reverts the shifts and leaves us with $\dot{\forall}\, (\varphi^{\star}_{\mathrm{id}})^{\diamond\perp} \vdash_2 \dot{\forall}\, \varphi$, resolved by the IH.
← Introducing the unused predicate quantifiers through (AllI′$_{\mathrm{p}}$) again reverts the shifts.

The cases for existential and second-order quantifiers are similar and the remaining ones are straightforward. □

The first property also still holds for the fixed backwards translation:

Corollary 7.30. $\Gamma \vdash_1 \varphi$ *implies* $\Gamma^{\blacklozenge\perp} \vdash_2 \varphi^{\blacklozenge\perp}$ *for all* φ *and* Γ.

Proof. Follows from Lemma 7.25, (Subst) and Theorem 7.23. □

With this final component established, the following diagram summarises the main steps of the deductive reduction:

$$\begin{tikzcd} (\mathcal{T}, C)^{\star} \vdash_1 \varphi^{\star} \arrow[d, "\text{Corollary 7.30}"'] & & \mathcal{T}, C \vdash_2 \varphi \\ (\mathcal{T}, C)^{\star\blacklozenge} \vdash_1 \varphi^{\star\blacklozenge} \arrow[rr, "\text{Lemma 7.25}"] & & (\mathcal{T}, C)^{\star\blacklozenge\perp} \vdash_2 \varphi^{\star\blacklozenge\perp} \arrow[u, "\text{Lemma 7.29}"'] \end{tikzcd}$$

Finally, by composition of these intermediate steps, we obtain the desired main result:

Theorem 7.31 (Deductive Reduction). $(\mathcal{T}, C)^{\star} \vdash_1 \varphi^{\star}$ *implies* $\mathcal{T} \vdash_2 \varphi$ *for all* $\mathcal{T}$ *and* φ.

Proof. By Corollary 7.30 we get $(\mathcal{T}, \mathcal{C})^{\star\blacklozenge\perp} \vdash_1 \varphi^{\star\blacklozenge\perp}$ and obtain $(\mathcal{T}, \mathcal{C}) \vdash_2 \varphi$ by Lemma 7.29. Since $\vdash_2 \mathcal{C}$ using (Compr), we get $\mathcal{T} \vdash_2 \varphi$. □

8 Consequences of the Reduction to FOL

We now employ the verified translation from FOL to SOL to show how characteristic properties transport from the former to the latter. Since the considered properties require classical assumptions in their general forms [19, 24], we give relative statements simply assuming the properties for FOL.

8.1 Completeness

We use the reduction to obtain second-order Henkin completeness from the first-order completeness theorem:

Theorem 8.1 (Completeness)**.** *Assuming that FOL is complete, then so is SOL with Henkin semantics.*

Proof. Follows by Theorem 7.17 and Theorem 7.31. □

8.2 Compactness

Compactness (i.e. the property that finitely satisfiable theories have a model) follows classically from completeness:

Theorem 8.2 (Compactness by Completeness)**.** *Assuming that FOL is complete, then SOL with Henkin semantics is compact under* LEM*.*

Proof. Let $\mathcal{T}$ be a second-order theory. Assume that all finite contexts $\Gamma \subseteq \mathcal{T}$ have a model. Since we assume LEM we can argue classically. Suppose there were no model of $\mathcal{T}$. Then $\mathcal{T} \vDash_2 \perp$ and by completeness (Theorem 8.1) $\mathcal{T} \vdash_2 \perp$. Hence, there is also a finite context $\Gamma \subseteq \mathcal{T}$ with $\Gamma \vDash_2 \perp$ and by soundness $\Gamma \vdash_2 \perp$. This is not possible, since by assumption $\Gamma \subseteq \mathcal{T}$ has a model. □

This is essentially the same proof that allowed us to refute strong completeness in Theorem 4.9. However, we can also directly derive Henkin compactness from first-order compactness, only requiring the semantic part of the reduction:

Theorem 8.3 (Compactness by Reduction)**.** *Assuming that FOL is compact, then so is SOL with Henkin semantics.*

Proof. Let $\mathcal{T}$ be a second-order theory. Assume that all finite contexts $\Gamma \subseteq \mathcal{T}$ have a Henkin model. To show that $\mathcal{T}$ also has a Henkin model it suffices to show that there is a first-order model $\mathcal{M}$ with $\mathcal{M} \vDash_1 (\mathcal{T}, \mathcal{C})^\star$, since then $\mathcal{M}^\diamond \vDash_2 \mathcal{T}$ by Lemma 7.10. By first-order compactness it is enough to verify that every finite $\Gamma^\star \subseteq (\mathcal{T}, \mathcal{C})^\star$ has a first-order model. We split this context into $\Gamma^\star = (\Gamma_\mathcal{T})^\star + (\Gamma_\mathcal{C})^\star$ with $\Gamma_\mathcal{T} \subseteq \mathcal{T}$ and $\Gamma_\mathcal{C} \subseteq \mathcal{C}$. By assumption, we know that $\Gamma_\mathcal{T}$ has a Henkin model $\mathcal{H}$ which gives us a first-order model $\mathcal{H}^\star \vDash_1 \Gamma_\mathcal{T}$ by Lemma 7.14. Since $\mathcal{H}^\star$ satisfies comprehension by construction, it is also a model of $(\Gamma_\mathcal{T})^\star + (\Gamma_\mathcal{C})^\star = \Gamma^\star$. □

8.3 Löwenheim-Skolem

Instead of separating the Löwenheim-Skolem theorems into an upward and downward direction, we use a version that combines both of them into one statement. That is, we say a logic has the Löwenheim-Skolem property if every theory with some infinite model has models of every infinite cardinality.[7]

Theorem 8.4 (Löwenheim-Skolem)**.** *Assuming that FOL has the Löwenheim-Skolem property, then so does SOL with Henkin semantics.*

Proof. Let $\mathcal{T}$ be a second-order theory with an infinite Henkin model $\mathcal{H}$. We have to show that for every infinite type X, there is also a model of $\mathcal{T}$ in bijection to X. It suffices to find a first-order model $\mathcal{M}$ of $(\mathcal{T}, \mathcal{C})^\star$ in bijection to X since the Henkin model $\mathcal{M}^\star$ preserves the domain. Thus, by the first-order Löwenheim-Skolem property it suffices to show that $(\mathcal{T}, \mathcal{C})^\star$ has some infinite first-order model. Since $\mathcal{H}^\star$ is infinite and also satisfies $(\mathcal{T}, \mathcal{C})^\star$, we are finished. □

We used this combined statement since it allows for an easier proof. Showing the upwards direction on its own using the reduction would be more complicated. Turning a Henkin model $\mathcal{H}$ into a first-order model $\mathcal{H}^\star$ can increase the cardinality, since the translated domain $D^\star$ also contains the predicate universe $\mathbb{U}$ (see Definition 7.5). Thus, one might need to "go down" first in order to end up with the desired cardinality. Our approach sidesteps this issue and produces an equivalent result.

9 Discussion

9.1 Coq Mechanisation

Our Coq mechanisation does not rely on any axioms and consists of 7900 lines of code, on top 1600 lines reused from the Coq Library of Undecidability Proofs [10]. Among the new code, 3600 lines are dedicated to the general setup of the second-order syntax, standard semantics, and natural deduction system. The development concerning PA_2 and its categoricity span 1600 lines with undecidability and incompleteness taking another 600 lines. The verification of Nour and Raffalli's reduction to FOL was the technically most challenging part, consisting of 2100 lines. The main difficulty lied in the deductive part of the reduction described in Section 7.3, in particular Theorem 7.23 and especially Lemma 7.29. Coming up with the right proof strategy to show Lemma 7.29 required us to capture the intuition that $\varphi^{\star\diamond}$ only adds unused quantifiers to φ. In the end, the main ingredient was Lemma 7.26 which allowed us to characterise the interaction between $\uparrow_\mathsf{i}\ \pi$, $\uparrow^n_\mathsf{p}\ \pi$ and substitutions. Moreover, working inside the deduction system with the de Bruijn encoding combined with the relatively complicated definition of $\varphi^\diamond$ relying on arity bounds etc. made the proofs quite tedious.

[7] We interpret this as beeing in bijection with every infinite type.

Our mechanisation of SOL is based on the existing FOL developments in [9, 20, 21]. In particular, we define signatures Σ as type classes allowing for the appropriate signature to be inferred based on context. The formalisation of predicate substitutions slightly differs from the presentation on paper in Definition 3.4. Instead of using substitutions $\varphi[\sigma]_p^n$ for a specific arity n, we employ substitutions $\varphi[\sigma]_p$ with $\sigma : \mathbb{N} \to \forall n.\, \mathfrak{P}_n$ that effect predicates of all arities and define the special case

$$\varphi[\sigma]_p^n := \varphi[\lambda im.\, \text{if } m = n \text{ then } \sigma\, i \text{ else } \mathsf{p}_i^m].$$

This simplifies the development in Section 7.3 since the substitutions dealing with different arities can be represented within a single operation.

While the presentation on paper focuses on SOL with predicate quantifiers, the mechanisation also incorporates function quantifiers (explicitly used in Lemma 5.13) and only restricts to the function-free fragment when necessary. See Appendix A for further details on function quantifiers.

9.2 Related Work

Mechanised first- and higher-order logic. While we are not aware of any previous mechanisations of SOL, there have been many formalisations concerned with first- or higher-order logics in various theorem provers like Isabelle/HOL [13, 14, 25], Mizar [3], Lean [40], and Coq [2, 9, 17, 21]. Our work on SOL is inspired by the Coq formalisation of FOL developed in [9] and [21]. We adopted their major design decisions of using a syntax with de Bruijn binders parametrised over an arbitrary signature. The second-order natural deduction introduced in Section 3 is also an extension of the first-order system presented in [9]. We also want to point to the work by Kirst and Smolka [22, 23] in which they investigate second-order ZF set theory in Coq, verifying that the axiomatisation is categorical for equipotent models.

Synthetic computability theory. The synthetic approach to computability theory was developed by Richman [32] and Bauer [1] and adapted to the type theory of Coq by Forster, Kirst, and Smolka [8]. Their work lays the foundation for the synthetic analysis of incompleteness and undecidability carried out in Section 5. Crucially, the synthetic undecidability of H_{10} has been verified by Larchey-Wendling and Forster [26] as part of the Coq Library of Undecidability Proofs [10], which serves as the starting point for our own reductions. The undecidability results mechanised in this paper are also intended as a contribution to this library.

Synthetic incompleteness. The computational account of incompleteness used in Section 5 circumventing the more involved mechanisation of explicit independent Gödel/Rosser sentences was employed by Kirst and Hermes [20], further benefitting from the synthetic approach. The initial reduction from H_{10} we use to establish undecidability of SOL is also drawn from their work.

Sources. Finally, we largely followed Shapiro's [33] account of SOL, and the method used in Section 7 to reduce SOL to FOL is due to Nour and Raffalli [29].

9.3 Future Work

While we have already contributed our undecidability results for SOL to the Coq Library of Undecidability Proofs [10], we still plan on merging the first-order completeness formalisation by Forster, Kirst, and Wehr [9] into our development. This would allow us to strengthen the relative statement "if FOL is complete then so is SOL" (Theorem 8.1) to a fully mechanised completeness result of SOL. Similarly, a future mechanisation of the first-order Löwenheim-Skolem theorem is needed to fully obtain Löwenheim-Skolem for SOL. In particular, a constructive analysis of Löwenheim-Skolem for FOL would be interesting in this context [19].

Forster, Kirst, and Wehr also analysed the constructiveness of FOL completeness in their work, focusing on the $\dot\forall, \dot\to, \dot\bot$-fragment. While the extension to full FOL requires LEM, completeness for the restricted fragment is equivalent to Markov's principle [16, 24]. It might be worthwhile to investigate whether our treatment of SOL and the reduction to FOL can be altered to work in this fragment, in particular regarding the dependency of comprehension on the existential quantifier. Furthermore, one can extend the FOL reduction to second-order Kripke models as demonstrated by Nour and Raffalli [29], which would allow us to also derive intuitionistic completeness from FOL.

Regarding PA_2, it would be interesting to investigate conservativity results concerning PA_1: Restricting Ind_2 to first-order definable properties and comprehension to first-order properties with second-order parameters results in the system called *arithmetical comprehension* (ACA_0) from reverse mathematics, which is a conservative extension of PA_1 [34].

Furthermore, it would also be worthwhile to explore second-order axiom systems apart from Peano arithmetic. For example, Shapiro shows that the categoricity proof of PA_2 can be extended to yield categoricity of second-order real analysis [33]. It would be interesting to investigate whether Shapiro's construction also works in our setting where the isomorphisms are not necessarily computable. Moreover, Kirst and Smolka provide a mechanisation of the categoricity of second-order ZF (for equipotent models) in Coq [22] that could be transported to our SOL backbone.

Finally, it would be interesting to mechanise results regarding the notion of *internal categoricity* popularised by Väänänen [41, 43], which draws from the fact that categoricity can be expressed in SOL and oftentimes proven internally via the deduction system $\vdash$ without relying on the external meta-theory. Verifying this would be a major challenge as it requires very complex derivations in $\vdash$. However, existing tooling for FOL like the proof mode developed in [18] could be extended to SOL and facilitate this task.

A Function Quantifiers

Besides predicate quantifiers, some authors also consider function quantifiers as part of SOL (for example [33, 42]). In the textbook setting they simply allow for more convenient notation and do not bring any additional expressive power with them, since they can be reduced back to predicates. However, the treatment of function quantifiers is more involved in our setting of constructive type theory. We define the syntactic fragment $\mathfrak{F}_2^F$ that extends $\mathfrak{F}_2$ with function quantifiers and variables:

Definition A.1 (SOL with Function Quantifiers)**.** *The types $\mathfrak{T}_2^F$ and $\mathfrak{F}_2^F$ of second-order terms and formulas with function quantifiers is defined by extending Definition 3.3 with*

$$t : \mathfrak{T}_2^F ::= \dots \mid \mathsf{f}_i^n\, \boldsymbol{t} \qquad (i, n : \mathbb{N}, \boldsymbol{t} : (\mathfrak{T}_2^F)^n)$$

$$\varphi, \psi : \mathfrak{F}_2^F ::= \dots \mid \dot\forall_{\mathsf{f}}^n \varphi \mid \dot\exists_{\mathsf{f}}^n \varphi \qquad (n : \mathbb{N})$$

The de Bruijn scoping works analogously to $\mathfrak{F}_2$. The most natural way to extend the standard semantics to $\mathfrak{F}_2^F$ is to interpret function quantifiers ranging over type theoretic functions $D^n \to D$. This is also the semantics used in Lemma 5.13:

Definition A.2 (Extended Standard Semantics)**.** *Given a model $\mathcal{M}$ with domain D, environments ρ are extended with an additional component $\rho_{\mathsf{f}} : \mathbb{N} \to \forall n. D^n \to D$ assigning values to function variables. Term evaluation and formula satisfiability from Definition 3.6 is extended to $\mathfrak{T}_2^F$ and $\mathfrak{F}_2^F$ by adding the following cases:*

$$[\![\mathsf{f}_i^n]\!]_\rho := \rho_{\mathsf{f}}\; i\; n \qquad \rho \vDash \dot\forall_{\mathsf{f}}^n \varphi := \forall f : D^n \to D.\, f \cdot \rho \vDash \varphi$$

Usually, second-order logic is defined in terms of classical set theory as meta-logic. There, n-ary functions are considered as the $(n+1)$-ary relation exhibiting the graph of the function. Hence, extending SOL with function quantifiers does not add any expressive power.[8] The same also holds for function symbols; they just serve as a notational convenience in the set-theoretic setting.

However, the concept of functions in type theory does not directly match up with functions in set theory. The function space $D^n \to D$ is underspecified and therefore this correspondence to relations is not available without axioms. As a result, the semantics given above behaves slightly differently than one would normally expect. For example, we can no longer show that validity in all models of PA_2 agrees (cf. Lemma 4.5). That is because totality and surjectivity of $\cong$ does not transport to functions. We cannot show that for all $f_1 : (D_1)^n \to D_1$ there is a $f_2 : (D_2)^n \to D_2$ with $f_1 \cong f_2$ as $\cong$ is not necessarily computable.[9]

One approach to remedy this issue and to get the same properties as the set-theoretic semantics in the literature is to interpret functions in a different way:

Definition A.3. *Let $X \rightsquigarrow Y$ be the type of total and functional relations from X to Y:*

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

where

- $\mathsf{total}\, F := \forall x. \exists y. F\,x\,y,$
- $\mathsf{functional}\, F := \forall x y y'. F\,x\,y \to F\,x\,y' \to y = y'.$

Interpreting functions as $D^n \rightsquigarrow D$ is more faithful to the traditional set-theoretic approach and leads to the following relational semantics:

Definition A.4 (Relational Standard Semantics)**.** *The function interpretation and variable assignment in a relational Model $\mathcal{M}_R$ have types $\mathcal{F}^\mathcal{I} : D^{|\mathcal{F}|} \rightsquigarrow D$ and $\rho_{\mathsf{f}} : \mathbb{N} \to \forall n. D^n \rightsquigarrow D$. Term evaluation turns into a relation $[\![\cdot]\!]_\rho^R : \mathfrak{T}_2^F \to D \to \mathbb{P}$ and we get formula satisfaction $\rho \vDash^R \varphi$ with*

$$[\![\mathsf{x}_i]\!]_\rho^R\, d := \rho_i\, i = d$$

$$[\![\mathcal{F}\,\boldsymbol{t}]\!]_\rho^R\, d := \exists \boldsymbol{v} : D^{|\mathcal{F}|}.\, [\![\boldsymbol{t}]\!]_\rho^R\, \boldsymbol{v} \land \mathcal{F}^\mathcal{I}\, \boldsymbol{v}\, d$$

$$\rho \vDash^R \mathcal{P}\,\boldsymbol{t} := \exists \boldsymbol{v} : D^{|\mathcal{P}|}.\, [\![\boldsymbol{t}]\!]_\rho^R\, \boldsymbol{v} \land \mathcal{P}^\mathcal{I}\, \boldsymbol{v}$$

$$\rho \vDash^R \dot\forall_{\mathsf{f}}^n \varphi := \forall f : D^n \rightsquigarrow D.\, f \cdot \rho \vDash^R \varphi$$

The remaining cases are defined in a similar way.

Fact A.5. *The evaluation relation $[\![\cdot]\!]_\rho$ is total and functional.*

The main downside of the relational semantics is that it is quite tedious to work with and mechanise because of the overhead introduced by always needing to invoke totality and functionality. Thus, we have not restated the complete categoricity development of Section 4.1 in terms of the relational semantics. Instead, we verified that validity agrees if we interpret the conventional models in the relational semantics, reusing the isomorphism $\cong$:

Lemma A.6 (Agreement for Relational Semantics)**.** *We can turn models $\mathcal{M}_1$ and $\mathcal{M}_2$ of PA_2 into relational models $\mathcal{M}_1^R$ and $\mathcal{M}_2^R$. Let ρ_1^R and ρ_2^R be relational environments with $\rho_1^R \cong \rho_2^R$. Then $\mathcal{M}_1^R, \rho_1^R \vDash \varphi$ iff $\mathcal{M}_2^R, \rho_2^R \vDash \varphi$ for all $\varphi : \mathfrak{F}_2^F$.*

Proof. Similar to the proof of Lemma 4.5, using the fact that $\cong$ is surjective and total for $(D_i)^n \rightsquigarrow D_i$. □

One way to bridge the gap between the two semantics is to alter our meta-theory by assuming unique choice:

Definition A.7. $\mathsf{UC} := \forall XY : \mathbb{T}. \forall R : X \rightsquigarrow Y. \forall x. \Sigma y. R\,x\,y$

Note that this is not a propositional axiom, but instead a stronger $\mathbb{T}$-level operator for the totality of the relations.

Fact A.8. *Assuming UC, one can turn $F : X \rightsquigarrow Y$ into $\hat{F} : X \to Y$. Similarly, one can turn $f : X \to Y$ into $\tilde{f} : X \rightsquigarrow Y$ with $\tilde{f} := \lambda xy. f\,x = y$. Lifting those translations to models and environments yields the following results:*

1. *$\mathcal{M}, \rho \vDash \varphi$ iff $\tilde{\mathcal{M}}, \tilde{\rho} \vDash \varphi$ for all models $\mathcal{M}$ and environments ρ.*

[8] One could also get rid of predicates and in turn express them using their characteristic function (provided there are two distinct individuals). Thus, functions and relations can be used interchangeably.

[9] This problem does not arise in Section 5.3 since the embedding is first-order.

2. *$\mathcal{M}_R, \rho \vDash \varphi$ iff $\hat{\mathcal{M}}_R, \hat{\rho} \vDash \varphi$ for all relational models $\mathcal{M}_R$ and environments ρ.*

Proof. We show (1) by induction on φ. (2) follows from (1) and the fact that $\tilde{\hat{\mathcal{M}}}_R$ and $\tilde{\hat{\rho}}$ behave the same as $\mathcal{M}_R$ and ρ. □

Henkin semantics can be extend to function quantifiers in a similar way, requiring a notion of function comprehension. However, we have not explored this in the paper as Nour and Raffalli's reduction only considers predicate quantifiers and would be very difficult to adapt to $\mathfrak{F}_2^F$.

B Natural Deduction

Below is the full definition of the second-order ND system:

Definition B.1 (Natural Deduction). *We represent the deduction system as an inductive predicate $\vdash\, : \mathcal{L}(\mathfrak{F}_2) \to \mathfrak{F}_2 \to \mathbb{P}$. Its rules are given by*

$$\textsc{Ctx}\ \frac{\varphi \in \Gamma}{\Gamma \vdash \varphi} \qquad \textsc{Exp}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi} \qquad \mathit{II}\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \dot{\to} \psi}$$

$$\mathit{IE}\ \frac{\Gamma \vdash \varphi \dot{\to} \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi} \qquad \mathit{CI}\ \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \dot{\wedge} \psi}$$

$$\mathit{CE}_1\ \frac{\Gamma \vdash \varphi \dot{\wedge} \psi}{\Gamma \vdash \varphi} \qquad \mathit{CE}_2\ \frac{\Gamma \vdash \varphi \dot{\wedge} \psi}{\Gamma \vdash \psi}$$

$$\mathit{DI}_1\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \dot{\vee} \psi} \qquad \mathit{DI}_2\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \dot{\vee} \psi}$$

$$\mathit{DE}\ \frac{\Gamma \vdash \varphi \dot{\vee} \psi \qquad \Gamma, \varphi \vdash \theta \qquad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}$$

$$\textsc{AllI}\ \frac{\Gamma[\uparrow] \vdash \varphi}{\Gamma \vdash \dot{\forall}\varphi} \qquad \textsc{AllE}\ \frac{\Gamma \vdash \dot{\forall}\varphi}{\Gamma \vdash \varphi[t]}$$

$$\textsc{Ex}I\ \frac{\Gamma \vdash \varphi[t]}{\Gamma \vdash \dot{\exists}\varphi} \qquad \textsc{Ex}E\ \frac{\Gamma \vdash \dot{\exists}\psi \qquad \Gamma[\uparrow], \psi \vdash \varphi}{\Gamma \vdash \varphi}$$

$$\textsc{All}I_p\ \frac{\Gamma[\uparrow]_p^n \vdash \varphi}{\Gamma \vdash \dot{\forall}_p^n \varphi} \qquad \textsc{All}E_p\ \frac{\Gamma \vdash \dot{\forall}_p^n \varphi}{\Gamma \vdash \varphi[P]_p^n}$$

$$\textsc{Ex}I_p\ \frac{\Gamma \vdash \varphi[P]_p^n}{\Gamma \vdash \dot{\exists}_p^n \varphi} \qquad \textsc{Ex}E_p\ \frac{\Gamma \vdash \dot{\exists}_p^n \psi \qquad \Gamma[\uparrow]_p^n, \psi \vdash_2 \varphi}{\Gamma \vdash \varphi}$$

$$\textsc{Peirce}\ \frac{}{\Gamma \vdash_c ((\varphi \dot{\to} \psi) \dot{\to} \varphi) \dot{\to} \varphi}$$

$$\textsc{Compr}\ \frac{P \text{ not free in } \varphi}{\Gamma \vdash \dot{\exists} P.\, \dot{\forall} x_1 ... x_n.\, P(x_1, ..., x_n) \dot{\leftrightarrow} \varphi}$$

*We use two versions of the deduction system denoted by $\vdash_c$ and $\vdash_i$ distinguishing between classical and intuitionistic logic. The (*Peirce*) rule is only available in the classical system $\vdash_c$ and may not be used in $\vdash_i$. We write $\vdash$ if the choice between $\vdash_c$ and $\vdash_i$ does not matter.*

References

[1] Andrej Bauer. 2006. First Steps in Synthetic Computability Theory. *Electronic Notes in Theoretical Computer Science* 155 (2006), 5–31. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

[2] Chad E. Brown. 2014. A semantics for intuitionistic higher-order logic supporting higher-order abstract syntax. (2014).

[3] Marco Caminati. 2010. Basic first-order model theory in Mizar. *Journal of Formalized Reasoning* 3 (01 2010).

[4] Thierry Coquand and Gérard Huet. 1986. *The calculus of constructions.* Ph.D. Dissertation. INRIA.

[5] Thierry Coquand and Bassel Mannaa. 2016. The Independence of Markov's Principle in Type Theory. (02 2016).

[6] Nicolaas G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)* 75, 5 (1972), 381–392.

[7] Richard Dedekind. 1888. *Was sind und was sollen die Zahlen?* Vieweg, Braunschweig. https://doi.org/10.24355/dbbs.084-200902200100-1

[8] 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* (Cascais, Portugal) *(CPP 2019)*. Association for Computing Machinery, New York, NY, USA, 38–51.

[9] Yannick Forster, Dominik Kirst, and Dominik Wehr. 2021. Completeness theorems for first-order logic analysed in constructive type theory: Extended version. *Journal of Logic and Computation* 31, 1 (01 2021), 112–151.

[10] Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. 2020. A Coq library of undecidable problems. In *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages.*

[11] Kurt Gödel. 1929. *Über die Vollständigkeit des Logikkalküls.*

[12] Kurt Gödel. 1931. Über Formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik* 38, 1 (1931), 173–198.

[13] John Harrison. 1998. Formalizing basic first order model theory. In *Theorem Proving in Higher Order Logics*, Jim Grundy and Malcolm Newey (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 153–170.

[14] John Harrison. 2006. Towards self-verification of HOL Light. In *Proceedings of the third International Joint Conference, IJCAR 2006 (Lecture Notes in Computer Science, Vol. 4130)*, Ulrich Furbach and Natarajan Shankar (Eds.). Springer-Verlag, Seattle, WA, 177–191.

[15] Leon Henkin. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15, 2 (1950), 81–91.

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

[17] Hugo Herberlin, Sunyoung Kim, and Gyesik Lee. 2017. Formalizing the meta-theory of first-order predicate logic. *Journal of the Korean Mathematical Society* 54 (01 2017), 1521–1536.

[18] Johannes Hostert, Mark Koch, and Dominik Kirst. 2021. A Toolbox for Mechanised First-Order Logic. *The Coq Workshop* (2021).

[19] Asaf Karagila. 2014. *Downward Löwenheim-Skolem theorems and choice principles.* Technical Report. http://karagila.org/wp-content/uploads/2012/10/Lowenheim-Skolem-and-Choice.pdf

[20] Dominik Kirst and Marc Hermes. 2021. Synthetic Undecidability and Incompleteness of First-Order Axiom Systems in Coq. In *ITP.*

[21] Dominik Kirst and Dominique Larchey-Wendling. 2020. Trakhtenbrot's Theorem in Coq: A Constructive Approach to Finite Model Theory. *International Joint Conference on Automated Reasoning* (2020).

[22] Dominik Kirst and Gert Smolka. 2017. Categoricity Results for Second-Order ZF in Dependent Type Theory. In *ITP*.

[23] Dominik Kirst and Gert Smolka. 2018. Categoricity Results and Large Model Constructions for Second-Order ZF in Dependent Type Theory. *Journal of Automated Reasoning* 63 (2018), 415–438.

[24] Georg Kreisel. 1962. On weak completeness of intuitionistic predicate logic. *The Journal of Symbolic Logic* 27, 2 (1962), 139–158.

[25] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. 2016. Self-formalisation of higher-order logic. *Journal of Automated Reasoning* 56, 3 (2016), 221–259.

[26] Dominique Larchey-Wendling and Yannick Forster. 2019. Hilbert's Tenth Problem in Coq. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 131)*, Herman Geuvers (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 27:1–27:20. https://doi.org/10.4230/LIPIcs.FSCD.2019.27

[27] Maria Manzano. 1996. *Extensions of first-order logic*. Cambridge Tracts in Theoretical Computer Science, Vol. 19. Cambridge University Press.

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

[29] Karim Nour and Christophe Raffalli. 2003. Simple proof of the completeness theorem for second-order classical and intuitionistic logic by reduction to first-order mono-sorted logic. *Theoretical computer science* 308, 1-3 (2003), 227–237.

[30] Christine Paulin-Mohring. 1993. Inductive definitions in the system Coq rules and properties. In *Typed Lambda Calculi and Applications*, Marc Bezem and Jan Friso Groote (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 328–345.

[31] Stephen Read. 1997. Completeness and categoricity: Frege, Gödel and model theory. *History and Philosophy of Logic* 18, 2 (1997), 79–93.

[32] Fred Richman. 1983. Church's Thesis Without Tears. *The Journal of Symbolic Logic* 48, 3 (1983), 797–803.

[33] Stewart Shapiro. 1991. *Foundations without foundationalism: A case for second-order logic*. Vol. 17. Clarendon Press.

[34] Stephen G. Simpson. 2009. *Subsystems of Second Order Arithmetic* (2 ed.). Cambridge University Press.

[35] Kathrin Stark, Steven Schäfer, and Jonas Kaiser. 2019. Autosubst 2: Reasoning with Multi-Sorted de Bruijn Terms and Vector Substitutions *(CPP 2019)*. Association for Computing Machinery, New York, NY, USA, 166–180.

[36] Neil Tennant. 1990. *Natural logic*. Edinburgh University Press.

[37] The Coq Development Team. 2021. *The Coq Proof Assistant*. https://doi.org/10.5281/zenodo.4501022

[38] Jouko Väänänen. 2001. Second-order logic and foundations of mathematics. *Bulletin of Symbolic Logic* 7, 4 (2001), 504–520.

[39] Dirk Van Dalen. 1994. *Logic and structure*. Vol. 3. Springer.

[40] Luiz Viana. 2020. Proving the consistency of Logic in Lean. 1–8.

[41] Jouko Väänänen. 2015. Second-Order Logic and Set Theory. *Philosophy Compass* 10, 7 (2015), 463–478.

[42] Jouko Väänänen. 2021. Second-order and Higher-order Logic. In *The Stanford Encyclopedia of Philosophy* (Fall 2021 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[43] Jouko Väänänen and Tong Wang. 2012. Internal Categoricity in Arithmetic and Set Theory. *Notre Dame Journal of Formal Logic* 56 (01 2012).

[44] Ernest Zermelo. 1930. Über Grenzzahlen und Mengenbereiche. *Fundamenta Mathematicae* 16, 1 (1930), 29–47. http://eudml.org/doc/212506