Saarland University

Faculty of Mathematics and Computer Science

Master'sThesis

Modeling Peano Arithmetic in Constructive Type Theory

Undecidability and Tennenbaum's Theorem

Author
Marc Hermes

Advisors
Dominik Kirst
Prof. Dr. Moritz Weber

Reviewers
Prof. Dr. Moritz Weber
Prof. Dr. Gert Smolka

Submitted: 23th November 2021

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, 23$^{\text{th}}$ November, 2021

Abstract

Gödel's first incompleteness theorem entails that the first-order theory of Peano arithmetic (PA) and its consistent extensions admit a wealth of independent statements. By the completeness theorem then, PA cannot be categorical, meaning it does not posses a unique model up to isomorphism. A theorem by Stanley Tennenbaum however tells us that if we restrict our attention to computable models, first-order PA is categorical with regards to this class of models.

In this thesis we develop the first-order theory of PA inside of a constructive type theory to revisit and study Tennenbaum's theorem in this constructive setting. This approach allows for a synthetic viewpoint of computability and we can furthermore consistently assume Church's thesis, making it possible to farther abstract from many details in computability arguments. As an additional result, we establish the undecidability of PA via a reduction to the solvability of Diophantine equations. Both parts have been verified in the Coq proof-assistant.

Acknowledgements

I am deeply indebted to Prof. Dr. Moritz Weber who picked up on my interest for mathematical logic and gave me the opportunity to write a thesis on this topic. He brought the Programming Systems Lab (PSL) to my attention; a computer science chair "right across the street" involved with the Coq proof-assistant.

I am particularly grateful for the assistance given by Dominik Kirst, who took the time to answer the numerous questions I had when I started learning Coq, and was invaluable as an advisor during this thesis, most notably regarding his guidance and many helpful comments and discussions.

I would also like to express my special thanks to Prof. Dr. Gert Smolka who crucially made this interdisciplinary thesis possible and I am very grateful for everything I learned during his lectures and the time at the PSL.

Finally, I also want to thank Mirko Stappert and Friedrich Günther for their feedback on earlier drafts and fruitful discussions.

"Immediately after making this statement, Royal realized that it was true."
—Wes Anderson, *The Royal Tenenbaums*

Contents

Chapter 1

Introduction

The natural numbers are some of the most fundamental objects in mathematics with well established and time-proven formalizations like Peano arithmetic (PA). It gives a description of the natural numbers as a first-order theory, with axioms describing the computational behavior of numbers and how the truth of a property can be shown for all numbers. Most importantly PA succeeds in its goal to describe the natural numbers, as witnessed by the fact that $\mathbb{N}$ is a model of this first-order theory. But the success of PA goes much further. All of modern number theory can in principle be based on PA; integers $\mathbb{Z}$ and rational numbers $\mathbb{Q}$ can be defined inside of the theory. Fragments of second-order PA have been used as a basis to study the exact strength of theorems, by tracing back the necessary axioms to prove them, in a program called *reverse mathematics* [18]. Furthermore PA is expressive enough to encode many standard data structures, as well as recursive functions and computations, allowing the encoding of proofs about PA "inside of PA". This idea was developed by Gödel and used to fruition in his seminal incompleteness theorem (1931), showing that expressiveness of this magnitude also comes with drawbacks [20]. He explicitly constructed a sentence G in the language of PA which is true in $\mathbb{N}$ and yet neither G nor $\neg G$ can be deduced from the axioms; establishing that the theory is incomplete.

Another perspective on incompleteness arose shortly thereafter (1936) in the form of the then new theory of computation. Church [5] and Turing [51] independently gave a negative answer to the *Entscheidungsproblem*, which asked about the possibility of an algorithm which could decide whether any given mathematical statement was universally valid or not. Therefore, no matter the proposed algorithm, it must have a blind spot in the form of a formula φ such that it never yields a result upon inputting φ or $\neg\varphi$.

The incompleteness result has repercussions on the model theory of PA as well. By the completeness theorem, the unprovability of G entails that there must be a model $\mathcal{M}$ of PA in which G is not true. Since G is true in $\mathbb{N}$, the model $\mathcal{M}$ must therefore differ from $\mathbb{N}$. This shows that the grasp of PA on *what the natural numbers are* is

not quite as firm as maybe hoped for. Since the incompleteness theorem allows the same construction to be made on any consistent extension of PA, there is no hope for adding axioms with the goal to sieve out all models except the desired one, i.e. making the theory *categorical*.

The situation changes however if we restrict our attention to a subclass of models. In 1959 Stanley Tennenbaum [47] showed that requiring the models to be countable and computable, in the sense that their arithmetic operations are computable, they can no longer differ from the standard model. This further underlines the strangeness of non-standard models.

In this thesis we will look at a proof of this theorem, which is usually carried out in classical logic with set-theoretic foundations, and will reconsider it in a constructive type theory. This allows for a synthetic approach to computability [1, 39] and for consistently assuming anti-classical axioms like Church's thesis, to further streamline the treatment of computability. This investigation is accompanied by a mechanization of the presented proofs, which was realized in the Coq proof assistant.

Contributions: This thesis makes the following contributions:

- To the best of our knowledge it contains the first mechanized proof of Tennenbaum's Theorem.
- Mechanization and discussion of two additional variants of Tennenbaum's theorem [23, 32, 33].
- Identifying a finite fragment of PA which is shown undecidable. This entails a synthetic incompleteness proof of PA, further discussed in [28].
- The undecidability proof was also mechanized and been made part of the *Coq library of undecidability proofs* [16].

Structure: The thesis is divided into two parts which can be read independently.

The first part consists of Chapter 2 and gives an introduction into the constructive type theory (CTT) which serves as the meta-theory for the second part of the thesis. This is mainly geared towards readers unfamiliar with constructive logic and can safely be omitted by those familiar with the subject.

The second part starts with Chapter 3, where we define first-order logic inside of the meta-theory and use this to specify the first-order theory of PA which is then the object of study for the remaining chapters. Chapter 4 introduces the synthetic approach to computability theory available in CTT, which is then first used in Chapter 5 to show the undecidability of a finite fragment of PA. In Chapter 6 we present a constructive proof of Tennenbaum's theorem based on [44] followed by variants of the proof and the theorem based on [23, 32, 33].

Chapter 2

On Constructive Type Theory

2.1 Constructive Type Theory as a Foundation for Mathematics

When considering the foundations of mathematics, there are two main features which are desireable:

- it should provide a system surveyable enough to trust that it is free of contradictions,
- yet powerful enough to express all of the objects and intuitive notions that a mathematician might come up with.

Both points ensure that the mathematician is not limited in his practice, and can at the same time be confident of the correctness of his results. The most commonly used foundation certainly is Zermelo–Fraenkel set theory (ZF) [25]. The goal of set theory is to formalize the concept of a **set** by fixing intuitively appealing axioms about sets and the relation $\in$ on sets. These concepts are taken as *primitive* [1], all other concepts being defined in terms of *sets* and $\in$, and new theorems are derived from the axioms in the logical framework of first-order logic. There are however other foundations for mathematics. Relevant for this thesis are modern type theories, in which the concept of **functions** an **types** are taken as primitives. Contrary to set theory, where *every object* is a set, type theories formalize a way to separate objects into different bins by assigning them different types. This allows a restriction of functions such that they can only be applied to elements of one specific type. Maybe surprisingly this is not the case in set-theory. Here, any function can be applied to any object i.e. set.

To further illustrate the latter point, we look at the definition of functions in set theory. Intuitively, a function f from A to B is something that relates any *object* x

[1] We designate a symbol as *primitive* if it is part of the language of a theory by definition. This is in contrast to symbols which are introduced as a shorthand for new constructions / definitions inside of the theory.

from A to a unique object y from B, then usually written as $fx = y$ [2]. In set theory this idea of functions is made precise by saying that a function is a total, functional relation R_f on the sets A and B, where the relation R_f itself is a subset of the product of sets $A \times B$ [3], and $fx = y$ is then notation for $(x, y) \in R_f$. No matter the function f then, we can always ask the question wether it maps the set $\{\emptyset\}$ to $\emptyset$, since this is equivalent to $(\{\emptyset\}, \emptyset) \in R_f$.

For a second example showcasing that mathematical intuitions are not always met in set theory, we can look at the natural numbers. In the most common set-theoretic definition, the natural numbers are recursively defined by $0 := \emptyset$ and the successor function $Sn := n \cup \{n\}$. The existence of the set of natural numbers $\mathbb{N}$ is built into set theory, as it can be derived from the infinity axiom $\exists N.\, 0 \in N \wedge \forall n.\, n \in N \to Sn \in N$, stating the existence of a set N containing 0 and being closed under S. Sets with this property are also called **inductive**. The set $\mathbb{N}$ is then defined as the (by virtue of the axiom, non-empty) intersection of all inductive sets, and its uniqueness can be shown from the other axioms. By construction, every element of $\mathbb{N}$ is itself a set, which means that $20 \in 21$ is a perfectly fine (and in in this case provable) statement of ZF.

This last observation contradicts the intuition that it should not be possible to put two numbers into relation using $\in$; this should only work for sets. The underlying instinct is that sets and numbers are indeed of different *types*, and a relation like $\in$ should only be defined when the right object has the type *set*, and not for objects of type *natural number*. As mentioned before, this idea of assigning types to objects, to get the desired restrictions on relations and functions on them, is the fundamental idea behind **type theories** of any kind. Setting up a type theory usually means to start with a universe [4] of types $\mathbb{T}$, define some base types like the type of booleans $\mathbb{B}$, natural numbers $\mathbb{N}$ and function types. These are definitionally made part of $\mathbb{T}$. One then turns to describe how elements of these types are generated. To indicate that some object x has the type T the canonical notation is $x : T$ [5]. In the case of $\mathbb{N}$, one would add the axioms expressing that 0 has type $\mathbb{N}$ and that there is a successor function $S : \mathbb{N} \to \mathbb{N}$, which freely generates elements of $\mathbb{N}$ by applying S repeatedly; $S0, SS0, SSS0, \ldots$ Overall, the expressiveness of many type theories is on a level making them well-suited as a foundation for mathematics.

The foundational language is however not the only part usually going more or less unnoticed [3]. Another undeniable part of mathematical practice is to use deduction

[2] The more conventional notation is $f(x) = y$. We will stick however to the notation $fx = y$ as used in type theories to express the application of f to x.

[3] Where the product $\times$ is not primitive to set theory but defined using the axioms.

[4] "universe" is to be read as an intuitive notion, used to express a collection of types.

[5] Coming from set theory, it is helpful to think of $x : T$ as expressing $x \in T$. The correspondence is not exact, as illustrated by the arguments mentioned before.

–or more broadly logic– to get to conclusions and to formulate proofs. This too has been the subject to formalization and has lead to well studied logical systems [50] with precise notions of what constitutes a proof. Proofs are often presented in a **natural deduction** style, using *inference rules* of the form

$$\frac{A_1 \quad A_2 \quad \dots \quad A_n}{C}$$

to build proofs in the form of trees. An inference rule indicates that to infer the **goal** C, we need to provide **proof trees** for every **branch** $A_1, A_2, \dots, A_n$. We will mark rules with a double-line if we want to emphasize that they can be used to *close a branch*; meaning it needs no further justification. A complete proof is then a tree where every branch is closed by double-lines.

Example 2.1 *Given the following inference rules*

$$\frac{\dfrac{}{\mathsf{Yellow}} \quad \dfrac{}{\mathsf{Magenta}}}{\mathsf{Red}} \qquad \frac{\dfrac{}{\mathsf{Yellow}} \quad \dfrac{}{\mathsf{Red}}}{\mathsf{Orange}} \qquad \frac{\dfrac{}{\mathsf{Cyan}} \quad \dfrac{}{\mathsf{Magenta}}}{\mathsf{Blue}}$$

formalizing some color mixing, we can give a proof of Orange*:*

$$\frac{\dfrac{}{\mathsf{Yellow}} \quad \dfrac{\dfrac{}{\mathsf{Yellow}} \quad \dfrac{}{\mathsf{Magenta}}}{\mathsf{Red}}}{\mathsf{Orange}}$$

Using inference rules we can represent a fragment of the first-order deduction rules by

$$\frac{}{\Gamma \vdash P}\ (\mathsf{A}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\mathsf{I}_\to) \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ (\mathsf{E}_\to) \tag{2.1}$$

where $\Gamma \vdash P$ can be read as "with the information in the context Γ we can prove the proposition P". The **assumption rule** (A) indicates that given P is already part of the context Γ, we need not further justification in order to prove the goal P, allowing us to close this branch. The **implication introduction rule** ($\mathsf{I}_\to$) tells us how new information can get added to the context and introduces an implication into the goal. The **implication elimination rule** ($\mathsf{E}_\to$) is also called *modus ponens* and eliminates an implication from a branch. As an example we give a deduction of the proposition $(A \to (B \to A)) \to (A \to B) \to (A \to C)$ from the empty context. The proof-tree should be read from the bottom to the top:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \to (B \to C) \in \Gamma}\,(\mathsf{A})}{\Gamma \vdash A \to (B \to C)} \quad \dfrac{\dfrac{}{A \in \Gamma}\,(\mathsf{A})}{\Gamma \vdash A}}{\Gamma \vdash B \to C}\,(\mathsf{E}_\to) \quad \dfrac{\dfrac{\dfrac{}{A \to B \in \Gamma}\,(\mathsf{A})}{\Gamma \vdash A \to B} \quad \dfrac{\dfrac{}{A \in \Gamma}\,(\mathsf{A})}{\Gamma \vdash A}}{\Gamma \vdash B}\,(\mathsf{E}_\to)}{\Gamma \vdash C}\,(\mathsf{E}_\to)}{A \to (B \to C),\, A \to B \vdash A \to C}\,(\mathsf{I}_\to)}{A \to (B \to C) \vdash (A \to B) \to (A \to C)}\,(\mathsf{I}_\to)}{\vdash (A \to (B \to C)) \to (A \to B) \to (A \to C)}\,(\mathsf{I}_\to)$$

$$\text{where } \Gamma := A \to (B \to C),\, A \to B,\, A \tag{2.2}$$

The full set of deduction rules also includes rules for the other logical connectives plus quantifiers, and is given in Definition 3.4. Regarding the full rule set of first-order logic, one is of particular interest, namely **double negation elimination**:

$$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}\,(\mathsf{DNE})$$

since it allows us to do proofs by contradiction. If we want to prove the proposition A in context Γ, we can instead try to prove falsity $\bot$ with $\neg A$ added to the context.

$$\dfrac{\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash \neg\neg A}\,(\mathsf{I}_\to)}{\Gamma \vdash A}\,(\mathsf{DNE})$$

We can see this strategy in action in the proof of the following lemma:

Lemma 2.2 *There are irrational numbers $a, b \in \mathbb{R}$ such that a^b is rational.*

Proof For the purpose of a contradiction assume that for all irrational numbers a, b we have that a^b is irrational. Since $\sqrt{2}$ is irrational this implies that $\sqrt{2}^{\sqrt{2}}$ must be irrational. But then $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}}$ must also be irrational, leading to a contradiction, since $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$. □

There is something peculiar about the proof of Lemma 2.2; in the end, it does not explicitly construct any irrational numbers $a, b \in \mathbb{R}$ such that a^b is rational. Philosophically this can be regarded as unsatisfying; apparently there is no guarantee that we can extract numbers with a certain property from a proof that such numbers must exist. The conclusion to draw here is that the proofs, treated as objects by themselves, can be **non-informative** in the above sense. The reason that the presented proof is non-informative lies in the usage of DNE. The general correctness of this observation is best justified by the fact that removing DNE from the list of deduction rules, we

get the so called **intuitionistic logic** where all proofs do remain informative. This is due to the informative nature of the remaining deduction rules. As we will see in Section 2.2, the word "*informative*" here is not condemned to remain only an intuitive notion. We can make the information-content of proofs explicit: from every intuitionistic proof we can extract an algorithm (in the form of a λ-term) solving a corresponding problem.

It is now important to stress that by removing DNE as a logical rule, one leaves the confines of conventional mathematics, whose background logic is classical. This also means to leave behind equivalent principles such as the law of excluded middle (LEM) and proofs by contradiction. At first glance this seems like a heavy price to pay for the benefit of having informative proofs, since there can now be statements that no longer have any proof at all, whenever they unavoidably depend on DNE. But it also comes with an unexpected advantage, as now, there are also less formulas φ who's negation $\neg\varphi$ can be shown, opening the door for new mathematical theories with axioms that would have been inconsistent beforehand.

To make this concrete we give a few examples. Consider the set $\Delta := \{x \mid x^2 = 0\}$ as the set of *infinitesimal* values around 0. If we assume the **principle of microaffineness**

$$\forall(f\colon \Delta \to \mathbb{R})\ \exists! a \in \mathbb{R}\ \forall x \in \Delta.\, fx = a \cdot x + f\,0$$

as an axiom, it becomes possible to define the derivative of $f\colon \Delta \to \mathbb{R}$ to be the unique value a that the axiom provides and to easily extend this definition to functions $\mathbb{R} \to \mathbb{R}$. There is of course an obvious catch. Using classical reasoning we could immediately show that $x^2 = 0 \to x = 0$ i.e. $\Delta = \{0\}$, contradicting the uniqueness claim for a. The axiom is therefore incompatible with classical logic. In intuitionistic logic however this problem does not appear. The same proof implying $\Delta = \{0\}$ is no longer possible, and no other proofs can possibly be found, as is entailed by the existence of models showing the consistency of theories with the principle of microaffineness in intuitionistic logic. This opens up the intriguing theory of synthetic differential geometry [42], where infinitesimals are a valid concept and all functions $\mathbb{R} \to \mathbb{R}$ are smooth.

An example of crucial importance for this thesis comes in the form of **Church's thesis**, which makes the assertion that all functions $\mathbb{N} \to \mathbb{N}$ are computable, for any reasonable interpretation of *computable*. This statement is not compatible with classical logic, as by using LEM, it is relatively easy to construct a function $\mathbb{N} \to \mathbb{N}$ which solves the halting problem and can therefore not be computable. Using an intuitionistic logic however gives us the freedom to assume this as an axiom. We will come back to this and elaborate on the axiom in Chapter 6.

Maybe surprisingly, we find a similar example in analysis. Taking the simplest case

of a discontinuous function

$$fx := \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

and unfolding the set-theoretic definition of this function, we get that it is the relation $R_f = \{(0,1)\} \cup \{(x,0) \mid x \neq 0\}$ on $\mathbb{R} \times \mathbb{R}$. In intuitionistic logic one can still easily verify that this relation is functional and indeed discontinuous. But it is impossible to prove that R_f is total on the domain $\mathbb{R}$. Strictly speaking, it is therefore not a function. The conventional, classical proof of its totality would start with the case distinction $x = 0 \vee x \neq 0$, which is however not available since we cannot prove $\forall x \in \mathbb{R}. x = 0 \vee x \neq 0$ constructively. From a constructive proof of this statement, we could extract an algorithm which could be used to solve the halting problem [3]. Intuitionistically we will not be able to prove that any of the usual discontinuous functional relations are functions. Since it is also not possible to disprove that all functions are continuous, we can consistently assume this assertion as an axiom, similar in spirit to Church's thesis.

This thesis will part from the conventional mathematical framework in both of the aspects that we have now mentioned: The foundational language and the logic. The change in logic will be the most noticeable as proofs will only be able to use instances of LEM that we have proven beforehand and we will frequently come across statements of the form $\neg\neg A$, where we can no longer simply conclude A. When it comes to the foundational language, we will part from set theory and instead use a dependent type theory, in which, instead of sets, functions become the primitive notion. There are three main reasons for this:

- One goal of this thesis is to work in a constructive meta-theory, and (as we will see in Section 2.2.1) type theories come with a built-in intuitionistic logic.
- The constructive logic enables the usage of the aforementioned Church's Thesis, greatly simplifying the treatment of *computability*.
- All major parts of this thesis have been mechanized and checked in the Coq proof assistant, which is based on the type theory called *Calculus of Inductive Constructions* (CIC) [7, 35].

We will introduce the type-theoretic foundations in the coming sections. It should be stressed that when it comes to the readability of mathematical statements throughout this text, there will not be many noticeable differences compared to set-theoretic foundations.

2.2 A Primer on Type Theory

In the case of first-order ZF, the axioms are intended to describe the concept of a set, which is then taken as the primitive notion of discourse. In modern type theory

however the primitive notion is that of a *function* and is based on the λ-calculus that was developed by Church [4]. Our presentation of the type theory will be relatively short and will only represent the shadow of what would be CIC; outlining its essential features, without detailing every part. For more complete descriptions of simply-typed and dependent type theory, we refer to [21, 27, 40, 45, 50].

2.2.1 Simply-typed λ-Calculus

The simply-typed λ-calculus first introduces an enumerable list of type variables $A, B, C...$ and the following rules to build types:

- Every type variable $A, B, C, \ldots$ is a type and we write $A\colon \mathbb{T}$ to express that A is a type.
- For all types α, β we have the **function type** $\alpha \to \beta\colon \mathbb{T}$.

Next one specifies syntactic rules for inductively building terms based on an infinite set of variables $x, y, z, \ldots$ as follows:

- Every variable $x, y, z, \ldots$ is a λ-term.
- If s and t are λ-terms, then $(s\,t)$ is a λ-term.
- If x is a variable, A a type and s is a λ-term, then $\lambda x\colon A.\, s$ is a λ-term.

The expression $x\colon A$ is called a **type annotation** and expresses that the λ-term x has the type A. A list $x_1\colon A_1, \ldots, x_n\colon A_n$ of type annotations is called a **context** iff every appearing variable x_i appears at most once in the list. A term of the form $(s\,t)$ is to be read as an **application** of the term s to t and a term $\lambda x\colon A.\, s$ is an **abstraction**, intended to represent a function which can only be applied to terms of type A. In the application $(\lambda x\colon A.s)a$, where the abstraction is applied to another term $a\colon A$, the term a gets substituted into every occurrence of x in the body s. We denote a substitution like this with $s[x/a]$. For example, applying the abstraction $\lambda x\colon A.\,(x(yx))$ to $a\colon A$ will result in $(a(ya))$.

We will use the convention that parentheses in applications are left-associative, meaning $xyzw$ is to be understood as $(((xy)z)w)$. For convenience we will use the notation x^A for $x\colon A$, especially when writing abstractions, and we will often leave out the type annotations, when the type can be inferred from the context. If there are several abstractions in a row like in $\lambda x^A.\lambda y^B.\lambda z^C.\, s$ we will use the notation $\lambda x^A y^B z^C.s$.

The intended semantics making λ-terms behave like functions is formalized by the β-**reduction** relation $\succ_\beta$ on the terms. We already described the most important rule which is up to some details $(\lambda x^A.s)a \succ_\beta s[x/a]$. We will only illustrate conversions by giving an example.

Example 2.3 *Given the following two λ-terms:*

$$K := \lambda xy.\, x \qquad S := \lambda xyz.\, xz(yz) \tag{2.3}$$

we can compute the result of the application SKK:

$$\begin{aligned} SKK &= \big((\lambda xyz.\, xz(yz))K\big)K \\ &\succ_\beta \big(\lambda yz.\, Kz(yz)\big)K \\ &\succ_\beta \lambda z.\, Kz(Kz) \\ &= \lambda z.\, (\lambda xy.\, x)z\big(Kz\big) \\ &\succ_\beta \lambda z.\, (\lambda y.\, z)(Kz) \\ &\succ_\beta \lambda z.\, z \end{aligned}$$

The resulting term is often denoted I and has the behavior expected by an identity function; if applied to any term a, it returns it unchanged: $I\, a \succ_\beta a$.

I is also said to have a **normal form** as no further reductions can take place. We also include a second kind of reduction, called η-**reduction** which allows the reduction $(\lambda x.\, fx) \succ_\eta f$. This is again motivated by the semantical understanding of λ-terms as functions: it should make no difference whether we apply f or $\lambda x.\, fx$ to some term. These reduction rules can be combined into a relation $\succ_{\beta\eta}$, for which we simply write $\succ$. We can then introduce an equivalence relation on terms $\equiv$ such that $s \equiv t$ holds when both terms s, t reduce to the same normal form. It's crucial to note that every term reduces to a unique normal form, making the equivalence relation behave as expected [21].

That a certain term t has the type A under a context Γ will be called a **typing judgment** and is expressed by a ternary relation $\Gamma \vdash t\colon A$ [6]. We give inference rules specifying how typing judgments can be deduced:

$$\frac{\overline{(x\colon A) \in \Gamma}}{\Gamma \vdash x\colon A}\ (\mathsf{T}) \qquad \frac{\Gamma, (x\colon A) \vdash s\colon B}{\Gamma \vdash \lambda x^A.s\colon A \to B}\ (\mathsf{I}_\lambda) \qquad \frac{\Gamma \vdash s\colon (A \to B) \qquad \Gamma \vdash t\colon A}{\Gamma \vdash s\,t\colon B}\ (\mathsf{E}_\lambda)$$

A term t can be **typed** if there is a type A such that $\vdash t\colon A$, and a type A is called **inhabited** if there is a term t such that $\vdash t\colon A$. The typing relation is non-trivial as there are λ-terms t (e.g. $\lambda x.\, xx$) which cannot be typed, and types (e.g. $((A \to B) \to A) \to A$) which are not inhabited.

[6] We therefore use the notation $\Gamma \vdash$ in two separate instances here: one for writing deductive proofs in first-order logic and the second is for typing judgments. If desired, one could use a different symbol like $\Vdash$ to better differentiate between these two use-cases.

At this point we have to raise awareness about the extreme similarities between the above rules and the rules for the implicational fragment of natural deduction that we saw in Equation (2.1):

$$\frac{\overline{A \in \Gamma}}{\Gamma \vdash A}\ (\mathsf{A}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\mathsf{I}_{\to}) \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ (\mathsf{E}_{\to})$$

The observation that the rules for typing judgments and the deduction rules of intuitionistic logic are in direct correspondence is known as the **Curry-Howard-deBruijn isomorphism** (CHdBI) [46, 50] and underlines how the implicational fragment of propositional logic can be found as part of the type theory we have presented so far:

$$(\mathsf{A}) \leftrightarrow (\mathsf{T}) \qquad (\mathsf{I}_{\to}) \leftrightarrow (\mathsf{I}_{\lambda}) \qquad (\mathsf{E}_{\to}) \leftrightarrow (\mathsf{E}_{\lambda}) \tag{2.4}$$

We will soon add rules to the type theory which will then allow us to extend this isomorphism, prompting a viewpoint called **propositions as types** (PAT). The isomorphism states that a proposition P has a deduction in intuitionistic logic precisely when P, treated as a type, is inhabited. Syntactically, any typing judgement of P immediately yields a deductive proof of P simply by reducing any appearing annotation $t\colon A$ in the deduction to only A. But the other direction also works out. From every intuitionistic proof of a proposition P we can extract an inhabitant of the type P. We will exemplify this by looking the deduction Equation (2.2) of the proposition $(A \to (B \to C)) \to (B \to C) \to (A \to C)$, and show how we can extract an inhabitant. We start by replacing all of the deduction rules by typing rules, as indicated by Equation (2.4). Since we do not know any of the terms that should appear in the deduction, we initially just leave them blank.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{(\_\colon A \to (B \to C)) \in \Gamma}}{\Gamma \vdash \_\colon A \to (B \to C)}\ (\mathsf{T}) \quad \dfrac{\overline{(\_\colon A) \in \Gamma}}{\Gamma \vdash \_\colon A}\ (\mathsf{T})}{\Gamma \vdash \_\colon B \to C}\ (\mathsf{E}_{\lambda}) \quad \dfrac{\dfrac{\overline{(\_\colon A \to B) \in \Gamma}}{\Gamma \vdash \_\colon A \to B}\ (\mathsf{T}) \quad \dfrac{\overline{(z\colon A) \in \Gamma}}{\Gamma \vdash \_\colon A}\ (\mathsf{T})}{\Gamma \vdash \_\colon B}\ (\mathsf{E}_{\lambda})}{\Gamma \vdash \_\colon C}\ (\mathsf{E}_{\lambda})}{(\_\colon A \to (B \to C)), (\_\colon A \to B) \vdash \_\colon A \to C}\ (\mathsf{I}_{\lambda})}{(\_\colon A \to (B \to C)) \vdash \_\colon (A \to B) \to (A \to C)}\ (\mathsf{I}_{\lambda})}{\vdash \_\colon (A \to (B \to C)) \to (A \to B) \to (A \to C)}\ (\mathsf{I}_{\lambda})$$

$$\text{where } \Gamma := (\_\colon A \to (B \to C)), (\_\colon A \to B), (\_\colon A)$$

Next we simply assign names to the terms that are in the context Γ:

$$\text{where } \Gamma := (x\colon A \to (B \to C)), (y\colon A \to B), (z\colon A)$$

Focusing on the left part of the deduction, we can see that this enforces how the blanks in the (T) rule applications must be filled:

$$\dfrac{\dfrac{\dfrac{}{(x\colon A \to (B \to C)) \in \Gamma}}{\Gamma \vdash x\colon A \to (B \to C)}\,(\mathsf{T}) \qquad \dfrac{\dfrac{}{(y\colon A) \in \Gamma}}{\Gamma \vdash y\colon A}\,(\mathsf{T})}{\Gamma \vdash \_\colon B \to C}\,(\mathsf{E}_\lambda)$$

The (E_λ) rule then also dictates how the blank at the very bottom has to be filled:

$$\dfrac{\dfrac{\dfrac{}{(x\colon A \to (B \to C)) \in \Gamma}}{\Gamma \vdash x\colon A \to (B \to C)}\,(\mathsf{T}) \qquad \dfrac{\dfrac{}{(y\colon A) \in \Gamma}}{\Gamma \vdash y\colon A}\,(\mathsf{T})}{\Gamma \vdash xy\colon B \to C}\,(\mathsf{E}_\lambda)$$

Traversing the whole tree in this fashion, from top to bottom, the rules enforce how every blank space must be filled, producing a completely annotated tree.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(x\colon A \to (B \to C)) \in \Gamma}}{\Gamma \vdash x\colon A \to (B \to C)}\,(\mathsf{T}) \quad \dfrac{\dfrac{}{(z\colon A) \in \Gamma}}{\Gamma \vdash z\colon A}\,(\mathsf{T})}{\Gamma \vdash xz\colon B \to C}\,(\mathsf{E}_\lambda) \quad \dfrac{\dfrac{\dfrac{}{(y\colon A \to B) \in \Gamma}}{\Gamma \vdash y\colon A \to B}\,(\mathsf{T}) \quad \dfrac{\dfrac{}{(z\colon A) \in \Gamma}}{\Gamma \vdash z\colon A}\,(\mathsf{T})}{\Gamma \vdash yz\colon B}\,(\mathsf{E}_\lambda)}{\Gamma \vdash xz(yz)\colon C}\,(\mathsf{E}_\lambda)}{(x\colon A \to (B \to C)),\, (y\colon A \to B) \vdash \lambda z.\, xz(yz)\colon A \to C}\,(\mathsf{I}_\lambda)}{(x\colon A \to (B \to C)) \vdash \lambda yz.\, xz(yz)\colon (A \to B) \to (A \to C)}\,(\mathsf{I}_\lambda)}{\vdash \lambda xyz.\, xz(yz)\colon (A \to (B \to C)) \to (A \to B) \to (A \to C)}\,(\mathsf{I}_\lambda)$$

We can recognize the constructed term $\lambda xyz.\, xz(yz)$ as the term S defined in Equation (2.3). By the above judgment we have therefore shown that S can be typed, and more specifically, that it is an inhabitant of the type $(A \to (B \to C)) \to (B \to C) \to (A \to C)$. Terms of type P which are extracted from intuitionistic proofs of P in the above fashion will be called **proof-terms**. Recalling that every λ-term is a function, this makes precise in which sense every intuitionistic proof contains the construction of an algorithm.

Whenever we want to show that a type is inhabited, we will usually leave out terms t in any $\Gamma \vdash t\colon A$ during the judgment; therefore essentially just giving intuitionistic proofs of P. For some time, we will list the extracted proof-terms at the end of the proofs, to make the reader aware of the computational content of the given proof.

2.2.2 Dependent Type Theory

The type theory sketched so far is a system that is quite open in the sense that it can easily be extended by new types. This is achieved by adding new primitive symbols for types and terms, rules specifying how terms of the type are formed and

how they can be used. The latter is achieved by giving **elimination rules**. We start by introducing primitives for binary **product-types** $\times$ and **sum-types** $+$:

$$\frac{\Gamma \vdash A\colon \mathbb{T} \qquad \Gamma \vdash B\colon \mathbb{T}}{\Gamma \vdash A \times B\colon \mathbb{T}} \qquad\qquad \frac{\Gamma \vdash A\colon \mathbb{T} \qquad \Gamma \vdash B\colon \mathbb{T}}{\Gamma \vdash A + B\colon \mathbb{T}}$$

Whenever we introduce new rules like in the above, the context Γ is implicitly assumed to be quantified over. Next we add primitives for terms of the respective types, as well as their **eliminators** π_1, π_2 and E_+:

$$\frac{\Gamma \vdash a\colon A \qquad \Gamma \vdash b\colon B}{\Gamma \vdash (a,b)\colon A \times B} \qquad \frac{\Gamma \vdash f_1\colon A \to C \qquad \Gamma \vdash f_2\colon B \to C}{\Gamma \vdash E_+\, f_1\, f_2\colon A + B \to C}$$

$$\frac{\Gamma \vdash p\colon A \times B}{\Gamma \vdash \pi_1\, p\colon A} \qquad \frac{\Gamma \vdash a\colon A}{\Gamma \vdash i_1 a\colon A + B}$$

$$\frac{\Gamma \vdash p\colon A \times B}{\Gamma \vdash \pi_2\, p\colon B} \qquad \frac{\Gamma \vdash b\colon B}{\Gamma \vdash i_2 b\colon A + B}$$

On the left, we first have the introduction rule for the product $\times$, followed by two elimination rules, and on the right we first have the elimination rule for the sum $+$, followed by the two introduction rules [7] . This order puts a highlight on the duality of the two constructs. We also add conversion rules

$$\pi_1\,(a,b) \succ a \qquad E_+\, f_1\, f_2\,(i_1 a) \succ f_1\, a$$
$$\pi_2\,(a,b) \succ b \qquad E_+\, f_1\, f_2\,(i_2 b) \succ f_2\, b$$

which tell us that π_1, π_2 can be used to project out entries form a pair (a,b) and that $E_+\, f_1\, f_2\, t$ becomes an application of either f_1 or f_2, depending on wether the term $t\colon A + B$ was produced by the injection $i_1\colon A \to A + B$ or $i_2\colon B \to A + B$. These types further extend the CHdBI. This can be seen by comparing the rules for $\times$ and $+$ with with the corresponding rules for $\wedge$ and $\vee$ respectively. Using the product type we then define the usual shorthand $A \leftrightarrow B := (A \to B) \times (B \to A)$. We can now represent propositional statements like $(A \wedge B \to C) \leftrightarrow (A \to (B \to C))$ and show that the corresponding type is inhabited.

Lemma 2.4 *The type* $(A \times B \to C) \leftrightarrow (A \to (B \to C))$ *is inhabited.*

Proof We start by showing that the left part of the product in the $\leftrightarrow$ type is inhabited, where we use $\Gamma := (f\colon A \times B \to C), (a\colon A), (b\colon B)$:

[7] One can loosely draw a parallel to category theory here. Product-types correspond to the product $A \times B$ of two objects A, B in some category $\mathcal{C}$, with the usual projections $\pi_1\colon A \times B \to A$ and $\pi_2\colon A \times B \to B$. Just like in the case for products in a category, the indicated rules for the product-types tell us that from a pair $p\colon A \times B$ we can project out the terms $\pi_1\, p\colon A$ and $\pi_2\, p\colon B$. Similarly, sum-types correspond to coproducts $A \coprod B$ with injections $i_1\colon A \to A \coprod B$ and $i_2\colon B \to A \coprod B$.

$$\dfrac{\dfrac{\Gamma \vdash A \times B \to C \qquad \dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \times B}}{\Gamma \vdash C}}{\vdash (A \times B \to C) \to A \to (B \to C)}$$

All of the branches can now be closed with the information in Γ and we get the proof-term $L := \lambda f^{A \times B \to C} a^A b^B.\, f(a, b)$.

For the right part we set $\Gamma := (f\colon A \to (B \to C)), (p\colon A \times B)$ and get

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash A \to (B \to C)} \qquad \dfrac{\dfrac{}{\Gamma \vdash A \times B}}{\Gamma \vdash A}}{\Gamma \vdash B \to C} \qquad \dfrac{\dfrac{}{\Gamma \vdash A \times B}}{\Gamma \vdash B}}{\dfrac{\Gamma \vdash C}{\vdash (A \to (B \to C)) \to A \times B \to C}}$$

where all branches can again be closed and we get the proof-term

$$R := \lambda f^{A \to (B \to C)} p^{A \times B}. f(\pi_1\, p)(\pi_2\, p).$$

The type in question is then inhabited by the term (L, R). □

The functions R and L constructed in Lemma 2.4 can be used to transform a **cascaded** function $f\colon A \to B \to C$ into its **curried** counterpart $Rf\colon A \times B \to C$ and vice versa.

Next we make a major addition in the form of **dependent product types** (also called **Π-types**), which can be formed according to the rule

$$\dfrac{\Gamma \vdash A\colon \mathbb{T} \qquad \Gamma, (x\colon A) \vdash B\colon \mathbb{T}}{\Gamma \vdash \Pi x^A.\, B\colon \mathbb{T}}$$

We give a rule specifying which terms are to be annotated as Π-types as well as the elimination rule

$$\dfrac{\Gamma, (x\colon A) \vdash s\colon B}{\Gamma \vdash \lambda x^A.s\colon \Pi x^A.\, B} \qquad \dfrac{\Gamma \vdash f\colon \Pi x^A.\, B \qquad \Gamma \vdash a\colon A}{\Gamma \vdash fa\colon B[a/x]}$$

In a dependent product type $\Pi x^A.\, B$, the type B is now allowed to depend on the term $x\colon A$. A Π-type should therefore be thought of as a generalized function type, where the type B of the output can now depend on the input element x of type A. To emphasize this we could also write $\Pi x^A.\, B(x)$ but it also becomes visible in the elimination rule, where we do a substitution $B[a/x]$. This is in contrast to the previously encountered function types $A \to B$, where the output was always of type B, no matter the input. The elimination rule tells us how a term of type $\Pi x^A.\, B$ can be used.

The added Π-types also have their corresponding representative in logic via the CHdBI: They correspond to the ∀ quantified statements. Again one can compare the respective deduction rules too see this connection. This leads to a further extension of the internal logic of the type theory, now also encompassing higher-order intuitionistic logic. The connection justifies the reading of a type like $\Pi x^A.B(x)$ [8] as "for every term x of type A we have $B(x)$".

Apart from the Π-types we have just seen, we also introduce **dependent sum types** (also called Σ-types). Here the introduction rules are

$$\frac{\Gamma \vdash A\colon \mathbb{T} \qquad \Gamma, (x\colon A) \vdash B\colon \mathbb{T}}{\Gamma \vdash \Sigma(x\colon A).\, B\colon \mathbb{T}}$$

$$\frac{\Gamma \vdash a\colon A \qquad \Gamma \vdash b\colon B[a/x]}{\Gamma \vdash \langle a, b\rangle\colon \Sigma\, x^A.\, B}$$

with elimination rule

$$\frac{\Gamma \vdash f\colon \Pi x^A.\, (B \to C)}{\Gamma \vdash E_\Sigma f\colon (\Sigma\, x^A.\, B) \to C}$$

By the construction rule, to get a term $\langle a, b\rangle$ of type $\Sigma\, x^A.\, B$ we need to provide a term $a\colon A$, referred to as a **witness**, and a verification that $B[a/x]$ is inhabited. This is analogous to giving existence proofs, establishing the existence of some element a having the property expressed by the predicate $B(x)$. A Σ-type like $\Sigma\, x^A.\, B(x)$ therefore can be read as an existence statement: "there is a term x of type A with the property $B(x)$".

From this point onwards, we also want to relax typing in the sense that

- convertible terms should be considered equal during typing judgments:

$$\frac{\Gamma \vdash t\colon A \qquad t \equiv t' \qquad \Gamma \vdash t'\colon A}{\Gamma \vdash t'\colon A}$$

- and this should hold analogously for convertible types:

$$\frac{\Gamma \vdash t\colon A \qquad A \equiv A' \qquad \Gamma \vdash A'\colon \mathbb{T}}{\Gamma \vdash t\colon A'}$$

This allows us to exchange convertible terms or types during a typing judgement, and we will freely make use of this without making the use of the rules explicit.

[8] This corresponds to the notation $\forall x \in A :\ B(x)$ in set theory.

2.2.3 Base Types

We now want to add some base types to the type theory, similarly to how the natural numbers are axiomatically added to ZF. Indeed we will also start by adding the **type of natural numbers**. The formation rules introducing the primitive symbols $\mathbb{N}, 0$ and S together with their typing judgments are:

$$\frac{}{\Gamma \vdash \mathbb{N} : \mathbb{T}} \qquad \frac{}{\Gamma \vdash 0 : \mathbb{N}} \quad \frac{}{\Gamma \vdash S : \mathbb{N} \to \mathbb{N}}$$

We further add a primitive symbol $E_{\mathbb{N}}$ with typing rule:

$$\frac{\Gamma \vdash P : \mathbb{N} \to \mathbb{T} \qquad \Gamma \vdash p_0 : P\,0 \qquad \Gamma \vdash R : \Pi n^{\mathbb{N}}.\, P\,n \to P(Sn)}{\Gamma \vdash E_{\mathbb{N}}\,P\,p_0\,R : \Pi n^{\mathbb{N}}.\, P\,n}$$

We will often write $E_{\mathbb{N}}\ \_\ p_0\,R$ instead of $E_{\mathbb{N}}\,P\,p_0\,R$, when P can be inferred, and will do so for all other eliminators to come.

The elimination rule of $E_{\mathbb{N}}$ should be reminiscent of the **induction principle** for natural numbers. To show that for all $n : \mathbb{N}$ we have $P\,n$, we need to provide a term of $P\,0$ (i.e. the base case), and a term for $\Pi n^{\mathbb{N}}.\, P\,n \to P(Sn)$ (the inductive step). Further adding the conversion rules $E_{\mathbb{N}}\,P\,p_0\,R\,0 \succ 0$ and $E_{\mathbb{N}}\,P\,p_0\,R\,(Sn) \succ R\,n\,(E_{\mathbb{N}}\,P\,p_0\,R\,n)$ for the eliminator –describing its computational behavior– we have now implemented induction and recursion on natural numbers. To give an example of its use, we show how addition on the natural numbers can now be defined, which ought to satisfy the following two equations:

$$\begin{aligned} \mathsf{Add}\,0\,m &\equiv m \\ \mathsf{Add}\,(Sn)\,m &\equiv S(\mathsf{Add}\,n\,m) \end{aligned}$$

We can abstract over m in these equations to get:

$$\begin{aligned} \mathsf{Add}\,0 &\equiv \lambda y.\, y \\ \mathsf{Add}\,(Sn) &\equiv \lambda y.\, S((\mathsf{Add}\,n)\,y) = (\lambda x A y.\, S(A\,y))\,n\,(\mathsf{Add}\,n) \end{aligned}$$

where we can read off that with $P := \lambda n^{\mathbb{N}}.\, \mathbb{N} \to \mathbb{N}$, $p_0 := \lambda y.\, y$ and $R := \lambda x A y.\, S(A\,y)$ the term $\mathsf{Add} := E_{\mathbb{N}}\,P\,p_0\,R$ has the desired computational behavior of addition, and with the elimination principle one can check that the term indeed has the desired type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$.

We can now prove that $2 := SS0$ is of type $\mathbb{N}$

$$\dfrac{\dfrac{}{\vdash S : \mathbb{N} \to \mathbb{N}} \qquad \dfrac{\dfrac{}{\vdash S : \mathbb{N} \to \mathbb{N}} \qquad \dfrac{}{\vdash 0 : \mathbb{N}}}{\vdash S0 : \mathbb{N}}}{\vdash SS0 : \mathbb{N}}$$

and that $\mathsf{Add}\,2\,2$ reduces to $4 := SSSS0$:

$$\mathsf{Add}\,2\,2 = \mathsf{Add}(SS0)(SS0) \succ S\,\mathsf{Add}(S0)(SS0) \succ SS\,\mathsf{Add}\,0\,(SS0) \succ SSSS0 = 4.$$

Following natural numbers, we introduce the **type of booleans**:

$$\frac{}{\Gamma \vdash \mathbb{B} : \mathbb{T}} \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \mathbb{B}} \quad \frac{}{\Gamma \vdash \mathsf{ff} : \mathbb{B}} \qquad \frac{\Gamma \vdash P : \mathbb{B} \to \mathbb{T} \qquad \Gamma \vdash p_{\mathsf{tt}} : P\,\mathsf{tt} \qquad \Gamma \vdash p_{\mathsf{ff}} : P\,\mathsf{ff}}{\Gamma \vdash E_{\mathbb{B}}\,P\,p_{\mathsf{tt}}\,p_{\mathsf{ff}} : \Pi b^{\mathbb{B}}.\,P\,b} \tag{2.5}$$

with conversion rules $E_{\mathbb{B}}\,P\,p_{\mathsf{tt}}\,p_{\mathsf{ff}}\,\mathsf{tt} \succ p_{\mathsf{tt}}$ and $E_{\mathbb{B}}\,P\,p_{\mathsf{tt}}\,p_{\mathsf{ff}}\,\mathsf{ff} \succ p_{\mathsf{ff}}$, which make the eliminator behave like "$\mathsf{if}\ b\ \mathsf{then}\ p_{\mathsf{tt}}\ \mathsf{else}\ p_{\mathsf{ff}}$". We can easily define negation on booleans by $\neg_{\mathbb{B}} := (E_{\mathbb{B}}\ \_\ \mathsf{ff}\ \mathsf{tt})$ and can further use $\neg_{\mathbb{B}}$ to define a term $\mathsf{eqd}_{\mathbb{B}} : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$ satisfying the equations

$$\begin{aligned} \mathsf{eqd}_{\mathbb{B}}\,\mathsf{tt}\,\mathsf{tt} &\equiv \mathsf{tt} & \mathsf{eqd}_{\mathbb{B}}\,\mathsf{ff}\,\mathsf{ff} &\equiv \mathsf{tt} \\ \mathsf{eqd}_{\mathbb{B}}\,\mathsf{tt}\,\mathsf{ff} &\equiv \mathsf{ff} & \mathsf{eqd}_{\mathbb{B}}\,\mathsf{ff}\,\mathsf{tt} &\equiv \mathsf{ff} \end{aligned}$$

therefore telling us whether two booleans are equal or not. The four equations from above are summarized by the following two:

$$\mathsf{eqd}_{\mathbb{B}}\,\mathsf{tt} \equiv \lambda x.\,x \qquad \mathsf{eqd}_{\mathbb{B}}\,\mathsf{ff} \equiv \neg_{\mathbb{B}}$$

leading to the conclusion that we should define

$$\mathsf{eqd}_{\mathbb{B}} := E_{\mathbb{B}}\ \_\ (\lambda x.x)(\neg_{\mathbb{B}}). \tag{2.6}$$

Next we introduce the **empty type**:

$$\frac{}{\Gamma \vdash \mathbb{0} : \mathbb{T}} \qquad \frac{\Gamma \vdash P : \mathbb{0} \to \mathbb{T}}{\Gamma \vdash E_{\mathbb{0}}\,P : \Pi f^{\mathbb{0}}.\,P\,f}$$

This type has no rules introducing any term and plays a similar role as the empty set in set theory and falsity in logic. This is by virtue of its elimination rule, which allows the construction of a term for any type, provided we ever have a term of the empty type.

Lemma 2.5 *Given a term $f : \mathbb{0}$ we can construct a term of any type A.*

Proof We show that $E_{\mathbb{0}}(\lambda x.A)f$ is a term of type A:

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \mathbb{0} \vdash A : \mathbb{T}}{\vdash (\lambda x.\,A) : \mathbb{0} \to \mathbb{T}}}{\vdash E_{\mathbb{0}}\,(\lambda x.\,A) : \Pi z^{\mathbb{0}}.\,(\lambda x.\,A)f}}{\vdash E_{\mathbb{0}}\,(\lambda x.\,A) : \Pi z^{\mathbb{0}}.\,A} \qquad \vdash f : \mathbb{0}}{\vdash E_{\mathbb{0}}\,(\lambda x.\,A)f : A[f/z]}}{\vdash E_{\mathbb{0}}\,(\lambda x.\,A)f : A}$$

Where both branches are now closed by the assumptions. □

Disregarding the left branch, which essentially only checks for correct typing, the deduction in Lemma 2.5 establishes

$$\frac{\vdash f : \mathbb{0}}{\vdash E_{\mathbb{0}}\,(\lambda x.\, A) f : A}$$

which is the CHdBI equivalent to the explosion rule in Definition 3.4.

Next we introduce two data-types, starting with the **list type** over a given $X : \mathbb{T}$:

$$\frac{}{\Gamma \vdash \mathcal{L}(X) : \mathbb{T}} \qquad \frac{}{\Gamma \vdash [\,] : \mathcal{L}(X)} \qquad \frac{}{\Gamma \vdash \mathsf{Cons} : X \to \mathcal{L}(X) \to \mathcal{L}(X)}$$

where $[\,]$ is to be understood as the empty list and we write $x :: \ell$ for $\mathsf{Cons}\, x\, \ell$, which stands for appending an element x to the list ℓ. The eliminator for lists over X is

$$\frac{\Gamma \vdash P : \mathcal{L}(X) \to \mathbb{T} \qquad \Gamma \vdash p_0 : P\,[\,] \qquad \Gamma \vdash C : \Pi\, x\, \ell.\, P\, \ell \to P(x :: \ell)}{\Gamma \vdash E_{\mathcal{L}(X)}\, P\, p_0\, C : \Pi \ell.\, P\, \ell}$$

Lists are generally useful whenever we want to store finitely many elements of some type. They can also be used to give a definition of finite sets and finiteness for types. We will adapt the conventional notation and write lists as $[23, 11, 21]$ instead of $23 :: 11 :: 21 :: [\,]$.

The next type we introduce will allow the definition of partial functions, meaning functions which are allowed to return no value on some inputs. For example, we might want to define a function $\mathsf{give} : \mathbb{N} \to \mathcal{L}(X) \to X$ which takes a natural number n as well as a list $\ell : \mathcal{L}(X)$ and returns the element at position n of the list. Problems arise however if n exceeds the number of elements in the list, or even worse, if the list is empty; $\mathsf{give}\, n\, [\,]$ cannot be defined for any n. We therefore define an **option type** over X with the possibility to return the empty value $\emptyset$.

$$\frac{}{\Gamma \vdash \mathcal{O}(X) : \mathbb{T}} \qquad \frac{}{\Gamma \vdash \emptyset : \mathcal{O}(X)} \qquad \frac{}{\Gamma \vdash \mathsf{Some} : X \to \mathcal{O}(X)}$$

$$\frac{\Gamma \vdash P : \mathcal{O}(X) \to \mathbb{T} \qquad \Gamma \vdash p_0 : P\,\emptyset \qquad \Gamma \vdash f : \Pi x.\, P(\mathsf{Some}\ x)}{\Gamma \vdash E_{\mathcal{O}(X)}\, P\, p_0\, f : \Pi s.\, P\, s}$$

These rules state that any term of $\mathcal{O}(X)$ is either the empty value $\emptyset$ (also called **none**) or is carrying some term $x : X$ packaged in $\mathsf{Some}\, x$. This allows us to define a function $\mathsf{give} : \mathbb{N} \to \mathcal{L}(X) \to \mathcal{O}(X)$ by recursion, returning $\emptyset$ in the case where the list is empty:

$$\begin{aligned} \mathsf{give}\, n\, [\,] &:= \emptyset \\ \mathsf{give}\, 0\, (x :: \ell) &:= \mathsf{Some}\, x \\ \mathsf{give}\, (Sn)\, (x :: \ell) &:= \mathsf{give}\, n\, \ell \end{aligned}$$

The above definition of the function can be turned into one that in defined in terms of the eliminators $E_{\mathcal{L}(X)}$ and $E_{\mathbb{N}}$.

We do want to implement one last feature to the type theory, and it is best illustrated by pointing out an inconvenience about the function give. The function we have presented above only works on lists of type $\mathcal{L}(X)$, meaning for a fixed type X. We cannot apply it to a list of type $\mathcal{L}(\mathbb{N})$. So we should have written give_X to be precise, and we would need to define $\mathsf{give}_{\mathbb{N}}$ in the other case as well. Of course it is also clear that the definitions will not differ much, apart from the type that the list runs over. So what we really want to define is a version of give where the type of the list can also be indicated at the start:

$$\mathsf{give}\colon \Pi(X\colon \mathbb{T}).\, \mathbb{N} \to \mathcal{L}(X) \to \mathcal{O}(X)$$

But once we check whether this type of give is well typed, we run into a problem:

$$\frac{\vdash \mathbb{T}\colon \mathbb{T} \qquad (X\colon \mathbb{T}) \vdash \mathbb{N} \to \mathcal{L}(X) \to \mathcal{O}(X)}{\vdash \Pi(X\colon \mathbb{T}).\, \mathbb{N} \to \mathcal{L}(X) \to \mathcal{O}(X)}$$

For the left branch we need to show $\vdash \mathbb{T}\colon \mathbb{T}$, which at the present moment is not possible. The question whether it is a good idea to assume $\vdash \mathbb{T}\colon \mathbb{T}$ (i.e. that "the type of all types is a type") should be reminiscent of the –known to be problematic– question about the "set of all sets". Indeed it turns out that naively adding $\vdash \mathbb{T}\colon \mathbb{T}$ introduces an inconsistency into the theory [19, 24]. There are however resolutions to this problem. In the CIC, an infinite, cumulative hierarchy of type universes $\mathbb{T}_i$ satisfying $\mathbb{T}_i\colon \mathbb{T}_j$ for $i \leq j$ is used, together with a necessary adaption of the typing rule for Π-types. It is then however possible to hide these details from any user of the theory. From a practical standpoint therefore, this allows us to simply use $\vdash \mathbb{T}\colon \mathbb{T}$ as a typing judgment.

The final version of give is then a **polymorphic** function, meaning a function which can also take inputs of type $\mathbb{T}$. Apart from a whole range of new function definitions, this also allows us to formulate statements involving all list types, by starting with $\forall X^{\mathbb{T}} \ell^{\mathcal{L}(X)} \ldots$, and instead of adding rules to define list and option types for every type, we can formulate rules incorporating this quantification, here illustrated for option types:

$$\frac{}{\Gamma \vdash \mathcal{O}\colon \mathbb{T} \to \mathbb{T}} \qquad \frac{}{\Gamma \vdash \emptyset\colon \Pi X^{\mathbb{T}}.\, \mathcal{O}(X)} \qquad \frac{}{\Gamma \vdash \mathsf{Some} : \Pi X^{\mathbb{T}}.\, X \to \mathcal{O}(X)}$$

2.2.4 Propositions as Types

With the types we have introduced up to this point, we can finally complete our PAT viewpoint between type theory and higher order intuitionistic logic. Due to the similarity of respective inference rules we have the following correspondences:

Type Theory	$\to$	$\times$	$+$	$\mathbb{0}$	$\mathcal{O}(\mathbb{0})$	Π	Σ
Intuitionistic Logic	$\to$	$\wedge$	$\vee$	$\bot$	$\top$	$\forall$	$\exists$

We can therefore represent any mathematical proposition by some type, using the translation of symbols indicated above [9]. We will now start to take the freedom to only give the usual text-style mathematical proofs for propositions and typing judgments; outlining only the most important parts which could then be fleshed out to full constructions. We will still give detailed deductions whenever we feel that they should be instructive, but will start fanning them out over time. Similarly, because of the close connection of the Π-types and $\forall$ quantification from the PAT viewpoint, we will use them interchangeably, and will only use $\forall$ starting Chapter 3.

2.2.5 The Type of Propositions

We could try to reintroduce classical reasoning into the type theory by adding the law of excluded middle in the form of a new rule:

$$\frac{}{\Gamma \vdash D \colon \forall A^{\mathbb{T}}.\, A + (A \to \mathbb{0})} \tag{2.7}$$

which claims that there is a term D which can yield, for every type A, a term of type A or a term of type $A \to \mathbb{0}$. Contrary to the other principles we have added, there is however no computational justification for this. A λ-term of this type would yield a solution for some version of the halting problem. Adding this non-computational principle would therefore destroy the computational interpretation of the whole type theory, as there is no more guarantee that every proof only employs informative reasoning principles. There is however a different and safer way to enable the inclusion of non-computational principles. For this, we introduce a new type universe $\mathbb{P}$[10] of propositions at the very bottom of the hierarchy, which is used to tag propositions which do not admit a computational interpretation. The inference rules for this type are then controlling in which way non-computational statements can still be used in conjunction with the rest of the type theory. This separates terms of type $\mathbb{P}$ from other types, leaving the type theory as a whole intact. Concerning the inference rules, we introduce the symbols $\vee$ and $\exists$ as primitives into the type theory [11], with the rules:

$$\frac{\Gamma \vdash A \colon \mathbb{P} \qquad \Gamma \vdash B \colon \mathbb{P}}{\Gamma \vdash A \vee B \colon \mathbb{P}} \qquad \frac{\Gamma \vdash a \colon A}{\Gamma \vdash o_1 a \colon A \vee B}$$

$$\frac{\Gamma \vdash h_1 \colon A \to Q \qquad \Gamma \vdash h_2 \colon B \to Q \qquad \Gamma \vdash Q \colon \mathbb{P}}{\Gamma \vdash E_\vee\, h_1\, h_2 \colon A \vee B \to Q} \qquad \frac{\Gamma \vdash b \colon B}{\Gamma \vdash o_2 b \colon A \vee B}$$

[9] Negations $\neg A$ are defined as $A \to \bot$ in the logic and correspond to the types $A \to \mathbb{0}$ in the type theory.

[10] This is not the historical reason for the introduction of $\mathbb{P}$.

[11] We highlighted earlier that $+$ and Σ on the level of the type theory correspond to $\vee$ and $\exists$ respectively in intuitionistic logic. We are now reusing the same symbols $\vee$ and $\exists$, but for new constructs on the level of the type theory, which do not correspond to anything in the logic.

$$\frac{\Gamma \vdash A\colon \mathbb{T} \qquad \Gamma, (x\colon A) \vdash P\colon \mathbb{P}}{\Gamma \vdash \exists x^A.\, P\colon \mathbb{P}} \qquad \frac{\Gamma \vdash a\colon A \qquad \Gamma \vdash h\colon P[a/x]}{\Gamma \vdash (a, h)\colon \exists x^A.\, P}$$

$$\frac{\Gamma \vdash f\colon \Pi x^A.\, (P \to Q) \qquad \Gamma \vdash Q\colon \mathbb{P}}{\Gamma \vdash E_\exists f\colon (\exists x^A.\, P) \to Q}$$

where, in contrast to the very similar rules for $+$ and Σ-types, the elimination rules for $\vee$ and $\exists$ make sure that, since $A \vee B$ and $\exists x.\, P\, x$ are tagged by $\mathbb{P}$ and therefore to be considered as non-informative, they can only be used to prove other non-informative propositions $Q\colon \mathbb{P}$. We can still prove non-informative statements from their informative counterpart. For example, given $A, B\colon \mathbb{P}$ we have a proof of $A + B \to A \vee B$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{\overline{(a\colon A) \vdash a\colon A}}}{(a\colon A) \vdash o_1 a\colon A \vee B}}{\vdash \lambda a.\, o_1 a\colon A \to A \vee B} \qquad \dfrac{\dfrac{\dfrac{\overline{\overline{(b\colon B) \vdash b\colon B}}}{(b\colon B) \vdash o_2 b\colon A \vee B}}{\vdash \lambda b.\, o_2 b\colon B \to A \vee B}}{\vdash E_+(\lambda a.\, o_1 a)(\lambda b.\, o_2 b)\colon A + B \to A \vee B}$$

But we will not be able to prove $A \vee B \to A + B$, since we cannot show $\vdash A + B\colon \mathbb{P}$. This is just as intended: $A \vee B : \mathbb{P}$ is not supposed to have computational content, contrary to $A + B$ which is supposed to carry a construction of either A or B.

We can now use $\mathbb{P}$ to formulate the law of excluded middle as well as other conventional logical notions and principles of classical and intuitionistic logic:

Definition 2.6

- $\mathsf{definite}\,(P\colon \mathbb{P}) := P \vee \neg P$
- $\mathsf{stable}\,(P\colon \mathbb{P}) := \neg\neg P \to P$
- $\mathsf{Definite}\,(X\colon \mathbb{T})(p : X \to \mathbb{P}) := \forall x.\, \mathsf{definite}\,(p\, x)$
- $\mathsf{Stable}\,(X\colon \mathbb{T})(p : X \to \mathbb{P}) := \forall x.\, \mathsf{stable}\,(p\, x)$
- $\mathsf{LEM} := \forall P\colon \mathbb{P}.\, \mathsf{definite}\ P$ *(Law of Excluded Middle)*
- $\mathsf{DNE} := \forall P\colon \mathbb{P}.\, \mathsf{stable}\ P$ *(Double Negation Elimination)*
- $\mathsf{MP} := \forall (f\colon \mathbb{N} \to \mathbb{N}).\, \mathsf{stable}\,(\exists n.\, fn = 0)$ *(Markov's Principle)*

LEM as stated above cannot be proven inside of our type theory and neither can its negation [53]. Therefore if we desire to use classical reasoning, we can now do so, since

$$\frac{}{\Gamma \vdash \ell\colon \mathsf{LEM}}$$

can consistently be added as a rule. Note that LEM differs from Equation (2.7) as it only claims to have a decision of P or $\neg P$ for every proposition $P\colon \mathbb{P}$ instead of

every type $P\colon \mathbb{T}$. The fact that LEM is not provable does not entail that there are no definite propositions; we will soon encounter some examples. Constructively $\neg\neg P$ does not yield a proof of P; it only states that it is impossible for P to be false, assuring that it is in the realm of the possible to find a proof of P. We will therefore also read $\neg\neg P$ as "**potentially** P" [2].

We can make use of the constructive framework to study how classical reasoning principles relate to one another. For example we have:

Fact 2.7 *Every definite proposition is stable.*

Proof Let $P\colon \mathbb{P}$ be definite. Assuming $\neg\neg P$ we need to show P. Using that P is definite, we get two cases:

- If P, then we are done.
- If $\neg P$ then combined with $\neg\neg P$ we get $\bot$ and therefore P by the explosion rule.

□

Fact 2.8 LEM *is equivalent to* DNE.

Proof The implication from left to right follows by Fact 2.7. For the converse assume some $P\colon \mathbb{P}$, then using $D := P \vee \neg P$ we have a proof of $\neg\neg(P \vee \neg P)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg D \vdash D \to \bot}}{\neg D, P \vdash \bot}}{\neg D \vdash \neg P}}{\neg D \vdash P \vee \neg P} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg D, P \vdash P}}{\neg D, P \vdash P \vee \neg P}}{\neg D \vdash D \to \bot}}{\neg D \vdash \bot}}{\vdash \neg\neg(P \vee \neg P)}$$

and therefore $P \vee \neg P$ by DNE. □

To finish of this section, we will use $\mathbb{P}$ to introduce a notion of equality for every type. For every $A\colon \mathbb{T}$ and $a\colon A$ we add rules

$$\frac{}{\Gamma \vdash \mathsf{eq}_A\, a\colon A \to \mathbb{P}} \qquad \frac{}{\Gamma \vdash \mathsf{refl}_A\, a\colon \mathsf{eq}_A\, a\, a} \qquad \frac{\Gamma \vdash P\colon A \to \mathbb{T} \qquad \Gamma \vdash h\colon P\, a}{\Gamma \vdash E_{\mathsf{eq}}\, a\, P\, h\colon \Pi x^A.\, \mathsf{eq}_A\, a\, x \to P\, x}$$

where we will usually write $x =_A y$ instead of $\mathsf{eq}_A\, x\, y$ and $x = y$ if the type A can be inferred. The elimination rule E_{eq} realizes substitution under equality: Given a proof of $P\, a$ and given a proof of the equality $x = a$, it allows us to conclude $P\, x$. With the equality type we can now express a whole range of interesting mathematical statements. To illustrate this, we start by showing the disjointness of the terms in $\mathbb{B}$ and finish by showing the agreement of $=_{\mathbb{B}}$ and $\mathsf{eqd}_{\mathbb{B}}$.

Lemma 2.9 (Disjointness) *The term* tt *is not equal to the term* ff.

Proof This is shown by verifying that the type $\mathsf{ff} = \mathsf{tt} \to \mathbb{0}$ is inhabited. The idea of the proof is that if we have $\mathsf{ff} = \mathsf{tt}$, then $\mathbb{0} \equiv (E_{\mathbb{B}}\_\,\mathbb{0}\,\mathbb{N})\,\mathsf{tt} \equiv (E_{\mathbb{B}}\_\,\mathbb{0}\,\mathbb{N})\,\mathsf{ff} \equiv \mathbb{N}$ showing that under this particular assumption, $\mathbb{0}$ is inhabited.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash P\colon \mathbb{B} \to \mathbb{T} \qquad \vdash \mathbb{0}\colon P\,\mathsf{tt} \qquad \vdash \mathbb{N}\colon P\,\mathsf{ff}}{\vdash \Pi b^{\mathbb{B}}.\, P\,b}}{\vdash (E_{\mathbb{B}}\,P\,\mathbb{0}\,\mathbb{N})\colon \Pi b^{\mathbb{B}}.\,\mathbb{T}}}{\vdash (E_{\mathbb{B}}\,P\,\mathbb{0}\,\mathbb{N})\colon \mathbb{B} \to \mathbb{T}} \qquad \dfrac{\vdash \mathbb{N}}{\vdash (E_{\mathbb{B}}\,P\,\mathbb{0}\,\mathbb{N})\,\mathsf{ff}}}{\vdash \Pi x^{\mathbb{B}}.\,\mathsf{ff} = x \to (E_{\mathbb{B}}\,P\,\mathbb{0}\,\mathbb{N})\,x} \qquad \dfrac{}{\vdash \mathsf{tt}\colon \mathbb{B}}}{\vdash \mathsf{ff} = \mathsf{tt} \to (E_{\mathbb{B}}\,P\,\mathbb{0}\,\mathbb{N})\,\mathsf{tt}}}{\vdash \mathsf{ff} = \mathsf{tt} \to \mathbb{0}}$$

The right branch $\vdash \mathbb{N}$ can be finished by providing any natural number e.g. $23\colon \mathbb{N}$. For the left branch to work out, we simply choose $P := \lambda x^{\mathbb{B}}.\mathbb{T}$. The final proof-term is $E_{\mathsf{eq}}\,\mathsf{ff}\,\big(E_{\mathbb{B}}(\lambda x^{\mathbb{B}}.\mathbb{T})\,\mathbb{0}\,\mathbb{N}\big)\,23\,\mathsf{tt}$. □

Before we can show the agreement of the two equalities on $\mathbb{B}$, we need to verify that equality $=$ satisfies the expected symmetry and transitivity laws.

Lemma 2.10 *For any type* A *and* $x, y, z\colon A$ *we have*

1. $x = y \to y = x$ *(Symmetry)*
2. $x = y \to y = z \to x = z$ *(Transitivity)*

Proof 1. For this we need to show that there is a term of type $\Pi x^A y^A.\, x = y \to y = x$. By choosing $P := \lambda a^A.\, \mathsf{eq}_A\, a\, x$ we get:

$$\dfrac{\dfrac{(x\colon A) \vdash P\colon A \to \mathbb{P} \qquad \dfrac{(x\colon A) \vdash \mathsf{eq}_A\, x\, x}{(x\colon A) \vdash P\, x}}{(x\colon A) \vdash \Pi y^A.\, \mathsf{eq}_A\, x\, y \to P\, y}}{\vdash \Pi x^A y^A.\, \mathsf{eq}_A\, x\, y \to \mathsf{eq}_A\, y\, x}$$

where both branches can now be closed by making use of the rules for equality. The resulting proof-term is $E_{\mathsf{eq}}\, x\, (\lambda a^A.\, \mathsf{eq}_A\, a\, x)(\mathsf{refl}_A\, x)$.

2. We show that the type $\Pi x^A y^A z^A.\, x = y \to y = z \to x = z$ is inhabited. Using $P := \lambda a^A.\, a = z \to x = z$ we have

$$\dfrac{\dfrac{(x,y,z\colon A)\vdash P\colon A\to\mathbb{T}}{(x,y,z\colon A)\vdash \Pi a^A.\, x=a\to P\,a}\qquad \dfrac{\dfrac{(x,y,z\colon A)\vdash x=z\to x=z}{(x,y,z\colon A)\vdash P\,x}}{(x,y,z\colon A)\vdash \Pi a^A.\, x=a\to P\,a}\qquad (x,y,z\colon A)\vdash y\colon A}{\dfrac{\dfrac{(x,y,z\colon A)\vdash (\Pi a^A.\, x=a\to P\,a)\,y}{(x,y,z\colon A)\vdash x=y\to P\,y}}{\vdash \Pi x^A y^A z^A.\, x=y\to y=z\to x=z}}$$

where all branches can now be closed easily.
This yields the proof-term $\lambda x^A y^A z^A.\, E_{\mathsf{eq}}\,x\,(\lambda a^A.\, a=z\to x=z)(\lambda e^{x=z}.e)\,y$. □

Fact 2.11 (Case Analysis) *For every $b\colon\mathbb{B}$ either $b=\mathsf{tt}$ or $b=\mathsf{ff}$.*

Proof We show that $\Pi b^{\mathbb{B}}.\,(b=\mathsf{tt})+(b=\mathsf{ff})$ is inhabited. This follows immediately by using the elimination rule for $\mathbb{B}$:

$$\dfrac{\vdash \_\colon\mathbb{B}\to\mathbb{T}\qquad \dfrac{\vdash \mathsf{ff}=\mathsf{ff}}{\vdash (\mathsf{ff}=\mathsf{tt})+(\mathsf{ff}=\mathsf{ff})}\qquad \dfrac{\vdash \mathsf{tt}=\mathsf{tt}}{\vdash (\mathsf{tt}=\mathsf{tt})+(\mathsf{tt}=\mathsf{ff})}}{\vdash \Pi b^{\mathbb{B}}.\,(b=\mathsf{tt})+(b=\mathsf{ff})}$$

Lemma 2.12 *For any $x,y\colon\mathbb{B}$, we have $x=y$ if and only if $\mathsf{eqd}_{\mathbb{B}}\,x\,y$ returns tt.*

Proof To construct a term of type $\Pi x^{\mathbb{B}} y^{\mathbb{B}}.\,\mathsf{eqd}_{\mathbb{B}}\,x\,y=\mathsf{tt}\leftrightarrow x=y$ we use case analysis on both x and y giving us the four cases seen in Equation (2.5). For every case it is easy to check that the equivalence holds. □

2.3 The Coq Proof-Assistant

In the previous sections, we sketched out major parts of the *calculus of inductive constructions* (CIC), a type theory which makes up the foundation of the proof-assistant Coq. Coq is an open-source software, first released in 1989, and is based on CIC which was most notably developed by Thierry Coquand, Gérard Huet and Christine Pauline-Mohring [7, 35]. The majority is implemented in the programming language OCaml and Coq itself constitutes a dependently typed functional programming language with the feature that it has the strong normalization property; meaning every definable function terminates. The main focus of Coq however lies in its use as a logical system. It allows the definition of mathematical objects, formulation of statements and most importantly proofs. The user can use commands and tactics to execute proof steps, while the proof assistant keeps track of these actions and displays the current proof goal as well as the available assumptions. Since Coq comes with an extensive standard library covering basic results on natural numbers, rationals, reals and various

data types, it is often possible to finish proof steps with the use of these included results or some basic automation. For example, boolean formulas and some intuitionistic propositions can automatically be proven by respective automation tactics. A finished proof can then be checked for correctness and the user will be informed about possible mistakes and their location in the proof.

Chapter 3

First-Order Logic

In Chapter 2 we have described a constructive type theory (CTT), which we will now use as the meta-theory for all further mathematical work. The goal is to study some results about the first-order theory of **Peano arithmetic** (PA) and also more specifically **Heyting arithmetic** (HA), which has the same axiomatization, but uses intuitionistic first-order logic. We describe first-order logic inside of CTT, by inductively defining formulas, terms and the deduction system. We then define a semantics for this logic, which uses Tarski-models and interprets given formulas over the respective domain of the model. The type of natural numbers $\mathbb{N}$ will then naturally be a model of HA. Having finished this setup, we can then use our meta-theory to reason about proofs and semantic entailment of first-order logic, which will be important for both results that we want to investigate: The undecidability of PA in Chapter 5 and Tennenbaum's theorem in Chapter 6.

Before specializing to one theory, we keep the definition of first-order logic general and fix some arbitrary signature $\Sigma = (\mathcal{F}; \mathcal{P})$.

3.1 Syntax and Natural Deduction

Definition 3.1 *We define terms* $t\colon \mathsf{tm}$ *and formulas* $\varphi\colon \mathsf{fm}$ *inductively.*

$$s, t\colon \mathsf{tm} ::= x_n \mid f\, v \qquad (f\colon \mathcal{F},\ n\colon \mathbb{N},\ v\colon \mathsf{tm}^{|f|})$$
$$\alpha, \beta\colon \mathsf{fm} ::= P\, v \mid \alpha \dot{\to} \beta \mid \alpha \mathbin{\dot{\wedge}} \beta \mid \alpha \mathbin{\dot{\vee}} \beta \mid \dot{\forall}\, \alpha \mid \dot{\exists}\, \beta \qquad (P\colon \mathcal{P},\ v\colon \mathsf{tm}^{|P|}).$$

Where $|f|$ *and* $|P|$ *are the arities of the function symbol* f *and predicate symbol* P *respectively.*

Remark 3.2 If we assume that the signature contains a unary function symbol f and a predicate for equality $\doteq$, one of the formulas we can form is

$$\dot{\forall}\, \dot{\forall}\, fx_1 \doteq fx_0 \dot{\to} x_1 \doteq x_0$$

Here, we are using the so called de Bruijn [10] indices to realize the binding of variables to quantifiers. For the reader who is not accustomed to this notation, we will now

illustrate how this binding works by showing how to recover the more conventional notation which uses *named* quantifiers $\dot{\forall} y$, $\dot{\exists} y$:

- Choose a variable x_k which still has an index k and replace it by a variable name y which is not yet in the formula.
- Move up the structure of the formula until k quantifiers have been passed; y will now be bound to the next quantifier.
- If the next quantifier is already named $Q\, z$, go back and change y to z.
- If the next quantifier Q is unnamed, change it to $Q\, y$.
- If there is no next quantifier nothing needs to be done.
- Repeat until no indexed variable remains in the formula.

If we start with the first x_0 in our example from above and use y as the new variable name, we get

$$\dot{\forall}\, \dot{\forall} y.\, f x_1 \doteq f y \mathrel{\dot{\to}} x_1 \doteq x_0.$$

Continuing with the second x_0 we get

$$\dot{\forall}\, \dot{\forall} y.\, f x_1 \doteq f y \mathrel{\dot{\to}} x_1 \doteq y.$$

and repeating the procedure on the remaining x_1's with new name x we get the more familiar looking

$$\dot{\forall} x\, \dot{\forall} y.\, f x \doteq f y \mathrel{\dot{\to}} x \doteq y.$$

For the rest of the text we will use both notations interchangeably, with a preference for the conventional one.

Definition 3.3 *Given a variable assignment $\sigma\colon \mathbb{N} \to \mathsf{tm}$ we recursively define **substitution** on terms by $x_k[\sigma]{:=}\sigma\, k$, $f\, v{:=}f(v[\sigma])$ and extend this definition to formulas by*

$$\begin{aligned} \bot[\sigma] &:= \bot & (\alpha \mathbin{\dot{\square}} \beta)[\sigma] &:= \alpha[\sigma] \mathbin{\square} \beta[\sigma] \\ (P\, v)[\sigma] &:= P\, (v[\sigma]) & (\dot{\nabla}\, \varphi)[\sigma] &:= \nabla(\varphi[\sigma]) \end{aligned}$$

where $\dot{\square}$ is any logical connective and $\dot{\nabla}$ any quantifier from the signature. By $\uparrow$ we designate the substitution $\lambda k.\, x_{Sk}$ shifting all variable indices up by one.

Definition 3.4 (Natural Deduction) *We define intuitionistic natural deduction $\vdash\colon \mathcal{L}\mathsf{fm} \to \mathsf{fm} \to \mathbb{P}$ inductively by the rules*

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

$$\frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \mathbin{\dot{\wedge}} \psi} \qquad \frac{\Gamma \vdash \phi \mathbin{\dot{\wedge}} \psi}{\Gamma \vdash \phi} \qquad \frac{\Gamma \vdash \phi \mathbin{\dot{\wedge}} \psi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \mathbin{\dot{\vee}} \psi} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \mathbin{\dot{\vee}} \psi} \qquad \frac{\Gamma \vdash \phi \mathbin{\dot{\vee}} \psi \quad \Gamma, \phi \vdash \theta \quad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}$$

$$\frac{\Gamma[\uparrow] \vdash \phi}{\Gamma \vdash \dot{\forall} \phi} \qquad \frac{\Gamma \vdash \dot{\forall} \phi}{\Gamma \vdash \phi[t]} \qquad \frac{\Gamma \vdash \phi[t]}{\Gamma \vdash \dot{\exists} \phi} \qquad \frac{\Gamma \vdash \dot{\exists} \phi \quad \Gamma[\uparrow], \phi \vdash \psi[\uparrow]}{\Gamma \vdash \psi}$$

where we get the classical variant by adding the rule

$$\frac{\Gamma \vdash \dot{\neg}\dot{\neg}\phi}{\Gamma \vdash \phi}$$

To clarify the distinction between the two, we write $\vdash_i$ for intuitionistic natural deduction and $\vdash_c$ for the classical one. In situations where we can be agnostic about what deduction we are using, we will simply write $\vdash$.

Fact 3.5 *If the signature Σ is enumerable, then so are the type of formulas* fm *and the type of deductions $\Gamma \vdash \varphi$.*

Definition 3.6 *A formula φ is said to be* ***bounded*** *by $N : \mathbb{N}$ if for the highest unbound variable x_n we have $n \leq N$, and* ***closed*** *if it is bounded by 0; meaning all variables are bound by quantifiers.*

Fact 3.7 *If $\varphi : \mathsf{fm}$ is bound by $N : \mathbb{N}$ and there is a substitution $\sigma : \mathbb{N} \to \mathsf{tm}$ such that $\Gamma \vdash \varphi[\sigma]$, then $\Gamma \vdash \exists^N \varphi$.*

3.2 Semantics

We will use the standard Tarski semantics for first-order logic. This means we use **models** which consist of a type D designating a domain, and interpret the functions and predicates symbols of the signature $(\mathcal{F}, \mathcal{P})$ as functions and predicates on the domain type. We can then extend this interpretation to formulas by also specifying how the logical symbols are supposed to be interpreted in the meta-logic. Here we make a first important choice. We will place all interpretations of logical symbols on the level of $\mathbb{P}$. Let's contrast this with one other choice we have available: Namely placing everything on type level. This means instead of interpreting e.g. the formula $\alpha \mathbin{\dot{\vee}} \beta$ as the proposition "$\mathcal{M} \vDash \alpha$ or $\mathcal{M} \vDash \beta$" we interpret it as a sum type $\mathcal{M} \vDash \alpha + \mathcal{M} \vDash \beta$, leading to a more restrictive model theory. For now we will stick to the placement into $\mathbb{P}$, and will come back to the other option in Section 6.8.2.

Definition 3.8 (Tarski Semantics) *A **model** $\mathcal{M}$ consists of a type D designating its domain together with functions $f^{\mathcal{M}} \colon D^{|f|} \to D$ and $P^{\mathcal{M}} \colon D^{|P|} \to \mathbb{P}$ giving every symbol of the signature an interpretation. We will abuse Notation and also use $\mathcal{M}$ to refer to the domain. In this context, functions $\rho \colon \mathbb{N} \to \mathcal{M}$ will be called environments and are used as variable assignments to recursively give interpretations to terms:*

$$\widehat{\rho}(x_k) := \rho\, k \qquad \widehat{\rho}(f\, v) := f^{\mathcal{M}}(\widehat{\rho}(v)) \qquad (v \colon \mathsf{tm}^n)$$

and extend this to formulas by

$$\begin{aligned} \mathcal{M} \vDash_\rho P\, v &:= P^{\mathcal{M}}(\widehat{\rho}(v)) & \mathcal{M} \vDash_\rho (\alpha \mathrel{\dot{\to}} \beta) &:= \mathcal{M} \vDash_\rho \alpha \to \mathcal{M} \vDash_\rho \beta \\ \mathcal{M} \vDash_\rho \alpha \mathrel{\dot{\wedge}} \beta &:= \mathcal{M} \vDash_\rho \alpha \wedge \mathcal{M} \vDash_\rho \beta & \mathcal{M} \vDash_\rho \alpha \mathrel{\dot{\vee}} \beta &:= \mathcal{M} \vDash_\rho \alpha \vee \mathcal{M} \vDash_\rho \beta \\ \mathcal{M} \vDash_\rho \dot{\forall}\, \alpha &:= \forall x \colon D.\ \mathcal{M} \vDash_{x;\rho} \alpha & \mathcal{M} \vDash_\rho \dot{\exists}\, \alpha &:= \exists x \colon D.\ \mathcal{M} \vDash_{x;\rho} \alpha \end{aligned}$$

*where $x;\rho$ is defined by $(x;\rho)\, 0 := x$, $(x;\rho)(Sn) := \rho\, n$ and is simply appending x as the first element of the environment ρ. We then say that a formula φ **holds in the model** $\mathcal{M}$ and write $\mathcal{M} \vDash \varphi$ if for every environment ρ we have $\mathcal{M} \vDash_\rho \varphi$. We extend this notation by writing $\mathcal{M} \vDash \Gamma$ if every formula in the list Γ holds in $\mathcal{M}$ and $\mathcal{M} \vDash \mathcal{T}$ iff $\forall \varphi.\, \mathcal{T}\, \varphi \to \mathcal{M} \vDash \varphi$ for a theory $\mathcal{T} \colon \mathsf{fm} \to \mathbb{P}$.*

Fact 3.9 (Soundness) *For any context Γ and formula φ we have that $\Gamma \vdash_i \varphi$ implies $\Gamma \vDash \varphi$.*

Remark 3.10 If $\varphi(x)$ is a unary formula and $m \colon \mathcal{M}$ we will write $\mathcal{M} \vDash \varphi(m)$ instead of $\forall \rho.\, \mathcal{M} \vDash_{m;\rho} \varphi(x)$. Analogously for any n-ary formula.

3.3 Peano Arithmetic

Having set up the machinery of first-order logic, we will now use this to study Peano arithmetic (PA) as a first-order theory. It's signature consists of symbols for the constant zero, the successor function, addition, multiplication and equality:

$$(\mathcal{F}_{\mathsf{PA}}; \mathcal{P}_{\mathsf{PA}}) := (O,\ \mathsf{S}\_\ ,\ \_ \oplus \_\ ,\ \_ \otimes \_\ ;\ \_ \doteq \_).$$

The finite core of PA axioms consists of statements characterizing the successor function:

$$\text{Disjointness}: \ \dot{\forall} x.\, Sx \doteq O \mathrel{\dot{\to}} \bot \qquad \text{Injectivity}: \ \dot{\forall} xy.\, Sx \doteq Sy \mathrel{\dot{\to}} x \doteq y$$

as well as addition and multiplication:

$$\begin{aligned} \oplus\text{-base}: \ & \dot{\forall} x.\, O \oplus x \doteq x & \oplus\text{-recursion}: \ & \dot{\forall} xy.\, (\mathsf{S}x) \oplus y \doteq S(x \oplus y) \\ \otimes\text{-base}: \ & \dot{\forall} x.\, O \otimes x \doteq O & \otimes\text{-recursion}: \ & \dot{\forall} xy.\, (\mathsf{S}x) \otimes y \doteq y \oplus x \otimes y \end{aligned}$$

We then get the full (and infinite) axiomatization of PA by adding the axiom scheme of induction, which in our meta-theory is a type-theoretic function on formulas:

$$\lambda\varphi.\,\varphi[O] \dot{\to} (\dot{\forall}\, x.\,\varphi[x] \dot{\to} \varphi[Sx]) \dot{\to} \dot{\forall}\, x.\,\varphi[x]$$

If instead of the induction scheme we add the axiom $\dot{\forall}\, x.\, x = O \mathbin{\dot{\vee}} \dot{\exists}\, y.\, x = Sy$, we get the theory Q known as **Robinson arithmetic**.

Definition 3.11 *We recursively define a function* $\overline{\cdot}\colon \mathbb{N} \to \mathsf{tm}$ *by* $\overline{0} := O$ *and* $\overline{n+1} := \mathsf{S}\overline{n}$*, giving every natural number a representation as a term. Any term* t *which is of the form* $\overline{n}$ *will be called* ***numeral****.*

Remark 3.12 For any function $f\colon X \to \mathbb{N}$ we will extend our notation and write $\overline{f}$ for the composition $f \circ \overline{\cdot}$ of f with the numeral function $\overline{\cdot}\colon \mathbb{N} \to \mathsf{tm}$.

In order to have the usual behavior we expect from $\doteq$, we also add the following axioms to PA:

$$\begin{aligned}
\text{Reflexivity}: &\ \dot{\forall}\, x.\, x \doteq x\\
\text{Symmetry}: &\ \dot{\forall}\, xy.\, x \doteq y \dot{\to} y \doteq x\\
\text{Transitivity}: &\ \dot{\forall}\, xyz.\, x \doteq y \dot{\to} y \doteq z \dot{\to} x \doteq z\\
\mathsf{S}\text{-equality}: &\ \dot{\forall}\, xy.\, x \doteq y \dot{\to} \mathsf{S}x \doteq \mathsf{S}y\\
\oplus\text{-equality}: &\ \dot{\forall}\, xyuv.\, x \doteq u \mathbin{\dot{\wedge}} y \doteq v \dot{\to} x \oplus y \doteq u \oplus v\\
\otimes\text{-equality}: &\ \dot{\forall}\, xyuv.\, x \doteq u \mathbin{\dot{\wedge}} y \doteq v \dot{\to} x \otimes y \doteq u \otimes v.
\end{aligned}$$

We also use further notations; one for expressing **less than** $x \mathbin{\dot{<}} y := \dot{\exists}\, k.\, \mathsf{S}(x \oplus k) \doteq y$ one for **less or equal** $x \mathbin{\dot{\leq}} y := \dot{\exists}\, k.\, x \oplus k \doteq y$ and one for **divisibility** $x \mid y := \dot{\exists}\, k.\, x \otimes k \doteq y$.

The formulas of PA can be classified based on the quantifier blocks they contain, giving them a natural hierarchy. We will only consider two levels of this hierarchy: the very base level Δ_0 only containing formulas wih bounded quantifiers and the level Σ_1 containing formulas from Δ_0 with arbitrarily many existential quantifiers in front of them.

Definition 3.13 *We inductively define the predicate* $\Delta_0 : \mathsf{fm} \to \mathbb{P}$ *by*

$$\frac{}{\Delta_0\, \bot} \qquad \frac{}{s \doteq t} \qquad \frac{\Delta_0\, \alpha \qquad \Delta_0\, \beta}{\Delta_0\, (\alpha \mathbin{\dot{\square}} \beta)}$$

where $\dot{\square}$ *is any logical connective. A* Δ_0 *formula preceded by* $n\colon \mathbb{N}$ *existential quantifiers will be called* Σ_1*.*

Lemma 3.14 *For any $\sigma\colon \mathbb{N} \to \mathsf{tm}$, the formula $\phi[\sigma]$ is Δ_0 if ϕ is Δ_0.*

Proof By induction on the formula ϕ. □

Closed Δ_0 formulas correspond to statements with no quantifiers and only involving equations with no free variables. For this type of formulas we can either proof them or their negation.

Theorem 3.15 *For any closed Δ_0 formula φ we have $\mathsf{PA} \vdash \varphi$ or $\mathsf{PA} \vdash \dot\neg\varphi$.*

Proof We proceed by induction on the formula.

- If $\varphi = \bot$ then we can show $\mathsf{PA} \vdash \dot\neg\bot$.
- If $\varphi = \alpha \mathbin{\dot\square} \beta$ for some connective $\dot\square$, then by the induction hypothesis we have that $\mathsf{PA} \vdash \alpha$ or $\mathsf{PA} \vdash \dot\neg\alpha$ as well as $\mathsf{PA} \vdash \beta$ or $\mathsf{PA} \vdash \dot\neg\beta$. For every connective, this leads to 4 cases which are easily handled.
- The case $\varphi = (s \doteq t)$ is covered by a result we will later show in Corollary 5.7.

The case where φ has some quantifier does not occur, since by Definition 3.13 Δ_0 formulas do not have quantifiers. □

Lemma 3.16 ($\boldsymbol{\Delta_0}$-Absoluteness) *Let $\mathcal{M} \vDash \mathsf{PA}$ and φ be any closed Δ_0 formula, then $\mathbb{N} \vDash \varphi \to \mathcal{M} \vDash \varphi$.*

Proof By Theorem 3.15 we have either $\mathsf{PA} \vdash \varphi$ or $\mathsf{PA} \vdash \dot\neg\varphi$. Since $\mathbb{N} \vDash \varphi$ we must have $\mathsf{PA} \vdash \varphi$ and therefore $\mathcal{M} \vDash \varphi$ by soundness. □

Lemma 3.17 *Let $\varphi_0(x, y)$ be a Δ_0 formula, then there is a Δ_0 formula $\gamma_0(z)$ such that $\mathsf{PA} \vdash \dot\exists x \mathbin{\dot\exists} y.\, \varphi_0(x, y) \mathrel{\dot\leftrightarrow} \dot\exists z.\, \gamma_0(z)$.*

Proof Sketch We let $\gamma_0(z) := \dot\exists x < z \mathbin{\dot\exists} y < z.\, \varphi_0(x, y)$. If we have $\dot\exists x \mathbin{\dot\exists} y.\, \varphi_0(x, y)$ then $z := x + y + 1$ shows $\dot\exists z.\, \gamma_0(z)$ and conversely, if $\dot\exists x < z \mathbin{\dot\exists} y < z.\, \varphi_0(x, y)$ then it is trivial to show $\dot\exists x \mathbin{\dot\exists} y.\, \varphi_0(x, y)$. □

Corollary 3.18 ($\boldsymbol{\dot\exists}$ Compression) *For any formula $\dot\exists^n \varphi_0$ with φ_0 being Δ_0, there is a Δ_0 formula γ_0 such that $\mathsf{PA} \vdash \dot\exists^n \varphi \mathrel{\dot\leftrightarrow} \dot\exists \gamma_0$.*

Proof We do a proof by induction on n, starting with $n = 1$ and using Lemma 3.17 in the inductive step. □

Lemma 3.19 *For any closed Σ_1 formula φ we have $\mathbb{N} \vDash \varphi \leftrightarrow \mathsf{PA} \vdash \varphi$.*

Proof By Corollary 3.18 there is a Δ_0 formula φ_0 such that $\dot\exists \varphi_0$ is equivalent to φ. The assumption $\mathbb{N} \vDash \dot\exists \varphi_0$ then gives us $n\colon \mathbb{N}$ with $\mathbb{N} \vDash \varphi_0(\overline{n})$. By Lemma 3.16 we then have $\mathsf{PA} \vdash \varphi_0(\overline{n})$ giving us $\mathsf{PA} \vdash \dot\exists \varphi_0$. The converse direction follows by soundness. □

Corollary 3.20 ($\boldsymbol{\Sigma_1}$ Absoluteness) *Let $\mathcal{M} \vDash \mathsf{PA}$ and φ be any closed Σ_1 formula, then $\mathbb{N} \vDash \varphi \to \mathcal{M} \vDash \varphi$.*

Chapter 4

Synthetic Computability

Usually, to talk about the notion of *computability*, one would develop a model of computation inside of set theory, like the Turing machine model, which can then be used to define what a computable function is. It then becomes possible to define notions like **decidability**. A predicate $p \subseteq \mathbb{N}$ would be considered to be *decidable* if there is a Turing machine computing the function $f : \mathbb{N} \to \{0, 1\}$ with $fx = 1 \leftrightarrow x \in p$.

When it comes to Coq, every *concrete* functions as defined by the user, in the absence of any additional axioms, can in principle be shown to be computable in a chosen model of computation, even in an automated way, by external tools [14]; meaning from the outside we can observe the computability of every function defined inside of the constructive type theory we work in. This then justifies a viewpoint where we simply consider every function inside of the type theory as computable, allowing many definitions, like the one for decidability, to be simplified. A predicate $p : \mathbb{N} \to \mathbb{P}$ is now called decidable if there exists a function $f : \mathbb{N} \to \mathbb{B}$ such that $fx = \mathsf{tt} \leftrightarrow p\,x$. Here we did not need to require the computability of f and to make reference to a concrete model of computation. This approach to the theory of computable functions is called **synthetic computability theory** [1, 39]. We can even go one step further and assume an axiom which then internalizes this viewpoint and will come back to this in Section 6.1.

We can now define notions of computability theory in this setting:

Definition 4.1

- $\mathsf{dec}(P : \mathbb{P}) := \Sigma\, b^{\mathbb{B}}.\, P \leftrightarrow b = \mathsf{tt}$
- $\mathsf{Dec}(p : X \to \mathbb{P}) := \exists (f : X \to \mathbb{B})\, \forall x.\, (p\,x \leftrightarrow fx = \mathsf{tt})$
- $\mathsf{Enum}(p : \mathbb{N} \to \mathbb{P}) := \exists (f : \mathbb{N} \to \mathbb{N})\ \forall x.\, p\,x \leftrightarrow \exists n.\, f\,n = S\,x$
- $\mathsf{Eq}(X : \mathbb{T}) := \forall x\,y :\, X.\, \mathsf{dec}(x = y)$

Furthermore, a type X is called ***discrete*** *if* $\mathsf{Eq}\,X$ *holds,* ***enumerable*** *if there is a function* $f\colon \mathbb{N} \to \mathcal{O}(X)$ *with* $\forall x\,\exists n.\, fn = \mathsf{Some}\,x$*, and* ***data type*** *if both hold.*

Fact 4.2 *For every proposition P, if* $\mathsf{dec}\,P$ *then* $P \vee \neg P$.

Proof By assumption there is a boolean $b\colon \mathbb{B}$ such that $P \leftrightarrow b = \mathsf{tt}$. By case analysis on b get two cases.

- If $b = \mathsf{tt}$ then P follows, hence we can show $P \vee \neg P$.
- In the case $b = \mathsf{ff}$, assume we had P, then this would imply $b = \mathsf{tt}$ and therefore $\mathsf{tt} = \mathsf{ff}$ which gives a contradiction. This shows $\neg P$ allowing us to prove $P \vee \neg P$.

□

Fact 4.3 *For any proposition P we have* $\mathsf{dec}(\neg P) \to \mathsf{stable}\,P \to \mathsf{dec}\,P$.

Lemma 4.4 *For any $x, y\colon \mathbb{N}$ we have* $\mathsf{dec}(x = y)$.

Proof We do an induction on $x\colon \mathbb{N}$ to prove the statement $\forall y.\, \mathsf{dec}(x = y)$.

- In the base case $x = 0$ we need to show $\mathsf{dec}(0 = y)$ for arbitrary y. Using case distinction for natural numbers, we know that either $y = 0$ or $y = Sz$ for some $z\colon \mathbb{N}$. The first case $\mathsf{dec}(0 = 0)$ is easy to verify. In the second case we need to show $\exists b\colon \mathbb{B}.\, 0 = Sz \leftrightarrow b = \mathsf{tt}$ which holds for the choice $b := \mathsf{ff}$.
- In the inductive step we nee to show $\forall y.\, \mathsf{dec}(Sx = y)$ while having the induction hypothesis $\forall y.\, \mathsf{dec}(x = y)$ at our disposal. We again proceed by a case distinction on y. The $y = 0$ case, $\mathsf{dec}(Sx = 0)$ is handled as above and in the remaining case we need to show $\mathsf{dec}(Sx = Sy)$. We know use the induction hypothesis by which we know that there is a boolean $b\colon \mathbb{B}$ such that $x = y \leftrightarrow b = \mathsf{tt}$. Therefore $Sx = Sy \leftrightarrow x = y \leftrightarrow b = \mathsf{tt}$, proving the claim. □

Chapter 5

Undecidability of Peano Arithmetic

In this chapter we will establish the undecidability of the first order theory of PA. More concretely, by means of a reduction, we will show that neither validity nor provability for arbitrary PA formulas can be decided. We will see that PA can naturally express the solvability of an arbitrary Diophantine equation $p = q$ by a formula $\varphi_{p,q}$ and if validity or provability were decidable for arbitrary formulas, we could decide it for $\varphi_{p,q}$ and therefore decide whether $p = q$ has a solution or not. The solvability problem of Diophantine equations however is famously know to be undecidable [30, 31, 29].

We will start off by showing that PA can do computations on numerals, which will be needed in the verification of the reduction. We then show an overall stronger undecidability result by identifying a finite fragment of PA which already turns out to be undecidable.

5.1 Computing on Numerals

We will now see that PA can do simple computations on numerals. This means that if we take two natural numbers e.g. 5 and 7 and add them to get 12, then PA can deductively show this result on the respective numerals i.e. we can show $\mathsf{PA} \vdash \overline{5} \oplus \overline{7} \doteq \overline{12}$.

Lemma 5.1 *For any $x, y \colon \mathbb{N}$ we have $\mathsf{PA} \vdash \overline{x + y} \doteq \overline{x} \oplus \overline{y}$.*

Proof We proceed by induction on $x \colon \mathbb{N}$.

In the base case we need to show $\mathsf{PA} \vdash \overline{y} \doteq \overline{0} \oplus \overline{y}$ which immediately follows by the $\oplus$-base axiom of PA. The successor case is shown by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{PA} \vdash \overline{x + y} \doteq \overline{x} \oplus \overline{y}}\,\text{Ind. hyp.}}{\mathsf{PA} \vdash \mathsf{S}\overline{x + y} \doteq \mathsf{S}(\overline{x} \oplus \overline{y})}\,\mathsf{S}\text{-equality}}{\mathsf{PA} \vdash \mathsf{S}\overline{x + y} \doteq (\mathsf{S}\overline{x}) \oplus \overline{y}}\,\oplus\text{-recursion}}{\mathsf{PA} \vdash \overline{Sx + y} \doteq \overline{Sx} \oplus \overline{y}}\,\text{Def. of numerals}$$

□

Lemma 5.2 *For any $x, y \colon \mathbb{N}$ we have $\mathsf{PA} \vdash \overline{x \cdot y} \doteq \overline{x} \otimes \overline{y}$.*

Proof Again we proceed by induction on $x \colon \mathbb{N}$.

In the base case we need to show $\mathsf{PA} \vdash \overline{0} \doteq \overline{0} \otimes \overline{y}$ which immediately follows by the $\otimes$-base axiom of PA. The successor case is shown by

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{PA} \vdash \overline{y} \doteq \overline{y}}\,\text{Reflexivity} \quad \dfrac{}{\mathsf{PA} \vdash \overline{x \cdot y} \doteq \overline{x} \otimes \overline{y}}\,\text{Ind. hyp.}}{\mathsf{PA} \vdash \overline{y} \oplus \overline{x \cdot y} \doteq \overline{y} \oplus \overline{x} \otimes \overline{y}}\,\oplus\text{-equality}}{\mathsf{PA} \vdash \overline{y} \oplus \overline{x \cdot y} \doteq (\mathsf{S}\overline{x}) \otimes \overline{y}}\,\otimes\text{-recursion}}{\mathsf{PA} \vdash \overline{y + x \cdot y} \doteq (\mathsf{S}\overline{x}) \otimes \overline{y}}\,\text{Lemma 5.1}}{\mathsf{PA} \vdash \overline{Sx \cdot y} \doteq \overline{Sx} \otimes \overline{y}}\,\text{Def. of numerals}$$

□

We can think of the previous results as homomorphism properties of the numeral function $\overline{\cdot}$.

Fact 5.3 *For $x, y \colon \mathbb{N}$ with $x = y$ we have $\mathsf{PA} \vdash \overline{x} \doteq \overline{y}$.*

Lemma 5.4 *For $x, y \colon \mathbb{N}$ with $x \neq y$ we have $\mathsf{PA} \vdash \dot{\neg}\overline{x} \doteq \overline{y}$*

Proof We do an induction on $x \colon \mathbb{N}$. In the base case we need to show that $0 \neq y$ implies $\mathsf{PA} \vdash \dot{\neg}\overline{0} \doteq \overline{y}$. In the case $y = 0$ this follows trivially and in the $y = Sz$ case this follows by the disjointness axiom.

In the inductive step we know that $Sx \neq y$ and need to show $\mathsf{PA} \vdash \dot{\neg}\,\overline{Sx} \doteq \overline{y}$. Here the $y = 0$ case follows by the disjointness axiom. In the $y = Sz$ case we get

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{PA} \vdash \dot{\neg}\overline{x} \doteq \overline{z}}\,\text{Ind. Hyp.}}{\overline{x} \doteq \overline{z},\, \mathsf{PA} \vdash \bot}}{\overline{Sx} \doteq \overline{Sz},\, \mathsf{PA} \vdash \bot}\,\text{Injectivity}}{\mathsf{PA} \vdash \dot{\neg}\overline{Sx} \doteq \overline{Sz}}$$

□

We can now use these results to show that PA can either show a deduction for the equality or apartness of two numerals.

Theorem 5.5 *If $x, y \colon \mathbb{N}$ then $\mathsf{PA} \vdash \overline{x} \doteq \overline{y}$ + $\mathsf{PA} \vdash \dot{\neg}\overline{x} \doteq \overline{y}$.*

Proof By Lemma 4.4 we either have $x = y$ or $x \neq y$ and the claim therefore follows by using Fact 5.3 and Lemma 5.4. □

Our goal is to extend this decidability result of equality to any closed term, by first showing that every closed term is equal to some numeral.

Lemma 5.6 *If* $t\colon \mathsf{tm}$ *is closed then there is* $n\colon \mathbb{N}$ *with* $\mathsf{PA} \vdash t \doteq \overline{n}$.

Proof We proceed by induction on the closed term t. Since t is closed, it cannot be equal to a variable x_k and we therefore only need to consider the cases where t is build up from function symbols and other closed terms. The only function symbols in PA are $O, \mathsf{S}, \oplus$ and $\otimes$.

- If $t = O$ then we clearly have $\mathsf{PA} \vdash t \doteq \overline{0}$.
- Now assume $t = \mathsf{S}t_0$. Since t_0 is closed we have by induction that there is $n_0\colon \mathbb{N}$ with $\mathsf{PA} \vdash t_0 \doteq \overline{n_0}$. From this we straightforwardly get $\mathsf{PA} \vdash t \doteq \overline{Sn_0}$.
- If $t = t_1 \oplus t_2$ then t_1, t_2 must be closed and by induction we get $n_1, n_2\colon \mathbb{N}$ with $\mathsf{PA} \vdash t_1 \doteq \overline{n_1}$ and $\mathsf{PA} \vdash t_2 \doteq \overline{n_2}$. We then have

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{PA} \vdash t_2 \doteq \overline{n_2}} \quad \dfrac{}{\mathsf{PA} \vdash t_1 \doteq \overline{n_1}}}{\mathsf{PA} \vdash t_1 \oplus t_2 \doteq \overline{n_1} \oplus \overline{n_2}}\,\oplus\text{-equality}}{\mathsf{PA} \vdash t \doteq \overline{n_1} \oplus \overline{n_2}}}{\mathsf{PA} \vdash t \doteq \overline{n_1 + n_2}}\,\text{Lemma 5.1}$$

- The case $t = t_1 \otimes t_2$ is analogous to the previous one. □

Corollary 5.7 *If* $s, t\colon \mathsf{tm}$ *are closed then* $\mathsf{PA} \vdash t \doteq s \;+\; \mathsf{PA} \vdash \dot{\neg} t \doteq s$.

Proof By Lemma 5.6 s, t are provably equal to numerals. Therefore the claim follows from Theorem 5.5. □

5.2 Undecidability

We will now focus on the finite fragment $\mathsf{U} \subseteq \mathsf{PA}$ whose only axioms are the equality axioms, as well as the base-case and recursion axioms for $\oplus$ and $\otimes$. We immediately remark that $\mathbb{N}$ is still a model of U and U is still strong enough to show Lemma 5.1 and Lemma 5.2.

Definition 5.8 *We inductively define polynomials with natural number coefficients, by building them up using constants, variables, addition and multiplication:*

$$p, q ::= \mathsf{a}_n \mid \mathsf{var}\ k \mid \mathsf{add}\ p\ q \mid \mathsf{mult}\ p\ q \qquad (n, k\colon \mathbb{N})$$

A tuple (p, q) *of polynomials then designates a Diophantine equation* $p = q$. *We recursively define the canonical evaluation* $[\![\cdot]\!]_\sigma$ *of polynomials under a variable assignment*

$\sigma\colon \mathbb{N} \to \mathbb{N}$ *by:*

$$\begin{aligned} [\![\mathsf{a}_n]\!]_\sigma &:= n & [\![\mathsf{add}\ p\ q]\!]_\sigma &:= [\![p]\!]_\sigma + [\![q]\!]_\sigma \\ [\![\mathsf{var}\ k]\!]_\sigma &:= \sigma\, k & [\![\mathsf{mult}\ p\ q]\!]_\sigma &:= [\![p]\!]_\sigma \cdot [\![q]\!]_\sigma \end{aligned}$$

We then say that a Diophantine equation $p = q$ has a solution iff there exists an assignment $\sigma\colon \mathbb{N} \to \mathbb{N}$ such that $[\![p]\!]_\sigma = [\![q]\!]_\sigma$.

Definition 5.9 *We recursively define an embedding of polynomials p as terms $p^*\colon \mathsf{tm}$ in the signature of* PA

$$\begin{aligned} \mathsf{a}_n{}^* &:= \overline{n} & (\mathsf{add}\ p\ q)^* &:= p^* \oplus q^* \\ (\mathsf{var}\ k)^* &:= x_k & (\mathsf{mult}\ p\ q)^* &:= p^* \otimes q^* \end{aligned}$$

Definition 5.10 *A Diophantine equation (p, q) can now be expressed as the* PA *formula by $p^* \doteq q^*$. If x_N is the greatest variable appearing in the formula $p^* \doteq q^*$ then $\varphi_{p,q} := \dot{\exists}^N p^* \doteq q^*$ is a closed formula internally expressing that the Diophantine equation has a solution. This will be our embedding of equations into* PA*.*

Next we will see that this way of embedding is sensible, by noting that p^* evaluates to the expected result in the standard model $\mathbb{N}$.

Lemma 5.11 *For every environment $\sigma\colon \mathbb{N} \to \mathbb{N}$ we have $\widehat{\sigma}(p^*) = [\![p]\!]_\sigma$.*

Proof We show this by induction on p.

- If $p = \mathsf{a}_n$ we have $\widehat{\sigma}(\mathsf{a}_n{}^*) = \widehat{\sigma}(\overline{n}) = n = [\![\mathsf{a}_n]\!]_\sigma$.
- If $p = \mathsf{var}\ k$ then $\widehat{\sigma}((\mathsf{var}\ k)^*) = \widehat{\sigma}(x_k) = \sigma(k) = [\![\mathsf{var}\ k]\!]_\sigma$.
- If $p = \mathsf{add}\ a\ b$ we have

$$\widehat{\sigma}((\mathsf{add}\ a\ b)^*) = \widehat{\sigma}(a^* \oplus b^*) = \widehat{\sigma}(a^*) + \widehat{\sigma}(b^*) \overset{\text{Ind.}}{=} [\![a]\!]_\sigma + [\![b]\!]_\sigma = [\![\mathsf{add}\ a\ b]\!]_\sigma$$

- The case $p = \mathsf{mult}\ a\ b$ is analogous to the previous one. □

We will now see that we get a similar result in the deduction system. Consider that p has the syntactic representation $p^* = \overline{2} \otimes x_1 \otimes x_1 \oplus x_0 \oplus \overline{5}$. If we are now given a variable assignment $\sigma\colon \mathbb{N} \to \mathbb{N}$ with $\sigma(0) = 4$, $\sigma(1) = 2$ we can evaluate the polynomial p in the usual way to get

$$[\![p]\!]_\sigma = 2\sigma(1)^2 + \sigma(0) + 5 = 2 \cdot 2 \cdot 2 + 4 + 5 = 17$$

or we can use σ and substitute its values as numerals into p^*

$$p^*[\overline{\sigma}] = \overline{2} \otimes \overline{\sigma(1)} \otimes \overline{\sigma(1)} \oplus \overline{\sigma(0)} \oplus \overline{5} = \overline{2} \otimes \overline{2} \otimes \overline{2} \oplus \overline{4} \oplus \overline{5}$$

By previous results we know that U is strong enough to derive that the latter is equal to the numeral $\overline{17}$. This observation leads us to the natural extension of the homomorphism property of $\overline{\cdot}$ to polynomials:

Lemma 5.12 *Let p be a polynomial and $\sigma\colon \mathbb{N} \to \mathbb{N}$ a variable assignment, then we have $\mathsf{U} \vdash p^*[\overline{\sigma}] \doteq \overline{[\![p]\!]_\sigma}$*

Proof We do an induction on p.

- If $p = \mathsf{a}_n$ we have

$$\dfrac{\dfrac{\dfrac{}{\mathsf{U} \vdash \overline{n} \doteq \overline{n}}\,\text{Reflexivity}}{\mathsf{U} \vdash \overline{n}\,[\overline{\sigma}] \doteq \overline{n}}\,\text{Subst.}}{\mathsf{U} \vdash {\mathsf{a}_n}^*[\overline{\sigma}] \doteq \overline{[\![\mathsf{a}_n]\!]_\sigma}}\,\text{Def.}$$

- If $p = \mathsf{var}\ k$ we have

$$\dfrac{\dfrac{\dfrac{}{\mathsf{U} \vdash \overline{\sigma(k)} \doteq \overline{\sigma(k)}}\,\text{Reflexivity}}{\mathsf{U} \vdash (x_k)[\overline{\sigma}] \doteq \overline{\sigma(k)}}\,\text{Subst.}}{\mathsf{U} \vdash (\mathsf{var}\ k)^*[\overline{\sigma}] \doteq \overline{[\![\mathsf{var}\ k]\!]_\sigma}}\,\text{Def.}$$

- If $p = \mathsf{add}\ a\ b$ we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Ind. Hyp.}\ \dfrac{}{\mathsf{U} \vdash a^*[\overline{\sigma}] \doteq \overline{[\![a]\!]_\sigma}} \qquad \dfrac{}{\mathsf{U} \vdash b^*[\overline{\sigma}] \doteq \overline{[\![b]\!]_\sigma}}\,\text{Ind. Hyp.}}{\mathsf{U} \vdash a^*[\overline{\sigma}] \oplus b^*[\overline{\sigma}] \doteq \overline{[\![a]\!]_\sigma} \oplus \overline{[\![b]\!]_\sigma}}\,\oplus\text{-equality}}{\mathsf{U} \vdash a^*[\overline{\sigma}] \oplus b^*[\overline{\sigma}] \doteq \overline{[\![a]\!]_\sigma + [\![b]\!]_\sigma}}\,\text{Lemma 5.1}}{\mathsf{U} \vdash (a^* \oplus b^*)[\overline{\sigma}] \doteq \overline{[\![a]\!]_\sigma + [\![b]\!]_\sigma}}\,\text{Subst.}}{\mathsf{U} \vdash (\mathsf{add}\ a\ b)^*[\overline{\sigma}] \doteq \overline{[\![\mathsf{add}\ a\ b]\!]_\sigma}}\,\text{Def.}$$

- The case where $p = \mathsf{mult}\ a\ b$ is analogous to the previous one. □

With this result we can now verify that when substituting the solution of a Diophantine equation into its syntactical representation, U is strong enough to recognize it as a solution.

Lemma 5.13 *If (p, q) has the solution $\sigma\colon \mathbb{N} \to \mathbb{N}$ then $\mathsf{U} \vdash (p^* \doteq q^*)[\overline{\sigma}]$.*

Proof By assumption we have $s := [\![p]\!]_\sigma = [\![q]\!]_\sigma$ and therefore

$$\dfrac{\dfrac{\dfrac{}{\mathsf{U} \vdash p^*[\overline{\sigma}] \doteq \overline{[\![p]\!]_\sigma}}\,\text{Lemma 5.12} \quad \dfrac{}{\mathsf{U} \vdash \overline{[\![q]\!]_\sigma} \doteq q^*[\overline{\sigma}]}\,\text{Lemma 5.12}}{\mathsf{U} \vdash p^*[\overline{\sigma}] \doteq \overline{s}}\,\text{Def.} \quad \dfrac{}{\mathsf{U} \vdash \overline{s} \doteq q^*[\overline{\sigma}]}\,\text{Def.}}{\dfrac{\mathsf{U} \vdash p^*[\overline{\sigma}] \doteq q^*[\overline{\sigma}]}{\mathsf{U} \vdash (p^* \doteq q^*)[\overline{\sigma}]}\,\text{Subst.}}\,\text{Transitivity}$$

□

Corollary 5.14 *If (p, q) has a solution then $\mathsf{U} \vdash \varphi_{p,q}$.*

Proof Let σ be a solution of (p, q), then by Lemma 5.13 we have $\mathsf{U} \vdash (p^* \doteq q^*)[\overline{\sigma}]$ and by Fact 3.7 the substitution $\overline{\sigma}$ is therefore a witness for $\mathsf{U} \vdash \exists^N p^* \doteq q^*$. □

Lemma 5.15 *(p, q) has a solution iff $\mathsf{U} \vDash \varphi_{p,q}$.*

Proof If (p, q) has a solution, then by Corollary 5.14 we have $\mathsf{U} \vdash \varphi_{p,q}$ and therefore $\mathsf{U} \vDash \varphi_{p,q}$ by soundness.

Conversely, if we have $\mathsf{U} \vDash \varphi_{p,q}$, then in particular $\mathbb{N} \vDash \varphi_{p,q}$ which means there is an environment $\rho \colon \mathbb{N} \to \mathbb{N}$ such that $\mathbb{N} \vDash_\rho p^* \doteq q^*$. We then know that ρ is a solution for (p, q) due to Lemma 5.11. □

Lemma 5.16 *(p, q) has a solution iff $\mathsf{U} \vdash_i \varphi_{p,q}$.*

Proof The first direction is simply Corollary 5.14. The converse follows from soundness and Lemma 5.15. □

Note that in Lemma 5.16 we have to restrict provability to intuitionistic provability $\vdash_i$, since without the additional assumption of LEM, the constructive type theory only allows a proof of soundness for $\vdash_i$. Since $\varphi_{p,q}$ is Σ_1, we could use a Friedman translation to extract a classical proof $\mathsf{U} \vdash_c \varphi_{p,q}$ form the intuitionistic proof $\mathsf{U} \vdash_i \varphi_{p,q}$, therefore generalizing Lemma 5.15 and the following results.

Theorem 5.17 *For every theory $\mathcal{T}$ with $\mathsf{U} \subseteq \mathcal{T}$ and $\mathbb{N} \vDash \mathcal{T}$ we have that validity $\lambda\varphi.\, \mathcal{T} \vDash \varphi$ and provability $\lambda\varphi.\, \mathcal{T} \vdash_i \varphi$ are undecidable. So in particular this holds for PA.*

Proof Lemma 5.15 and Lemma 5.16 respectively show that the decidability of $\lambda\varphi.\, \mathsf{U} \vDash \varphi$ and $\lambda\varphi.\, \mathsf{U} \vdash_i \varphi$ would imply the decidability of the satisfiability of Diophantine equations, which is known to be undecidable. Since $\mathsf{U} \subseteq \mathcal{T}$, this result then extends to $\mathcal{T}$. □

Corollary 5.18 *Validity $\lambda\varphi.\, \vDash \varphi$ and provability $\lambda\varphi.\, \vdash_i \varphi$ for first-order logic are undecidable over the signature of PA.*

Proof Let $\bigwedge \mathsf{U}$ denote the formula which is the conjunction of the finitely many axioms of U. Since we then have

$$\vdash_i (\bigwedge \mathsf{U} \dot{\to} \varphi) \;\Leftrightarrow\; \mathsf{U} \vdash_i \varphi \quad \text{and} \quad \vDash (\bigwedge \mathsf{U} \dot{\to} \varphi) \;\Leftrightarrow\; \mathsf{U} \vDash \varphi$$

the result follows from Theorem 5.17. □

Chapter 6

Tennenbaum's Theorem

When we think about models of PA, the first thing that comes to mind are the natural numbers $\mathbb{N}$, and one could certainly say that the sole purpose of PA is to accurately describe this model. The world of PA models is however larger than this. One way to show this is to add a new constant c to the language of PA and get an extended theory PA^* by adding the axiom scheme comprising of the axioms $n < c$ for every $n \colon \mathbb{N}$. Every finite subset of these new axioms will still have $\mathbb{N}$ as a model; we only need to make sure that the interpretation of the constant c is big enough to satisfy the finitely many inequalities $n_1 < c, \ldots, n_k < c$. By the classical compactness theorem then, there must also be a model for the full theory PA^*. This is then easily verified to be a model of PA not isomorphic to $\mathbb{N}$, as the interpretation of c can have no counterpart in $\mathbb{N}$. These kinds of non-standard models can be so large that it becomes impossible to enumerate all of their elements or perform computations on them algorithmically. This raises the question whether there is a sweet spot in this regard: Is there a model of PA which is *non-standard*, in the sense that it is not isomorphic to $\mathbb{N}$, and yet still keeps the arithmetical operations computable?

For a first take on the question we can assume that here *computable* means having a Turing machine computing the function. If we then take e.g. the addition function $\oplus^{\mathcal{M}} \colon \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ we can however not immediately describe a Turing machine that computes it. Since Turing machines manipulate finite strings on a tape, we cannot hand it elements of a generic model $\mathsf{PA} \vDash \mathcal{M}$ as input. We need to relate every element of $\mathcal{M}$ to a string over the tape alphabet, which then enables us to hand them as a digestible input to the machine. This immediately leads us to the conclusion that the models under consideration should be countable if all its elements are to be encoded without ambiguity, which is necessary if we want a machine computing $\oplus^{\mathcal{M}}$ on all possible input pairs $x, y \colon \mathcal{M}$. So the question narrows down to countable non-standard models with computable operations.

An answer to this question was first given by Stanley Tennenbaum in [47] and is now known as Tennenbaum's theorem. It states that for any countable non-standard

model $\mathcal{M} \vDash \mathsf{PA}$ there is no way to encode its elements as natural numbers such that either the addition or multiplication operation on $\mathcal{M}$ is computable on the codes.

The first proof of Tennenbaum's theorem we present is based on the article [44] by Peter Smith and has four main ingredients:

1. In Lemma 6.28 we saw a straightforward encoding of finite sets as natural numbers. In a non-standard model this encoding can surprisingly be extended to also encode infinite sets.

2. In a non-standard model, any predicate which holds on all numerals will also hold on some non-standard element (Lemma 6.46).

3. A result from the theory of computation (Section 6.2) tells us that there are pairs of recursively enumerable sets which cannot be separated by a recursive set.

4. In a *computable* PA model divisibility of elements by a standard element is decidable.

The proof then very roughly proceeds as follows: assuming we are in a non-standard model $\mathcal{M} \vDash \mathsf{PA}$, and given a pair of inseparable r.e. sets $A, B \subseteq \mathbb{N}$, 2. tells us that there is a non-standard element $e \in \mathcal{M}$ such that for all elements below e, the sets are disjoint. The set $X := \{x \in \mathcal{M} \mid x \in A \land x < e\}$ turns out to separate A and B and can therefore not be recursive. By 1. the set can however be coded by one element $c \in \mathcal{M}$, such that $x \in X \leftrightarrow \pi_x \mid c$. If addition $\oplus^{\mathcal{M}}$ of the model were computable, then $\pi_x \mid c$ would turn out to be recursive, which is not possible as it would entail recursiveness of X.

The framework in [44] takes set theory as a foundation, classical logic for reasoning and uses Turing machines as the computational model. Here we want to treat the theorem in constructive type theory, adding to existing work [37, 33, 32] by not only giving another formal treatment of the theorem in a constructive setting, but by also mechanizing and verifying these results in the Coq proof assistant.

Compared to the setting Smith uses, our approach differs in the following points:

- The foundation uses constructive type theory, having a built-in intuitionistic logic which we use as the meta-logic for the development of first-order logic.

- We still use Tarski semantics for the first-order logic, but with the model domains being types. This gives it an overall different flavor as by Definition 3.8 e.g. function symbols are interpreted as functions in the type theory.

- We will not use a concrete model of computation, but instead assume Church's thesis (Section 6.1), stating that every function $\mathbb{N} \to \mathbb{N}$ inside of our chosen

type theory is computable.

- In our formulation of the theorem we will not have to assume that multiplication or addition of the model are computable, as this follows from the previous two points.

After the presentation of the first proof, we will also see that by choosing even stricter constructive semantics, as was done by McCarty in [33], the categoricity of Heyting arithmetic can be shown rather quickly, giving a variant of Tennenbaum's theorem.

6.1 Church's Thesis

Eli: "Let's not make this any more difficult than it already is."
— Wes Anderson, *The Royal Tenenbaums*

Since Tennenbaum's theorem is concerned with models of PA whose operations are computable, we necessarily need to specify the model or notion of computability which will be used. Making reference to a computable function then means we are making reference to a concrete description of the function in this computation model. In [44] the choice was to consider Turing machines.

In our constructive setting, there is however another approach. Instead of explicitly working with a model of computation in this way, we can simplify the treatment by assuming Church's thesis as an axiom. Church's thesis is an informal statement which in its usual form expresses that every function $f : \mathbb{N} \to \mathbb{N}$ we would *intuitively* consider to be *computable*, has a representation in the Turing machine (or any equivalent) model. The idea is that if we have a class of functions that we intuitively believe to only consist of computable functions, we can commit to this belief by assuming Church's thesis as an axiom, formulated in such a way that it applies to this class of functions. We could not consistently assume Church's thesis in a classical logic as it is easy to construct a non-recursive function $\mathbb{N} \to \mathbb{N}$ using the law of excluded middle. As discussed in Chapter 4 we are using the constructive type theory as a setting for a synthetic theory of computation, therefore subscribing to the assumption that every function internal to the type theory is computable. We therefore assume the following formulation:

Definition 6.1 (Church's Thesis) *By* CT *we mean the statement that every function* $f : \mathbb{N} \to \mathbb{N}$ *in* CTT *is computable in a formal model of computation (e.g. Turing machines or* μ*-recursive functions).*

Remark 6.2 When it comes to Coq, every *concrete* function as e.g. defined by the user can in principle be shown to be computable in a chosen model of computation. In a proof however, say for a statement of the form $\forall(f : \mathbb{N} \to \mathbb{N}).\, P\, f$, we would start by assuming we have some $f : \mathbb{N} \to \mathbb{N}$. The function f then has no concrete definition, making it impossible to produce a representation of it in the model of computation.

It is in a situation like this then, that we make use of (and need to assume) Church's thesis in order to get a representation.

To see the usefulness of Church's Thesis in our context, we first remark that we can get the very helpful Axiom 6.4 by combining it with the following standard result [43]

Proposition 6.3 *If $f\colon \mathbb{N} \to \mathbb{N}$ is a μ-recursive function, then there is a binary Σ_1 formula $\varphi_f(x,y)$ such that*

$$\forall n.\, \mathsf{Q} \vdash \dot{\forall} y.\, \varphi_f(\overline{n}, y) \leftrightarrow \overline{f\, n} \doteq y$$

This should be read as: Q can verify that φ_f is an object level representation of f.

Axiom 6.4 (CT) *For every function $f\colon \mathbb{N} \to \mathbb{N}$ there is a Σ_1 formula $\varphi_f(x,y)$ with $\forall n, \mathsf{Q} \vdash \dot{\forall} y.\, \varphi_f(\overline{n}, y) \leftrightarrow \overline{f\, n} \doteq y$.*

Axiom 6.4 then is our chosen formulation of Church's thesis for this context. In principle, Proposition 6.3 could be derived, as was done (in a slightly different formulation) in [34].

Remark 6.5 Using Axiom 6.4 we get the existence of a formula φ_π representing the injective prime function π we will construct in Corollary 6.27. This avoids going through some lengths in explicitly constructing and deductively verifying the correctness of a formula φ_π by hand, as would otherwise be necessary.

Definition 6.6 *Let $p\colon \mathbb{N} \to \mathbb{P}$, then we call p **weakly representable** if there is a unary Σ_1 formula $\varphi_p(x)$ with $\forall x.\, p\, x \leftrightarrow \mathsf{Q} \vdash \varphi_p(\overline{x})$ and **strongly representable** if for every $x\colon \mathbb{N}$ we have $p\, x \to \mathsf{Q} \vdash \varphi_p(\overline{x})$ and $\neg p\, x \to \mathsf{Q} \vdash \dot{\neg}\varphi_p(\overline{x})$.*

We will then further assume the **representability theorem** [38] as an axiom.

Axiom 6.7 (RT) *Any enumerable (decidable) predicate on $\mathbb{N}$ is weakly (strongly) representable.*

Using CT, the representability theorem can in principle be shown, and we will give a sketch of its proof. The proof was however not fully mechanized, which is the reason we assume it as an axiom.

Proof Sketch (of Axiom 6.7) If p is enumerable there is a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\forall x.\, p\, x \leftrightarrow \exists n.\, fn = Sx$ and by Axiom 6.4 there is a binary Σ_1 formula $\varphi_f(x,y)$ representing f. We then define $\varphi_p(x) := \dot{\exists}\, n.\, \varphi_f(n, \mathsf{S}x)$ giving us

$$\begin{aligned}
\mathsf{Q} \vdash \varphi_p(\overline{x}) &\iff \mathsf{Q} \vdash \dot{\exists}\, n.\, \varphi_f(n, \mathsf{S}\overline{x}) \\
&\iff \exists n\colon \mathbb{N}.\, \mathsf{Q} \vdash \varphi_f(\overline{n}, \mathsf{S}\overline{x}) \\
&\iff \exists n\colon \mathbb{N}.\, \mathsf{Q} \vdash \overline{fn} \doteq \mathsf{S}\overline{x} \\
&\iff \exists n\colon \mathbb{N}.\, fn = Sx \iff p\, x
\end{aligned}$$

where in the second equivalence we made use of the fact that $\dot{\exists}\, n.\, \varphi_f$ is Σ_1. This shows that p is weakly representable.

If p is decidable there is a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\forall x.\, p\, x \leftrightarrow fx = 0$ and by Axiom 6.4 there is a binary Σ_1 formula $\varphi_f(x, y)$ representing f. We then define $\varphi_p(x) := \varphi_f(x, \overline{0})$. By making use of Fact 5.3 we get

$$p\, x \implies fx = 0 \implies \mathsf{Q} \vdash \overline{fx} \doteq \overline{0} \implies \mathsf{Q} \vdash \varphi_f(\overline{x}, \overline{0}) \implies \mathsf{Q} \vdash \varphi_p(\overline{x})$$

and similarly

$$\neg p\, x \implies fx \neq 0 \implies \mathsf{Q} \vdash \dot{\neg}(\overline{fx} \doteq \overline{0}) \implies \mathsf{Q} \vdash \dot{\neg}\varphi_f(\overline{x}, \overline{0}) \implies \mathsf{Q} \vdash \dot{\neg}\varphi_p(\overline{x})$$

by usage of Lemma 5.4. This shows that p is strongly representable. □

6.2 Inseparable r.e. Sets

To show Tennenbaum's theorem we will use a result from the theory of computation stating the existence of **recursively enumerable** (r.e.) sets which cannot be separated by a recursive set. In our context we use predicates instead of sets and use the synthetic notions of r.e. and recursiveness to express inseparability:

Definition 6.8 *A pair $A, B\colon \mathbb{N} \to \mathbb{P}$ of predicates is called **inseparable** iff*

- *A and B are enumerable,*
- *they are disjoint in the sense that $\forall n.\, \neg(A\, n \wedge B\, n)$*
- *there is no decidable $D\colon \mathbb{N} \to \mathbb{P}$ which includes A i.e. $\forall n.\, A\, n \to D\, n$ and is disjoint from B i.e. $\forall n.\, \neg(B\, n \wedge D\, n)$.*

Lemma 6.9 *There is a pair $A, B\colon \mathbb{N} \to \mathbb{P}$ of disjoint enumerable predicates which cannot be separated by predicates of the form $\lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \varphi(\overline{n})$ with completeness property $\forall n.\, \mathsf{Q} \vdash \varphi(\overline{n}) \vee \mathsf{Q} \vdash \dot{\neg}\varphi(\overline{n})$.*

Proof By Fact 3.5 we know that there is an enumeration $\Phi_n : \mathsf{fm}\ (n\colon \mathbb{N})$ of all formulas. We can therefore define the disjoint predicates $A := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \dot{\neg}\, \Phi_n(\overline{n})$ and $B := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \Phi_n(\overline{n})$. A and B are then enumerable by Fact 3.5 since Q is finite. Regarding the inseparability we need to show that for any predicate $\lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \varphi(\overline{n})$ with the completeness property, having the two statements

$$\forall n.\, A\ n \to \mathsf{Q} \vdash \varphi(\overline{n}) \qquad\qquad \forall n.\, \neg(B\, n \wedge \mathsf{Q} \vdash \varphi(\overline{n}))$$

leads to a contradiction. Using the fact that we have an enumeration of formulas, we get $d\colon \mathbb{N}$ with $\Phi_d = \varphi$. Using this and unfolding the definition of A and B we have

$$\forall n.\, \mathsf{Q} \vdash \dot{\neg}\, \Phi_n(\overline{n}) \to \mathsf{Q} \vdash \Phi_d(\overline{n}) \qquad\qquad \forall n.\, \neg(\mathsf{Q} \vdash \Phi_n(\overline{n}) \wedge \mathsf{Q} \vdash \Phi_d(\overline{n})) \tag{6.1}$$

Next we use the completeness property of $\lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \varphi(\overline{n})$. If $Q \vdash \varphi(\overline{d})$ then $\mathsf{Q} \vdash \Phi_d(\overline{d})$ and therefore the right statement in the above gives us $\bot$. If $\neg\, \mathsf{Q} \vdash \varphi(\overline{d})$ then we have $\neg\, \mathsf{Q} \vdash \Phi_d(\overline{d})$ and therefore $\mathsf{Q} \vdash \dot{\neg}\, \Phi_d(\overline{d})$ by the decidability property of Φ_d. This then leads to a contradiction due to the left statement in Equation (6.1). □

Corollary 6.10 *Assuming* RT*, there is a pair* $\alpha(x), \beta(x)$ *of unary* Σ_1 *formulas such that* $A := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \alpha(\overline{n})$ *and* $B := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \beta(\overline{n})$ *cannot be separated by predicates of the form* $\lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \varphi(\overline{n})$ *with completeness property* $\forall n.\, \mathsf{Q} \vdash \varphi(\overline{n}) \vee \mathsf{Q} \vdash \dot{\neg}\varphi(\overline{n})$.

Proof By Lemma 6.9 there are disjoint enumerable predicates A, B which cannot be separated in the indicated way. By the enumerability part of RT we then get the desired formulas from them. □

Corollary 6.11 *Assuming* RT*, there is a pair* $A, B \colon \mathbb{N} \to \mathbb{P}$ *of inseparable predicates.*

Proof By Lemma 6.9 there are disjoint enumerable predicates A, B which cannot be separated by formulas with the completeness property. By RT any decidable predicate D gives raise to such a formula wherefore D cannot separate A and B. □

Corollary 6.12 *Assuming* RT*, there is a pair* $\alpha(x), \beta(x)$ *of unary* Σ_1 *formulas such that* $A := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \alpha(\overline{n})$ *and* $B := \lambda n^{\mathbb{N}}.\, \mathsf{Q} \vdash \beta(\overline{n})$ *are inseparable.*

Proof By using RT on Lemma 6.9. □

6.3 Some Number Theory and Finite Coding

We will later need some facts about prime numbers and will now develop these results on the level of our type theory, which provides us with the natural number type $\mathbb{N}$. We presuppose that sensible definitions for basic arithmetic operations have been given in the type theory, as was done for addition in Section 2.2.3.

We set out to show the infinitude of primes and start by showing a classic theorem by Euclid.

Theorem 6.13 (Euclid) *For any* $x, q \colon \mathbb{N}$ *we can show*

$$\Sigma\, d\, r \colon \mathbb{N}.\ x = d \cdot q + r \wedge (0 < q \to\ r < q)$$

Proof If $q = 0$, we can choose $d := 0$ and $r := x$. Now knowing $q > 0$, we do an induction on $x \colon \mathbb{N}$. In the base case $x = 0$, the choice $d := 0$, $r := 0$ does the job. In the inductive case we know that $x = d' \cdot q + r'$ and using Lemma 4.4 we consider two cases:

- If $r' = q - 1$ then $Sx = S(d' \cdot q + q - 1) = (Sd') \cdot q$. Choosing $d := Sd'$ and $r := 0$ therefore proves the statement.

- If $r' \neq q-1$ then $Sx = S(d' \cdot q + r') = d' \cdot q + (Sr')$. We can then choose $d := d'$ and $r := Sr'$. In this case the fact $r = Sr' < q$ follows from $r' < q$ and $Sr' \neq q$. □

The next lemma shows that d, r as constructed by Theorem 6.13 are unique.

Lemma 6.14 *Given $q, r_1, r_2, d_1, d_2 \colon \mathbb{N}$ with $r_1, r_2 < q$, then $d_1 q + r_1 = d_2 q + r_2$ implies $d_1 = d_2$ and $r_1 = r_2$.*

Proof We distinguish three cases.

- If $d_1 = d_2$ then $d_1 q + r_1 = d_1 q + r_2$ which shows $r_1 = r_2$.
- If $d_1 < d_2$ then $(Sd_1)q \leq d_2 q + r_2$ and because of $r_1 < q$ we have $d_1 q + r_1 < (Sd_1)q$. Combined this gives $d_1 q + r_1 < d_2 q + r_2$, in contradiction to $d_1 q + r_1 = d_2 q + r_2$.
- If $d_2 < d_1$ we get a contradiction, just like in the second case. □

Proposition 6.15 *For any $x, q \colon \mathbb{N}$, we can extract the **divisor** $\mathsf{div}_q\, x := d$ and **modulus** $\mathsf{mod}_q\, x := r$ of x with respect to q from the proof of Theorem 6.13 and can immediately follow:*

1. $x = (\mathsf{div}_q\, x) \cdot q + (\mathsf{mod}_q\, x)$
2. $0 < q \to \mathsf{mod}_q\, x < q$
3. $0 < q \leq x \to \mathsf{mod}_q\, x < x$.

Definition 6.16 *A number $a \colon \mathbb{N}$ is divisible by $b \colon \mathbb{N}$, written $b \mid a$ if $\Sigma\, n^{\mathbb{N}}.\ a = n \cdot b$ [1]. Then $p \colon \mathbb{N}$ is a **prime** number if $p > 1$ and for every $a, b \colon \mathbb{N}$, $p \mid a \cdot b$ implies $p \mid a$ or $p \mid b$.*

Definition 6.17 *$n \colon \mathbb{N}$ is called **irreducible** if $n > 1$ and for any $x < n$, $x \mid n$ implies $x = 1$. We write $\mathsf{irred}\, n$.*

Lemma 6.18 irred *is decidable.*

Proof Since the quantification is bounded and equality as well as divisibility are decidable, the whole predicate is decidable. □

Theorem 6.19 *For any $n \colon \mathbb{N}$, we have $\mathsf{irred}\, n + (n > 1 \to \Sigma\, x < N.\ x \mid n \ \wedge\ x \neq 1)$*
.

[1] In the Coq development we usually used the equivalent statement $\mathsf{mod}_b\, a = 0$

Proof We use Lemma 6.18 and the fact that the right side is equivalent to the negation of $\mathsf{irred}\, n$. □

Corollary 6.20 *For any* $n\colon \mathbb{N}$*, we have* $\mathsf{irred}\, n \;+\; (n > 1 \to\; \Sigma\, x, y < N.\; x \cdot y = n)$.

Theorem 6.21 *Every* $n\colon \mathbb{N}$ *is prime iff it is irreducible.*

Proof (adapted from [52]) It is easy to show that every prime is irreducible, so we only show the converse. Assuming that x is irreducible we try to show $x \mid ab \;\Rightarrow\; x \mid a \;\vee\; x \mid b$ by strong induction on a. We distinguish two cases:

- $(x \leq a)$ By Theorem 6.13 we have $a = d \cdot x + r$ with $r < x$, implying $r < a$. Since we have
$$x \mid ab \implies x \mid db \cdot x + rb \implies x \mid rb$$
the induction hypothesis can be applied on r giving us $x \mid r \;\vee\; x \mid b$, which shows $x \mid a \;\vee\; x \mid b$.
- $(x > a)$ It is trivial for $a = 0$, so assume $a > 0$. In this case Theorem 6.13 gives us $x = d \cdot a + r$ with $r < a$. Since $x \mid ab$ there is a k with $kx = ab$. So we have
$$x \mid xb \implies x \mid dab + rb \implies x \mid dk \cdot x + rb \implies x \mid rb$$
again making it possible to apply the induction hypothesis on r to get $x \mid r \;\vee\; x \mid b$. If $x \mid b$ we are done. If $x \mid r$ we must actually have $r = 0$ since $r < a < x$. Thus $x = da$ and by the irreducibility of x we get $a = x \;\vee\; a = 1$ showing $x \mid a \;\vee\; x \mid b$ respectively. □

Using this we can show that any number above 1 has some prime factor.

Lemma 6.22 *If* $n > 1$ *then* $\Sigma\, x\colon \mathbb{N}.\, \mathsf{prime}\, x \wedge x \mid n$.

Proof We do strong induction on n. By Corollary 6.20 n is already prime or we have $x, y < n$ with $x \cdot y = n$. In the latter case the induction hypothesis gives us a prime factor of x which is then also a prime factor of n. □

Definition 6.23 *We define the factorial* $x!$ *recursively by* $0! := 1$ *and* $(Sx)! := (Sx) \cdot x!$.

Fact 6.24 *For any* $x\colon \mathbb{N}$*, we have* $0 < x!$ *and* $x > 0 \;\to\; x \mid x!$.

Lemma 6.25 *For any* $0 < y \leq x$ *we have* $y \mid x!$.

Proof By induction on x with y generalized. The base case is trivial, since there is no such y. For the induction step, we distinguish between the cases $y = Sx$ or $y \leq x$, which are then both shown by usage of the induction hypothesis and Fact 6.24. □

Theorem 6.26 (Infinitude of Primes) $\forall n\colon \mathbb{N}.\, \Sigma\, x > n.\, \mathsf{prime}\, x.$

Proof Let $n\colon \mathbb{N}$ be given. Since $n! + 1 > 1$ we can use Corollary 6.20 to get a prime factor k of $n! + 1$. If $k \leq n$ we would have $\mathsf{mod}_k\, n! = 0$ by Fact 6.24 and $\mathsf{mod}_k\, (n! + 1) = 0$ leading to the contradictory statement $\mathsf{mod}_k\, 1 = 0$. Hence k is prime with $k > n$ as desired. □

From the above intuitionistic proof we can extract a function $\pi'\colon \mathbb{N} \to \mathbb{N}$ which only produces prime numbers and has the additional property $\forall n.\, \pi'\, n > n$.

Corollary 6.27 *The recursively defined function* $\pi : \mathbb{N} \to \mathbb{N}$

$$\pi\, 0 := \pi'\, 0 \quad ; \quad \pi\, (Sn) := \pi'(\pi\, n)$$

is injective and produces only prime numbers.

Proof By the property of π' for any $n > 0$ we have $\pi\, (Sn) = \pi'(\pi\, n) > \pi\, n$ showing that π is strictly increasing and therefore injective. □

We will now see an application of the previously (Corollary 6.27) defined injective prime-function π. It enables us to encode finite sets of natural numbers. To illustrate this, we take the finite set $S := \{3, 8, 14, 21\}$ and encode it by a single codenumber $c\colon \mathbb{N}$ by multiplying prime numbers given by the function π.

$$c := (\pi\, 3) \cdot (\pi\, 8) \cdot (\pi\, 14) \cdot (\pi\, 21)$$

If we want to know if $k \in S$, we simply need to check whether $\pi\, k \mid c$.

In particular, we can use this to encode finite sets of the form $\{u < n \mid p\, u\}$ where p is a definite predicate and $n\colon \mathbb{N}$ some bound.

Lemma 6.28 (Finite Coding) *If* $p\colon \mathbb{N} \to \mathbb{P}$ *is definite, then for every bound* $n\colon \mathbb{N}$ *there is a code* $c\colon \mathbb{N}$ *such that*

$$\forall u^{\mathbb{N}}.\big(u < n \to (p\, u \leftrightarrow \pi\, u \mid c)\big) \wedge \big(\pi\, u \mid c \to u < n\big).$$

Proof We do a proof by induction on n. For $n = 0$ we can choose $c = 1$. For the successor case: if $\neg p\, n$ we can simply take the code c given by the induction hypothesis, otherwise if $p\, n$ we multiply the given c with $\pi\, n$. In both cases the separate parts of the conjunction are checked by making use of the prime property as well as the injectivity of π. □

6.4 Basic Peano Arithmetic

In this section we will state and proof results about PA in some generic model $\mathcal{M} \vDash \mathsf{PA}$. This is in contrast to Section 6.3, where we worked in the specific model $\mathcal{M} = \mathbb{N}$, to facilitate some proofs. For the remainder of the whole chapter, $\mathcal{M}$ will always designate a PA model.

Proposition 6.29 *For any $\mathcal{N} \vDash \mathsf{PA}$, we can recursively define a function $\mu : \mathbb{N} \to \mathcal{N}$ by $\mu\, 0 := O^{\mathcal{N}}$ and $\mu\,(n+1) := \mathsf{S}^{\mathcal{N}}(\mu\, n)$. An image point $e \colon \mathcal{N}$ of μ is called **standard number** and we will also use the notation $\mathsf{std}\, e$ to express this. We further have*

- *$\widehat{\rho}(\overline{n}) = \mu\, n$ for any $n \colon \mathbb{N}$ and environment $\rho \colon \mathbb{N} \to \mathcal{N}$.*
- *μ is an injective homomorphism and therefore an **embedding** of $\mathbb{N}$ into $\mathcal{N}$.*

Proof The fact $\widehat{\rho}(\overline{n}) = \mu\, n$ easily follows by induction on $n \colon \mathbb{N}$. The homomorphism property can then be concluded from this fact and the homomorphism property of the numeral function, by using soundness on the results of Lemma 5.1 and Lemma 5.2. It remains to show injectivity. For this we show $\forall y.\ \mu\, x = \mu\, y \to x = y$ by induction on x. Both the base case and inductive step are handled by case analysis on y. □

Usually we would have to write $\mathsf{S}^{\mathcal{M}}, \oplus^{\mathcal{M}}, \otimes^{\mathcal{M}}, \dot{-}^{\mathcal{M}}$ for the interpretations of the respective symbols in a model $\mathcal{M}$. For better readability we will however take the freedom to overload the symbols $S, +, \cdot, =$ to also refer to these interpretations.

From this point on, this section can be skipped on a first reading, as all of the results that will follow are "obviously" true in PA and their proofs do not illuminate the proof of Tennenbaum's theorem.

Lemma 6.30 *For all $x, y \colon \mathcal{M}$ we have*

$$\begin{array}{ccc} 0 = Sx \to \bot & x + 0 = x & x \cdot 0 = 0 \\ Sx = Sy \to x = y & x + Sy = S(x + y) & x \cdot (Sy) = x + x \cdot y \end{array}$$

Proof These follow straight from the axioms via soundness. □

Lemma 6.31 *For all $x, y, z \colon \mathcal{M}$ we have the statements concerning addition and multiplication*

$$\begin{array}{cc} x + y = y + x & x \cdot y = y \cdot x \\ (x + y) + z = x + (y + z) & (x \cdot y) \cdot z = x \cdot (y \cdot z) \\ x + y = 0 \to x = 0 \wedge y = 0 & x + z = y + z \to x = y \\ x = 0 \vee \exists y \colon \mathcal{M}.\, x = Sy & x = y \vee x \neq y \end{array}$$

and the following results for inequalities:

$$\begin{array}{cc} \neg\, x < 0 & Sx < Sy \leftrightarrow x < y \\ x < Sy \leftrightarrow x \leq y & x < Sy \leftrightarrow x < y \vee x = y \\ x < y \vee x = y \vee y < x & x < y \to x \neq y \\ x < y \to y \leq z \to x < z & x \leq y \to y \leq z \to x \leq z \\ x < y \to x + z < y + z & x \leq y \to x + z \leq y + z \end{array}$$

All of the statements in Lemma 6.31 have relatively straightforward proofs in $\mathbb{N}$ and require at most induction on one of the numbers. On paper these proofs then just translate to any other model, although they become more technical. For illustration purposes we will elaborate on these details in all of the remaining proofs of this section and refer the reader to the mechanized proofs if they are interested in more.

Lemma 6.32 (Euclid) *For every $q, x \colon \mathcal{M}$ there exist $d, r \colon \mathcal{M}$ such that $x = d \cdot q + r \wedge (0 < q \to r < q)$.*

Proof From a result of Lemma 6.31 we know that q is zero or the successor of some $q' \colon \mathcal{M}$. If $d = 0$ then $d := \overline{0}$ and $r := x$ prove the claim. In the case that $q = Sq'$ we proceed by induction on $x \colon \mathcal{M}$. In the base case we need to show

$$\mathcal{M} \vDash \dot{\exists}\, d\, r.\, \overline{0} \doteq d \otimes q \oplus r \;\; \dot{\wedge} \;\; (\overline{0} \mathrel{\dot{<}} q \mathrel{\dot{\to}} r \mathrel{\dot{<}} q)$$

which holds for $d := \overline{0}$ and $r := \overline{0}$. In the induction step the induction hypothesis gives us $d', r' \colon \mathcal{M}$ such that

$$x = d' \cdot q + r' \;\wedge\; (\overline{0} < q \to r' < q)$$

Using the decidability of equality shown in Lemma 6.31 we can now distinguish two cases:

- If $r' = q'$ then by choosing $d := Sd'$ and $r := \overline{0}$ we have

  $$d \cdot q + r = Sd' \cdot q = d' \cdot q + q = d' \cdot q + Sq' = S(d' \cdot q' + r') = Sx$$

  and $\overline{0} < q \to \overline{0} < q$ as desired.

- If $r' \neq q'$ we choose $d := d'$ and $r := Sr'$ to get

  $$d \cdot q + r = d' \cdot q + Sr' = S(d \cdot q' + r') = Sx.$$

  We are left to show $Sr' = r < q = Sq'$ which is equivalent to $r' < q'$ by a result from Lemma 6.31. By the induction hypothesis we know that $r' < Sq'$ and therefore $r' \leq q'$. We now have enough information to show easily show the goal in every case trichotomy on r' and q' gives us. □

Next we show uniqueness of this decomposition.

Lemma 6.33 *For all $q, r_1, r_2, d_1, d_2 \colon \mathcal{M}$ we have*

$$r_1 < q \to d_1 < d_2 \to d_1 \cdot q + r_2 = d_2 \cdot q + r_2 \to \bot.$$

Proof Given $r_1 < q$ we have $d_1 \cdot q + r_1 < d_1 \cdot q + q$ by the monotonicity of addition. From $d_1 < d_2$ we get

$$\begin{aligned} Sd_1 \leq d_2 &\implies (Sd_1) \cdot q \leq d_2 \cdot q \\ &\implies (Sd_1) \cdot q \leq d_2 \cdot q + r_2 \implies d_1 \cdot q + q \leq d_2 \cdot q + r_2 \end{aligned}$$

By Lemma 6.31 the inequalities can be combined to give us $d_1 \cdot q + r_1 < d_2 \cdot q + r_2$ and therefore a contradiction to the assumption $d_1 \cdot q + r_1 = d_2 \cdot q + r_2$. □

Lemma 6.34 *For all $q, r_1, r_2, d_1, d_2 \colon \mathcal{M}$ we have*

$$r_1 < q \to r_2 < d_2 \to d_1 \cdot q + r_2 = d_2 \cdot q + r_2 \to r_1 = r_2 \wedge d_1 = d_2.$$

Proof By trichotomy we have $d_1 < d_2$ or $d_1 = d_2$ or $d_2 < d_1$ where the cases with strict inequalities can immediately be excluded since they yield contradictions due to Lemma 6.33. We therefore have $d_1 = d_2$, further entailing

$$d_1 \cdot q + r_1 = d_2 \cdot q + r_2 \implies d_1 \cdot q + r_1 = d_1 \cdot q + r_2 \implies r_1 = r_2$$

by the cancellation property of addition shown in Lemma 6.31. □

Lemma 6.35 *For any $x, y \colon \mathbb{N}$ we have $x < y$ iff $\mathcal{M} \vDash \overline{x} \mathrel{\dot{<}} \overline{y}$.*

Proof If $x < y$ there is $k \colon \mathbb{N}$ with $S(x + k) = y$ showing $\mathcal{M} \vDash \mathsf{S}\overline{x} \oplus \overline{k} \doteq \overline{y}$ by Lemma 5.1 and therefore witnessing $\mathcal{M} \vDash \overline{x} \mathrel{\dot{<}} \overline{y}$. For the converse implication, if there is $e \colon M$ with $S(\mu\, x + e) = \mu\, y$ then e must be a standard number $e = \mu\, k$, giving us $\mu\,(S(x + k)) = \mu\, y$ and therefore $S(x + k) = y$ by the homomorphism property and injectivity of μ. □

Lemma 6.36 *For every quantifier-free binary Δ_0 formula $\varphi(x, y)$ we have*

$$\mathcal{M} \vDash \forall x\, y.\, \varphi(x, y) \mathrel{\dot{\vee}} \dot{\neg}\varphi(x, y)$$

Proof We do an induction on Δ_0 φ.

- If $\varphi = \bot$ we have $\mathcal{M} \vDash \dot{\neg}\bot$.
- If $\varphi = (s \doteq t)$ then the claim follows from the decidability of equality shown in Lemma 6.31.

- If $\varphi = \alpha \mathbin{\dot{\square}} \beta$ then α, β are themselves binary and Δ_0 and so the induction hypothesis applies to them, giving us enough information to show the claim no matter the particular connective $\dot{\square}$. □

Lemma 6.37 *For every quantifier-free binary Δ_0 formula $\varphi(x, y)$ we have*

$$\mathcal{M} \vDash \forall b\, y.\, (\dot{\exists}\, x.\, x \mathbin{\dot{<}} b \mathbin{\dot{\wedge}} \varphi(x, y)) \mathbin{\dot{\vee}} \dot{\neg}(\dot{\exists}\, x.\, x \mathbin{\dot{<}} b \mathbin{\dot{\wedge}} \varphi(x, y))$$

Proof We do an induction on b. The base case $b = 0$ is then straightforward as we can show the right side of the disjunction. In the induction step we get two cases from the induction hypothesis. In the first case we have that $\mathcal{M} \vDash \dot{\exists}\, x.\, x \mathbin{\dot{<}} b \mathbin{\dot{\wedge}} \varphi(x, y)$ and can therefore show $\mathcal{M} \vDash \dot{\exists}\, x.\, x \mathbin{\dot{<}} \mathsf{S} b \mathbin{\dot{\wedge}} \varphi(x, y)$ in the goal. In the second case we have $\mathcal{M} \vDash \dot{\neg}\, \dot{\exists}\, x.\, x \mathbin{\dot{<}} b \mathbin{\dot{\wedge}} \varphi(x, y)$. By Lemma 6.36, we have $\mathcal{M} \vDash \varphi(b, y) \mathbin{\dot{\vee}} \dot{\neg}\varphi(b, y)$ and depending on the case we can either show $\mathcal{M} \vDash \dot{\exists}\, x.\, x \mathbin{\dot{<}} \mathsf{S} b \mathbin{\dot{\wedge}} \varphi(x, y)$ or $\mathcal{M} \vDash \dot{\neg}\, \dot{\exists}\, x.\, x \mathbin{\dot{<}} \mathsf{S} b \mathbin{\dot{\wedge}} \varphi(x, y)$. □

Lemma 6.38 *For every $x, y \colon \mathbb{N}$ we have $x < y$ iff $\mathcal{M} \vDash \overline{x} \mathbin{\dot{<}} \overline{y}$.*

Proof We do an induction over $y \colon \mathbb{N}$ with x quantified; the base case being trivial. In the induction step we need to show the equivalence of $x < Sy$ and $\mathcal{M} \vDash \overline{x} \mathbin{\dot{<}} \overline{Sy}$. This is again trivial in the case where $x = 0$. If $x = Sz$ then $Sz < Sy$ which is equivalent to $z < y$, furthermore by the induction hypothesis equivalent to $\mathcal{M} \vDash \overline{z} \mathbin{\dot{<}} \overline{y}$ and finally equivalent to $\mathcal{M} \vDash \overline{Sz} \mathbin{\dot{<}} \overline{Sy}$ by a result from Lemma 6.31. □

6.5 Standard Models

We will now define standard models of PA and record some basic results about them.

Definition 6.39 *$\mathcal{M}$ is a **standard model** iff the homomorphism μ as defined in Proposition 6.29 is surjective.*

With a few steps we can show that $\mathcal{M}$ is standard iff there is a bijective homomorphism $\phi \colon \mathbb{N} \to \mathcal{M}$. We will accordingly write $\mathcal{M} \cong \mathbb{N}$ if this is the case. For this purpose, we start by showing that μ is essentially the only homomorphism from $\mathbb{N}$ to $\mathcal{M}$ we need to care about, since it is unique up to functional extensionality:

Lemma 6.40 *Let $\phi \colon \mathbb{N} \to \mathcal{M}$ be a homomorphism, then $\forall x.\, \phi\, x = \mu\, x$.*

Proof By induction on x and using the fact that both are homomorphisms. □

Lemma 6.41 *$\mathcal{M} \cong \mathbb{N} \iff \mathcal{M}$ is standard $\iff \forall e.\, \mathsf{std}\, e$.*

Proof Given $\mathcal{M} \cong \mathbb{N}$, there is an isomorphism $\phi \colon \mathbb{N} \to \mathcal{M}$. Since ϕ is surjective, Lemma 6.40 implies that μ must also be surjective. For the converse: if μ is surjective, it is an isomorphism since it is injective by Proposition 6.29. The second equivalence simply holds by definition. □

Having seen that every model contains a unique embedding of $\mathbb{N}$, one could ask whether there is a formula φ which could define and pick out precisely the standard numbers in $\mathcal{M}$. Lemma 6.42 gives an answer to this question:

Lemma 6.42 *$\mathcal{M} \cong \mathbb{N}$ iff there is a unary formula $\varphi(x)$ with*

$$\forall e.\ \big(\mathsf{std}\, e \leftrightarrow \mathcal{M} \vDash \varphi(e)\big).$$

Proof If $\mathcal{M} \cong \mathbb{N}$, then the formula $x = x$ shows the claim. For the converse, given a formula φ with the stated property, we certainly have $\mathcal{M} \vDash \varphi(\overline{0})$ since $\mu\, 0$ is a standard number, and clearly $\mathcal{M} \vDash \varphi(x) \to \mathsf{std}\, x \to \mathsf{std}\,(Sx) \to \mathcal{M} \vDash \varphi(Sx)$. Thus by induction in the model, we have $\mathcal{M} \vDash \forall x.\, \varphi(x)$, which is equivalent to $\forall e.\, \mathsf{std}\, e$. □

Remark 6.43 We can extract a more general statement from Lemma 6.42: Given any predicate $P : \mathcal{M} \to \mathbb{P}$ which holds on $\mu\, 0$ and $\forall x\colon \mathcal{M}.\, P\, x \to P\,(Sx)$, we have $\forall e.\, P\, e$ iff there is a unary formula φ with $\forall e.\ \big(P\, e \leftrightarrow M \vDash \varphi(e)\big)$.

6.6 Overspill and Infinite Coding

In this section we will examine some properties of PA models which differ from $\mathbb{N}$ and conclude with the two results Lemma 6.46 and Lemma 6.48 which are essential for the proof of Tennenbaum's theorem as the latter allows the potential encoding of some infinite subsets.

Lemma 6.44 *For any $e\colon \mathcal{M}$, we have $\neg\mathsf{std}\, e$ iff $\forall n\colon \mathbb{N}.\ \mu\, n < e$.*

Proof Assume $\neg\mathsf{std}\, e$, then by trichotomy of $<$ we have $\mu\, n < e$ or $\mu\, n = e$ or $e < \mu\, n$. Since the last two cases would imply $\mathsf{std}\, e$ we must have $\mu\, n < e$. For the reverse implication, assume $\mu\, k = e$ for some $k\colon \mathbb{N}$. Then the assumption $\forall n.\, \mu\, n < e$ immediately gives us a contradiction for e.g. $n = k + 1$. □

Definition 6.45 *We will say that $\mathcal{M}$ is **not standard** and write $\mathcal{M} \ncong \mathbb{N}$ iff $\mathcal{M}$ is not isomorphic to $\mathbb{N}$. We will call it **non-standard** iff there is $e\colon M$ such that $\neg\mathsf{std}\, e$ and founded on the result of Lemma 6.44 we will often use the notation $\mathbb{N} < e\colon \mathcal{M}$ for this.*

Note that we have "non-standard → not standard". The converse implication however, does not hold constructively in general. It does hold in a negative context, which is the situation we will later often encounter.

Using Lemma 6.42 we now get a notable result telling us that if a formula holds on all standard elements, then –classically speaking– it's truth value will "overspill" into the rest of the model, meaning at least one non-standard element will also satisfy the formula.

Lemma 6.46 (Overspill) *If $\mathcal{M} \ncong \mathbb{N}$ and $\varphi(x)$ is a unary formula with $\mathcal{M} \vDash \phi(\mu\, n)$ for every $n\colon \mathbb{N}$ then*

1. $\neg \big(\forall e\colon \mathcal{M}.\, (\mathcal{M} \vDash \phi(e)) \to \mathsf{std}\, e\big)$
2. $\mathsf{Stable}\ \mathsf{std} \implies \neg\neg\, \exists e\colon \mathcal{M}.\, \neg\, \mathsf{std}\, e \ \wedge\ \mathcal{M} \vDash \phi(e)$
3. $\mathsf{DNE} \implies \exists e\colon \mathcal{M}.\, \neg\, \mathsf{std}\, e \ \wedge\ \mathcal{M} \vDash \phi(e)$.

Proof 1. Assuming $\forall e\colon \mathcal{M}.\, (\mathcal{M} \vDash \phi(e)) \to \mathsf{std}\, e$ and combining it with our assumption that φ holds on all numerals, Lemma 6.42 implies $\mathcal{M} \cong \mathbb{N}$, giving us a contradiction.

For 2. we note that stability of std gives us the implication

$$\big(\neg \exists e\colon \mathcal{M}.\neg \mathsf{std}\, e \ \wedge\ \mathcal{M} \vDash \varphi(e)\big) \implies \big(\forall e\colon \mathcal{M}.\, (\mathcal{M} \vDash \varphi(e)) \to \mathsf{std}\, e\big)$$

and we therefore get a contradiction in the same way as in 1. and 3. immediately follows from the second statement. □

In Section 6.3 we had seen that in $\mathbb{N}$ it is relatively easy to encode finite sets. If we try to do the same in PA we first of all need to express the statement on the object level. This now comes with the difficulty of finding a first-order formula which expresses that a number is prime. Fortunately we can now make use of the earlier Remark 6.5 telling us that by making use of CT and more specifically Axiom 6.4, we have a formula φ_π representing the function π. This then allows us to express the desired statement and get the following version of the coding result in our PA-model $\mathcal{M}$:

Lemma 6.47 *If φ is a binary Δ_0 formula, then for any $n\colon \mathbb{N}$ we have*

$$\mathcal{M} \vDash \dot\forall b\ \dot\exists c\ \dot\forall u \dot< \overline{n}.\ \big(\dot\exists z \dot< b.\ \varphi(z,u)\big) \dot\leftrightarrow\ \dot\exists p.\ \varphi_\pi(u,p) \mathbin{\dot\wedge} p \mid c$$

Proof Let $b\colon \mathcal{M}$ be given, then by Lemma 6.37 the predicate $\lambda m\colon \mathbb{N}.\, \mathcal{M} \vDash \dot\exists z \dot< b.\ \varphi(z, \overline{m})$ is propositionally decidable and therefore Lemma 6.28 gives us $c\colon \mathbb{N}$ coding the predicate up to the bound n. We will now show that $\overline{c}$ then also proves the claim

$$\mathcal{M} \vDash \dot\forall u \dot< \overline{n}.\ \big(\dot\exists z \dot< b.\ \varphi(z,u)\big) \dot\leftrightarrow\ \dot\exists p.\ \varphi_\pi(u,p) \mathbin{\dot\wedge} p \mid \overline{c}$$

Given $u\colon \mathcal{M}$ with $\mathcal{M} \vDash u \dot< \overline{n}$ we conclude that u must be a standard number $u = \mu\, k$ for some $k\colon \mathbb{N}$. We then have the equivalence

$$\pi\, k \mid c \iff \mathcal{M} \vDash \dot\exists z \dot< b.\ \varphi(z, \overline{k})$$

since c is coding the predicate on the right. Finally we also have the equivalence $\pi\, k \mid c \iff \mathcal{M} \vDash \dot\exists p.\ \varphi_\pi(\overline{k}, p) \mathbin{\dot\wedge} p \mid \overline{c}$ due to φ_π being the representation of π. □

Looking at the coding statement we have just shown we can recognize that it is a formula which is satisfied for every standard number. Therefore it is susceptible to an application of Overspill, which gets us the surprising result that in models that are not standard (and up to a $\neg\neg$) we can get rid of the bound n, making it possible to encode infinite sets by a single element of the model.

Lemma 6.48 *If $\mathcal{M} \ncong \mathbb{N}$ and* std *is stable, then for any binary Δ_0 formula φ we have*

$$\neg\neg\, \forall b\colon \mathcal{M}\ \exists c\colon \mathcal{M}\ \forall u\colon \mathbb{N}.\ \ \mathcal{M} \vDash \left(\dot\exists\, z \dot< b.\ \varphi(z, \overline{u})\right) \dot\leftrightarrow\ \dot\exists p.\ \varphi_\pi\,(\overline{u}, p)\ \dot\wedge\ p \mid c$$

Proof By Lemma 6.47 and Overspill Lemma 6.46 we potentially have a non-standard $e\colon \mathcal{M}$ with

$$\mathcal{M} \vDash \dot\forall b\ \dot\exists c\ \dot\forall u \dot< e.\ \left(\dot\exists\, z \dot< b.\ \varphi(z, u)\right) \dot\leftrightarrow\ \dot\exists p.\ \varphi_\pi\,(u, p)\ \dot\wedge\ p \mid c$$

and thus for any $b\colon \mathcal{M}$ there is a code $c\colon \mathcal{M}$ such that for any $u\colon \mathcal{M}$ with $u < e$ we potentially have

$$\mathcal{M} \vDash \left(\dot\exists\, z \dot< b.\ \varphi(z, u)\right) \dot\leftrightarrow\ \dot\exists p.\ \varphi_\pi\,(u, p)\ \dot\wedge\ p \mid c$$

So in particular because of Lemma 6.44 $u < e$ holds whenever u is a standard number, which then shows the claim. □

6.7 Tennenbaum's Theorem

We now show the last missing ingredient to Tennenbaum's theorem by establishing that divisibility by a standard number is decidable if the model is a data type.

Lemma 6.49 *If $\mathcal{M}$ is a data type, $e\colon \mathcal{M}$ and $n > 0$ then $\mathcal{M} \vDash \overline{n} \mid e$ is decidable.*

Proof Let $n\colon \mathbb{N}$ with $n > 0$ and $e\colon \mathcal{M}$ be given. Since $\mathcal{M}$ is enumerable we get a surjective function $g : \mathbb{N} \to \mathcal{M}$ and since $\mathcal{M}$ has decidable equality the statement $e = q \cdot \mu\, n + \mu\, r$ is decidable, as is $\mu\, 0 < \mu\, n \to \mu\, r < \mu\, n$ by Lemma 6.38. We can therefore use the witness operator of $\mathbb{N}$ in combination with the Euclidean Lemma 6.32 to get

$$\Sigma\, r, k\colon \mathbb{N}.\, e = \mu\, n \cdot (g\, k) + \mu\, r\ \wedge\ (\mu\, 0 < \mu\, n \to \mu\, r < \mu\, n)$$

So we have access to $r, k\colon \mathbb{N}$ satisfying the above conjunction. We now get two cases:

- If $r = 0$ we have $e = \mu\, n \cdot (g\, k)$ and therefore $\mathcal{M} \vDash \overline{n} \mid e$.
- If $r \neq 0$ we can show $\neg\mathcal{M} \vDash \overline{n} \mid e$. Since if we had $\mathcal{M} \vDash \overline{n} \mid e$ we get some $d\colon \mathcal{M}$ with $e = \overline{n} \cdot d$ which by the uniqueness Lemma 6.34 implies $r = 0$, a contradiction. □

Lemma 6.50 *If $\mathcal{M} \ncong \mathbb{N}$ with stable* std *then* $\neg\neg\exists c\colon \mathcal{M}.\, \neg\mathsf{Dec}(\lambda n.\, \mathcal{M} \vDash \overline{\pi\, n} \mid c)$.

Proof By Corollary 6.12 there are inseparable Σ_1 formulas $\exists z.\, \alpha(z, x)$ and $\exists z.\, \beta_0(z, x)$ where α_0, β_0 are binary Δ_0 formulas. Since the inseparable formulas are disjoint, we can show that this is also the case for α_0 and β_0:

$$\mathbb{N} \vDash \dot\forall\, x\, y\, z \dot< \overline{n}.\, \dot\neg(\alpha_0(y, x) \mathbin{\dot\wedge} \beta_0(z, x))$$

for every bound $n\colon \mathbb{N}$. By Lemma 3.16 we then get

$$\mathcal{M} \vDash \dot\forall\, x\, y\, z \dot< \overline{n}.\, \dot\neg(\alpha_0(y, x) \mathbin{\dot\wedge} \beta_0(z, x))$$

Using Overspill we therefore potentially have $e\colon \mathcal{M}$ with

$$\mathcal{M} \vDash \dot\forall\, x\, y\, z \dot< e.\, \dot\neg(\alpha_0(y, x) \mathbin{\dot\wedge} \beta_0(z, x))$$

showing the disjointness of α_0, β_0 when everything is bounded by e. We now define the predicate $X := \lambda n.\, \mathcal{M} \vDash \dot\exists\, z \dot< e.\, \alpha_0(z, \overline{n})$ and note that

- If $\mathbb{N} \vDash \dot\exists\, z.\, \alpha_0(z, \overline{n})$ there is $m\colon \mathbb{N}$ with $\mathbb{N} \vDash \alpha_0(\overline{m}, \overline{n})$ and $\mathcal{M} \vDash \alpha_0(\overline{m}, \overline{n})$ by Lemma 3.16. We therefore get Xn.
- Assume that $Xn \wedge \mathbb{N} \vDash \dot\exists\, z.\, \beta_0(z, \overline{n})$. Then similarly to above, there is $m\colon \mathbb{N}$ with $\mathcal{M} \vDash \beta_0(\overline{m}, \overline{n})$, showing $\mathcal{M} \vDash \dot\exists\, z \dot< e.\, \beta_0(z, \overline{m})$. Together with Xn this contradicts the disjointness of α_0, β_0 under the bound e.

Due to the inseparability of the given formulas, this shows that X cannot be decidable and by Lemma 6.48 there potentially is a code $c\colon \mathcal{M}$ with $Xn \Leftrightarrow \mathcal{M} \vDash \overline{\pi\, n} \mid c$. □

We now have everything in place to show Tennenbaum's Theorem. Compared to the original theorem this version is formulated positively, which makes a difference in our constructive setting.

Theorem 6.51 (Tennenbaum) *If $\mathcal{M} \vDash \mathsf{PA}$ is a data type and has stable* std *then $\mathcal{M} \cong \mathbb{N}$.*

Proof Assume $\mathcal{M} \ncong \mathbb{N}$ and try to show $\bot$. By Lemma 6.50 there is (we can remove the $\neg\neg$ since we are trying to show $\bot$) $c\colon \mathcal{M}$ with $\neg\mathsf{Dec}(\lambda n.\, \mathcal{M} \vDash \overline{\pi\, n} \mid c)$. This already leads to the desired contradiction, since Lemma 6.49 showed that divisibility is decidable if $\mathcal{M}$ is a data type. This establishes $\neg\neg\mathcal{M} \cong \mathbb{N}$, which is equivalent to $\mathcal{M} \cong \mathbb{N}$ due to the stability of std. □

6.8 Variants of Tennenbaum's Theorem

We will now look at two variants of Theorem 6.51, which will use a stronger notion of inseparable formulas, requiring an intuitionistic deductive proof that there is no intersection between the two formulas. We assume the existence of a formula pair like this as an axiom:

Axiom 6.52 *There are inseparable Σ_1 formulas $\alpha(x), \beta(x)$ for which we have $\mathsf{HA} \vdash_i \dot{\neg}\, \dot{\exists}\, x.\, \alpha(x) \mathbin{\dot{\wedge}} \beta(x)$. Formulas with this property are called* HA***-inseparable****.*

As stated by McCarty in [33], this should be derivable by carrying out the usual proof of the existence of inseparable predicates, but formalized on the object level of HA.

6.8.1 Circumventing Overspill

Inspection of the proof of Theorem 6.51 reveals that the dependency on the stability of std goes back to the use of the Overspill Lemma 6.46. As was suggested by Makholm [23] it is possible to circumvent the usage of Overspill completely. This gives us yet another version of Tennenbaum's theorem by eliminating one of the assumptions, ending however with a weaker conclusion.

This version requires a deeper rendering of Lemma 6.48, where the quantification over n is not on the meta level, but on the object level. This can certainly be shown by following the same proof idea as in Lemma 6.28. As this deduction was not mechanized in the thesis we will simply take it as an assumption.

Theorem 6.53 *Assuming that for every binary Δ_0 formula $\alpha(x, y)$ we have*

$$\mathsf{PA} \vdash \dot{\forall}\, n\, b\, \dot{\exists}\, c\, \dot{\forall}\, u \mathbin{\dot{<}} n.\, \big(\dot{\exists}\, z \mathbin{\dot{<}} b.\, \alpha(z, u)\big) \mathbin{\dot{\leftrightarrow}} \dot{\exists} p.\, \varphi_\pi(u, p) \mathbin{\dot{\wedge}} p \mid c$$

then for any data type $\mathcal{M} \vDash \mathsf{PA}$ we have $\forall e.\, \neg\neg\, \mathsf{std}\, e$.

Proof Assume we have $e\colon \mathcal{M}$ with $\neg\mathsf{std}\, e$ and HA-inseparable Σ_1 formulas $\exists z.\, \alpha_0(z, x)$ and $\exists z.\, \beta_0(z, x)$ given by Axiom 6.52, where α_0, β_0 are binary Δ_0 formulas. For the predicate $X := \lambda u\colon \mathbb{N}.\, \mathcal{M} \vDash \dot{\exists}\, z \mathbin{\dot{<}} e.\, \alpha_0(z, \overline{u})$ we then have

- If $\mathbb{N} \vDash \dot{\exists}\, z.\, \alpha_0(z, \overline{n})$ there is $m\colon \mathbb{N}$ with $\mathbb{N} \vDash \alpha_0(\overline{m}, \overline{n})$ and $\mathcal{M} \vDash \alpha_0(\overline{m}, \overline{n})$ by Lemma 3.16. We therefore get Xn.
- Assume that $Xn \wedge \mathbb{N} \vDash \dot{\exists}\, z.\, \beta_0(z, \overline{n})$. Then similarly to above, there is $m\colon \mathbb{N}$ with $\mathcal{M} \vDash \beta_0(\overline{m}, \overline{n})$, showing $\mathcal{M} \vDash \dot{\exists}\, z \mathbin{\dot{<}} e.\, \beta_0(z, \overline{m})$. Together with Xn this contradicts however the deductive disjointness property of the HA-inseparable formulas α_0 and β_0.

Due to the inseparability of the given Σ_1 formulas, this shows that X is not decidable.

By our assumption and soundness we then get

$$\mathcal{M} \vDash \dot{\exists}\, c\, \dot{\forall}\, u \mathbin{\dot{<}} e.\, \big(\dot{\exists}\, z \mathbin{\dot{<}} e.\, \alpha_0(z, u)\big) \mathbin{\dot{\leftrightarrow}} \dot{\exists} p.\, \varphi_\pi(u, p) \mathbin{\dot{\wedge}} p \mid c$$

So there is a code $c\colon \mathcal{M}$ such that the predicate $X := \lambda u\colon \mathbb{N}.\, \mathcal{M} \vDash \dot{\exists}\, z \mathbin{\dot{<}} e.\, \alpha_0(z, \overline{u})$ is coded by it, meaning that for every $u\colon \mathbb{N}$ we have

$$Xu \iff \mathcal{M} \vDash \dot{\exists} p.\, \varphi_\pi(\overline{u}, p) \mathbin{\dot{\wedge}} p \mid c$$

and by Lemma 6.49 we can therefore conclude that X is decidable, giving us a contradiction. □

6.8.2 Variant by McCarty

We will now look at another constructive proof of Tennenbaum's theorem that is due to Charles McCarty [33, 32]. More specifically this version is concerned with HA instead of PA. This is not the only difference in the setup however. McCarty also assumes a more constructive and therefore more restrictive model theory. His chosen semantics is still à la Tarski, but he strengthens it by basing it on intuitionistic set theory IZF. This greatly restricts the class of models under consideration. As was mentioned before our definition of Tarski semantics in Definition 3.8, in the type theory we can replicate this shift to a more more constructive set theory by putting the interpretations of first-order propositional symbols from $\mathbb{P}$ to $\mathbb{T}$. For this section we will assume a change like this has taken place in the definition of $\vDash$. Most importantly this means that disjunctions $\alpha \mathbin{\dot\wedge} \beta$ are now interpreted as informative sum types $\mathcal{M} \vDash \alpha + \mathcal{M} \vDash \beta$.

Lemma 6.54 *For any unary formula $\varphi(x)$ we have*

$$\mathcal{M} \vDash \dot\forall x. \dot\neg\dot\neg \dot\forall y < x. \varphi(y) \mathbin{\dot\vee} \dot\neg\varphi(y).$$

Proof We proceed by induction on x. If $x = 0$ the statement trivially holds. In the inductive step we need to show $\mathcal{M} \vDash \dot\neg\dot\neg \dot\forall y < \mathsf{S}x. \varphi(y) \mathbin{\dot\vee} \dot\neg\varphi(y)$. Since single double negated instances of excluded middle can be shown constructively, we have $\mathcal{M} \vdash \dot\neg\dot\neg(\varphi(x) \mathbin{\dot\vee} \dot\neg\varphi(x))$ and combined with the induction hypothesis this proves the goal. □

Theorem 6.55 *Assuming* MP*, every* HA *model is isomorphic to* $\mathbb{N}$.

Proof Let $\mathcal{M} \vDash \mathsf{HA}$ and α, β be HA-inseparable formulas given by Axiom 6.52. Considering the predicate $X := \lambda n. \mathcal{M} \vDash \alpha(\overline{n})$ we have:

- $\mathbb{N} \vDash \alpha(\overline{n}) \to Xn$ by Corollary 3.20.
- Assuming $Xn \wedge \mathbb{N} \vDash \beta(\overline{n})$ we get $\mathcal{M} \vDash \beta(\overline{n})$ again by absoluteness, but together with Xn it contradicts $\mathsf{HA} \vdash_i \dot\neg \dot\exists x. \alpha(x) \mathbin{\dot\wedge} \beta(x)$.

This shows that X is separating the formulas α, β and can therefore not be decidable. For the purpose of getting a contradiction, we now assume $\mathbb{N} < e \colon \mathcal{M}$. Using Lemma 6.54 on α and instantiating x with e we get

$$\neg\neg\, \mathcal{M} \vDash \dot\forall y < e. \alpha(y) \mathbin{\dot\vee} \dot\neg\alpha(y).$$

We are trying to prove $\bot$ so we can drop the $\neg\neg$ in the above, and since any standard number $\overline{n}$ is smaller than e we can conclude $\mathcal{M} \vDash \alpha(\overline{n}) \mathbin{\dot\vee} \dot\neg\, \alpha(\overline{n})$, showing that there is a decision $\mathcal{M} \vDash \alpha(\overline{n}) + \neg\mathcal{M} \vDash \alpha(\overline{n})$ for every $n \colon \mathbb{N}$. This means we have $\mathsf{Dec}\, X$, in contradiction to the earlier statement about X, therefore showing $\neg\, \mathbb{N} < e \colon \mathcal{M}$, which is equivalent to $\forall e. \neg\neg\, \mathsf{std}\, e$. Because of MP we can then conclude $\mathcal{M} \cong \mathbb{N}$. □

Chapter 7

Conclusion

7.1 Discussion

Having seen three variants of Tennenbaum's theorem, we will now turn to discuss them. To start off, we note that because of the definiteness of equality shown in Lemma 6.31 and due to Fact 4.3, we can conclude that the models in question only need decidable apartness $\neq$ instead of decidable equality.

In his paper [33], McCarty pointed out that a weak version of CT, namely

Definition 7.1 (Weak Church's Thesis) *Every function $f : \mathbb{N} \to \mathbb{N}$ is potentially (i.e. $\neg\neg$) computable.*

suffices for his proof. Adapted to our setting this would amount to assuming

Axiom 7.2 (WCT) *For every function $f : \mathbb{N} \to \mathbb{N}$ there potentially exists a Σ_1 formula $\varphi_f(x, y)$ with $\forall n. \mathsf{Q} \vdash \dot{\forall} y.\, \varphi_f(\overline{n}, y) \dot{\leftrightarrow} \overline{f\, n} \doteq y$.*

instead of CT. This leads to a version of Corollary 6.12 only expressing the potential existence of a pair of inseparable formulas, which still suffices for the proof of Lemma 6.50 since it also only asserts a potential existence. We can therefore strengthen the overall results, since we only need the weaker Axiom 7.2.

In the proofs of all three variants of Tennenbaum's theorem we made use of the global assumptions CT and RT. For now we will leave them out in the statements of the respective variants, so that they read:

1. (Tennenbaum) For every enumerable apartness type $\mathcal{M} \vDash \mathsf{PA}$ with stable std we have $\forall e.\, \mathsf{std}\, e$.
2. (Makholm) For every enumerable apartness type $\mathcal{M} \vDash \mathsf{PA}$ we have $\forall e. \neg\neg\, \mathsf{std}\, e$.
3. (McCarty) Assuming a more constructive model theory, for every $\mathcal{M} \vDash \mathsf{HA}$ we have $\forall e. \neg\neg\, \mathsf{std}\, e$.

On the surface, they all differ quite substantially from the original statement of Tennenbaum's theorem as they do not mention the computability of any operations. But as mentioned in the introduction of Chapter 6 this owes to the fact that our models are situated in a constructive type theory, making the operations of the model computable. In the presence of MP all versions imply that the model is isomorphic to $\mathbb{N}$ under the respective assumptions, but as presented here, in the absence of MP, we can make out differences in their conclusions. We see that the variants due to Makholm and McCarty only show $\forall e.\neg\neg\mathsf{std}\, e$. Most notably, the variant by Makholm tells us how far we can at least get if we drop the stability assumption of std in the first variant. This raises the question about the gap between the two statements, i.e. what is the additional assumption needed to make a proof of "Makholm $\to$ Tennenbaum" possible. One guess in this direction is that in the presence of CT the stability of std for every data type could imply MP–or in the absence of CT– implies the version of Markov's principle which says that every μ-recursive functions which potentially terminates does indeed terminate.

Looking at the proofs of variants 1 and 3 we can see that Lemma 6.49 is responsible for the assumptions on the domain type of the model. In order to get the lemma, we needed the type to have decidable apartness as well as an enumerator. It might be worthwhile to study PA models with decidable numeral divisibility in their own right, to see whether there are interesting equivalent characterizations. The question about the most general preconditions that the type of the model has to satisfy in order to make a proof of the theorem possible, also remains open.

Next, we want to mention that we can give an alternative proof of Lemma 6.49, if instead of being enumerable, the type of $\mathcal{M}$ has a **witness operator**. A witness operator for a type A is a function $\mathcal{W}\colon \forall p^{A\to\mathbb{P}}.\,(\forall x.\,p\,x + \neg p\,x) \to (\exists x.\,p\,x) \to \Sigma\, x.\,p\,x$, which turns any propositional satisfiability proof of a decidable predicate into an informative proof with a concrete witness.

Lemma 7.3 *If $\mathcal{M}$ is discrete and a has a witness operator, then for any $e\colon \mathcal{M}$ and $0 < n$ the proposition $\mathcal{M} \vDash \overline{n} \mid e$ is decidable.*

Proof Let $n\colon \mathbb{N}$ with $n > 0$ and $e\colon \mathcal{M}$ be given. By Lemma 6.32 we propositionally have the existence of $q, r\colon \mathcal{M}$ with

$$e = q \cdot \mu\, n + r \quad \text{and} \quad 0 < \mu\, n \to r < \mu\, n$$

The decidability of the above conjunction follows from the assumption that $\mathcal{M}$ has decidable equality and Lemma 6.35. We can use the witness operator of $\mathcal{M}$ to get computational witnesses q, r. Since $r < \mu\, n$, the element r must be a standard number $r = \mu\, r'$. We can now consider the cases $r' = 0$ or $r' \neq 0$ and finish up in the same way as in the proof of Lemma 6.49. □

This enables a proof of Theorem 6.51 for discrete types which posses a witness operator. We can still push this a bit further. Again, we can weaken discreteness to apartness and inspecting the proof of Theorem 6.51 further, with regards to possible double negations we could introduce, we can see that it suffices for Lemma 7.3 to show $\neg\neg\forall e^{\mathcal{M}},\, 0 < n.\, \mathsf{dec}(\mathcal{M} \vDash \overline{n} \mid e)$. Note that this lemma is of the form $A_1 \wedge A_2 \to C$ which also shows $\neg\neg(A_1 \wedge A_2) \to \neg\neg C$. Since we only need it to conclude with $\neg\neg C$ the lemma can therefore be weakened to:

Lemma 7.4 *If $\mathcal{M}$ is potentially an apartness type with a witness operator, then potentially for any $e\colon \mathcal{M}$ and $0 < n$ the proposition $\mathcal{M} \vDash \overline{n} \mid e$ is decidable.*

Overall this leaves us with the following strengthened version of Theorem 6.51:

Corollary 7.5 *If $\mathcal{M} \vDash \mathsf{PA}$ has stable std and is potentially an apartness type with a witness operator, then $\mathcal{M} \cong \mathbb{N}$.*

Assuming LEM, the model existence theorem [11, 17] gives the construction of a classical enumerable model of the theory PA^*, which is PA together with an axioms enforcing the existence of some element bigger then any numeral (see the introduction of Chapter 6). By the above corollary then, this model cannot have decidable apartness. Note here that the combined usage of LEM and CT is most likely unproblematic in Coq [12] whereas they are inconsistent in systems which cannot differentiate between total functional relations and functions (cf. [49]).

7.2 Coq Mechanization

The mechanization of first-order logic was based on previous developments and starts by the definition of inductive types for terms, formulas and the natural deduction system using deBruijn indices to realize binding of variables to quantifiers. Proofs in PA that are easy on paper can be hard to turn into fully mechanized natural deductions, mainly because of numerous steps and context management needed for simple substitutions during a deduction. To avoid this in the mechanization, whenever possible, we instead verified that a statement holds in every PA model, which brought the management of the proof back to the more comfortable level of the proof assistant.

The mechanization of the undecidability of PA comprises 1650 lines of code (loc), 750 loc for specifications and 900 loc for the lemmas and their proofs, which are represented by the content of Chapter 5. The mechanization of Tennenbaum's theorem as presented in Chapter 6 took about 4200 loc, divided into 1100 loc for specifications and 3100 loc for proofs.

We want to note here that the proof by McCarty as presented in Theorem 6.55 was not mechanized. Switching all of the necessary definitions from $\mathbb{P}$ to $\mathbb{T}$ introduces many typing problems requiring substantial changes in the original project. We did

mechanize the proof by additionally assuming $\forall A\, B\colon \mathbb{P}.\, A \vee B \to A + B$, showing that the outlined proof indeed works out and it does remain as a future goal to go through with the change from $\mathbb{P}$ to $\mathbb{T}$.

7.3 Related Work

Presentations of first-order logic in the context of proof-checking have already been discussed and used by Shankar in [41], Paulson [36] and O'Connor [34], where they were used to investigate Gödel's incompleteness theorems. Our mechanization of first-order logic follows work based on [15, 17]. Undecidability of the satisfiability of Diophantine equations goes back to the work of Davis, Putnam, Robinson and Matiyasevich [8, 9], and was mechanized in Coq by Larchey-Wendling and Forster [29] which is part of a bigger library of mechanized undecidability results [16]. Classical presentations of Tennenbaum's theorem can be found by Smith [44] and Kaye [26], and further investigations into Tennenbaum phenomena were done by Godziszewski and Hamkins in [48]. Constructive treatments have been done by McCarty [32, 33], and Plisko [37]. For a general account of the Church-Turing thesis we refer to [6], for the more specific use in constructive settings the book by Troelstra [49] and for further investigations into CT and its connections to other axioms of synthetic computability theory this work [13] by Forster.

7.4 Future Work

We mentioned before, that by performing our investigation of Tennenbaum's theorem in a constructive type theory, all functions are naturally considered computable. The statement of the theorem therefore no longer mentions computability of addition or multiplication. To bring this back into play we could assume the existence of an abstract universal machine $T : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathcal{O}(\mathbb{N})$ [13]. The computability of addition and multiplication can then be expressed in reference to T. Accordingly, CT then needs to be adapted to only apply to functions $\mathbb{N} \to \mathbb{N}$ which are computable by T and it should again be possible to only assume the weaker WCT version.

In the Coq development, we only considered extensional models, meaning the equality symbol $\doteq$ of PA was always interpreted as the equality on the domain type of the model. This simplified the treatment by enabling the usage of the rewriting capabilities provided by the proof assistant. We are confident that this restriction is not necessary and could in principle be removed.

We did not properly formalize the arithmetic hierarchy on formulas. For a more complete definition of Δ_0 formulas, one would like to include a rule allowing bounded quantifiers in Δ_0 formulas. To simplify matters, we also usually assumed that Σ_1 formulas already have the form $\dot{\exists}\varphi_0$ where φ_0 is a Δ_0 formula. Overall we deem a proper mechanization of the arithmetic hierarchy with convenient definitions as a project on its own.

In Section 6.8.1 we saw how the overspill lemma could be avoided with some stronger assumptions. The deductive proof appearing as an assumption of Theorem 6.53 is certainly provable and one possible way to establish that a deduction must exist comes via the usage of the completeness theorem for classical first-order logic. The completeness theorem however requires the addition of classical reasoning principles to the type theory, which is in conflict with the assumption of CT. A direct mechanized natural deduction proof seems like a feasible future goal, as there are recently developed tools aiding in the construction of first-order deductive proofs [22].

Appendix A

Appendix

As an overview, we again present all of the axioms of PA and the two subsystems that are in use in this text; Robinson arithmetic Q and the undecidable fragment U from Chapter 5.

U only contains the axioms:

$$\begin{aligned}
&\dot\forall\, x.\, O \oplus x \doteq x\\
&\dot\forall\, xy.\, (\mathsf{S}x) \oplus y \doteq S(x \oplus y)\\
&\dot\forall\, x.\, O \otimes x \doteq O\\
&\dot\forall\, xy.\, (\mathsf{S}x) \otimes y \doteq y \oplus x \otimes y.
\end{aligned}$$

Robinson arithmetic Q has the additional axioms:

$$\begin{aligned}
&\dot\forall\, x.\, Sx \doteq O \mathrel{\dot\to} \bot\\
&\dot\forall\, xy.\, Sx \doteq Sy \mathrel{\dot\to} x \doteq y\\
&\dot\forall\, x.\, x = O \mathrel{\dot\vee} \dot\exists\, z.\, x = Sz
\end{aligned}$$

and we get PA by further adding

$$\lambda\varphi.\, \varphi[O] \mathrel{\dot\to} (\dot\forall\, x.\, \varphi[x] \mathrel{\dot\to} \varphi[Sx]) \mathrel{\dot\to} \dot\forall\, x.\, \varphi[x].$$

Bibliography

[1] Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006.

[2] Andrej Bauer. Intuitionistic mathematics for physics, 2008. URL `http://math.andrej.com/2008/08/13/intuitionistic-mathematics-for-physics/`.

[3] Andrej Bauer. Five stages of accepting constructive mathematics. *Bulletin of the American Mathematical Society*, 54(3):481–498, 2017.

[4] Alonzo Church. A set of postulates for the foundation of logic. *Annals of mathematics*, pages 346–366, 1932.

[5] Alonzo Church. A note on the Entscheidungsproblem. *The journal of symbolic logic*, 1(1):40–41, 1936.

[6] B. Jack Copeland. The Church-Turing Thesis. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2020 edition, 2020.

[7] Thierry Coquand and Gérard Huet. *The calculus of constructions*. PhD thesis, INRIA, 1986.

[8] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics*, pages 425–436, 1961.

[9] Martin Davis, Yuri Matijasevič, and Julia Robinson. Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution. American Math. Soc Providence, R. I, 1976.

[10] Nicolaas G. de Bruijn. *Lambda calculus notation with nameless dummies, a tool forautomatic formula manipulation, with application to the Church-Rosser theorem. Indagationes Mathematicae*, 34:381–392, 1972.

[11] Melvin Fitting. Model existence theorems for modal and intuitionistic logics. *The journal of symbolic logic*, 38(4):613–627, 1973.

[12] Yannick Forster. Church's thesis and related axioms in coq's type theory. *arXiv preprint arXiv:2009.00416*, 2020.

[13] Yannick Forster. Church's Thesis and Related Axioms in Coq's Type Theory. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-95977-175-7. doi: 10.4230/LIPIcs.CSL.2021.21. URL `https://drops.dagstuhl.de/opus/volltexte/2021/13455`.

[14] Yannick Forster and Fabian Kunze. A certifying extraction with time bounds from Coq to call-by-value λ-calculus. *arXiv preprint arXiv:1904.11818*, 2019.

[15] 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*, pages 38–51, 2019.

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

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

[18] Harvey Friedman. Some systems of second order arithmetic and their use. In *Proceedings of the international congress of mathematicians (Vancouver, BC, 1974)*, volume 1, pages 235–242. Citeseer, 1975.

[19] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Éditeur inconnu, 1972.

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

[21] J Roger Hindley. *Basic simple type theory*. Number 42. Cambridge University Press, 1997.

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

[23] Henning Makholm (https://math.stackexchange.com/users/14366/hmakholm-

left-over monica). Tennenbaum's theorem without overspill. Mathematics Stack Exchange. URL `https://math.stackexchange.com/q/649457`. (version: 2014-01-24).

[24] Antonius JC Hurkens. A simplification of Girard's paradox. In *International Conference on Typed Lambda Calculi and Applications*, pages 266–278. Springer, 1995.

[25] Thomas Jech. *Set theory*. Springer Science & Business Media, 2013.

[26] Richard Kaye. Tennenbaum's theorem for models of arithmetic. *Set Theory, Arithmetic, and Foundations of Mathematics. Ed. by J. Kennedy and R. Kossak. Lecture Notes in Logic. Cambridge*, pages 66–79, 2011.

[27] Dominik Kirst. *Foundations of Mathematics: A Discussion of Sets and Types*, 2018.

[28] Dominik Kirst and Marc Hermes. Synthetic Undecidability and Incompleteness of First-Order Axiom Systems in Coq. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-188-7. doi: 10.4230/LIPIcs.ITP.2021.23. URL `https://drops.dagstuhl.de/opus/volltexte/2021/13918`.

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

[30] Yuri V. Matijasevič. *Enumerable sets are Diophantine. Soviet Mathematics: Doklady*, 11:354–357, 1970.

[31] Yuri V. Matiyasevich. *On Hilbert's Tenth Problem. Expository Lectures 1*, 2000. URL `https://mathtube.org/sites/default/files/lecture-notes/Matiyasevich.pdf`.

[32] Charles McCarty. Variations on a thesis: intuitionism and computability. *Notre Dame Journal of Formal Logic*, 28(4):536–580, 1987.

[33] Charles McCarty. Constructive validity is nonarithmetic. *The Journal of Symbolic Logic*, 53(4):1036–1041, 1988. ISSN 00224812. URL `http://www.jstor.org/stable/2274603`.

[34] Russell O'Connor. Essential incompleteness of arithmetic verified by Coq. In

International Conference on Theorem Proving in Higher Order Logics, pages 245–260. Springer, 2005.

[35] Christine Paulin-Mohring. Inductive definitions in the system Coq rules and properties. In *International Conference on Typed Lambda Calculi and Applications*, pages 328–345. Springer, 1993.

[36] Lawrence C Paulson. A mechanised proof of Gödel's incompleteness theorems using nominal Isabelle. *Journal of Automated Reasoning*, 55(1):1–37, 2015.

[37] Valerii Egorovich Plisko. Constructive formalization of the Tennenbaum theorem and its applications. *Mathematical notes of the Academy of Sciences of the USSR*, 48(3):950–957, 1990.

[38] Panu Raatikainen. Gödel's Incompleteness Theorems. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2021 edition, 2021.

[39] Fred Richman. Church's thesis without tears. *The Journal of symbolic logic*, 48 (3):797–803, 1983.

[40] Egbert Rijke. Introduction to homotopy type theory. *Lecture notes*, 2018.

[41] Natarajan Shankar. *Proof-checking metamathematics (theorem-proving)*. PhD thesis, The University of Texas at Austin, 1986.

[42] Michael Shulman. Synthetic differential geometry, 2006.

[43] Peter Smith. *An introduction to Gödel's theorems*. Cambridge University Press, 2013.

[44] Peter Smith. Tennenbaum's theorem. 2014.

[45] Gert Smolka. *Modeling and Proving in Computational Type Theory Using the Coq Proof Assistant*. 2021.

[46] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Elsevier, 2006.

[47] Stanley Tennenbaum. Non-archimedean models for arithmetic. *Notices of the American Mathematical Society*, 6(270):44, 1959.

[48] Michał Tomasz Godziszewski and Joel David Hamkins. Computable quotient presentations of models of arithmetic and set theory. *arXiv e-prints*, pages arXiv–1702, 2017.

[49] Anne S Troelstra. *Metamathematical investigation of intuitionistic arithmetic and analysis*, volume 344. Springer Science & Business Media, 1973.

[50] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory*. Number 43. Cambridge University Press, 2000.

[51] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1): 230–265, 1937.

[52] Jaap van Oosten. Gödel's incompleteness theorems. *Departement of Mathematics, Utrecht University*, 2015.

[53] Benjamin Werner. Sets in types, types in sets. In *International Symposium on Theoretical Aspects of Computer Software*, pages 530–546. Springer, 1997.