Saarland University
Faculty of Mathematics and Computer Science

Gödel's Theorem Without Tears

Essential Incompleteness in Synthetic Computability

Bachelor's Thesis

Author
Benjamin Peters

Supervisor
Prof. Dr. Gert Smolka

Advisor
Dominik Kirst

Reviewers
Prof. Dr. Gert Smolka
Prof. Dr. Markus Bläser

Submitted: 13th June 2022

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 13$^{\text{th}}$ June, 2022

Abstract

Gödel published his groundbreaking first incompleteness theorem in 1931, stating that a large class of formal logics admits independent sentences which are neither provable nor refutable. This result, in conjunction with his second incompleteness theorem, established the impossibility of resolving Hilbert's program, which proposed a possible path towards a single formal system unifying all of mathematics. Gödel's incompleteness result was strengthened further by Rosser in 1936 regarding the conditions imposed on the formal systems.

Computability theory, which also originated in the 1930s, was quickly applied to formal logics by Turing, Kleene, and others to yield incompleteness results similar in strength to Gödel's original theorem, but weaker than Rosser's strengthening. These proofs have become folklore in computer science. Kleene later found a stronger proof of incompleteness using computability theory, yielding an incompleteness result as strong as Rosser's, which is, however, much lesser-known than the folklore proof.

In this thesis, we work in constructive type theory to reformulate Kleene's incompleteness results abstractly in the setting of synthetic computability theory, assuming a form of Church's thesis which internalises the fact that all functions in such a setting are computable. This extremely succinct reformulation showcases the simplicity of the computational argument while staying formally entirely precise, a combination hard to achieve in typical textbook presentations. As an application, we instantiate the abstract result to first-order logic to derive essential incompleteness of Robinson arithmetic.

This thesis is accompanied by a Coq mechanisation including all mentioned results and based on existing libraries of undecidability proofs and first-order logic.

Acknowledgements

First and foremost, I want to thank my advisor, Dominik Kirst, for his advice and support over the course of this project. I am immensely thankful for him giving me the opportunity to work on this exciting project, and providing me with a perfect balance between independence and additional support, as well as numerous hours of fruitful discussions and helpful feedback.

I also would like to thank Professor Smolka for allowing me to write my Bachelor's thesis at his group, as well as for introducing me to computational logic through his lectures.

I thank Marc Hermes for being a great person to share an office with, and him as well as Johannes Hostert for their repeated support with Coq and first-order logic.

In addition, I thank Arthur Correnson, Marc Hermes, Nico Mansion, and Niklas Mück for proof-reading this thesis.

Finally, I want to express my gratitude towards Professor Smolka and Professor Bläser for reviewing this thesis.

Contents

1 Introduction

In 1931, Gödel [17] published his seminal first incompleteness theorem, stating that the *Principia Mathematica* (PM), an early formal logic by Whitehead and Russel attempting to unify the foundations of mathematics [35], and a related class of formal logics are incomplete. In particular, he showed how to construct sentences that are neither provable nor refutable in these formal logics and all their effective and sound[1] extensions, that is, extensions that only show true sentences and have enumerable provability. Such sentences are called independent.

Gödel also presents his second incompleteness theorem in the same paper, stating that those formal logics additionally cannot show their own consistency. These results disrupted the mathematical community by showing that there cannot be a single unifying foundation for all of mathematics, which was commonly assumed up until that point. In fact, Gödel discovered both incompleteness theorems while attempting to contribute to *Hilbert's program*, which proposed a path towards such a unifying foundation, after earlier attempts suffered from paradoxes and inconsistencies [69]. Over time, however, dealing with formal systems of different strengths became an important part of investigating the foundations of mathematics, averting the foundational crisis feared by many mathematicians in the 1930s.

Gödel's incompleteness results have been (and are still) interpreted in philosophy in a wide variety of ways [49] and have often been exposed to a general audience through popular-scientific interpretations.[2] Gödel's actual incompleteness proof, however, is both technical and difficult to understand intuitively, and has been misinterpreted in popular culture, by philosophers, and mathematicians [16].

A modern computer scientist's view on incompleteness is typically completely different from the one gained through Gödel's proof. Incompleteness of effective, sound, and powerful enough formal logics, such as PM, can be regarded as following directly from the undecidability of the halting problem. Kleene published the idea to apply computability theory to show incompleteness in 1936 [26] and later continued working on this approach [27]. A similar proof idea was also mentioned by Turing in 1936 [64] in his seminal work on the undecidability of the *Entscheidungsproblem*. Post claimed to have discovered similar abstract incompleteness results in 1921, even before computability theory was developed in the 1930s, although they were only published in 1941 [48].[3] The core ideas of Kleene's proofs are easy to explain compactly and are easy to understand just using one of the most rudimentary results from computability theory, the undecidability of the halting problem. Today his proof can essentially be regarded as folklore.

[1] Gödel also showed this for ω-consistent extensions, a weaker, less semantic assumption.

[2] See, for example, `https://youtu.be/HeQX2HjkcNo`, as well as [21, 58].

[3] See also [59].

In 1936, a few years after Gödel published his results, Rosser [53] found a way, known today as *Rosser's trick*, to enhance Gödel's proof to yield incompleteness even of consistent but unsound and effective extensions, that is, extensions that cannot derive an internal contradiction, but may still show false sentences, and have enumerable provability.

Kleene's early result is close in strength to Gödel's, and therefore weaker than the one obtained by Rosser. Additionally, Kleene's folklore proof does not explicitly construct an independent sentence. Unfortunately, Rosser's modification of Gödel's proof cannot directly be applied to Kleene's proof.

Nevertheless, Kleene later improved upon his approach in 1951 [28], showing results that can be considered even more general than the one gained from the Gödel-Rosser proof. Kleene's strengthened result only requires slightly more computability theory and yields the same form of incompleteness for PM as the Gödel-Rosser proof. Interestingly, to instantiate Kleene's strengthened result to first-order logic just using the assumptions from the weaker proof, Rosser's trick can be applied. Kleene's improved approach to incompleteness is much lesser-known than one might expect, despite its much more abstract (and therefore general) approach.[4] Kleene features both proofs prominently in his books [29, 30].

Actually formalising Gödel's first incompleteness theorem completely is hard. Both showing that the formal logic in question can represent its own provability and using this fact to obtain an independent sentence is tedious. Mechanisations of Gödel's first incompleteness theorem have long been used to benchmark both the power of certain proof assistants or theorem provers, as well as computer-assisted proofs in general [54, 42, 19, 45, 46, 47, 24]. Most mechanisations we know of are based on the Gödel-Rosser approach.

Even when approaching incompleteness using computability theory, however, formalisation and particularly mechanisation remain difficult, since one still has to deal with the details of a concrete model of computation, such as Turing machines or μ-recursive functions. Using a shortcut via *synthetic computability theory* [50, 3, 12], pioneered by Richman in his seminal paper "Church's Thesis Without Tears" [50], Kirst and Hermes [24] recently formalised Kleene's folklore proof of incompleteness without these difficulties. By using the calculus of inductive constructions (CIC) [6, 44] as their meta-logic, underlying the Coq proof assistant [62], arguments related to computability can be simplified greatly. In particular, in a constructive logic such as CIC, only computable functions can be defined. Therefore quantifiers can be interpreted as only ranging over computable functions. This makes it possible to define properties like decidability and enumerability synthetically, that is, without referring to a specific model of computation.

Additionally, in CIC we can assume axioms such as Church's thesis [33, 63, 9, 11], internalising the notion that all functions are computable, allowing us to explore even more computability theory without referring to a model of computation. We assume Church's thesis to allow us to formalise and mechanise Kleene's form of incompleteness

[4] In fact, we were made aware of these results by an e-mail by Anatoly Vorobey on the Foundations of Mathematics mailing list [67] asking for references to these proofs, referencing both a blog post by Scott Aaronson [1] as well as an anonymous Stackexchange reply [65], all of which mentioned how little known this result is.

in different strengths in synthetic computability, culminating in essential incompleteness both of abstract formal systems and of first-order arithmetic.

1.1 Contributions

This thesis's contributions consist of four main parts:

- We re-examine Kleene's incompleteness results from a modern perspective by interpreting them abstractly using synthetic computability theory, while additionally obtaining analogous undecidability results, improving upon and extending the results by Kirst and Hermes [24]. In particular, we attempt to give an intuitive but precise reformulation of Kleene's proofs using synthetic computability theory.
- We instantiate these results to a mechanised representation of first-order logic with the axiomatisation of Robinson's Q by using Rosser's trick, building upon existing work on the DPRM theorem by Larchey-Wendling and Forster [34] to obtain the required representability assumptions.
- By applying Rosser's trick in a more general setting we obtain a form of Church's thesis for Robinson arithmetic under the assumption of Church's thesis for μ-recursive functions, as well as other representability properties.
- We give a full mechanisation of the incompleteness results using the Coq proof assistant, assuming different forms of Church's thesis. In particular, we are able to mechanise the abstract incompleteness proof in its strongest form in only around 150 lines of code, since the synthetic approach abstracts away the tedious parts of the proof, such as Gödelisations and computability proofs. All mechanised theorems are linked with the digital version of this thesis.

1.2 Outline

In Chapter 2, we first introduce the basics of the calculus of inductive constructions (CIC), and give some preliminary definitions and proofs in synthetic computability theory. Then we formalise an abstract version of Kleene's approach to incompleteness in different strengths in Chapter 3. After introducing a formalisation of first-order logic in CIC in Chapter 4, we instantiate the abstract incompleteness results to first-order logic with the axiomatisation of Robinson arithmetic Q in Chapter 5 and show additional representability theorems for Q in Chapter 6. We conclude this thesis in Chapter 7 by discussing the mechanisation as well as related and future work.

2 Computational Type Theory

The results in this thesis are formalized and largely mechanised in the framework of the calculus of inductive constructions (CIC) [6, 44] as implemented by the Coq proof assistant [62]. CIC is a constructive type theory and features both an impredicative universe of propositions as well as a countably infinite hierarchy of computational types universes.

We begin by outlining the basics of CIC. We then introduce synthetic computability and, in particular, Church's thesis, and show some basic results from computability theory. Finally, we introduce μ-recursive functions as a model of computation as well as their variant of Church's thesis.

2.1 Constructive Type Theory

We write $x : X$ to denote that x is of type X. CIC distinguishes between a hierarchy of predicative type universes $\mathbb{T}_1 : \mathbb{T}_2 : \dots$ and an impredicative universe of propositions $\mathbb{P} \subset \mathbb{T}_1$. We will omit the indices of type universes for readability.

CIC supports defining types in $\mathbb{T}$ and $\mathbb{P}$ inductively. While inductive types in $\mathbb{T}$ may contain computational information, this does not hold for inductive types in $\mathbb{P}$. In particular, elimination of inductively constructed values from $\mathbb{P}$ into $\mathbb{T}$ is only allowed in heavily restricted cases, preventing us from extracting any information from proofs.

Dependent function types $\forall x : X.\, U$, where U may refer to x, are primitive in CIC. Non-dependent function types $A \to B$ are defined using dependent function types $\forall x : X.\, B$, where x does not occur in B. We write $\lambda x : X.\, v$ to denote a function of type $\forall x : X.\, U$ (or its non-dependent counterpart). Such functions can be defined by strict structural recursion, which guarantees that all functions terminate.

We give definitions of some basic inductive types used throughout this thesis and the accompanying mechanisation.

- The type of natural numbers:

  $$\mathbb{N} : \mathbb{T} \ ::= 0 : \mathbb{N} \mid S : \mathbb{N} \to \mathbb{N}$$

  We write 1 for $S\,0$, 2 for $S\,S\,0$, and so on. Addition $+$, subtraction $-$, and multiplication $\cdot$ are defined recursively.

- The type of Booleans:

  $$\mathbb{B} : \mathbb{T} \ ::= \mathit{tt} : \mathbb{B} \mid \mathit{ff} : \mathbb{B}$$

  We write $!b$ for Boolean negation.

- The type of pairs:

$$\mathrm{product}(X, Y : \mathbb{T}) : \mathbb{T} \ ::= \mathrm{pair} : X \to Y \to \mathrm{product}\, X\, Y$$

  We write $X \times Y$ for $\mathrm{product}\, X\, Y$ and (x, y) for $\mathrm{pair}\, x\, y$.

- The type of dependent pairs:

$$\mathrm{sig}(X : \mathbb{T})(p : X \to \mathbb{T}) : \mathbb{T} \ ::= \mathrm{ex} : \forall(x : X).\, px \to \mathrm{sig}\, X\, p$$

  We write $\Sigma x.\, px$ for $\mathrm{sig}\, X\, p$ and $(x, y)_p$ for $\mathrm{sig}\, p\, x\, y$. Note that, given $p : X \to \mathbb{P}$, a dependent pair $\Sigma x.\, px$ can also be interpreted as a (computationally accessible) value x together with a proof of a property px.

- The option type:

$$\mathcal{O}(X : \mathbb{T}) : \mathbb{T} \ ::= \ \mathrm{Some} : X \to \mathcal{O}(X) \mid \mathrm{None} : \mathcal{O}(X)$$

  We write $\ulcorner x \urcorner$ for $\mathrm{Some}\, x$.

- The type of lists:

$$\mathcal{L}(X : \mathbb{T}) : \mathbb{T} \ ::= \ \mathrm{nil} : \mathcal{L}(X) \mid \mathrm{cons} : X \to \mathcal{L}(X) \to \mathcal{L}(X)$$

  We write $x :: L$ or L, x for $\mathrm{cons}\, x\, L$, and define a membership predicate $x \in L$ recursively.

- The type of vectors:

$$\mathcal{V}(X : \mathbb{T}, n : \mathbb{N}) : \mathbb{T} \ ::= \ \mathrm{nil} : \mathcal{V}X0 \mid \mathrm{cons} : X \to \mathcal{V}Xn \to \mathcal{V}X(Sn)$$

  We overload the notations for lists and use them for vectors as well.

Many typical logical operators and constants, that is, falsity $\bot$, truth $\top$, conjunction $\wedge$, disjunction $\vee$, and existentials $\exists x.\, px$, are defined inductively in $\mathbb{P}$. Negation is defined as $\neg P \ := \ P \to \bot$, and equivalence $A \leftrightarrow B$ is defined as $(A \to B) \wedge (B \to A)$. Universal quantification is represented by dependent function types, and implication is represented by non-dependent function types.

The logic induced by these operators is intuitionistic, that is, in particular, the law of excluded middle $\mathsf{LEM} := \forall(P : \mathbb{P}).\, P \vee \neg P$ is independent. It may, however, be assumed to obtain a classical logic.

We represent predicates on a type X as functions $X \to \mathbb{P}$. We also write $x \in P$ for Px. The complement of a predicate is defined by $\overline{P} := \lambda x.\, \neg Px$. Given predicates P_1 and P_2 on X, P_1 is a sub-predicate of P_2 if $P_1 \subseteq P_2 \ := \ \forall x.\, P_1\, x \ \to \ P_2\, x$, and P_1 and P_2 are disjoint if $\forall x.\, \neg(P_1\, x \wedge P_2\, x)$.

2.2 Synthetic Computability Theory

Without assuming certain additional axioms, all functions that can be defined in a constructive logic such as CIC are computable. Therefore all quantifiers ranging over

functions can be interpreted as only ranging over computable functions, which allows us to formulate concepts from computability theory without referring to a concrete model of computation. This makes formalising and especially mechanising results in computability theory much easier than in a typical textbook setting. This approach to computability theory is called synthetic computability theory [50, 3].

2.2.1 Basic Synthetic Notions

We succinctly define some rudimentary notions from computability theory in CIC, as presented in [12].

Definition 2.1. *A predicate $P : X \to \mathbb{P}$ is decidable if there is a function $f : X \to \mathbb{B}$, a decider of P, such that:*

$$\forall x.\, Px \;\leftrightarrow\; fx = tt$$

If there is no such function, P is undecidable. P is enumerable if there is a function $f : \mathbb{N} \to \mathcal{O}(X)$, an enumerator of P, such that:

$$\forall x.\, Px \;\leftrightarrow\; \exists k.\, fk = \ulcorner x \urcorner$$

P is co-enumerable if $\overline{P}$ is enumerable. A type $X : \mathbb{T}$ is enumerable if the predicate $\lambda(x : X).\, \top$ is enumerable.

Definition 2.2 (Discreteness). *A type X is discrete if the predicate $\lambda(x_1, x_2) : (X \times X).\, x_1 = x_2$ is decidable.*

Lemma 2.3. *Any decidable predicate $P : X \to \mathbb{P}$ on an enumerable type is both enumerable and co-enumerable.*

Proof. Let $f : X \to \mathbb{B}$ be a decider of P and let $g : \mathbb{N} \to \mathcal{O}(X)$ be an enumerator of P. Define functions $h_1, h_2 : \mathbb{N} \to \mathcal{O}(X)$ as follows:

$$h_1 n := \begin{cases} \ulcorner x \urcorner & \text{if } gn = \ulcorner x \urcorner \text{ and } fx = tt \\ \text{None} & \text{otherwise} \end{cases}$$

$$h_2 n := \begin{cases} \ulcorner x \urcorner & \text{if } gn = \ulcorner x \urcorner \text{ and } fx = ff \\ \text{None} & \text{otherwise} \end{cases}$$

Then h_1 is an enumerator and h_2 is a co-enumerator of P.

Note that analogous proofs for concrete models of computations, particularly for Turing machines, are often considerably harder, since defining functions in such models and showing their correctness formally tends to be very tedious.

2.2.2 Partial Functions

All functions we can define within our type theory are total. We do, however, also need to consider partial functions, which we represent using step-indices as a form of "fuel" for the computation.

Definition 2.4 (Partial functions). *A partial value $p : \operatorname{Part} Y$ is a step-indexed function $p : \mathbb{N} \to \mathcal{O}(Y)$ which is agnostic to the step-index, that is:*

$$\forall k_1 k_2 y_1 y_2.\, p\, k_1 = \ulcorner y_1 \urcorner \;\to\; p\, k_2 = \ulcorner y_2 \urcorner \;\to\; y_1 = y_2$$

We write $p \rhd y$ if $\exists k.\, p\, k = \ulcorner y \urcorner$. A partial function $f : X \rightharpoonup Y$ is a function $f : X \to \operatorname{Part} Y$. A partial function is total if $\forall x.\, \exists y.\, fx \rhd y$.

Remark 2.5. *Partial values can also be defined to be monotonic with respect to the step-index, that is:*

$$\forall k_1 k_2 y.\, p\, k_1 = \ulcorner y \urcorner \;\to\; k_2 \geq k_1 \;\to\; p\, k_2 = \ulcorner y \urcorner$$

It is easy to show that any monotonic partial value is agnostic. The converse direction is not generally true. Nevertheless, given an agnostic partial value $p : \operatorname{Part} Y$ it is possible to find a monotonic partial value $p' : \operatorname{Part} Y$ such that $\forall y.\, p \rhd y \;\leftrightarrow\; p' \rhd y$.

It is not obvious how to obtain a function from a total partial function, that is, a partial function that is additionally total, since it is not generally possible to extract a computable witness from existentiality proofs. In particular, we cannot eliminate a proposition, such as $\exists y.\, fx \rhd y$, into a type, such as $\Sigma x.\, fx \rhd y$. In CIC, however, it is possible to work around this for decidable predicates on enumerable types by doing a form of bounded search.

Lemma 2.6. *Let $P : X \to \mathbb{P}$ be a decidable predicate on an enumerable type X. Then $\exists x.\, Px$ implies $\Sigma x.\, Px$.*

Proof. See [37].

In general, enumerability of the predicate suffices.

Lemma 2.7. *Let $X, Y : \mathbb{T}$ and $f : X \rightharpoonup Y$ be a total partial function. There is a function $g : X \to Y$ such that:*

$$\forall xy.\, fx \rhd y \;\leftrightarrow\; gx = y$$

Proof. Let $p : \operatorname{Part} Y$ such that $\exists y.\, p \rhd y$. It suffices to show $\Sigma y.\, p \rhd y$. We have

$$\exists k.\, \exists y.\, pk = \ulcorner y \urcorner,$$

to which we can apply Lemma 2.6, since $\exists y.\, pk = \ulcorner y \urcorner$ is decidable.

Using this result we will, from now on, identify partial total partial functions with functions.

2.2.3 Church's Thesis

While synthetic computability suffices to formalize some aspects of computability theory, it is not powerful enough to, for example, give common negative results, such as undecidability of the halting problem. This is because it is consistent to assume certain

non-constructive axioms, such as the axiom of choice [68, 9], in CIC, which contradicts such undecidability results.

To work around these limitations, we assume different variations of the axiom of "Church's thesis" [33, 63], internalising the fact that all functions are computable by stating that some function is universal for all functions of a certain type. This universal function can either be abstract (that is, existentially quantified) or be an interpreter of an at least Turing-complete model of computation. While this form of Church's thesis is compatible with LEM (and therefore with classical reasoning) because of the split between impredicative and predicative universes, some consistent axioms, such as the axiom of choice, destroy the computational interpretation of functions and are therefore incompatible with Church's thesis.

Definition 2.8 (Enumerability of partial functions (EPF)). *Let X be a type. A function $\theta : \mathbb{N} \to (\mathbb{N} \rightharpoonup X)$ is universal for all partial functions $\mathbb{N} \rightharpoonup X$ if:*

$$\forall f : \mathbb{N} \rightharpoonup X.\, \exists c.\, \forall xy.\, fx \rhd y \;\leftrightarrow\; \theta cx \rhd y$$

We define the axiom "enumerability of partial functions", written EPF_X, as follows: There exists[5] a function θ that is universal for all partial functions to X. We only consider $\mathsf{EPF}_{\mathbb{N}}$ and $\mathsf{EPF}_{\mathbb{B}}$ in this thesis.

Given a partial function $f : \mathbb{N} \rightharpoonup X$, we call the witness obtained by EPF_X the code of f.

This form of Church's thesis is due to Richman [50] and was applied to CIC by Forster [11, 10], who also gives arguments for its consistency.

Lemma 2.9. $\mathsf{EPF}_{\mathbb{N}}$ *implies* $\mathsf{EPF}_{\mathbb{B}}$.

Proof. A partial function $f : \mathbb{N} \rightharpoonup \mathbb{B}$ can be easily represented as a function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ by choosing an invertible embedding from $\mathbb{B}$ to $\mathbb{N}$.

The converse also holds [9]. It is, however, more difficult to show.

Assuming $\mathsf{EPF}_{\mathbb{B}}$, we can now show that the halting problem for θ is undecidable. In particular, our proof is informative: Given any (not necessarily total) partial function agreeing with the halting problem whenever it halts, we construct an input on which it diverges.

Definition 2.10 (Halting problem). *Assume $\mathsf{EPF}_{\mathbb{B}}$. The self-halting problem (for $\mathbb{B}$) is defined as*

$$\mathsf{H} := \lambda x.\, \exists y.\, \theta xx \rhd y.$$

Lemma 2.11. *Assume $\mathsf{EPF}_{\mathbb{B}}$. Let $f : \mathbb{N} \rightharpoonup \mathbb{B}$ be a partial function such that:*

$$\forall x.\, fx \rhd tt \;\leftrightarrow\; x \in \mathsf{H}$$

There is some input x such that $\forall b.\, fx \not\rhd b$.

[5] This existential quantifier can be represented using either $\exists$ or Σ. Both variants are consistent, although the former suffices for our purposes since all of our main results are propositions.

Proof. Choose

$$g : \mathbb{N} \rightharpoonup \mathbb{B}, gx := \begin{cases} tt & \text{if } fx \rhd ff \\ \text{undefined} & \text{if } fx \rhd tt \text{ or } fx \text{ diverges} \end{cases}$$

and let c be the code of g. We have

$$fc \rhd ff \;\leftrightarrow\; gc \rhd tt \;\leftrightarrow\; (\exists y.\, gc \rhd y) \;\leftrightarrow\; (\exists y.\, \theta cc \rhd y) \;\leftrightarrow\; c \in \mathsf{H} \;\leftrightarrow\; fc \rhd tt$$

Therefore $\forall b.\, fc \not\rhd b$.

Corollary 2.12 (H is undecidable). *Assuming* $\mathsf{EPF}_{\mathbb{B}}$, H *is undecidable.*

Proof. Let $f : \mathbb{N} \to \mathbb{B}$ be a function deciding H. The induced total partial function $\overline{f} : \mathbb{N} \rightharpoonup \mathbb{B}$ obviously agrees with H and must therefore diverge on an input, which contradicts totality.

We will now consider another problem from computability theory: recursively inseparable sets, which were originally considered by Kleene [28]. We will follow the same approach as for the halting problem, giving an informative divergence proof to show the non-existence of a total function.

Definition 2.13. *A partial function* $f : \mathbb{N} \rightharpoonup \mathbb{B}$ *recursively separates two disjoint predicates* $P_1, P_2 : \mathbb{N} \to \mathbb{P}$ *if:*

$$P_1 x \;\to\; fx \rhd tt \qquad P_2 x \;\to\; fx \rhd ff$$

A total function $f : \mathbb{N} \to \mathbb{B}$ *recursively separates* P_1 *and* P_2 *if its induced partial function does so. If there is no total function recursively separating* P_1 *and* P_2*, they are recursively inseparable.*

Lemma 2.14. *Assume* $\mathsf{EPF}_{\mathbb{B}}$*. Given any function* $f : \mathbb{N} \rightharpoonup \mathbb{B}$ *that recursively separates* $P_1 := \lambda x.\, \theta xx \rhd tt$ *and* $P_2 := \lambda x.\, \theta xx \rhd ff$*, there is some* c *such that* $\forall b.\, fc \not\rhd b$*.*

Proof. Choose

$$g : \mathbb{N} \rightharpoonup \mathbb{N}, g\, x := \begin{cases} !fx & \text{if } fx \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

and let c be the code of g. We have

$$fc \rhd tt \;\leftrightarrow\; gc \rhd ff \;\leftrightarrow\; \theta cc \rhd ff \;\leftrightarrow\; P_2\, x \;\to\; fc \rhd ff$$

and

$$fc \rhd ff \;\leftrightarrow\; gc \rhd tt \;\leftrightarrow\; \theta cc \rhd tt \;\leftrightarrow\; P_1\, x \;\to\; fc \rhd tt$$

Therefore, $\forall b.\, fc \not\rhd b$.

Corollary 2.15. *Assuming* $\mathsf{EPF}_{\mathbb{B}}$*,* $P_1 := \lambda x.\, \theta xx \rhd tt$ *and* $P_2 := \lambda x.\, \theta xx \rhd ff$ *are recursively inseparable.*

To apply these results to a concrete model of computation, we assume an interpreter for this model to be universal for all partial functions. We use μ-recursive functions as our machine model, as described in [34]. However, any Turing-complete model of computation would suffice. Details on how μ-recursive functions are represented in CIC and implemented in Coq are not important for this thesis as we largely rely on existing results.

Definition 2.16. *Let $\theta^\mu : \mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{N}$ be an interpreter for μ-recursive functions encoded as natural numbers.*[6]

We define the axiom "EPF for μ-recursive functions", written $\mathsf{EPF}^\mu_{\mathbb{N}}$, as follows: The interpreter of μ-recursive functions θ^μ is universal for all partial functions.

Lemma 2.17. *Assuming $\mathsf{EPF}^\mu_{\mathbb{N}}$, if a predicate $P : \mathbb{N} \to \mathbb{P}$ is enumerable, it is also μ-enumerable, that is, there is some c such that:*

$$\forall x.\, Px \;\leftrightarrow\; \exists y.\, \theta^\mu\, cx \rhd y$$

The converse of this statement holds as well.

Lemma 2.18. $\mathsf{EPF}^\mu_{\mathbb{N}}$ *implies* $\mathsf{EPF}_{\mathbb{N}}$.

Note that our definition of enumerability by a μ-recursive function is closer to a notion of semi-decidability. Both notions are, however, equivalent, since $\mathbb{N}$ is enumerable.

[6] In the mechanisation, μ-recursive functions are represented as an indexed inductive type $\mathcal{R}$. To obtain suitable conversions between $\mathcal{R}$ and $\mathbb{N}$, we use Lemma 2.6 as well as enumerability and discreteness of $\mathcal{R}$, which were shown by Johannes Hostert. His mechanisation of these facts is distributed along with the mechanisation of the rest of our work.

3 Abstract and Synthetic Incompleteness

In this chapter, we give an intuitive but precise abstract reformulation of the abstract incompleteness proofs by Kleene, as he describes them in his books [29, 30]. To obtain these results without referring to a concrete model of computation we use synthetic computability theory and assume a form of Church's thesis.

Our abstract representation of formal systems attempts to be as simple as possible while still being able to capture the essence of Kleene's incompleteness proofs. In particular, we do not model semantic properties such as soundness (c.f. [24]), or any but the most fundamental syntactic properties (c.f. [47]).

Formalising Kleene's folklore incompleteness proof using the halting problem for this notion of formal system turns out to be easy, but only yields a weaker incompleteness result than the original Gödel-Rosser proof of incompleteness. In particular, it requires a representability property usually only fulfilled by sound formal logics as well as their sound extensions.

We additionally give another, later incompleteness proof also due to Kleene of essential incompleteness of certain formal systems, that is, incompleteness of all consistent extensions.

We first introduce a notion of abstract formal systems and show that provability is decidable in complete formal systems. We then present a proof of undecidability, incompleteness, and the existence of an independent sentence under the assumptions of weak representability of the halting problem and $\mathsf{EPF}_{\mathbb{B}}$, following Kleene's folklore proof.

Afterwards we give the strengthened versions of these proofs following Kleene's improved approach, yielding essential undecidability and essential incompleteness, as well as the existence of independent sentences in all consistent extensions. This approach assumes strong separability of two recursively inseparable predicates and $\mathsf{EPF}_{\mathbb{B}}$.

3.1 Abstract Formal Systems

We introduce an abstract notion of formal systems and show that in complete formal systems, provability is decidable.

Definition 3.1 (Formal systems). *A formal system* $\mathrm{FS} = (S, \neg, \vdash_{\mathrm{FS}})$ *consists of a type of logical sentences* $S : \mathbb{T}$*, a negation function* $\neg : S \to S$ *and a provability predicate* $\vdash_{\mathrm{FS}} : S \to \mathbb{P}$ *fulfilling the following properties:*

- S *is discrete.*
- $\vdash_{\mathrm{FS}}$ *is enumerable.*

- FS *is consistent:* $\forall s.\, \neg(\vdash_{\mathrm{FS}} s \ \wedge \ \vdash_{\mathrm{FS}} \neg s)$

We write $\mathrm{FS} \vdash s$ *for* $\vdash_{\mathrm{FS}} s$. FS *is complete if:*

$$\forall s.\, \mathrm{FS} \vdash s \ \vee \ \mathrm{FS} \vdash \neg s$$

A formal system that is not complete is incomplete. A formal system is decidable if its provability predicate is decidable. A sentence s is independent in FS *if:*

$$\mathrm{FS} \nvdash s \wedge \mathrm{FS} \nvdash \neg s$$

Typically, formal logics with a form of negation are formal systems in this sense. In particular, we show that a particular natural deduction system for first-order logic over any enumerable and consistent axiomatisation is a formal system in Chapter 4.

We do not consider ω-consistency or (variants of) soundness abstractly since they either require a notion of quantification or a notion of semantic truth, respectively. Both would complicate the definition of formal systems considerably and are not required for our main results, but could be used to generalise the undecidability and incompleteness proofs using the halting problem.

Definition 3.2 (Extensions). *Let* $\mathrm{FS} = (S, \neg, \vdash_{\mathrm{FS}})$ *and* $\mathrm{FS}' = (S, \neg, \vdash_{\mathrm{FS}'})$ *be two formal systems only differing in their provability predicates. We say that* FS' *is an extension of* FS *if*

$$\forall s.\, \mathrm{FS} \vdash s \ \rightarrow \ \mathrm{FS}' \vdash s.$$

A formal system of which all extensions are incomplete is essentially incomplete.

Note that if a sentence is independent in an extension of a formal system, it is also independent in the formal system itself.

Fact 3.3 (Decidability). *Any complete formal system is decidable.*

Proof. Let $f : S \rightharpoonup \mathbb{B}$ be a partial function, that, given a sentence s, enumerates all provable sentences, checks whether they are s or $\neg s$ and returns tt or ff respectively. Note that even without completeness, f fulfils:

$$\forall s.\, f\, s \rhd tt \ \leftrightarrow \ \mathrm{FS} \vdash s$$
$$\forall s.\, f\, s \rhd ff \ \leftrightarrow \ \mathrm{FS} \vdash \neg s$$

By completeness, f is total, and therefore decides provability (see Lemma 2.7).

3.2 Folklore Proof Using Soundness

We present the well-known folklore proof of incompleteness.

Definition 3.4 (Weak representability). *Let* $\mathrm{FS} = (S, \neg, \vdash)$ *be a formal system and* $P : X \to \mathbb{P}$ *be a predicate. A representation function* $r : X \to S$ *weakly represents* P *if*

$$\forall x.\, Px \ \leftrightarrow \ \mathrm{FS} \vdash rx.$$

In this case we say that FS *weakly represents* P *or that* P *is weakly representable in* FS.

Remark 3.5. *Note that even if a formal system represents P, its extensions do not have to, since a (consistent) formal system might still show false statements, that is, it might be unsound. Weak representability only preserves along sound extensions, which we cannot express using our abstract representation of formal systems.*

In particular, assume r weakly represents P in FS*. Let x be such that $\neg Px$ and therefore* $\mathrm{FS} \nvdash rx$*. An extension* FS' *of* FS *might show* $\mathrm{FS}' \vdash rx$*, and therefore r might not weakly represent P in* FS*.*

Even showing weak representability properties of formal systems is usually done using forms of soundness to show the direction from right to left.

A representation function r weakly representing a predicate P can be understood as a many-one reduction from P to provability, which motivates the following result:

Lemma 3.6 (Decidability). *A predicate weakly representable in a decidable formal system is decidable.*

3.2.1 Anonymous Incompleteness

The results shown until now trivially yield undecidability and incompleteness of formal systems weakly representing H.

Fact 3.7 (Undecidability). *Assuming* $\mathsf{EPF}_{\mathbb{B}}$*, any formal system that weakly represents the self-halting problem* H *is undecidable.*

Proof. By Lemma 3.6 and Corollary 2.12.

Fact 3.8 (Anonymous incompleteness). *Assuming* $\mathsf{EPF}_{\mathbb{B}}$*, any formal system that weakly represents the self-halting problem* H *is incomplete.*

Proof. By Lemma 3.6, Fact 3.3, and Corollary 2.12.

The incompleteness result shown by Kirst and Hermes [24] is very similar to this one, except that instead of showing that completeness implies falsity, they show that completeness yields a decider for the halting problem of a Turing complete model of computation, which does not yield falsity without assuming, for example, $\mathsf{EPF}^{\mu}_{\mathbb{N}}$. They do, however, consider an abstract notion of formal systems that incorporates soundness, and are therefore able to abstractly consider incompleteness up to sound extensions.

3.2.2 Informative Incompleteness

The most obvious way to strengthen Fact 3.8 is to explicitly construct an independent sentence. To do this, we use the informative version of the undecidability of the halting problem.

Theorem 3.9 (Informative incompleteness). *Assume* $\mathsf{EPF}_{\mathbb{B}}$*. Let* $\mathrm{FS} = (S, \neg, \vdash)$ *be a formal system and $r : \mathbb{N} \to S$ be a representation function that weakly represents the self-halting problem* H*. There is some c such that rc is independent in* FS*.*

Proof. Let f be the partial function $f : S \rightharpoonup \mathbb{B}$ constructed in the proof of Fact 3.3. Consider the function $g : \mathbb{N} \rightharpoonup \mathbb{B}$ defined by $gx := f(rx)$. It fulfils

$$\forall x.\, gx \rhd tt \;\leftrightarrow\; \mathsf{H}x.$$

By Lemma 2.11 there is an input c on which g diverges, and therefore $\mathrm{FS} \nvdash rc$ and $\mathrm{FS} \nvdash \neg rc$.

3.3 Strengthened Proof Using Consistency

As explained earlier, to obtain incompleteness for unsound but consistent formal systems we need a different form of representability.

Definition 3.10 (Strong separability). *Let* $\mathrm{FS} = (S, \neg, \vdash)$ *be a formal system,* $X : \mathbb{T}$*, and* $P_1, P_2 : X \to \mathbb{P}$*. A representation function* $r : X \to S$ *strongly separates* P_1 *and* P_2 *if*

$$\forall x.\, P_1\, x \;\to\; \mathrm{FS} \vdash rx \quad \wedge \quad P_2\, x \;\to\; \mathrm{FS} \vdash \neg rx.$$

In this case we say that FS *strongly separates* P_1 *and* P_2 *or that* P_1 *and* P_2 *are strongly separable in* FS*.*

As opposed to weak representability, strong separability is preserved along all extensions of formal systems. In particular, we do not pose any restrictions on the provability of rx if $\neg P_1\, x$ and $\neg P_2\, x$.

Lemma 3.11. *If a formal system strongly separates two predicates, all its extensions do as well.*

Weak representability and strong separability are otherwise difficult to compare. Strong separability is stronger in the sense that it gives us refutability and not just "positive" provability. It is, however, possible to show the following facts:

- If a complete formal system weakly represents a predicate P_1, it also strongly separates P_1 and P_2 if P_1 and P_2 are disjoint.
- If a formal system strongly separates a predicate P and its complement $\overline{P}$, it also weakly represents P.[7]

Assuming strong separability of two recursively inseparable predicates, we obtain undecidability and incompleteness.

Fact 3.12. *Assuming* $\mathsf{EPF}_{\mathbb{B}}$*, any formal system* $\mathrm{FS} = (S, \neg, \vdash)$ *that strongly separates*

$$P_1 := \lambda x.\, \theta xx \rhd tt \qquad P_2 := \lambda x.\, \theta xx \rhd ff$$

is undecidable.

[7] It even strongly represents P as defined in Section 6.2, which can be seen by generalising Lemma 6.2 to arbitrary formal systems.

Proof. Let r be the representation function strongly separating P_1 and P_2. Let $f : \mathbb{N} \to \mathbb{B}$ be the function that, given x, decides whether $\mathrm{FS} \vdash rx$. Now, f recursively separates P_1 and P_2, which contradicts Corollary 2.15.

Theorem 3.13. *Assume* $\mathsf{EPF}_{\mathbb{B}}$. *Let* $\mathrm{FS} = (S, \neg, \vdash)$ *be a formal system and* $r : \mathbb{N} \to S$ *be a representation function that strongly separates the following predicates:*

$$P_1 := \lambda x.\, \theta xx \rhd tt \qquad P_2 := \lambda x.\, \theta xx \rhd ff$$

There is some c *such that* rc *is independent in* FS.

Proof. Let f be the partial function $f : S \rightharpoonup \mathbb{B}$ constructed in the proof of Fact 3.3. Consider the function $g : \mathbb{N} \rightharpoonup \mathbb{B}$ defined by $gx := f(rx)$. It recursively separates P_1 and P_2. We can construct an input c on which it diverges by Lemma 2.14, and therefore $\mathrm{FS} \nvdash rc$ and $\mathrm{FS} \nvdash \neg rc$.

Corollary 3.14 (Essential incompleteness). *Assume* $\mathsf{EPF}_{\mathbb{B}}$. *Let* $\mathrm{FS} = (S, \neg, \vdash)$ *be a formal system and* $r : \mathbb{N} \to S$ *be a representation function that strongly separates the following predicates:*

$$P_1 := \lambda x.\, \theta xx \rhd tt \qquad P_2 := \lambda x.\, \theta xx \rhd ff$$

For any extension of FS *there is some* c *such that* rc *is independent in the extension, that is,* FS *is essentially incomplete. Any extension of* FS *is undecidable.*

3.4 Conclusion

There are multiple advantages of Kleene's over Gödel's approach to incompleteness:

- Kleene's results are more general, or rather, much easier to formulate abstractly.
- Kleene's result also yields essential undecidability as an intermediate step.
- Kleene's approach is easier to understand intuitively. Everything can be shown using only basic tools from computability theory.
- Kleene's results are much easier to formalize (and mechanize) abstractly without resorting to hand-waving computability or provability assumptions by working in synthetic computability.

However, Kleene's strengthened result does lose some elegance in comparison to the folklore proof, especially when viewed in conjunction with the instantiation and the proof of strong separability in Chapter 5.

4 First-Order Logic

In Chapter 3, we described an abstract approach to incompleteness of formal systems. Our next goal is to instantiate it to first-order logic [66], in particular to first-order logic over the axiomatisation of *Robinson arithmetic* [52]. Most of our definitions of first-order logic and related notions are part of a larger effort to mechanise first-order logic in the Coq proof assistant [25, 24]. They are also part of the Coq library of undecidability proofs [15].

Most of the definitions presented here are standard and can be skipped by a reader familiar with first-order logic. The embedding into constructive type theory is natural.

We first define syntax, semantics, and syntactic provability of first-order logic and show the soundness of our deduction system. We then define the theories of *Robinson arithmetic* (or *Robinson's* Q) as well as *Heyting* and *Peano arithmetic*, and show them sound with respect to the standard model of natural numbers. We then give definitions for the first level of the arithmetical hierarchy and show some of their properties, in particular Σ_1-completeness. We define the axiom of completeness for first-order logic and show some facts for working with it. Finally, we instantiate the abstract formal systems from Chapter 3 to first-order logic.

4.1 Syntax

We represent formulas and terms as inductive types. Typically, the syntax of first-order logic is parametrized over a signature, that is, two finite types of function and predicates symbols respectively, as well as their arities. Our definition immediately instantiates it to the signature of Peano arithmetic. Its function symbols are $0, \sigma, +$, and $\cdot$ with arities of $0, 1, 2$, and 2 respectively, and its only predicate symbol is $=$ with arity 2.

Definition 4.1 (Syntax). *Let $\mathcal{V} := \mathbb{N}$ be the type of variables. The types of terms $\mathcal{T}$ and formulas $\mathcal{F}$ are defined inductively by:*

$$\begin{array}{rcll} t, s : \mathcal{T} & ::= & x \mid 0 \mid \sigma t \mid t + s \mid t \cdot s & x \in \mathcal{V} \\ \varphi, \psi : \mathcal{F} & ::= & \bot \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid \forall x.\, \varphi \mid \exists x.\, \varphi \mid t = s & x \in \mathcal{V} \end{array}$$

We define the following derived notions:

$$\begin{array}{rcl} \neg\varphi & := & \varphi \to \bot \\ \varphi \leftrightarrow \psi & := & (\varphi \to \psi) \wedge (\psi \to \varphi) \end{array}$$

We also define an embedding of natural numbers into terms:

$$\begin{aligned} \overline{\cdot} &: \mathbb{N} \to \mathcal{T} \\ \overline{0} &:= 0 \\ \overline{Sn} &:= \sigma\overline{n} \end{aligned}$$

Terms of the form $\overline{n}$ for some n are called numerals.

Definition 4.2. *A variable x is bound in a formula φ if it only occurs in subexpressions of φ of the form $\exists x.\, \psi$ or $\forall x.\, \psi$. If x is not bound in φ, we say x occurs freely in x. A formula is closed if it only contains bound variables.*

On paper, we assume that all bound and free variables are pairwise different in any mathematical context, that is, a definition, proof, etc. This is also known as the Barendregt convention [2]. If a formula violates the Barendregt convention, its bound variables can be consistently renamed such that it does. This convention greatly simplifies working with quantifiers and substitutions.

Since it is not clear how to assume the Barendregt convention when mechanising, we represent variables using de Bruijn indices [8] in that case. While allowing us to deal with variables formally, it makes formulas much harder to read. We do not consider lemmas on substitutions in this thesis, even though they become crucial during mechanisation, particularly when dealing with statements on quantifiers.

Definition 4.3 (Environment). *An environment on T is a function $\rho : \mathcal{V} \to T$. We define updates $\rho[x \mapsto v]$ as follows:*

$$\begin{aligned} (\rho[x \mapsto v])\, y &:= v && \text{if } x = y \\ (\rho[x \mapsto v])\, y &:= \rho y && \text{if } x \neq y \end{aligned}$$

Definition 4.4. *Parallel substitution on formulas $\cdot[\cdot] : \mathcal{F} \to (\mathcal{V} \to \mathcal{T}) \to \mathcal{F}$ as well as terms $\cdot[\cdot] : \mathcal{T} \to (\mathcal{V} \to \mathcal{T}) \to \mathcal{T}$ is defined as follows:*

$$\begin{aligned} \bot[\rho] &:= \bot \\ (\varphi \wedge \psi)[\rho] &:= \varphi[\rho] \wedge \psi[\rho] \\ (\varphi \vee \psi)[\rho] &:= \varphi[\rho] \vee \psi[\rho] \\ (\varphi \to \psi)[\rho] &:= \varphi[\rho] \to \psi[\rho] \\ (\forall x.\, \varphi)[\rho] &:= \forall x.\, \varphi[\rho[x \mapsto x]] \\ (\exists x.\, \varphi)[\rho] &:= \exists x.\, \varphi[\rho[x \mapsto x]] \\ (t = s)[\rho] &:= t[\rho] = s[\rho] \end{aligned} \qquad \begin{aligned} 0[\rho] &:= 0 \\ (\sigma t)[\rho] &:= \sigma t[\rho] \\ (t + s)[\rho] &:= t[\rho] + s[\rho] \\ (t \cdot s)[\rho] &:= t[\rho] \cdot s[\rho] \end{aligned}$$

Single-point substitution is defined as $\varphi[x \mapsto t] := \varphi[\mathrm{id}[x \mapsto t]]$. Given a formula φ with one free variable x or two free variables x, y, respectively, we write $\varphi(a) := \varphi[x \mapsto b]$ or $\varphi(a, b) := \varphi[x \mapsto a][y \mapsto b]$ for substituting in terms a or a and b. We define $t(a)$ for terms t analogously.

4.2 Semantics

We use standard Tarski semantics for first-order logic [61]. This means that every formula is assigned a meaning in $\mathbb{P}$ by interpreting the logical connectives as their meta-level counterparts in type theory, and equality as well as terms using a so-called model.

Definition 4.5 (Model). *A model M consists of a carrier type D and interpretations of the function and predicate symbols:*

$$\begin{aligned} 0_M &: D \\ \sigma_M &: D \to D \\ +_M, \cdot_M &: D \to D \to D \\ =_M &: D \to D \to \mathbb{P} \end{aligned}$$

We will use M to refer to the carrier D as well as the model itself.

Definition 4.6 (Axiomatisation). *An axiomatisation $T : \mathcal{F} \to \mathbb{P}$ is a predicate on formulas. It is enumerable if T is enumerable.*

Definition 4.7 (Tarski semantics). *A model M satisfies a formula φ in an environment $\rho : \mathcal{V} \to M$ if $M \vDash_\rho \varphi$ with $\vDash$ defined inductively by:*

$$\begin{aligned} M \vDash_\rho \bot &:= \bot \\ M \vDash_\rho \varphi \wedge \psi &:= (M \vDash_\rho \varphi) \wedge (M \vDash_\rho \psi) \\ M \vDash_\rho \varphi \vee \psi &:= (M \vDash_\rho \varphi) \vee (M \vDash_\rho \psi) \\ M \vDash_\rho \varphi \to \psi &:= (M \vDash_\rho \varphi) \to (M \vDash_\rho \psi) \\ M \vDash_\rho \exists x.\, \varphi &:= \exists y.\, M \vDash_{\rho[x \mapsto y]} \varphi \\ M \vDash_\rho \forall x.\, \varphi &:= \forall y.\, M \vDash_{\rho[x \mapsto y]} \varphi \\ M \vDash_\rho t = s &:= [\![t]\!]_\rho =_M [\![s]\!]_\rho \end{aligned} \qquad \begin{aligned} [\![x]\!]_\rho &:= \rho x \\ [\![0]\!]_\rho &:= 0_M \\ [\![\sigma t]\!]_\rho &:= \sigma_M [\![t]\!]_\rho \\ [\![t + s]\!]_\rho &:= [\![t]\!]_\rho +_M [\![s]\!]_\rho \\ [\![t \cdot s]\!]_\rho &:= [\![t]\!]_\rho \cdot_M [\![s]\!]_\rho \end{aligned}$$

Let T be an axiomatisation. We write:

$$\begin{aligned} M \vDash \varphi &:= \forall \rho.\, M \vDash_\rho \varphi \\ M \vDash T &:= \forall \varphi \in T.\, M \vDash \varphi \\ T \vDash \varphi &:= \forall M \vDash T.\, M \vDash \varphi \end{aligned}$$

A model is extensional if for any $x, y \in M$ we have $x =_M y \leftrightarrow x = y$.

Note that without additional assumptions $\vDash$ yields intuitionistic first-order semantics since our meta-logic is intuitionistic. We obtain classical semantics when assuming LEM.

Definition 4.8 (Standard model). *The standard model $\mathbb{N}$ has the type $\mathbb{N}$ as its carrier and the canonical meta-level interpretations of $0, \sigma, +, \cdot$, and $=$.*

Definition 4.9. *An axiomatisation T is sound (with respect to the standard model) if $\mathbb{N} \vDash T$.*

It is possible to differentiate between classically and intuitionistically sound axiomatisations. We will, however, only work with theories that are both classically and intuitionistically sound.

4.3 Natural Deduction

While Tarski semantics give us a notion of correctness of formulas, they do not give us a computationally useful notion of provability. We use a natural deduction calculus to fill this gap.

Definition 4.10 (Provability). *Provability* $\Gamma \vdash \varphi$ *for* $\Gamma : \mathcal{L}(\mathcal{F}), \varphi : \mathcal{F}$ *is defined inductively by:*

$$\frac{\varphi \in \Gamma}{\Gamma \vdash \varphi} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi} \qquad \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi} \qquad \frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \qquad \frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \chi \quad \Gamma, \psi \vdash \chi}{\Gamma \vdash \chi}$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\, \varphi} \qquad \frac{\Gamma \vdash \forall x.\, \varphi}{\Gamma \vdash \varphi[x \mapsto t]} \qquad \frac{\Gamma \vdash \varphi[x \mapsto t]}{\Gamma \vdash \exists x.\, \varphi} \qquad \frac{\Gamma \vdash \exists x.\, \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}$$

We do not spell out restrictions on variable occurrences for the rules on quantifiers, fully relying on the Barendregt convention. We can only do this since Γ *is a finite context. To work with potentially infinite axiomatisations* T *we define* $T \vDash \varphi$ *as* $\exists \Gamma.\, (\forall \varphi \in \Gamma.\, \varphi \in T) \wedge \Gamma \vDash \varphi$.

Lemma 4.11 (Soundness). *We have for any axiomatisation* T*:*

$$T \vdash \varphi \;\to\; T \vDash \varphi$$

Proof. By induction on the derivation of $T \vdash \varphi$.

Note that we use soundness to refer to a property of axiomatisations or to a property of this deduction system.

Definition 4.12 (Consistency). *An axiomatisation* T *is consistent if* $T \nvdash \bot$.

Definition 4.13 (ω-consistency). *An axiomatisation* T *is* ω*-consistent if for any formula* φ *we have either* $T \nvdash \exists k.\, \varphi(k)$ *or* $\exists x.\, T \nvdash \neg\varphi(\overline{x})$.

Note that any sound axiomatisation is ω-consistent, and that any ω-consistent axiomatisation is consistent.

Definition 4.14. *Let* T *be an axiomatisation. We use* T^c *to refer to the classical closure of the axiomatisation, that is,* T *with all instances of Peirce's law:*

$$\forall x_1, \ldots, x_n.\, ((\varphi \;\to\; \psi) \;\to\; \varphi) \;\to\; \varphi$$

The mechanisation treats classical provability by presenting two different versions of the deduction system, separated by a binary flag.

Lemma 4.15. *Let* T *be an axiomatisation. Assuming* LEM*, if* T *is sound,* T^c *is also sound.*

4.4 Robinson and Peano Arithmetic

The usual axiomatisation of natural numbers in first-order logic is Heyting arithmetic, or its classical version, Peano arithmetic. We mostly work with a simpler axiomatisation called Robinson arithmetic [52], which is finite but much weaker than Heyting arithmetic. In particular, it does not include induction.

Definition 4.16 (Robinson arithmetic). *The axiomatisation of Robinson arithmetic (or Robinson's* Q*) consists of the following axioms:*

$$
\begin{array}{ll}
(ER) & \forall x. \quad x = x \\
(ES) & \forall xy. \quad x = y \;\to\; y = x \\
(ET) & \forall xyz. \quad x = y \;\to\; y = z \;\to\; x = z \\
(CS) & \forall xy. \quad x = y \;\to\; \sigma x = \sigma y \\
(CA) & \forall xyzw. \, x = y \;\to\; z = w \;\to\; x + z = y + w \\
(CM) & \forall xyzw. \, x = y \;\to\; z = w \;\to\; x \cdot z = y \cdot w
\end{array}
\qquad
\begin{array}{ll}
(AZ) & \forall x. \quad 0 + x = x \\
(AR) & \forall xy. \, (\sigma x) + y = \sigma(x + y) \\
(MZ) & \forall x. \quad 0 \cdot x = 0 \\
(MR) & \forall xy. \, (\sigma x) \cdot y = y + x \cdot y \\
(ZS) & \forall x. \quad 0 \neq \sigma x \\
(CD) & \forall x. \quad x = 0 \vee (\exists y. \, x = \sigma y) \\
(SI) & \forall xy. \, \sigma x = \sigma y \;\to\; x = y
\end{array}
$$

In the mechanisation, Robinson arithmetic is usually represented as a finite context $\mathsf{Q}' : \mathcal{L}(\mathcal{F})$ and only converted to the respective axiomatisation $\lambda\varphi. \, \varphi \in Q_{\mathcal{L}}$ when necessary.

Definition 4.17 (Peano arithmetic). *The axiomatisation of Heyting arithmetic* HA *consists of the axioms of Robinson arithmetic except* (CD) *but including all instances of the induction scheme:*

$$(I\varphi) \quad \varphi(0) \;\to\; (\forall x. \, \varphi(x) \;\to\; \varphi(\sigma x)) \;\to\; \forall x. \, \varphi(x)$$

The axiomatisation of Peano arithmetic PA *is the classical closure* HA^c*.*

Note that HA subsumes Q (and PA subsumes Q^c) because (CD) can easily be derived using the induction schemes.

Many properties that hold in Peano or Heyting arithmetic cannot be shown with Robinson arithmetic. In particular, this holds for commutativity and associativity of addition or multiplication. While this can already be regarded as a form of incompleteness, we will also show incompleteness of consistent and enumerable extensions $T \supseteq \mathsf{Q}$, which may contain these properties as axioms.

Lemma 4.18. Q *and* HA *are sound.*

Corollary 4.19 (Consistency). Q *and* HA *are consistent, that is,* $\mathsf{Q} \nvdash \bot$ *and* $\mathsf{HA} \nvdash \bot$*.*

4.5 Arithmetical Hierarchy

In Chapter 5 we will mostly deal with Σ_1-formulas [4] due to a property called Σ_1-completeness. We show Q-decidability of formulas only containing bounded quantifiers and completeness of Σ_1-formulas.

Definition 4.20 (Δ_0-formulas). *Let T be an axiomatisation. A formula φ is T-decidable if $T \vdash \varphi[\rho] \vee T \vdash \neg\varphi[\rho]$ for any substitution ρ such that $\varphi[\rho]$ is closed.*

If φ is Q-decidable we say that φ is Δ_0 or a Δ_0-formula, also written $\varphi \in \Delta_0$.

Note that a formula that is Q-decidable is also Q^c-decidable.

Definition 4.21. *We define two derived comparison operators for first-order formulas as follows:*

$$x \leq y := \exists z.\, y = x + z$$
$$x \leq' y := \exists z.\, y = z + x$$

We need two versions of comparisons to accommodate to the absence of commutativity in Q.

Lemma 4.22. *For any closed term t there is an $n : \mathbb{N}$ such that:*

$$\mathsf{Q} \vdash t = \overline{n}$$

Proof. By induction on t.

Fact 4.23. *The following formulas are Δ_0:*

1. *propositional formulas (including falsity),*
2. *equations $a = b$, where a and b are terms,*
3. *bounded quantifiers $\forall x \leq y.\, \varphi$, $\exists x \leq y.\, \varphi$ or $\exists x \leq' y.\, \varphi$, where y is a variable other than x, and $\varphi \in \Delta_0$.*

Proof. 1. Trivial.

2. By applying Lemma 4.22 to a and b and induction on either numeral.

3. Bounded quantifiers can be shown equivalent to finite conjunction or disjunction, which can be shown Q-decidable. The actual proof is, from a technical perspective, by far the most challenging presented in this thesis, requiring many lemmas on equality, addition, and comparisons, such as, for terms a, b, c, formulas φ, and natural numbers t:

   a) $\mathsf{Q} \vdash a = b \to c(a) = w(b)$
   b) $\mathsf{Q} \vdash a = b \to \varphi(a) \to \varphi(b)$
   c) $\mathsf{Q} \vdash \forall x.\, x + 0 = \overline{t} \to x = \overline{t}$
   d) $\mathsf{Q} \vdash \overline{t} + 0 = \overline{t}$
   e) $\mathsf{Q} \vdash \forall x.\, x \leq 0 \to x = 0$
   f) $\mathsf{Q} \vdash \forall x.\, x \leq' 0 \to x = 0$
   g) $\mathsf{Q} \vdash \forall x.\, x = \overline{t} \vee x \neq \overline{t}$
   h) $\mathsf{Q} \vdash \forall xy.\, x + S\, y = S\, \overline{t} \leftrightarrow x + y = \overline{t}$
   i) $\mathsf{Q} \vdash \forall x.\, x \leq \overline{t} \to x \leq S\, \overline{t}$
   j) $\mathsf{Q} \vdash \forall x.\, x \leq' \overline{t} \to x \leq' S\, \overline{t}$
   k) $\mathsf{Q} \vdash \overline{t} \leq \overline{t}$
   l) $\mathsf{Q} \vdash \overline{t} \leq' \overline{t}$
   m) $\mathsf{Q} \vdash \forall x.\, x \leq S\, \overline{t} \to x \neq S\, \overline{t} \to x \leq \overline{t}$
   n) $\mathsf{Q} \vdash \forall x.\, x \leq' S\, \overline{t} \to x \neq S\, \overline{t} \to x \leq' \overline{t}$

   They are shown directly or by induction on a formula, term, or numeral involved. Note that some of these, such as c), e), and f), are obvious using completeness (see Section 4.6) by Lemma 4.32.

Definition 4.24. *A formula is Σ_1 if it is of the form $\exists m_1, m_2, \ldots, m_n.\, \psi$ where $\psi \in \Delta_0$. A formula is Π_1 if it is of the form $\forall m_1, m_2, \ldots, m_n.\, \psi$ where $\psi \in \Delta_0$.*

These definitions of Δ_0, Σ_1, and Π_1 correspond to those by Mostowski [39] up to his presentation of provability. Δ_0 can also be defined purely syntactically, just as Σ_1 and Π_1, as done by Mück [40]. We believe that both definitions are equivalent up to equivalence in Q.

Lemma 4.25 ($\exists$ compression). *For any formula $\varphi \in \Sigma_1$ there is a formula $\psi \in \Delta_0$ such that:*

$$\mathsf{Q} \vdash \varphi \leftrightarrow \exists m.\, \psi$$

Proof. It suffices to show that we can compress two existential quantifiers, that is, for any $\varphi \in \Delta_0$:

$$\exists \psi \in \Delta_0.\, \mathsf{Q} \vdash (\exists xy.\, \varphi(x, y)) \leftrightarrow \exists z.\, \psi(z)$$

Choose:

$$\psi(z) := \exists x \leq z.\, \exists y \leq' z.\, \varphi(x, y)$$

The rest of this proof is done formally in Q. The direction from right to left is trivial. Let x, y be such that $\varphi(x, y)$. Choose $z := x + y$. Both bounds can easily be shown since our use of $\leq'$ accommodates the absence of commutativity.

Fact 4.26 (Σ_1-completeness). *Let $\varphi \in \Sigma_1$ be a closed formula. $\mathbb{N} \vDash \varphi$ implies $\mathsf{Q} \vdash \varphi$.*

Proof. By Lemma 4.25 we can assume $\varphi = \exists m.\, \psi$ for some $\psi \in \Delta_0$. We obtain $m \in \mathbb{N}$ and $\mathbb{N} \vDash \psi(\overline{m})$ by soundness. By the definition of Δ_0 and soundness, $\mathsf{Q} \vdash \psi(\overline{m})$ must hold.

Note that the converse holds by soundness.

Corollary 4.27 (Σ_1-witnesses). *Witnesses for closed Σ_1-formulas are always standard, that is, for any formula $\varphi \in \Sigma_1$ with a single free variable x:*

$$\mathsf{Q} \vdash \exists x.\, \varphi(x) \to \exists n.\, \mathsf{Q} \vdash \varphi(\overline{n})$$

Proof. By extracting a witness in $\mathbb{N}$ using soundness and reestablishing the formula using Σ_1-completeness.

4.6 Completeness

Completeness is a property of first-order logic that is not directly related to incompleteness. For classical first-order logic it states that if a formula is true in every model, it is provable. It is, however, not provable in our constructive meta-logic without additional assumptions [32, 13]. We will therefore consider it as an axiom.

We use it to explain results in Chapter 5 from a semantical perspective. It is not required as an assumption for our main results.

Definition 4.28 (Completeness). *The axiom of completeness for* Q *is defined as follows: For any formula* φ*, we have* $\mathsf{Q}^c \vdash \varphi$ *if and only if* $M \vDash \varphi$ *for every extensional model* $M \vDash \mathsf{Q}^c$*.*

Note that this formulation of completeness entails the assumption of classical soundness. We formulate completeness for extensional models to simplify the mechanisation, since extensionality allows us to use the Coq rewriting mechanism. Completeness does not hold for Q in place of Q^c when using Tarski semantics.

The following results and definitions are not directly related to completeness but will be helpful in proofs using completeness.

Lemma 4.29 (Absoluteness). *Let* $\varphi \in \Delta_0$ *be closed and* $M_1, M_2 \vDash \mathsf{Q}^c$ *be models. Then* $M_1 \vDash \varphi \to M_2 \vDash \varphi$*.*

Proof. Assume $M_1 \vDash \varphi$. We distinguish two cases:

1. If $\mathsf{Q} \vdash \varphi$, we obtain $M_2 \vDash \varphi$ by soundness.
2. If $\mathsf{Q} \vdash \neg\varphi$, we obtain $M_1 \vDash \neg\varphi$ by soundness, which contradicts the assumption.

Definition 4.30. *Let* M *be a model. We define a comparison operator inside models:*

$$x \leq_M y \;:=\; \exists z.\, x + z = y$$

Definition 4.31. *Let* $M \vDash \mathsf{Q}^c$ *be a model and* $x \in M$*. We call* x *standard if there is a number* $n \in \mathbb{N}$ *such that* $x = \llbracket \overline{n} \rrbracket$*.*

Lemma 4.32. *Let* $M \vDash \mathsf{Q}^c$ *be an extensional model,* $x, y \in M$*, and* $n \in \mathbb{N}$*. The following hold:*

1. *x is standard if and only if σx is standard.*
2. *$x + y$ is standard if and only if x and y are standard.*
3. *If $x \leq_M y$ and y is standard, x is also standard.*
4. *If x is non-standard, then $\llbracket \overline{n} \rrbracket \leq_M x$.*

Note that Lemma 4.32 also holds for non-extensional models. This would, however, prevent us from using Coq's rewriting mechanism during mechanisation, considerably complicating the proof.

4.7 Formal Systems

To conclude this chapter we instantiate the abstract formalism of formal systems from Chapter 3 to first-order logic.

Lemma 4.33. *Let* T *be an enumerable and consistent axiomatisation. Let*

$$\mathrm{FS}_T := (\mathcal{F}, \neg, \lambda\varphi.\, T \vdash \varphi).$$

FS_T *is a formal system. If* $T' \supseteq T$ *is a consistent and enumerable extension,* FS'_T *is an extension of* FS_T*.*

Definition 4.34. *Completeness, weak representability and strong separability of an enumerable axiomatisation T are defined as in the induced formal system FS_T. In our case, the representation functions will always be of the form $\lambda x.\, \varphi(x)$ where φ is a formula with a single free variable. We say T weakly Σ_1-represents or strongly Σ_1-separates if additionally $\varphi \in \Sigma_1$.*

5 Incompleteness of First-Order Logic

We are now almost ready to instantiate the abstract incompleteness results from Chapter 3 to the formalism of first-order logic, giving us essential incompleteness of Robinson arithmetic. This is one of the main results of this thesis, along with the abstract incompleteness proofs. At this point, we are only missing strong separability of disjoint and enumerable predicates in Q.

We use Rosser's trick to show that Robinson arithmetic strongly separates all disjoint and weakly Σ_1-representable predicates. Rosser's trick was used by Rosser [53] to weaken the preconditions of Gödel's original incompleteness proof [17].

Obtaining weak representability of (synthetically) enumerable predicates is not possible when only assuming a form of Church's thesis for an unspecified model of computation θ. Instead, we first show that Q weakly represents μ-enumerable predicates using an existing mechanisation of the DPRM theorem by Larchey-Wendling and Forster [34] and assume $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, making μ-enumerability and (synthetic) enumerability coincide.

We first show weak representability of μ-enumerable predicates using prior results. Then we explain the original Gödel-Rosser proof of incompleteness with a particular focus on the usage of Rosser's trick, which we then use to establish strong separability of disjoint and weakly representable predicates. Finally, we use these results to instantiate the abstract incompleteness and undecidability proofs from Chapter 3.

5.1 Weak representability

Weak representability of μ-enumerable predicates in Robinson arithmetic has already been mechanised by Kirst and Hermes [24], building upon work by Larchey-Wendling and Forster [34] mechanising the DPRM theorem [51, 7, 38]. We use a slightly simpler approach to this result by using Σ_1-completeness of Robinson arithmetic.

Theorem 5.1. *Let P be a μ-enumerable predicate. There is a formula $\varphi \in \Sigma_1$ that weakly represents P in Q.*

Proof. Using results on the DPRM theorem by Larchey-Wendling and Forster [34]. Diophantine equations with existentially quantified parameters can easily be embedded into first-order logic. Weak representability follows by soundness and Σ_1-completeness.

Kirst and Hermes [24] do, however, show a slightly more general result, giving weak representability in an even weaker axiomatisation than Robinson's Q.

Corollary 5.2. *Assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, any enumerable predicate is weakly representable.*

Proof. By Lemma 2.17.

This result can be used to instantiate the weaker abstract proofs of incompleteness.

Fact 5.3. *Assuming* $\mathsf{EPF}^{\mu}_{\mathbb{N}}$*, provability in* Q *is undecidable.*

Proof. By Fact 3.7 and Theorem 5.1.

Fact 5.4. *Assuming* $\mathsf{EPF}^{\mu}_{\mathbb{N}}$*, there is an independent* Σ_1*-sentence in* Q*.*

Proof. By Theorems 3.9 and 5.1.

Note that these results can also be obtained for all enumerable and sound (or even just ω-consistent, like Gödel's original result) extensions of Q, since such an extensions are, in particular, sound for Σ_1-formulas.

We have not mechanised these statements since they are subsumed by the results in Section 5.4.

5.2 Rosser's Trick for Gödel's Incompleteness Proof

We give a rough summary of Gödel's approach to his first incompleteness theorem based on [49] and show how Rosser strengthened this result to point out the parallels between Kleene's and the Gödel-Rosser approach to incompleteness.

Let $T \supseteq \mathsf{Q}$ be an enumerable and ω-consistent extension of Q. Gödel first arithmetises the deduction system, that is, he constructs a formula Prf with two free variables such that for any formula φ:

- If n is a Gödelisation of a proof of φ in T, then $T \vdash \mathrm{Prf}(\overline{\ulcorner \varphi \urcorner}, \overline{n})$.
- If n is not a Gödelisation of a proof of φ in T, then $T \vdash \neg\mathrm{Prf}(\overline{\ulcorner \varphi \urcorner}, \overline{n})$.

We use $\ulcorner \cdot \urcorner$ to refer to a Gödelisation of formulas. The choice of Gödelisation is not important, as long as it is "easy" to compute.

Next he defines a provability relation $\mathrm{Prov}(x) := \exists k.\, \mathrm{Prf}(x, k)$ that fulfils:

$$T \vdash \varphi \;\rightarrow\; T \vdash \mathrm{Prov}(\overline{\ulcorner \varphi \urcorner}) \tag{5.1}$$

Using the so-called "diagonal lemma" he then constructs a formula G_T such that:

$$T \vdash G_T \;\leftrightarrow\; \neg\mathrm{Prov}(\overline{\ulcorner G_T \urcorner}) \tag{5.2}$$

Informally, G_T states its own unprovability. Therefore, G_T is independent in T because:

- $T \nvdash G_T$: Assume $T \vdash G_T$. We can show $T \vdash \mathrm{Prov}(\overline{\ulcorner G_T \urcorner})$ by (5.1) and $T \vdash \neg\mathrm{Prov}(\overline{\ulcorner G_T \urcorner})$ by (5.2), which contradicts consistency. We do not need ω-consistency for this case.

- $T \nvdash \neg G_T$: Assume $T \vdash \neg G_T$. By consistency, there is no proof of G_T, and therefore $\forall k.\, T \vdash \neg\mathrm{Prf}(\overline{\ulcorner G_T \urcorner}, \overline{k})$. However, we also have $T \vdash \mathrm{Prov}(\overline{\ulcorner G_T \urcorner})$ by (5.2), which contradicts ω-consistency of T.

It is also possible to show $T \vdash \mathrm{Prov}(\overline{\ulcorner \varphi \urcorner}) \;\rightarrow\; T \vdash \varphi$ using ω-consistency.

Rosser gave a proof that consistency suffices for the existence of an independent sentence by using a modified provability relation:

$$\mathrm{Prov}'(x) := \exists k.\, \mathrm{Prf}(x, k) \wedge \forall k' \leq k.\, \neg\mathrm{Prf}(x, \mathrm{neg}(x))$$

The function neg negates a Gödelised formula. It is easy to define only using addition and multiplication when using a suitable Gödelisation.

While it is still possible to show that

$$T \vdash \varphi \;\rightarrow\; T \vdash \mathrm{Prov}'(\overline{\ulcorner \varphi \urcorner}), \tag{5.3}$$

we also obtain

$$T \vdash \neg\varphi \;\rightarrow\; T \vdash \neg\mathrm{Prov}'(\overline{\ulcorner \varphi \urcorner}). \tag{5.4}$$

That is, Prov' strongly separates the provable from the refutable formulas. The proofs of these properties are similar to the ones presented in Fact 5.6. By using the diagonal lemma to obtain a formula R_T such that

$$T \vdash R_T \;\leftrightarrow\; \neg\mathrm{Prov}'(\overline{\ulcorner \varphi \urcorner}), \tag{5.5}$$

we can show independence of R_T:

- $T \nvdash R_T$: Analogous to $T \nvdash G_T$.
- $T \nvdash \neg R_T$: We have $T \vdash \mathrm{Prov}'(\overline{\ulcorner R_T \urcorner})$ by (5.5) and $T \vdash \neg\mathrm{Prov}'(\overline{\ulcorner R_T \urcorner})$ by (5.4), which contradicts consistency.

This proof just needs consistency (as opposed to ω-consistency), and therefore yields incompleteness of all enumerable and consistent extensions of Q, since they also represent Prf as required.

5.3 Strong Separability of Disjoint Predicates

Rosser's trick cannot just be applied to provability, but all existentially representable predicates. We use it to give a proof of strong separability of disjoint and weakly Σ_1-representable predicates, based on [4].

Lemma 5.5 (Decidability of $\leq$). *Let $x \in \mathbb{N}$. Then*

$$\mathsf{Q} \vdash \forall y.\, \overline{x} \leq y \vee y \leq \overline{x}$$

Proof. By meta-level induction on x and object-level case distinction on y in the successor case.

Fact 5.6 (Rosser's trick). *Let $P_1, P_2 : \mathbb{N} \to \mathbb{P}$ be disjoint and weakly Σ_1-representable predicates. P_1 and P_2 are also strongly Σ_1-separable, that is, there is a formula φ_1 such that:*

$$P_1\, x \;\to\; \mathsf{Q} \vdash \varphi_1(\overline{x}) \tag{5.6}$$
$$P_2\, x \;\to\; \mathsf{Q} \vdash \neg\varphi_1(\overline{x}) \tag{5.7}$$

Proof. We additionally construct a formula φ_2 that strongly separates P_2 and P_1:

$$P_2\, x \;\to\; \mathsf{Q} \vdash \varphi_2(\overline{x}) \tag{5.8}$$
$$P_1\, x \;\to\; \mathsf{Q} \vdash \neg\varphi_2(\overline{x}) \tag{5.9}$$

Using Lemma 4.25, let $\psi_1, \psi_2 \in \Delta_0$ be such that:

$$P_1\, x \;\leftrightarrow\; \mathsf{Q} \vdash \exists k.\, \psi_1(\overline{x}, k) \tag{5.10}$$
$$P_2\, x \;\leftrightarrow\; \mathsf{Q} \vdash \exists k.\, \psi_2(\overline{x}, k) \tag{5.11}$$

Choose:

$$\varphi_1(x) := \exists k.\, \psi_1(x,k) \wedge \forall k' \leq k.\, \neg\psi_2(x,k')$$
$$\varphi_2(x) := \exists k.\, \psi_2(x,k) \wedge \forall k' \leq k.\, \neg\psi_1(x,k')$$

Now, φ_1 and φ_2 fulfil (5.6) through (5.9):

(5.6) Let $x : \mathbb{N}$ be such that $P_1\, x$. By (5.10) and soundness we have a $k \in \mathbb{N}$ such that $\mathbb{N} \vDash \psi_1(\overline{x}, \overline{k})$. By Σ_1-completeness it suffices to show $\mathbb{N} \vDash \varphi_1(\overline{x}, \overline{k})$. By choosing k, the first conjunct is trivial. For the second one, let $k' \leq k$ be such that $\mathbb{N} \vDash \psi_2(\overline{x}, \overline{k'})$. By Σ_1-completeness and (5.11) we have $P_2\, x$, which contradicts disjointness.

(5.8) Analogous to (5.6).

(5.7) Let $x : \mathbb{N}$ be such that $P_2\, x$. By (5.8) we have $\mathsf{Q} \vdash \varphi_2(\overline{x})$ and by Corollary 4.27 we have a $k_2 : \mathbb{N}$ such that $\mathsf{Q} \vdash \psi_2(\overline{x}, \overline{k_2}) \wedge \forall k_2' \leq \overline{k_2}.\, \neg\psi_1(\overline{x}, k_2')$. The rest of this proof is done formally in Q. Assume a k_1 such that $\psi_1(\overline{x}, k_1)$ and $\forall k_1' \leq k_1.\, \neg\psi_2(\overline{x}, k_1')$. We are done by doing a case distinction on whether $\overline{k_2} \leq k_1$ or $k_1 \leq \overline{k_2}$ using Lemma 5.5 and instantiating one of the quantified assumptions.

(5.9) Analogous to (5.7).

Corollary 5.7. *Assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, any two disjoint and enumerable predicates are strongly Σ_1-separable.*

5.3.1 Illustrative Proof Using Completeness

We give a semantic interpretation of the proof of Fact 5.6 assuming (the axiom of) completeness, based on [43]. It gives a different perspective on the results from the last section. In particular, we give another proof of (5.9) for Q^c instead of Q, since completeness only applies to classical theories:

Proof (Alternate proof of (5.9)). Assume $P_1\, x$ for some $x : \mathbb{N}$. By $P_1\, x$ and Corollary 4.27, we have a $k_1 : \mathbb{N}$ such that $\mathsf{Q}^c \vdash \psi_1(\overline{x}, \overline{k_1})$ and therefore $\mathbb{N} \vDash \psi_1(\overline{x}, \overline{k_1})$ by soundness.

Let $M \vDash Q^c$ be a model. By completeness, it suffices to assume $M \vDash \psi_2(\overline{x}, k_2)$ and $\forall k' \leq k_2.\, \neg(M \vDash \psi_2(\overline{x}, k'))$ for some $k_2 : M$ and derive a contradiction.

Now, k_2 must be non-standard, because otherwise we would have $M \vDash \psi_1(\overline{x}, \overline{k_1})$ and therefore $\mathbb{N} \vDash \psi_2(\overline{x}, \overline{k_2})$ by absoluteness, which yields a contradiction by Σ_1-completeness, weak representability, and disjointness of P_1 and P_2.

Therefore $\overline{k_1} \leq k_2$ by Lemma 4.32, with which we can instantiate the bounded quantifier and obtain a contradiction.

Semantic proofs tend to be easier to find and easier to understand intuitively. Unfortunately, translating semantic into purely syntactic proofs to avoid the assumption of completeness is sometimes difficult. Particularly in this case, the semantic proof is very different from the syntactic one.

5.4 Main Results

We can now use the stronger abstract incompleteness results to show essential undecidability and incompleteness of first-order logic over the axiomatisation of Robinson arithmetic.

Theorem 5.8 (Essential undecidability). *Assuming* $\mathsf{EPF}^{\mu}_{\mathbb{N}}$*, provability in* Q *and all its consistent extensions* $T \supseteq \mathsf{Q}$ *is undecidable, that is,* Q *is essentially undecidable.*

Proof. By Corollaries 3.14 and 5.7. In particular, $\lambda x.\, \theta x x \rhd b$ is enumerable for any $b : \mathbb{B}$. We obtain $\mathsf{EPF}_{\mathbb{B}}$ by Lemmas 2.9 and 2.18.

Theorem 5.9 (Essential incompleteness). *Assuming* $\mathsf{EPF}^{\mu}_{\mathbb{N}}$*,* Q *and all its consistent extensions* $T \supseteq \mathsf{Q}$ *have an independent and closed* Σ_1 *formula, that is,* Q *is essentially incomplete.*

Proof. Analogous to Theorem 5.8.

Note that both theorems could be shown without assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$ because it is admissible, that is, all functions it is instantiated with can, in principle, directly be implemented using μ-recursive functions. This would, however, require proving the abstract incompleteness results from Chapter 3 for μ-recursive functions and first-order logic, which is not the goal of this thesis.

We can also obtain incompleteness of Q^c by showing Q^c is consistent. We cannot show this immediately because it is not sound without the assumption of LEM. One way to show this is to use a Friedman translation (c.f. [20]) to show that consistency of Q^c is equivalent to consistency of Q.

6 Further Representability Results

We can not only use Rosser's trick to obtain strong separability, but also other, more powerful representability results. Some of them have, for example, been assumed and used by Hermes and Kirst [20].

In this chapter we strengthen the statement of strong separability slightly, give a proof of strong representability of decidable predicates, and derive a form of Church's thesis for Robinson's Q from $\mathsf{EPF}^{\mu}_{\mathbb{N}}$.

Note that we have not yet mechanised the results in this chapter. We expect their mechanisation to be tedious, but not difficult.

6.1 Improved Strong Separability

We improve on the statement of Fact 5.6 by finding additional properties of the formulas φ_1 and φ_2 constructed during the proof.

Fact 6.1. *Let $P_1, P_2 : \mathbb{N} \to \mathbb{P}$ be disjoint and weakly Σ_1-representable predicates. There are formulas $\varphi_1, \varphi_2 \in \Sigma_1$ such that:*

- *P_1 and P_2 are weakly represented by φ_1 and φ_2, respectively:*

$$P_1\, x \;\leftrightarrow\; \mathsf{Q} \vdash \varphi_1(\overline{x}) \tag{6.1}$$
$$P_2\, x \;\leftrightarrow\; \mathsf{Q} \vdash \varphi_2(\overline{x}) \tag{6.2}$$

- *P_1 and P_2 are strongly separated by φ_1, that is, in addition:*

$$P_1\, x \;\to\; \mathsf{Q} \vdash \neg\varphi_2(\overline{x}) \tag{6.3}$$

- *P_2 and P_1 are strongly separated by φ_2, that is, in addition:*

$$P_2\, x \;\to\; \mathsf{Q} \vdash \neg\varphi_1(\overline{x}) \tag{6.4}$$

- *φ_1 and φ_2 can be shown disjoint internally by HA:*

$$\mathsf{HA} \vdash \forall x.\, \neg(\varphi_1(x) \wedge \varphi_2(x)) \tag{6.5}$$

Proof. The proofs of (6.1) through (6.4) are by Fact 5.6 or obvious by the definitions of φ_1 and φ_2. We show (6.5) by first showing that $\mathsf{HA} \vdash \forall xy.\, x \le y \vee y \le x$ by object-level induction on x and then instantiating one of the bounded quantifiers in the assumptions to derive a contradiction. We do not know of a way to show (6.5) in Q, in particular since $\mathsf{Q} \nvdash \forall xy.\, x \le y \vee y \le x$ can be shown by giving an appropriate model.

The deep disjointness property (6.5) was assumed by [20].

6.2 Strong Representability

While enumerable predicates correspond exactly to weakly representable predicates, decidable (or rather, enumerable and co-enumerable) predicates correspond exactly to strongly representable predicates. Furthermore, strong representability of decidable predicates is a corollary of strong separability of disjoint and enumerable predicates.

Lemma 6.2. *Let P be a predicate and P as well as $\overline{P}$ be weakly Σ_1-representable. There is a formula $\varphi \in \Sigma_1$ (or $\varphi \in \Pi_1$) that strongly represents P, that is:*

$$Px \;\to\; \mathsf{Q} \vdash \varphi(x) \qquad\qquad \neg Px \;\to\; \mathsf{Q} \vdash \neg\varphi(x)$$

Proof. For $\varphi \in \Sigma_1$, apply Fact 5.6 to P and $\overline{P}$. For $\varphi \in \Pi_1$, apply it to $\overline{P}$ and P instead, negate the resulting Σ_1-formula and obtain an equivalent Π_1-formula by using that $\mathsf{Q} \vdash (\neg\exists x.\, \psi(x)) \;\leftrightarrow\; \forall x.\, \neg\psi(x)$ for any formula ψ.

Note that any formula strongly representing a definite predicate P, that is, $\forall x.\, Px \vee \neg Px$, is Q-decidable. Note that any decidable predicate is definite.

Corollary 6.3. *Assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, any enumerable and co-enumerable (and particularly any decidable) predicate is strongly representable.*

6.3 Church's Thesis for Robinson Arithmetic

To work with computability theory and accompanying representability theorems in first-order logic synthetically, a form of Church's thesis for Robinson arithmetic is desirable. For instance, a formulation for total functions was assumed in [20]. We give a proof of a form of Church's thesis for Q ($\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$) for both total and partial functions, assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$.

To do this we first need to show a form of bounded binary quantification to be Q-decidable.

Lemma 6.4. *Let φ be Q-decidable. Binary bounded quantifiers $\forall xy.\, x + y \leq z \;\to\; \varphi$, where z is a variable other than x and y, are Q-decidable.*

Proof. Similar to Fact 4.23, particularly case *3.*

Fact 6.5. *Let $f : \mathbb{N} \rightharpoonup \mathbb{N}$ be a partial function such that the graph of f, that is, $\{(x, y) \mid fx \rhd y\}$, is weakly Σ_1-representable.*[8] *There is a $\varphi \in \Sigma_1$ such that:*

$$fx \rhd y \;\to\; \mathsf{Q} \vdash \forall y'.\, \varphi(\overline{x}, y') \;\leftrightarrow\; y' = \overline{y}$$

Proof. By Lemma 4.25, let $\psi \in \Delta_0$ be such that:

$$fx \rhd y \;\leftrightarrow\; \mathsf{Q} \vdash \exists k.\, \psi(x, y, k)$$

[8] Weak representability can be generalised to predicates of arbitrary arity by using a pairing function

Choose

$$\Phi(x,y,k) := \psi(x,y,k) \wedge \forall y'k'.\, y' + k' \leq y + k \;\to\; \psi(x,y',k') \;\to\; y' = y$$
$$\varphi(x,y) := \exists k.\, \Phi(x,y,k).$$

Assume $fx \rhd y$. The proof of $\mathsf{Q} \vdash y' = y \;\to\; \forall y'.\, \varphi(\overline{x}, y')$ is similar to the proofs of (5.6) and (5.8) in Fact 5.6.

The rest of this proof is done formally in Q, except when stated otherwise. Assume y', k' such that $\Phi(\overline{x}, y', k')$. By $fx \rhd y$ and the direction from right to left we also have $\varphi(\overline{x}, \overline{y})$ and therefore by Corollary 4.27 a $k \in \mathbb{N}$ such that $\Phi(\overline{x}, \overline{y}, \overline{k})$. We are done by doing a case distinction on whether $\overline{y+k} \leq y' + k'$ or $y' + k' \leq \overline{y+k}$ using Lemma 5.5.

Corollary 6.6 (Church's thesis for Q ($\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$)). *Let $f : \mathbb{N} \rightharpoonup \mathbb{N}$ be a partial function. Assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, there is a formula $\varphi \in \Sigma_1$ such that:*

$$fx \rhd y \;\to\; \mathsf{Q} \vdash \forall y'.\, \varphi(\overline{x}, y') \;\leftrightarrow\; y' = \overline{y}$$

Proof. The graph of a partial function is synthetically enumerable and therefore weakly Σ_1-representable by Corollary 5.2.

The converse of this statement, that is, $\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$ implies $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, also appears to hold. $\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$ appears to be practical and powerful way to unify representability properties of Q. In particular, assuming $\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$, it is possible to show weak and strong representability of decidable and enumerable predicates respectively (c.f. [20]), as well as strong separability.

It can also be useful to consider Church's thesis only for total functions, leading to the following simplified form:

Corollary 6.7. *Let $f : \mathbb{N} \to \mathbb{N}$. Assuming $\mathsf{EPF}^{\mu}_{\mathbb{N}}$, there is a formula $\varphi \in \Sigma_1$ such that:*

$$Q \vdash \forall y'.\, \varphi(\overline{x}, y') \;\leftrightarrow\; y' = \overline{fx}$$

This alternative form of $\mathsf{CT}^{\mathsf{Q}}_{\mathbb{N}}$ was also assumed by [20].

7 Conclusion

In this thesis we first gave abstract incompleteness proofs in different strengths for abstract formal systems with a negation function, following Kleene. The strongest version states essential incompleteness (and, by a related proof, essential undecidability) of formal systems strongly separating certain enumerable and disjoint predicates.

Secondly, we instantiated our results to first-order logic over the axiomatisation of Robinson arithmetic. We used a mechanisation of the DPRM theorem to obtain weak representability of enumerable predicates in Robinson's Q and then used Rosser's trick to show strong separability of disjoint and enumerable predicates.

Lastly we used strong separability and Rosser's trick to obtain other, more powerful representability results.

7.1 Discussion

The different variants of incompleteness theorems we considered throughout this thesis can be classified along two axes: The strength of the result (anonymous incompleteness or an independent sentence), and the strength of the assumptions (soundness or consistency). Both our main abstract and instantiated incompleteness results are of the strongest type, that is, they assume consistency and construct an independent sentence.

7.2 Mechanization

The mechanisation[9] consists of two main parts: The abstract incompleteness proofs and their instantiation to first-order logic. The former consists of only around 400 lines of code, which can be reduced to around 150 when only considering the strongest incompleteness proofs, while the latter adds around 2250 lines of code. The development is based on the Coq library of undecidability proofs (CLUP) [15], from which additional code, particularly on synthetic computability, first-order logic, and the DPRM theorem, is used.

Mechanising and working with partial functions and Church's thesis is straightforward. The paper proofs, however, tend to follow a slightly different structure than their mechanised counterparts, in particular when dealing with equivalences, such as in Lemma 2.11. Additionally, definitions of partial functions in Coq, as for example in Fact 3.3, tend to be slightly unnatural since they have to propagate step-indices explicitly when using

[9] The accompanying mechanisation can be accessed at `https://github.com/uds-psl/coq-synthetic-incompleteness/tree/bachelor`.

other step-indexed functions or enumerators. This could be avoided by using the abstract interface by Forster [10]. Otherwise, the mechanisation of Chapters 2 and 3 is notably unremarkable.

Mechanising the instantiation to first-order logic, however, was a lot more work. We build upon an existing mechanisation of first-order logic by Kirst et al. [25] that includes most fundamental definitions and lemmas for working with first-order logic. As opposed to the definitions presented in this text, it defines formulas and terms to be quantified over a signature, that is, types of predicate and function symbols and their corresponding arities, explicitly defines classical provability as a part of the deduction system, and uses de Bruijn indices instead of explicit naming. While the former two differences did not affect the mechanisation other than requiring some boilerplate code, the latter repeatedly caused us problems. Mechanising structures that include binders, such as predicate logic or programming languages, is well known to be much more tedious than dealing with them on paper, where many lemmas on and properties of substitutions are largely glossed over. On paper we avoided much of this by explicitly working with the Barendregt convention.

Notably, a lot of work (almost half of the mechanisation of the instantiation, by lines of code) went into mechanising Q-decidability of bounded quantification and Σ_1-completeness due to the technicality of these results.

We relied heavily on the first-order proof mode for Coq by Koch, as described in [22], allowing us to use tactics similar to the ones included with Coq to show statements within first-order logic. The proof mode also provides translations between a de Bruijn representation of logical formulas and a named representation, which greatly improves the ergonomics of working with first-order logic. This project would have been much more tedious if we did not have the proof mode available.

7.3 Related Work

Mechanisations of Gödel's incompleteness theorems. The earliest mechanisation of Gödel's first incompleteness theorem was developed by Shankar in 1994 [54] using Nqthm [5], also called the Boyer-Moore theorem prover, a proof assistant based on Lisp. He does not mechanise incompleteness of arithmetic, but of a finite set theory, which simplifies encoding recursive structures, such as formulas and proofs, immensely. His development consists of around 20 000 lines of code. A mechanisation of incompleteness of first-order arithmetic, based on an axiomatisation similar to Robinson arithmetic, was first developed by O'Connor in 2005 [42] using Coq, consisting of almost 44 000 lines of code. Another mechanisation of incompleteness of arithmetic using HOL Light [18] was developed by Harrison in 2009 [19]. More recently, both of Gödel's incompleteness theorems were mechanised by Paulson in 2014 [45, 46] in around 12 000 lines of Isabelle [41] code. He showed incompleteness of a finite set theory slightly different from the one used by Shankar. To our knowledge, he was the first to give a complete mechanisation of Gödel's second incompleteness theorem, relying on a proof by Swierczkowski [60].

None of the mechanisations mentioned above used Kleene's approach to incompleteness. However, for example O'Connor used representability of primitive recursive functions as

an intermediate step to show weak representability of first-order provability, similar to Gödel's original proof.

Working with set theory instead of arithmetic considerably simplifies representing recursive structures within the logic itself, such as provability. We did not consider such problems in this thesis since we relied on a mechanisation of the DPRM theorem to obtain representability results.

Popescu and Traytel [47] mechanised incompleteness using the Gödel-Rosser approach abstractly in 2019 based on a much more complex notion of formal systems than ours, additionally incorporating substitutions, soundness, arithmetic, and more.

A weaker form of incompleteness for a subset of Robinson arithmetic in Coq was mechanised by Kirst and Hermes in 2021 [24], using Kleene's folklore proof and the DPRM theorem. Their result differs from ours in two ways: First, they do not obtain essential incompleteness since they rely on Kleene's early folklore proof using the halting problem (see Fact 3.8). Instead, they give an abstract notion of formal systems incorporating soundness, and use it to deduce incompleteness of all sound extensions of their axiomatisation. Secondly, their development does not deduce falsity from the assumption of incompleteness, instead constructing a decider for the halting problem of Turing machines, which also prevents them from constructing an independent sentence. This can, however, be considered a form of contradiction in synthetic computability theory.

Kirst and Hermes also mechanised an analogous incompleteness statement for a finite set theory by deriving undecidability using a reduction from the Post correspondence problem.

Synthetic computability theory. The basic principles of synthetic computability theory [50, 3] were first applied to CIC by Forster et al. [12]. A treatment of Church's thesis [33, 63] to enhance the expressivity and applicability of synthetic computability theory in CIC was developed by Forster [9, 11, 10].

The first proof of the DPRM theorem was finished in 1970 by Matiyasevitch [38] and mechanised by Larchey-Wendling and Forster [34], which is used as a source problem for undecidability proofs in the Coq library of undecidability proofs (CLUP) [15]. CLUP also contains a mechanised development of first-order logic [25].

Different approaches to Gödel's incompleteness theorems. The Gödel-Rosser approach to incompleteness was developed in the 1930s, primarily by Gödel [17] and Rosser [53]. Kleene presented his approach to incompleteness prominently in both of his books [29, 30], as well as multiple papers [26, 27, 28]. Turing mentioned similar ideas to show incompleteness in his seminal paper on the *Entscheidungsproblem* [64].

Different proofs of Gödel's first incompleteness theorem, among some abstract ones, have been considered by Smullyan [56, 57]. In particular, he also considers the strengthened version of Kleene's proof we considered in Chapter 3 abstractly, although with a slightly more complex notion of formal system.

Another attempt to formalise Gödel's incompleteness results "without (too many) tears" was developed by [55].

Our approach to incompleteness of arithmetic shares similarities with work by Beklemishev [4], who argues using an implicit model of computation. Another account of Gödel's incompleteness theorem was developed partially independently by Post [48].

7.4 Future Work

We have not yet mechanised the results from Chapter 6. We expect their mechanisation to be tedious but not difficult. Additionally, there might be other interesting representability properties we have not yet shown, particularly weak Π_1-representability of co-enumerable predicates.

We have not considered the conditions under which Rosser's trick is applicable abstractly. Doing this could simplify future instantiations of the stronger abstract incompleteness results, as long as the abstraction is sufficiently simple.

Our instantiation to first-order logic with Robinson's Q currently relies on a mechanisation of the DPRM theorem. The DPRM theorem, however, is a much stronger statement than we actually require, and is considerably harder to show. Using our mechanisation of Σ_1-completeness it appears feasible to mechanise weak representability of μ-enumerable predicates for our first-order logic directly by first finding formulas that weakly define, that is, a semantic notion analogous to weak representability, μ-enumerable predicates in the standard model. Similar approaches to have been taken by [42, 45].

While the first-order proof mode [22] was already very helpful in mechanising our results, it does not yet compare to its primary inspiration, the Iris[10] proof mode [31], in particular in regards to conversions between naming schemes, error messages, and reliability.

In Chapter 6, we showed that $\mathsf{EPF}_{\mathbb{N}}^{\mu}$ implies Church's thesis for Q. We expect the converse to be provable as well by first showing that, given any partial function "captured" by Q, its graph is μ-enumerable, which suffices for its μ-computability. Mechanising this fact, however, appears to be challenging because we would have to implement our first-order logic, that is, substitution, enumerability of provable formulas, etc., using μ-recursive functions. Automatic extractions of such functions for first-order logic, specifically into a lambda calculus, have already been investigated by Forster, Kirst and Wehr [13] using a tool by Forster and Kunze [14].

Our approach to mechanising Gödel's first incompleteness theorem does not immediately apply to Gödel's second incompleteness theorem as well, preventing us from mechanising it using our approach. In particular, it requires even deeper representability properties which, to our knowledge, cannot easily be obtained without inspecting their respective formulas explicitly, which we were able to avoid by using Rosser's trick. One small set of representability properties sufficient is known as the Hilbert-Bernays derivability conditions [36]. An abstract explanation of Gödel's second incompleteness theorem using computability theory could lead to a solution to this problem.

[10] A higher-order, shallowly embedded separation logic framework for Coq [23].

Bibliography

[1] Scott Aaronson. *Rosser's theorem via Turing machines*. Shtetl-Optimized. 21st July 2011. URL: https://scottaaronson.blog/?p=710 (visited on 28th Feb. 2022).

[2] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics 103. North-Holland, 1981.

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

[4] Lev Beklemishev. "Gödel incompleteness theorems and the limits of their applicability. I". In: *Russian Mathematical Surveys* 65 (2011), p. 857.

[5] Robert S. Boyer, Matt Kaufmann and J S. Moore. "The Boyer-Moore theorem prover and its interactive enhancement". In: *Computers & Mathematics with Applications* 29.2 (1995), pp. 27–62.

[6] Thierry Coquand and Gérard Huet. "The calculus of constructions". In: *Information and Computation* 76.2 (1988), pp. 95–120.

[7] Martin Davis, Hilary Putnam and Julia Robinson. "The decision problem for exponential Diophantine equations". In: *Annals of Mathematics* (1961), pp. 425–436.

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

[9] Yannick Forster. "Church's thesis and related axioms in Coq's type theory". In: *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*. Vol. 183. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 21:1–21:19.

[10] Yannick Forster. "Computability in Constructive Type Theory". PhD thesis. Saarland University, 2021. DOI: 10.22028/D291-35758.

[11] Yannick Forster. "Parametric Church's thesis: synthetic computability without choice". In: *International Symposium on Logical Foundations of Computer Science*. 2022, pp. 70–89.

[12] Yannick Forster, Dominik Kirst and Gert Smolka. "On synthetic undecidability in Coq, with an application to the Entscheidungsproblem". In: *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 2019, pp. 38–51.

[13] Yannick Forster, Dominik Kirst and Dominik Wehr. "Completeness theorems for first-order logic analysed in constructive type theory (extended version)". In: *Logical Foundations of Computer Science*. Springer, 2020, pp. 47–74.

[14] Yannick Forster and Fabian Kunze. "A certifying extraction with time bounds from Coq to call-by-value lambda calculus". In: *10th International Conference on Interactive Theorem Proving (ITP 2019)*. Vol. 141. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, 17:1–17:19.

[15] Yannick Forster et al. "A Coq library of undecidable problems". In: *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages*. 2020.

[16] Torkel Franzén. *Gödel's Theorem: An Incomplete Guide to its Use and Abuse*. Ak Peters Series. Taylor & Francis, 2005.

[17] Kurt Gödel. "Über Formal Unentscheidbare Sätze der Principa Mathematica und Verwandter Systeme I". In: *Monatshefte für Mathematik und Physik* 38 (1931), pp. 173–198.

[18] John Harrison. "HOL Light: a tutorial introduction". In: *Formal Methods in Computer-Aided Design*. Springer Berlin Heidelberg, 1996, pp. 265–269.

[19] John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[20] Marc Hermes and Dominik Kirst. "An analysis of Tennenbaum's theorem in constructive type theory". In: *7th International Conference on Formal Structures for Computation and Deduction*. 2022.

[21] Douglas R. Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid*. Basic Books Inc., 1979.

[22] Johannes Hostert, Mark Koch and Dominik Kirst. "A toolbox for mechanised first-order logic". In: *The Coq Workshop*. Vol. 2021. 2021.

[23] Ralf Jung et al. "Iris from the ground up: a modular foundation for higher-order concurrent separation logic". In: *Journal of Functional Programming* 28 (2018).

[24] Dominik Kirst and Marc Hermes. "Synthetic undecidability and incompleteness of first-order axiom systems in Coq". In: *ITP 2021*. 2021.

[25] Dominik Kirst et al. "A Coq library for mechanised first-order logic". In: *The Coq Workshop*. 2022.

[26] Stephen C. Kleene. "General recursive functions of natural numbers". In: *Mathematische Annalen* 112 (1936), pp. 727–742.

[27] Stephen C. Kleene. "Recursive predicates and quantifiers". In: *Transactions of the American Mathematical Society* 53 (1943), pp. 41–73.

[28] Stephen C. Kleene. "A symmetric form of Gödel's theorem". In: *The Journal of Symbolic Logic* 16.2 (1951), p. 147.

[29] Stephen C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

[30] Stephen C. Kleene. *Mathematical Logic*. Dover Publications, 1967.

[31] Robbert Krebbers, Amin Timany and Lars Birkedal. "Interactive proofs in higher-order concurrent separation logic". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. 2017, pp. 205–217.

[32] Georg Kreisel. “On weak completeness of intuitionistic predicate logic”. In: *The Journal of Symbolic Logic* 27.2 (1962), pp. 139–158.

[33] Georg Kreisel. “Mathematical logic”. In: *Journal of Symbolic Logic* 32.3 (1967), pp. 419–420.

[34] Dominique Larchey-Wendling and Yannick Forster. “Hilbert’s tenth problem in Coq (extended version)”. In: *Logical Methods in Computer Science* 18 (2022).

[35] Bernard Linsky and Andrew David Irvine. “Principia mathematica”. In: *The Stanford Encyclopedia of Philosophy*. Spring 2022 Edition. Metaphysics Research Lab, Stanford University, 2022.

[36] Martin H. Löb. “Solution of a problem of Leon Henkin”. In: *Journal of Symbolic Logic* 20.2 (1955), pp. 115–118.

[37] Yevgeniy Makarov and Jean-François Monin. *The Coq standard library. Library Coq.Logic.ConstructiveEpsilon*. URL: `https://coq.inria.fr/library/Coq.Logic.ConstructiveEpsilon.html` (visited on 13th May 2022).

[38] Yuri V. Matijasevič. “Enumerable sets are Diophantine”. In: *Soviet Mathematics: Doklady* 11 (1970), pp. 354–357.

[39] Andrzej Mostowski. “On definable sets of positive integers”. In: *Fundamenta Mathematicae* 34.1 (1947), pp. 81–112.

[40] Niklas Mück. “The Arithmetical Hierarchy, Oracle Computability, and Post’s Theorem in Synthetic Computability”. Unsubmitted. Bachelor’s thesis. Saarland University, 2022. URL: `https://ps.uni-saarland.de/~mueck/bachelor/thesis.pdf`.

[41] Tobias Nipkow, Lawrence C. Paulson and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Vol. 2283. Springer Science & Business Media, 2002.

[42] Russell O’Connor. “Essential incompleteness of arithmetic verified by Coq”. In: *Theorem Proving in Higher Order Logics* (2005), pp. 245–260.

[43] Sebastian Oberhoff. *How does one prove that Peano arithmetic can represent all partially computable functions?* Mathematics Stack Exchange. 7th Feb. 2020. URL: `https://math.stackexchange.com/q/3538168` (visited on 24th May 2022).

[44] Christine Paulin-Mohring. “Inductive definitions in the system Coq rules and properties”. In: *Typed Lambda Calculi and Applications*. Springer, 1993, pp. 328–345.

[45] Lawrence C. Paulson. “A machine-assisted proof of Gödel’s incompleteness theorems for the theory of hereditarily finite sets”. In: *The Review of Symbolic Logic* 7.3 (2014), pp. 484–498.

[46] Lawrence C. Paulson. “A mechanised proof of Gödel’s incompleteness theorems using nominal Isabelle”. In: *Journal of Automated Reasoning* 55 (June 2015), pp. 1–37.

[47] Andrei Popescu and Dmitriy Traytel. “A formally verified abstract account of Gödel’s incompleteness theorems”. In: *Automated Deduction – CADE 27*. Springer International Publishing, 2019, pp. 442–461.

[48] Emil L. Post. "Absolutely unsolvable problems and relatively undecidable propositions – acount of an anticipation". In: Springer, 1941, pp. 375–441.

[49] Panu Raatikainen. "Gödel's incompleteness theorems". In: *The Stanford Encyclopedia of Philosophy*. Spring 2022 Edition. Metaphysics Research Lab, Stanford University, 2022.

[50] Fred Richman. "Church's thesis without tears". In: *The Journal of Symbolic Logic* 48.3 (1983), pp. 797–803.

[51] Julia Robinson. "Existential definability in arithmetic". In: *Transactions of the American Mathematical Society* 72.3 (1952), pp. 437–449.

[52] Raphael Robinson. "An essentially undecidable axiom system". In: *Proceedings of the International Congress of Mathematics*. 1950, pp. 729–730.

[53] Barkley Rosser. "Extensions of some theorems of Gödel and Church". In: *Journal of Symbolic Logic* 1.3 (1936), pp. 87–91.

[54] Natarajan Shankar. *Metamathematics, Machines and Gödel's Proof*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.

[55] Peter Smith. *Gödel Without (Too Many) Tears*. Logic Matters, 2021.

[56] Raymond M. Smullyan. *Gödel's Incompleteness Theorems*. Oxford University Press, 1992.

[57] Raymond M. Smullyan. *Diagonalization and Self-Reference*. Clarendon Press, 1994.

[58] Raymond M. Smullyan. *The Godelian Puzzle Book: Puzzles, Paradoxes and Proofs*. Dover Publications, 2013.

[59] John Stillwell. "Emil Post and his anticipation of Gödel and Turing". In: *Mathematics Magazine* 77.1 (2004), pp. 3–14.

[60] Stanislaw Swierczkowski. "Finite sets and Gödel's incompleteness theorems". In: *Dissertationes Mathematicae* 422 (2003), pp. 1–58.

[61] Alfred Tarski. "The concept of truth in formalized languages". In: *Logic, Semantics, Metamathematics*. Oxford University Press, 1936, pp. 152–278.

[62] The Coq Development Team. *The Coq proof assistant*. Jan. 2022. DOI: 10.5281/zenodo.5846982.

[63] Anne S. Troelstra and Dirk van Dalen. *Constructivism in Mathematics, Vol 1*. ISSN. Elsevier Science, 1988.

[64] Alan M. Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.42 (1936), pp. 230–265.

[65] user21820. *Computability viewpoint of Godel/Rosser's incompleteness theorem*. Mathematics Stack Exchange. 31st Dec. 2021. URL: https://math.stackexchange.com/q/2486349 (visited on 22nd Mar. 2022).

[66] Dirk van Dalen. *Logic and Structure*. Fourth Edition. Springer, 2008.

[67] Anatoly Vorobey. *First incompleteness via computation: an explicit construction*. Foundations of Mathematics mailing list. URL: https://cs.nyu.edu/pipermail/fom/2021-September/022872.html (visited on 21st Feb. 2022).

[68] Benjamin Werner. “Sets in types, types in sets”. In: *International Symposium on Theoretical Aspects of Computer Software*. Springer, 1997, pp. 530–546.

[69] Richard Zach. “Hilbert’s program”. In: *The Stanford Encyclopedia of Philosophy*. Fall 2019 Edition. Metaphysics Research Lab, Stanford University, 2019.