WISB399 Bachelorscriptie (ECTS 7.5)

The Consequences of Gödel's Incompleteness Theorems for the Consistency of Mathematics

H. (Harm) Verheggen (ID: 6903967)
Supervisor: Dr. J. (Jaap) van Oosten
Utrecht University

June 19, 2024

Utrecht University Sharing science, *shaping tomorrow*

Preface

I would like to state a few things regarding this bachelor thesis. First, I assume that readers know common terms and definitions found in most preliminary courses of mathematical logic. Second, most results that are given without proof can be found in Van Oosten's lecture notes of *Basic Computability Theory* [6]. These notes form the skeleton for this text, hence including each proof with rigorous detail would make the whole too bloated. Naturally, results not stated in Van Oosten's will be referenced to their respective material.

I also would like to take a moment to thank the people who have supported me while I worked on my bachelor thesis. First, I want to express gratitude towards my parents for their endless love and encouragement. Without you I could not have gone to Utrecht University in the first place. Then there is my caring brother, who is always interested in how I am doing and what my current work entails. Additionally, I would like to thank fellow student Mohammed El Badaoui for taking time out of his busy schedule to provide useful criticism and sitting through my long ramblings. Finally, I want to acknowledge my thesis supervisor Dr. Jaap van Oosten for his pleasant collaboration and sage advice. Needless to say, I am responsible for any remaining errors.

Introduction

In 1931, Kurt Gödel (1906-1978) presented two theorems in an Austrian scientific journal. These were known as the first and second incompleteness theorem; together referred to as Gödel's Incompleteness Theorems. The theorems were a response to the *Principia Mathematica* (PM), which was a monumental work consisting of three volumes by Bertrand Russell and Alfred North Whitehead. The goal of PM was to formulate a system of axioms and rules of reasoning within which *all* of the mathematics known at the time could be stated and proved. Quite an ambitious goal, but it had some flaws.

The first flaw was revealed after Gödel assumed that PM satisfied a property he called *ω-consistency* ("omega-consistency"). Under the assumption that PM is ω-consistent, Gödel showed with his first incompleteness theorem that PM is *incomplete*, meaning that there exists a sentence in the language of PM that can neither be proved nor disproved within PM itself. We call such a sentence *independent* of PM. The second incompleteness theorem says that if PM is *consistent* – meaning that there is no sentence that can be *both* proved or disproved – then PM cannot prove its own consistency.

To provide some technical details, Gödel did actually carry out his arguments in a different system he called P. Regardless, it was clear that his two theorems were also applicable to PM. Another detail worth noting is the property of ω-consistency. This is a stronger property than consistency, and has additional flavour. However, J. Barkley Rosser proved in 1936 that the weaker assumption of consistency was enough to conclude that PM is incomplete.

Today, Gödel's Incompleteness Theorems are commonly used for formal systems within which a "certain amount of arithmetic" can be expressed and some "basic rules of arithmetic" can be proved. To state this more explicitly, any system whose language includes the language of elementary arithmetic and which has a fair amount of induction is one that meets the condition. Interestingly, the incompleteness theorems are also applicable to systems with sentences that do not directly state anything regarding natural numbers but rather refer to mathematical objects that can be used to *represent* the natural numbers.

For this thesis it is our aim to explore this more modern usage of Gödel's Incompleteness Theorems. To provide an overview, in Chapter 1 we will study a special family of functions known as *primitive recursive functions*. Roughly speaking, a primitive recursive function can be computed by a computer program whose loops are all for-loops. These will be relevant in Chapter 2 where we talk about the formal system of *Peano Arithmetic* (PA). It is within this same chapter that we learn that every primitive recursive function can be *represented* in PA by something called a Δ_1-formula, which will be crucial in the construction of the incompleteness theorems. Then, in Chapter 3, we will construct and discuss Gödel's Incompleteness Theorems. In particular, we will see that PA is incomplete and that it does not prove its own consistency. Finally, Chapter 4 will be more philosophical as we review some consequences and misconceptions of Gödel's work.

Contents

1 Primitive Recursion

The proof for the first incompleteness theorem makes use of a special family of functions called *primitive recursive functions.* Reason for our interest in these functions has to do with Gödel's arithmetization of formal languages, or *Gödel coding.* Using this method, Gödel was able to assign natural numbers to the terms, formulas, and – even – proofs of a formal language. This was crucial for his proofs of the incompleteness theorems.

For that reason we will concentrate on the theory that surrounds primitive recursive functions. And because natural numbers are important, we assume in this chapter that each variable stands for a member of the set $\mathbb{N} = \{0, 1, 2, \ldots\}$ with its usual addition $(+)$ and multiplication $(\cdot)$.

For reference, we have used Van Oosten [6], Smoryński [5], and Boolos, Burgess & Jeffrey [1].

1.1 Primitive Recursive Functions and Relations

When studying mathematics, one frequently encounters expressions that use variables, like x^2 or e^x. In both of these we speak of a number that varies on a variable x and a function that takes x as an input. However, we would like to be more precise when describing expressions in order to avoid ambiguity. For an example on where ambiguity might occur, take $x + y$. This can be interpreted in several ways. We will mention a few below:

a) A natural number (i.e. a constant).

b) A function that takes a 2-tuple (x, y) to give a number (i.e. of the form $\mathbb{N}^2 \to \mathbb{N}$).

c) A function that takes a 2-tuple (y, x) to give a number (i.e. of the form $\mathbb{N}^2 \to \mathbb{N}$).

d) A function that takes a 3-tuple (x, y, z) but only applies x and y to give a number (i.e. of the form $\mathbb{N}^3 \to \mathbb{N}$).

e) A function that only takes x and has y as a parameter to give a number (i.e. of the form $\mathbb{N} \to \mathbb{N}$).

The following definition helps us to distinguish between these interpretations.

Definition 1.1. Let $\vec{x}$ be a sequence of variables $x_1 \cdots x_k$ which may occur in an expression F. Then $\lambda\vec{x}.F$ denotes the function which takes the k-tuple $x_1 \cdots x_k$ and outputs $F(x_1, \ldots, x_k)$.

Example. Using Definition 1.1, we can, respectively, reformulate the previous interpretations of $x+y$ as: (a) $x+y$; (b) $\lambda xy.x+y$; (c) $\lambda yx.x+y$; (d) $\lambda xyz.x+y$; and (e) $\lambda x.x + y$.

Remark. In Definition 1.1 we used something called the λ-*notation*. This λ is there to specify that we are dealing with a function. Nevertheless, we will sometimes provide an expression of the form $F(\vec{x})$ without λ-notation and still call it a function if there is no ambiguity at play.

With a formal definition of functions at hand we will now discuss primitive recursive functions.

Definition 1.2. We call a function *primitive recursive* if it can be constructed by finitely many steps using the following rules:

F1). The *zero function* $Z = \lambda x.0$ is primitive recursive.

F2). The *successor function* $S = \lambda x.x + 1$ is primitive recursive.

F3). The *projection function* $P_i^k = \lambda x_1 \dots x_k.x_i$ is primitive recursive.

F4). If $G_1, \dots, G_l : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^l \to \mathbb{N}$ are primitive recursive, then so is $F : \mathbb{N}^l \to \mathbb{N}$ defined from $G_1, \dots, G_l$ and H by *composition*:

$$F(\vec{x}) = \lambda\vec{x}.H(G_1(\vec{x}), \dots, G_l(\vec{x}))$$

F5). If $G : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^{k+2} \to \mathbb{N}$ are primitive recursive, then so is $F : \mathbb{N}^{k+1} \to \mathbb{N}$ defined from G and H by *primitive recursion*:

$$\begin{aligned} F(0, \vec{x}) &= G(\vec{x}) \\ F(y + 1, \vec{x}) &= H(y, F(y, \vec{x}), \vec{x}) \end{aligned}$$

Remark. A special case occurs in (F5) for $k = 0$. In this scenario we assume that $H : \mathbb{N}^2 \to \mathbb{N}$ is a primitive recursive function and that there exists a $n \in \mathbb{N}$ such that $F : \mathbb{N} \to \mathbb{N}$ defined by

$$\begin{aligned} F(0) &= n \\ F(y + 1) &= H(y, F(y)) \end{aligned}$$

is also primitive recursive. However, note that we have not yet shown that any natural number can be given by a primitive recursive function. This will be done in the next example.

The functions Z, S and P_i^k together form what we will call *basic functions*. One may observe that these basic functions are, on their own, limited in their capability to construct other functions. We would need more tools in order to do this. Fortunately, this is where rules (F4) and (F5) come in, for they allow us to combine the basic functions into new primitive recursive functions. Thus, intuitively speaking, we may view rules (F1)-(F3) as the "building blocks" and rules (F4) and (F5) as the "cement."

Examples. In the following we will give a list of several primitive recursive functions. These were taking from Example 8.3 in Smoryński ([5], p. 59). It will be shown that these are indeed primitive recursive.

a). The *constant function* $C^k_n = \lambda x_1 \dots x_k.n$, for any $n \in \mathbb{N}$.

b). The *addition function* $A = \lambda xy.x + y$.

c). The *multiplication function* $M = \lambda xy.x \cdot y$.

d). The *exponentiation function* $E = \lambda xy.x^y$.

e). The *predecessor function* $\lambda x.\mathrm{pd}(x)$, which is defined by $\mathrm{pd}(0) = 0$, and $\mathrm{pd}(x) = x - 1$ if $x > 0$.

f). The *cut-off subtraction function* $\lambda xy.x \dot{-} y$, which is defined by $x \dot{-} y = x - y$ if $x \geq y$, and $x \dot{-} y = 0$ if $x < y$.

g). The *signum function* $\lambda x.\mathrm{sg}(x)$, which is defined by $\mathrm{sg}(0) = 0$, and $\mathrm{sg}(x) = 1$ if $x > 0$.

h). The *signum complement function* $\lambda x.\overline{\mathrm{sg}}(x)$, which is defined by $\overline{\mathrm{sg}}(0) = 1$, and $\overline{\mathrm{sg}}(x) = 0$ if $x > 0$.

Proof. (a) Note that the functions Z, S, P^k_1 are primitive recursive. Use these to define for any n the function

$$C^k_n(x_1, \dots, x_k) = S^n(Z(P^k_1(x_1, \dots, x_k)))$$

where S^n is defined by $S^0 = P^1_1$, and $S^n = S(S^{n-1})$ for $n > 0$. Using composition we see that S^n is primitive recursive for any $n \in \mathbb{N}$, hence C^k_n is primitive recursive.

(b) According to Definition 1.2(F5), primitive recursion in

$$\begin{aligned} A(x, 0) &= x + 0 &&= x &&= P^1_1(x) \\ A(x, y + 1) &= A(x, y) + 1 &&= S(A(x, y)) &&= S(P^3_2(y, A(x, y), x)) \end{aligned}$$

is applied to the *wrong* variable. Instead, define $B(y, x) = A(x, y)$ and do primitive recursion on y, or do

$$\begin{aligned} A(0, y) &= P^1_1(y) \\ A(x + 1, y) &= S(P^3_2(x, A(x, y), y)) \end{aligned}$$

(c) Using primitive recursion and the addition function A, we can compute $M(0, y) = 0 \cdot y = 0 = Z(y)$, and

$$\begin{aligned} M(x + 1, y) &= (x + 1) \cdot y \\ &= M(x, y) + y \\ &= A(M(x, y), y) \\ &= A(P^3_2(x, M(x, y), y), P^3_3(x, M(x, y), y)) \end{aligned}$$

(d) Observe: define $B(y,x) = E(x,y)$. Then $B(0,x) = 1 = S(Z(x))$, and

$$\begin{aligned} B(y+1,x) &= x \cdot B(y,x) \\ &= M(x, B(y,x)) \\ &= M(P^3_3(y, B(y,x), x), P^3_2(y, B(y,x), x)) \end{aligned}$$

(e) Observe: $\mathrm{pd}(0) = 0$, and $\mathrm{pd}(x+1) = x = P^2_1(x, \mathrm{pd}(x))$.
(f) Observe: define $B(y,x) = x \dot{-} y$. Then $B(0,x) = x = P^1_1(x)$, and

$$B(y+1,x) = \mathrm{pd}(B(y,x)) = \mathrm{pd}(P^3_2(y, B(y,x), x))$$

(g-h) Observe: $\overline{\mathrm{sg}}(x) = 1 \dot{-} x$ and $\mathrm{sg}(x) = 1 \dot{-} \overline{\mathrm{sg}}(x)$ □

This showcased a multitude of primitive recursive functions. But we can also compare natural numbers and see if any *relations* between them can be formally computed. This question can be tackled after introducing the following definition.

Definition 1.3. We call R a k-ary relation if it is a subset of $\mathbb{N}^k$. We write $R(x)$ if and only if $x \in R$, and $\neg R(x)$ if and only if $x \notin R$. Then $\chi_R : \mathbb{N}^k \to \mathbb{N}$ is the characteristic function of the k-ary relation R if the following holds:

$$\chi_R(\vec{x}) = \begin{cases} 0 & \text{if } R(x) \\ 1 & \text{if } \neg R(x) \end{cases}$$

Furthermore, we call R primitive recursive if its characteristic function is primitive recursive.

Remark. As Smoryński points out ([5], p. 61), this characteristic function is unusual compared to the one we commonly see in mathematics. He states that in *Recursion Theory* it is normal to swap out the roles of 0 and 1 such that 0 stands for "truth" and 1 for "falsehood." Smoryński argues that 0 is more noticeable among the natural numbers when doing recursion and, thus, deserve the honour to play the role of truth.

Examples. a) Consider the *absolute difference* function $\lambda xy.|x-y|$. A quick check for

$$|x-y| = x \dot{-} y + y \dot{-} x$$

shows that it is primitive recursive. Now, we may wonder whether the *equality relation*

$$\mathrm{eq} = \{(x,y) \mid x = y\}$$

is primitive recursive. To see that this is indeed the case, we can infer that its characteristic function yields $\chi_{\mathrm{eq}}(x,y) = 0$ if $x = y$, and $\chi_{\mathrm{eq}}(x,y) = 1$ if $x \neq y$. Since $|x-y| = 0$ when $x = y$, and $|x-y| > 0$ for $x \neq y$, we can use the signum function sg to find $\chi_{\mathrm{eq}}(x,y) = \mathrm{sg}(|x-y|)$. Because this is a composition of primitive recursive functions, we conclude that the equality relation is primitive recursive.

b) Let the *ordering relation* be given by

$$\mathrm{leq} = \{(x, y) \mid x \leq y\}$$

To show that this is primitive recursive, verify $\chi_{\mathrm{leq}}(x, y) = \mathrm{sg}(x \dot{-} y)$.

The next theorem covers some additional methods for constructing primitive recursive functions and relations.

Theorem 1.4.

i) If the function $F : \mathbb{N}^{k+1} \to \mathbb{N}$ is primitive recursive, then so are

$$\lambda \vec{x}z. \sum_{y<z} F(\vec{x}, y)$$
$$\lambda \vec{x}z. \prod_{y<z} F(\vec{x}, y)$$
$$\lambda \vec{x}z.(\mu y < z.F(\vec{x}, y) = 0)$$

The last of these is said to be defined by *bounded minimization*, and it produces the least $y < z$ such that $F(\vec{x}, y) = 0$; if such a $y < z$ does not exist, it outputs z.

ii) If R and S are primitive recursive k-ary relations, then so are $R \cap S$, $R \cup S$, $R - S$, and $\mathbb{N}^k - R$.

iii) If R is a primitive recursive $k + 1$-ary relation, then so are

$$R_\exists = \{(\vec{x}, z) \mid \exists y < z.R(\vec{x}, y)\}$$
$$R_\forall = \{(\vec{x}, z) \mid \forall y < z.R(\vec{x}, y)\}$$

Proof. (i) This is shown in Van Oosten ([6], p. 36).

(ii) Observe:

$$\begin{aligned}
\chi_{R \cap S} &= \lambda \vec{x}.\mathrm{sg}(\chi_R + \chi_S(\vec{x})) \\
\chi_{R \cup S} &= \lambda x.\chi_R(\vec{x}) \cdot \chi_S(\vec{x}) \\
\chi_{\mathbb{N}^k - R} &= \lambda x.\overline{\mathrm{sg}}(\chi_R(\vec{x})) \\
\chi_{R-S} &= \chi_{R \cap (\mathbb{N}^k - S)}
\end{aligned}$$

(iii) Observe:

$$\chi_{R_\exists} = \lambda \vec{x}z. \prod_{y<z} \chi_R(\vec{x}, y)$$

$$\chi_{R_\forall} = \lambda \vec{x}z.\mathrm{sg}\left(\sum_{y<z} \chi_R(\vec{x}, y)\right)$$

$\square$

The next proposition is an appetizer for a larger theorem. It introduces a way to define primitive recursive functions using "cases."

Proposition 1.5. [Definition by Cases] If $G_1, G_2, H : \mathbb{N}^k \to \mathbb{N}$ are primitive recursive functions, then so is $F : \mathbb{N}^k \to \mathbb{N}$ defined by

$$F(\vec{x}) = \begin{cases} G_1 & \text{if } H(\vec{x}) = 0 \\ G_2 & \text{if } H(\vec{x}) \neq 0 \end{cases}$$

Proof. We can verify

$$F(\vec{x}) = G_1(\vec{x})\overline{\text{sg}}(H(\vec{x})) + G_2(\vec{x})\text{sg}(H(\vec{x}))$$

□

This result can be extended for any number of pairwise disjoint cases. This is stated by Boolos & Jeffrey ([1], p. 74), which we have modified in our own terms below.

Theorem 1.6. [General Definition by Cases or GDC] Let $R_1, \ldots, R_n \subset \mathbb{N}^k$ be primitive recursive relations such that $R_i \cap R_j = \emptyset$ for all $i \neq j$. Suppose that $G_1, \ldots, G_n : \mathbb{N}^k \to \mathbb{N}$ are primitive recursive functions. Then the function $F : \mathbb{N}^k \to \mathbb{N}$ defined by

$$F(\vec{x}) = \begin{cases} G_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ & \vdots \\ G_n(\vec{x}) & \text{if } R_n(\vec{x}) \end{cases}$$

is also primitive recursive.

Proof. We have

$$F(\vec{x}) = \sum_{i=1}^{n} \text{mult}(G_i(\vec{x}), \chi_{R_i}(\vec{x}))$$

Since $R_1, \ldots, R_n$ are primitive recursive relations, it follows that $\chi_{R_1}, \ldots, \chi_{R_n}$ are also primitive recursive functions. Hence, for each $1 \leq i \leq n$, the function $\text{mult}(G_i(\vec{x}), \chi_{R_i}(\vec{x}))$ is primitive recursive. Using the addition function A to summarize all $\text{mult}(G_i(\vec{x}), \chi_{R_i}(\vec{x}))$ shows that F is primitive recursive. □

The next corollary makes GDC a bit more flexible.

Corollary 1.6.1. Let $R_1, \ldots, R_n \subset \mathbb{N}^k$ be primitive recursive relations such that $R_i \cap R_j = \emptyset$ for all $i \neq j$. Suppose that $G_1, \ldots, G_n, G_{n+1} : \mathbb{N}^k \to \mathbb{N}$ are primitive recursive functions. Then the function $F : \mathbb{N}^k \to \mathbb{N}$ defined by

$$F(\vec{x}) = \begin{cases} G_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ & \vdots \\ G_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ G_{n+1}(\vec{x}) & \text{else} \end{cases}$$

is also primitive recursive.

Proof. Here the case "else" means $R_{n+1}(\vec{x})$, where $R_{n+1} = \mathbb{N}^k - (R_1 \cup \ldots \cup R_n)$. Per Theorem 1.4(ii) we infer via induction that $R_1 \cup \ldots \cup R_n$ is primitive recursive, hence R_{n+1} is a primitive recursive relation. Then applying GDC completes the proof. $\square$

1.2 Pairing Functions and Coding

Although the encoding of formal languages becomes more important in Chapter 3, we will do some initial groundwork in this section. We do this now because we want to encode *sequences* of natural numbers into natural numbers in such a way that valuable operations, like taking the length of the sequence or finding its i-th component, are primitive recursive *in their codes*. To provide an example, given the code of a term t and a formula φ, which are, respectively, denoted by $\ulcorner t \urcorner$ and $\ulcorner \varphi \urcorner$, there exists a primitive recursive function such that $F(\ulcorner \varphi \urcorner, \ulcorner t \urcorner) = \ulcorner \varphi[t/v] \urcorner$. Regardless, for our first order of business we consider *pairing functions*.

Definition 1.7. A function $f : \mathbb{N}^2 \to \mathbb{N}$ is called a *pairing function* if it is bijective. If f is a pairing function, then we say that $f(x, y)$ codes the pair (x, y).

The pairing function we will focus primarily on is the *diagonal enumeration function* or *Cantor pairing function*, which is given by

$$\lambda xy.j(x, y) = \frac{1}{2}(x + y)(x + y + 1) + x = \frac{(x + y)^2 + 3x + y}{2}$$

So, $j(0, 0) = 0$, $j(0, 1) = 1$, $j(1, 0) = 2$, etc. For a visual perspective, view Figure 1 on the next page. Using this figure we can immediately see that j is a pairing function, for we can apply a combinatorial argument to say that each element $(x, y) \in \mathbb{N}^2$ gets paired with a unique number $z \in \mathbb{N}$. Furthermore, we can demonstrate that j is primitive recursive.

Theorem 1.8. The Cantor pairing function j is primitive recursive.

Proof. Define the function $T = \lambda x.\frac{x(x+1)}{2}$. Then $T(x + y) + x = j(x, y)$. Use primitive recursion and compute $T(0) = 0$ and $T(x + 1) = T(x) + S(x)$. Hence, T is primitive recursive, meaning j is primitive recursive via composition. $\square$

Notice for all $x, y \in \mathbb{N}$ that $x, y \leq j(x, y)$. This is convenient, for it allows us to define the *projection functions*

$$j_1(z) = \mu x \leq z.[\exists y \leq z.j(x, y) = z]$$
$$j_2(z) = \mu y \leq z.[\exists x \leq z.j(x, y) = z]$$

for all $z \in \mathbb{Z}$. Therefore, j_1 finds the smallest $x \leq z$ such that a $y \leq z$ exists with the property $j(x, y) = z$, and j_2 does the same but with the roles between x and y swapped. Playing around with these functions shows that the word "projection" is suitable, since we can quickly verify $j_1(j(x, y)) = x$ and $j_2(j(x, y)) = y$ for all $x, y \in \mathbb{N}$.

$$\begin{tikzcd}
\vdots & & & & \\
(0,3) \arrow[dr] & \iddots & & & \\
(0,2) \arrow[dr] & (1,2) \arrow[dr] & \iddots & & \\
(0,1) \arrow[dr] & (1,1) \arrow[dr] & (2,1) \arrow[dr] & \iddots & \\
(0,0) \arrow[u] & (1,0) \arrow[uul] & (2,0) \arrow[uuull] & (3,0) & \cdots
\end{tikzcd}$$

Figure 1: A visualization of the Cantor pairing function.

Additionally, we can demonstrate $j(j_1(z), j_2(z)) = z$ for all $z \in \mathbb{N}$, and that j_1, j_2 are primitive recursive functions. We provide a sketch instead of a proof: for the first part, simply by looking at Figure 1 we can infer $j(j_1(z), j_2(z)) = z$ for all $z \in \mathbb{N}$ since j is bijective. To see that j_1 and j_2 are primitive recursive, use primitive recursion. The result follows by using Figure 1, again, since it shows where "you go" in the computation based on previous results.

Now we have a bijective function that codes a pair of natural numbers into a natural number. But for future endeavours it would be ideal if we had a bijective function $\mathbb{N}^m \to \mathbb{N}$ for any $m \geq 1$ that codes tuples $(x_1, \ldots, x_m)$ of natural numbers into natural numbers. Fortunately, we can do that by using the Cantor pairing function j. This is done by the following.

Definition 1.9. For $m \geq 1$, define the *Cantor m-tuple function* $j^m : \mathbb{N}^m \to \mathbb{N}$ in the following manner:

$$j^1(z) = z$$
$$j^{m+1}(x_1, \ldots, x_m, x_{m+1}) = j(j^m(x_1, \ldots, x_m), x_{m+1})$$

In addition, its i-th *projection function* $j_i^m : \mathbb{N} \to \mathbb{N}$, where $1 \leq i \leq m$, satisfies

$$j^m(j_1^m(z), \ldots, j_m^m(z)) = z$$

for each $z \in \mathbb{N}$, and is given by:

$$j_1^1(z) = z$$
$$j_i^{m+1}(z) = \begin{cases} j_i^m(j_1(z)) & \text{if } 1 \leq i \leq m \\ j_2(z) & \text{if } i = m+1 \end{cases}$$

Remark. Note for $m = 1$ that $j^{m+1}(x, y) = j(x, y)$, and $j_1^{m+1}(z) = j_1(z)$ and $j_2^{m+1} = j_2(z)$. So, for $m = 1$ we get j and its projections back. Moreover, for each $m \geq 1$ we see that j^m is bijective since it is recursively defined by j.

Like the Cantor pairing function j and its projection functions j_1, j_2, the function j^m and its projections j_i^m are primitive recursive. This is stated, among other things, in the following lemma.

Lemma 1.10.

i) $j_i^m(j^m(x_1, \ldots, x_m)) = x_i$, for all $1 \leq i \leq m$.

ii) The functions j^m and j_i^m are primitive recursive.

Proof. (i) We do induction on m. Thus, start with $m = 1$. Then we must have $i = 1$, meaning $j_1^1(z) = z$. Next, suppose the statement holds for all $1 \leq i \leq m$ for some arbitrary m. Then we need to check for $m + 1$ if the statement holds for all $1 \leq i \leq m + 1$. If $1 \leq i \leq m$, then

$$\begin{aligned} j_i^{m+1}(j^{m+1}(x_1, \ldots, x_m, x_{m+1})) &= j_i^m(j_1(j^{m+1}(x_1, \ldots, x_m, x_{m+1}))) \\ &= j_i^m(j_1(j(j^m(x_1, \ldots, x_m), x_{m+1})))) \\ &= j_i^m(j^m(x_1, \ldots, x_m)) \\ &= x_i \end{aligned}$$

If $i = m + 1$, then

$$\begin{aligned} j_i^{m+1}(j^{m+1}(x_1, \ldots, x_m, x_{m+1})) &= j_2(j^{m+1}(x_1, \ldots, x_m, x_{m+1})) \\ &= j_2(j(j^m(x_1, \ldots, x_m), x_{m+1})) \\ &= x_{m+1} \end{aligned}$$

(ii) First do j^m. Intending to do induction, let $m = 1$. Then it is clear that $j^m(z) = j^1(z) = z$ is primitive recursive. Next, suppose j^m is primitive recursive for some arbitrary m. Then

$$j^{m+1}(x_1, \ldots, x_m, x_m) = j(j^m(x_1, \ldots, x_m), x_{m+1})$$

is a composition of primitive recursive functions, hence it is primitive recursive. For j_i^m, observe for $m = 1$ that $i = 1$ and thus $j_i^m(z) = j_1^1 = z$, which is primitive recursive. If $m > 1$, note that $j_i^m(z) = j_2((j_1)^k(z))$ where j_1 is iterated k-times with $k = |m - i - 1|$. Thus, it is a composition of primitive recursive functions, making it primitive recursive. Using Definition by Cases completes the proof. $\square$

Now we know that for every $m \geq 1$ the function j^m is bijective, and that j^m and j_i^m are primitive recursive. This means we can encode any m-tuple or *sequence* $(x_1, \ldots, x_m)$ of natural numbers into a natural number. We will denote such an encoding by the function $\langle x_1, \ldots, x_m \rangle$, which we will call the *code of the sequence*. In particular, the *empty sequence* $(-)$ will have the code $\langle \rangle$. We shall make this precise in the next definition.

Definition 1.11. [Code of the Sequence] Let $\langle\rangle = 0$ denote the *code of the empty sequence*, and for $m > 0$ we have the *code of the sequence*

$$\langle x_0, \ldots, x_{m-1}\rangle = j(m-1, j^m(x_0, \ldots, x_{m-1})) + 1$$

Remark. Observe that the code of the sequence is primitive recursive since it is a composition of primitive recursive functions. Furthermore, a sharp reader may have noticed that we start counting at 0 instead of 1. As Van Oosten states ([6], p. 40), this is conventional when coding arbitrary sequences, and it is also more consistent with the natural numbers that start at 0.

Previously we saw that the coding of the Cantor pairing function j was unique. This is also the case for the code of the sequence.

Lemma 1.12. For every $y \in \mathbb{N}$, either $y = 0$ or there exists a unique $m > 0$ and sequence $\langle x_0, \ldots, x_{m-1}\rangle$ such that $y = \langle x_0, \ldots, x_{m-1}\rangle$.

Proof. The case $y = 0$ is evident. Therefore, let $y > 0$. Observe that j and j^m produce unique solutions for their input values since they are bijective. Then $j(m-1, j^m(x_0, \ldots, x_{m-1}))$ is a composition of bijective functions, hence it is bijective. The function $\langle x_0, \ldots, x_{m-1}\rangle$ is simply that function but shifted by 1, which makes it bijective. Because of all this, each sequence $(x_0, \ldots, x_{m-1})$ corresponds with a unique number among

$$\{j(m-1, 0) + 1, j(m-1, 1) + 1, \ldots\}$$

This set is equivalent to

$$\{j(0, m), j(m, 0), j(m, 1), \ldots\}$$

with the same ordering. Using this list, we see that for each $y > 0$ there exists a unique $m > 0$ and sequence $(x_0, \ldots, x_{m-1})$ such that $y = \langle x_0, \ldots, x_{m-1}\rangle$. □

Now that we are able to assign codes for any arbitrary sequence of finite length, we may consider operations on sequences.

Definition 1.13. The function $\text{lh}(x)$ gives the length of the sequence with code x, and is given by

$$\lambda x.\text{lh}(x) = \begin{cases} 0 & \text{if } x = 0 \\ j_1(x-1) + 1 & \text{if } x > 0 \end{cases}$$

The functions $(x)_i$ gives the i-th element of the sequence with code x if $0 \leq i \leq \text{lh}(x)$, and 0 otherwise, and is given by

$$\lambda x.(x)_i = \begin{cases} j_{i+1}^{\text{lh}(x)}(j_2(x-1)) & \text{if } x > 0 \text{ and } 0 \leq i < \text{lh}(x) \\ 0 & \text{else} \end{cases}$$

These new functions are primitive recursive and possess some neat properties.

Theorem 1.14. The functions $\lambda x.\mathrm{lh}(x)$, $\lambda x.(x)_i$ are primitive recursive. In addition, we have $(\langle\,\rangle)_i = 0$ and $(\langle x_0, \dots, x_{m-1}\rangle)_i = x_i$, and for all $x \in \mathbb{N}$: either $x = 0$ or $x = \langle (x)_0, \dots, (x)_{\mathrm{lh}(x)-1}\rangle$.

Proof. To show that $\mathrm{lh}(x)$ and $(x)_i$ are primitive recursive, see that each case is given by a primitive recursive function and relation, and that the cases are pairwise disjoint. The result then follows from Corollary 1.6.1.

For the other properties, we first compute $(\langle\,\rangle)_i = (0)_i = 0$, and

$$\begin{aligned}
(\langle x_0, \dots, x_{m-1}\rangle)_i &= (j(m-1, j^m(x_0, \dots, x_{m-1})) + 1)_i \\
&= j_{i+1}^{\mathrm{lh}(x)}(j_2(j(m-1, j^m(x_0, \dots, x_{m-1})))) \\
&= j_{i+1}^{m}(j^m(x_0, \dots, x_{m-1})) \\
&= x_i
\end{aligned}$$

For the last part, case $x = 0$ is clear. Thus, assume $x > 0$. Then by Lemma 1.12 there is a unique $m > 0$ and sequence $(x_0, \dots, x_{m-1})$ such that $x = \langle x_0, \dots, x_{m-1}\rangle$. Using our previous computation implies $(x)_i = x_i$ for each i, which yields the desired result. □

Alongside finding the length or i-th component of a sequence, we can consider putting sequences after each other. This can be done with the following function.

Definition 1.15. Let $\lambda xy.x \star y$ denote the *concatenation function* with x and y being codes of sequences. It is given by the scheme:

$$\begin{gathered}
\langle\,\rangle \star y = y \\
x \star \langle\,\rangle = x \\
\langle (x)_0, \dots, (x)_{\mathrm{lh}(x)-1}\rangle \star \langle (y)_0, \dots, (y)_{\mathrm{lh}(y)-1}\rangle = \langle (x)_0, \dots, (x)_{\mathrm{lh}(x)-1}, (y)_0, \dots, (y)_{\mathrm{lh}(y)-1}\rangle
\end{gathered}$$

As might be expected, this function is primitive recursive.

Theorem 1.16. The concatenation function is primitive recursive.

Proof. We first define the *composition function* $\lambda xy.x \circ y$, which is given by $x \circ y = x \star \langle y\rangle$. To show that this is primitive recursive, consider two cases: $x = 0$ and $x = \langle (x)_0, \dots, (x)_{\mathrm{lh}(x)-1}\rangle$, which follows from Theorem 1.14. We find

$$\lambda xy.x \circ y = \begin{cases} \langle y\rangle & \text{if } x = 0 \\ \langle (x)_0, \dots, (x)_{\mathrm{lh}(x)-1}, y\rangle & \text{if } x > 0 \end{cases}$$

Using Definition by Cases we infer that this is primitive recursive. Next, define the function F with the scheme:

$$\begin{aligned}
F(0, x, y) &= x \\
F(w+1, x, y) &= F(w, x, y) \circ (y)_w
\end{aligned}$$

Each clause in the scheme is primitive recursive, hence via primitive recursion we infer that F is primitive recursive. Finally we put $x \star y = F(\mathrm{lh}(y), x, y)$, which is primitive recursive by composition. □

We end this chapter with course-of-values recursion. This method allows us to define $F(y+1, \vec{x})$ directly in terms of *all* its previous values $F(0, \vec{x}), \ldots, F(y, \vec{x})$.

Definition 1.17. Let $G : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^{k+2} \to \mathbb{N}$ be functions. The function $F : \mathbb{N}^{k+1} \to \mathbb{N}$ defined by the clauses

$$\begin{aligned} F(0, \vec{x}) &= G(\vec{x}) \\ F(y+1) &= H(y, j^{y+1}(F(0, \vec{x}), \ldots, F(y, \vec{x})), \vec{x}) \end{aligned}$$

is defined from G and H by *course-of-values recursion*

Using this definition we introduce the following theorem, the proof of which is given in Van Oosten ([6], p. 42).

Theorem 1.18. Suppose $G : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^{k+2} \to \mathbb{N}$ are primitive recursive functions and $F : \mathbb{N}^{k+1} \to \mathbb{N}$ is defined from G and H by course-of-values recursion. Then F is primitive recursive.

2 Peano Arithmetic

As was explained in the introduction, we will show that PA is incomplete and that it does not prove its own consistency using Gödel's Incompleteness Theorems. For that reason we will study PA and its properties. In particular, it will be revealed that a lot of elementary number theory can be carried out in PA.

For reference, this chapter uses Van Oosten [6] and Smoryński [5].

2.1 Basic Properties of PA

We focus on PA as a system of first-order logic, which is a theory of the language $L_{\text{PA}} = \{0, 1; +, \cdot\}$ where 0, 1 are constants, and $+, \cdot$ are binary function symbols satisfying the following axioms:

- $\forall x \neg(x + 1 = 0)$.
- $\forall xy(x + 1 = y + 1 \to x = y)$.
- $\forall x(x + 0 = 0)$.
- $\forall xy(x + (y + 1) = (x + y) + 1)$.
- $\forall x(x \cdot 0 = 0)$.
- $\forall xy(x \cdot (y + 1) = (x \cdot y) + x)$.
- $\forall \vec{x}[(\varphi(0, \vec{x}) \land \forall y(\varphi(y, \vec{x}) \to \varphi(y + 1, \vec{x}))) \to \forall \varphi(y, \vec{x})]$.

The last axiom considers formulas $\varphi(y, \vec{x})$ in L_{PA}. Axioms of this form are called *induction axioms*. The set of all induction axioms is called the *induction scheme*. This would produce an infinite amount of axioms, which are all contained in PA. This is crucial to note, for that would mean there are no finite L_{PA}-theories which have the same models as PA.

Interestingly, the set of natural numbers $\mathbb{N}$ together with the elements 0 and 1, and the usual addition $(+)$ and multiplication $(\cdot)$, is a model of PA. We call this the *standard model* of PA, and denote it by $\mathcal{N}$. However, there are also non-standard models of PA. To provide a popular example: define for every $n \in \mathbb{N}$ a term $\overline{n}$ of L_{PA} by recursion: $\overline{0} = 0$ and $\overline{n+1} = \overline{n} + 1$. Observe that this is not the identity function, for $\overline{3} = (((0 + 1) + 1) + 1)$. The terms $\overline{n}$ are what we call *numerals*, and they are something we will use later. Next, let c be a new constant of L_{PA}, and consider the language $L_{\text{PA}} \cup \{c\}$ with the theory

$$\{\text{axioms of PA}\} \cup \{\neg(c = n) \mid n \in \mathbb{N}\}$$

Every finite subset of this theory can be interpreted in $\mathbb{N}$. Hence, using the Compactness Theorem, we can infer that this is a consistent theory and therefore it has a model M, which has a *non-standard element* c^M.

Even though PA is given by quite simple axioms, the theory is surprisingly strong. However, it is precisely because of its strength that it comes with the weakness of being *incomplete*. Since it is our goal to reach the incompleteness theorems, we will develop some elementary number theory in PA. We start with a proposition that considers basic addition and multiplication. These properties can be proven via (double) induction.

Proposition 2.1.

i) $\mathrm{PA} \vdash \forall x(x = 0 \vee \exists y(x = y + 1))$.

ii) $\mathrm{PA} \vdash \forall xyz(x + (y + z) = (x + y) + z)$.

iii) $\mathrm{PA} \vdash \forall xy(x + y = y + x)$.

iv) $\mathrm{PA} \vdash \forall xyz(x + z = y + z \to x = y)$.

v) $\mathrm{PA} \vdash \forall xyz(x \cdot (y \cdot z) = (x \cdot y) \cdot z)$.

vi) $\mathrm{PA} \vdash \forall xy(x \cdot y = y \cdot x)$.

vii $\mathrm{PA} \vdash \forall xyz(x \cdot (y + z) = (x \cdot y) + (x \cdot z))$.

viii) $\mathrm{PA} \vdash \forall xyz(\neg(z = 0) \wedge x \cdot z = y \cdot z \to x = y)$.

Just like in ordinary elementary number theory, in PA we can formulate ordering of numbers. We will write $x < y$ for the formula $\exists z(x + (z + 1) = y)$. Alongside this abbreviation we shall use $\exists x < y\varphi$ and $\forall x < y\varphi$ for $\exists x(x < y \wedge \varphi)$ and $\forall x(x < y \to \varphi)$, respectively. Additionally, we write $x \leq y$ for $x = y \wedge x < y$ and $x \neq y$ for $\neg(x = y)$. The following two lemmas (Lemma 2.2 & 2.3) will use this new notation.

Lemma 2.2. [Least Number Principle or LNP]

$$\mathrm{PA} \vdash \exists w\varphi(w) \to \exists y(\varphi(y) \wedge \forall x(x < y \to \neg\varphi(x))))$$

Remark. We can read this statement as follows: for any formula φ, if there exists a w such that $\varphi(w)$, then there is a y such that $\varphi(y)$ and for all $x < y$ we have $\neg\varphi(x)$. Then the name "Least Number Principle" is suitable, for y can be viewed as the "least number" such that $\varphi(y)$ holds in PA.

Lemma 2.3. [Principle of Well-Founded Induction or PWFI]

$$\mathrm{PA} \vdash \forall w(\forall v < w\psi(v) \to \psi(w)) \to \forall w\psi(w)$$

Remark. Said differently: if for any w the statement $\forall v < w\psi(v) \to \psi(w)$ holds, then $\psi(w)$ holds for any w. This is intuitively clear, for if the first part is given then $w < w + 1$ would imply $\psi(w) \to \psi(w + 1)$.

2.2 Some Elementary Number Theory in PA

We embark on our journey of elementary number theory in PA with *Euclidean Division.*

Theorem 2.4. [Division with remainder]

$$\mathrm{PA} \vdash \forall xy(y \neq 0 \to \exists ab(x = a \cdot y + b \wedge 0 \leq b < y))$$

In addition, PA proves that such a and b are unique.

Remark. In Theorem 2.4, we call b the *remainder* of x on division by y, and a is the *integer* part of x divided by y.

We introduce some extra notation:

$$\begin{aligned} x|y &\equiv \exists z(x \cdot z = y) \\ \mathrm{irred}(x) &\equiv \forall v \leq x(v|x \to v = 1 \vee v = x) \\ \mathrm{prime}(x) &\equiv x > 1 \wedge \forall yz(x|(y \cdot z) \to x|y \vee x|z) \end{aligned}$$

In order to introduce another function symbol to PA, observe

$$\mathrm{PA} \vdash \forall xy\exists! z((z = 0 \wedge x < y) \vee x = z + y)$$

Then we may add the *subtraction* function symbol $-$ to the language, alongside the axiom

$$\forall xy((x < y \wedge x - y = 0) \vee (x = y + (x - y)))$$

This symbol is necessary for the proof of Bézout's Theorem, which is given in Van Oosten ([6], p. 62). With this stated, we describe some properties of primes in PA.

Proposition 2.5.

i) $\mathrm{PA} \vdash \forall x(x > 1 \to (\mathrm{irred}(x) \leftrightarrow \mathrm{prime}(x)))$

ii) $\mathrm{PA} \vdash \forall x(x > 1 \to \exists v(\mathrm{prime}(v) \wedge v|x))$

Remark. Proposition 2.5 is the conjunction of Propositions 4.4 and 4.5 from Van Oosten ([6], p. 59). This was done in order to make things more compact.

We will now define the least common multiple and greatest common divisor between two numbers. The greatest common divisor in particular will be useful to prove Bézout's Theorem for PA, which is crucial for Gödel's *coding of sequences* in PA.

To define the least common multiple, let $x, y \geq 1$. It is clear that $x|(x \cdot y)$ and $y|(x \cdot y)$. Then by LNP there would exist a least $w > 0$ such that $x|w$ and $y|w$. We denote this least w as $\mathrm{lcm}(x, y)$. From our definition it follows that $\mathrm{lcm}(x, y) \leq x \cdot y$. Using Euclidean Division we write $x \cdot y = a \cdot \mathrm{lcm}(x, y) + b$ with $0 \leq b < \mathrm{lcm}(x, y)$. We see that $x|b$ and $y|b$ would yield a contradiction if $b > 0$ due to the minimality of $\mathrm{lcm}(x, y)$. Hence, $x \cdot y = a \cdot \mathrm{lcm}(x, y)$ with a unique a. This a we will denote by $\mathrm{gcd}(x, y)$. If we write $\mathrm{lcm}(x, y) = y \cdot z$, we infer $x \cdot y = \mathrm{gcd}(x, y) \cdot y \cdot z$ and $\mathrm{gcd}(x, y)|x$. In a similar fashion, we can see that $\mathrm{gcd}(x, y)|y$. Finally, we say that x and y are *relatively prime* if $\mathrm{gcd}(x, y) = 1$.

Theorem 2.6. [Bézout's Theorem in PA]

$$\mathrm{PA} \vdash \forall xy \geq 1 \exists a \leq y, b \leq x(a \cdot x = b \cdot y + \gcd(x, y))$$

Suppose we have a sequence of numbers $x_0, \ldots, x_{n-1}$. We will define a number $m = \max(x_0, \ldots, x_{n-1} n)!$. Since $x_i < m \cdot (i+1) + 1$ for all i, the Chinese Remainder Theorem as provided by Smoryński ([5], p. 44) says we find a unique a such that

$$a \equiv x_i \mod (m \cdot (i+1) + 1)$$

Indeed, for each $i \neq j$ the numbers $m \cdot (i+1) + 1$ and $m \cdot (j+1) + 1$ are relatively prime. Otherwise, we would have a prime p that divides both. But then it would also divides their difference $m \cdot (j - i)$. And since p is prime, it follows that p divides m but also $m \cdot (i+1) + 1$, which causes a contradiction. That said, denote a by the pair (a, m). We will say that this *codes the sequence* $x_0, \ldots, x_{n-1}$ in PA.

The next theorem considers three important properties of coding sequences in PA. The first of these says that every x has a sequence that starts with x. The second mentions that each sequence can be extended. The third and final property is a technical condition which will be needed later. Before we state the theorem, let $\mathrm{rm}(x, y)$ denote the remainder of x on division by y, and $(a, m)_i$ denotes $\mathrm{rm}(a, m \cdot (i+1) + 1)$.

Theorem 2.7. [Properties of Sequences in PA]

i) $\mathrm{PA} \vdash \forall x \exists am((a, 0)_0 = x)$

ii) $\mathrm{PA} \vdash \forall xyam \exists bn(\forall i < y((a, m)_i = (b, n)_i) \wedge (b, n)_y = x)$

iii) $\mathrm{PA} \vdash \forall ami((a, m)_i \leq a)$

2.3 Recursive Functions Represented in PA

In the previous section we have seen that many parts of elementary number theory can be done in PA. For the following, we will show that primitive recursive functions can be *represented* in PA. This will be important for the *Diagonalization Lemma* (Lemma 3.2) in Chapter 3, which will be used in the construction of the first incompleteness theorem. Our method of representing functions in PA will be discussed shortly. But first, we provide some definitions.

Definition 2.8. Let $\varphi, \psi, \chi, \kappa$ be L_{PA}-formulas. We say that φ is a Δ_0-formula if all quantifiers are bounded in φ and it is of the form $\forall x < t$ or $\exists x < t$, where t is a term not containing x. If so, ψ is called a Σ_1-formula if it is of the form $\exists y_1 \ldots y_t \varphi$. Then $\chi(x_1, \ldots, x_k)$ is called a Δ_1-formula if both χ and $\neg\chi$ are equivalent to some ψ, and κ is called a Π_1-formula when it is of the form $\forall y_1 \ldots y_t \psi$. In all these cases, we write $\varphi \in \Delta_0$, $\psi \in \Sigma_1$, $\chi \in \Delta_1$, and $\kappa \in \Pi_1$.

An important property of Σ_1-formulas we will consider here is known as Σ_1-*Completeness.* It says that all closed Σ_1-formulas or Σ_1-*sentences* which are true in the standard model $\mathcal{N}$ are provable in PA and vice versa. Before we give this theorem, recall the definition of numerals as described on page 13 and view the following lemma.

Lemma 2.9.

i)	$(\mathrm{PA} \vdash \overline{n} + \overline{m} = \overline{k}) \iff n + m = k$	for all $n, m, k \in \mathbb{N}$
ii)	$(\mathrm{PA} \vdash \overline{n} \cdot \overline{m} = \overline{k}) \iff n \cdot m = k$	for all $n, m, k \in \mathbb{N}$
iii)	$(\mathrm{PA} \vdash \overline{n} < \overline{m}) \iff n < m$	for all $n, m, k \in \mathbb{N}$
iv)	$\mathrm{PA} \vdash \forall x(x < \overline{n} \leftrightarrow x = \overline{0} \vee \ldots \vee x = \overline{n-1})$	for all $n > 0$

Proof. (i) We do double induction, starting on m. So, first put $m = 0$. Then we need to show $(\mathrm{PA} \vdash \overline{n} + \overline{0} = \overline{k}) \iff n = k$. By the definition of numerals we get $\mathrm{PA} \vdash \overline{n} + \overline{0} = n$, hence we must demonstrate $(\mathrm{PA} \vdash \overline{n} = \overline{k}) \iff n = k$. For this, we do induction on k. For $k = 0$, it is clear that $(\mathrm{PA} \vdash \overline{n} = \overline{0}) \iff n = 0$ since $\mathrm{PA} \vdash \overline{n} = \overline{0} = 0$. Now assume that the statement holds for some arbitrary k. Then we need to show $(\mathrm{PA} \vdash \overline{n} = \overline{k+1}) \iff n = k + 1$. But $(\mathrm{PA} \vdash \overline{n} = \overline{k+1} = \overline{k} + 1) \iff n = k + 1$ since $(\mathrm{PA} \vdash \overline{n} = \overline{k}) \iff n = k$ by induction hypothesis. Now, assume $(\mathrm{PA} \vdash \overline{n} + \overline{m} = \overline{k}) \iff n + m = k$ holds for some arbitrary m. We need to show it also holds for $m + 1$. This is indeed the case, for $(\mathrm{PA} \vdash \overline{n} + \overline{m+1} = \overline{n} + \overline{m} + 1 = \overline{k} + 1) \iff n + m + 1 = k + 1$ per induction hypothesis.

(ii) We do induction on m. Thus, let $m = 0$. Per one of the axioms of PA we we know $\mathrm{PA} \vdash \overline{n} \cdot \overline{0} = \overline{n} \cdot 0 = 0$. From this we infer $(\mathrm{PA} \vdash \overline{k} = 0) \iff k = 0$. Next, suppose $(\mathrm{PA} \vdash \overline{n} \cdot \overline{m} = \overline{k}) \iff n \cdot m = k$ for some arbitrary m. Need to check for $m+1$. Observe $\mathrm{PA} \vdash \overline{n} \cdot (\overline{m+1}) = \overline{n} \cdot \overline{m} + \overline{n} \cdot 1 = \overline{k} + \overline{n}$. Per (i) and the induction hypothesis we conclude $(\mathrm{PA} \vdash \overline{n} \cdot (\overline{m+1}) = \overline{k} + \overline{n}) \iff n \cdot (m+1) = k + n$.

(iii) Just like with (ii), we do induction on m. For $m = 0$, the statement $(\mathrm{PA} \vdash \overline{n} < \overline{m}) \iff n < m$ holds vacuously since $\mathrm{PA} \vdash \forall x \neg(x + 1 = 0)$ and because there exists no $n \in \mathbb{N}$ such that $n < 0$. For the remainder, suppose $(\mathrm{PA} \vdash \overline{n} < \overline{m}) \iff n < m$ holds for some arbitrary m. Then we need to check if it is the same for $m + 1$. But $\mathrm{PA} \vdash \overline{n} < \overline{m+1} = \overline{m} + 1$, and by our induction hypothesis we have $\overline{n} < \overline{m}$ in PA if and only if $n < m$. Since $m < m + 1$, we infer the desired result.

(iv) Finally, we must show $\mathrm{PA} \vdash \forall x(x < \overline{n} \leftrightarrow x = \overline{0} \vee \ldots \vee x = \overline{n-1})$. This will be done via induction on n. For $n = 0$ the statement holds vacuously. Next, suppose the above statement holds for some $n \in \mathbb{N}$. Then we need to check if it also does for $n+1$. First, assume $x < \overline{n+1} = \overline{n} + 1$. Then $x < \overline{n}$ or $x = \overline{n}$. For $x < \overline{n}$ we use the induction hypothesis to infer $x = \overline{0}, x = \overline{1}, \ldots$, or $x = \overline{n-1}$. Hence, $x = \overline{0} \vee \ldots \vee x = \overline{n}$. The converse direction is evident. $\square$

Using Lemma 2.9 we can infer via induction on the L_{PA}-term $t(x_1, \ldots, x_k)$ with variables $x_1, \ldots, x_k$ the following: if $t^{\mathcal{N}}$ is the interpretation of t in $\mathcal{N}$, as a function $\mathbb{N}^k \to \mathbb{N}$, then for each $n_1, \ldots, n_k \in \mathbb{N}$ we have

$$\text{PA} \vdash t(\overline{n_1}, \ldots, \overline{n_k}) = \overline{t^{\mathcal{N}}(n_1, \ldots, n_k)}$$

This property along with Lemma 2.9 will help us to prove Σ_1-Completeness.

Theorem 2.10. [Σ_1-Completeness in PA] Let $\varphi \in \Delta_0$ (or $\varphi \in \Sigma_1$) be a formula with free variables $x_1, \ldots, x_k$, and $n_1, \ldots, n_k \in \mathbb{N}$. Then

$$\text{PA} \vdash \varphi(\overline{n_1}, \ldots, \overline{n_k}) \iff \mathcal{N} \models \varphi[n_1, \ldots, n_k]$$

Proof. We first start with induction on Δ_0-formulas. For the atomic formulas, these follow from our work in Lemma 2.9.

Next, assume $\varphi, \psi \in \Delta_0$ to be formulas with free variables $x_1, \ldots, x_k$ and $n_1, \ldots, n_k \in \mathbb{N}$ such that

$$\begin{aligned} \text{PA} \vdash \varphi(\overline{n_1}, \ldots, \overline{n_k}) &\iff \mathcal{N} \models \varphi[n_1, \ldots, n_k] \\ \text{PA} \vdash \psi(\overline{n_1}, \ldots, \overline{n_k}) &\iff \mathcal{N} \models \psi[n_1, \ldots, n_k] \end{aligned}$$

Then we need to show that the theorem holds for $\varphi \wedge \psi, \varphi \vee \psi, \varphi \to \psi, \forall x < t\varphi$, and $\exists x < t\varphi$. Firstly, these are all Δ_0 since φ and ψ are; this can be shown via induction.

For $\varphi \wedge \psi$, this is true in $\mathcal{N}$ if and only if both of them are true in $\mathcal{N}$. By induction hypothesis we find proofs for both φ and ψ in PA, which holds precisely when their conjunction has a proof in PA.

Checking $\varphi \vee \psi$, this is true in $\mathcal{N}$ if and only if either one of them is true in $\mathcal{N}$. By induction hypothesis we get proofs for either φ or ψ in PA, which holds precisely when their disjunction has a proof in PA.

Next is $\varphi \to \psi$. This is true in $\mathcal{N}$ if and only if either $\neg\varphi$ or ψ are true in $\mathcal{N}$. By induction hypothesis, either $\neg\varphi$ or ψ is provable in PA, which precisely holds when $\varphi \to \psi$ is provable in PA.

Then there is $\forall x < t\varphi$, whose x is a bounded variable. Note that this is equivalent to $\forall x(x < t \to \varphi(x))$. Then use $\text{PA} \vdash t(\overline{n_1}, \ldots, \overline{n_k}) = \overline{t^{\mathcal{N}}(n_1, \ldots, n_k)}$ in conjunction with $\text{PA} \vdash \forall x(x < t \leftrightarrow x = \overline{0} \vee \ldots \vee x = \overline{t^{\mathcal{N}} - 1})$ to find

$$\text{PA} \vdash \forall x < t\varphi \leftrightarrow \varphi(\overline{0}) \wedge \ldots \wedge \varphi(\overline{t^{\mathcal{N}} - 1})$$

From this we infer

$$\begin{aligned} \mathcal{N} \models \forall x < t\varphi &\iff \mathcal{N} \models \varphi(0) \wedge \ldots \wedge \varphi(t^{\mathcal{N}} - 1) \\ &\iff \text{PA} \vdash \varphi(\overline{0}) \wedge \ldots \wedge \varphi(\overline{t^{\mathcal{N}} - 1}) \\ &\iff \text{PA} \vdash \forall x < t\varphi \end{aligned}$$

Finally, we have $\exists x < t\varphi$ which has a x that is a bound variable. Remember that this is equivalent to $\exists x(x < t \wedge \varphi(x))$. Then in a similar fashion as the previous case, we have

$$\text{PA} \vdash \exists x < t\varphi \leftrightarrow \varphi(\overline{0}) \vee \ldots \vee \varphi(\overline{t^{\mathcal{N}} - 1})$$

We use this to conclude

$$\begin{aligned}\mathcal{N} \models \exists x < t\varphi &\iff \mathcal{N} \models \varphi(0) \vee \ldots \vee \varphi(t^{\mathcal{N}} - 1)\\ &\iff \text{PA} \vdash \varphi(\overline{0}) \vee \ldots \vee \varphi(\overline{t^{\mathcal{N}} - 1})\\ &\iff \text{PA} \vdash \exists x < t\varphi\end{aligned}$$

For the last part, we show that the statement holds for any Σ_1-formula. To do this, let $\varphi \in \Sigma_1$ and $\psi \in \Delta_0$ such that $\varphi \equiv \exists x\psi$, which is enough to prove the theorem. Since $\mathcal{N}$ is a model of PA, we know $\text{PA} \vdash \varphi \implies \mathcal{N} \models \varphi$ automatically. For the converse direction, since $\psi \in \Delta_0$ we find

$$\begin{aligned}\mathcal{N} \models \varphi &\implies \mathcal{N} \models \exists x\psi(x)\\ &\implies \exists n.\mathcal{N} \models \psi(n)\\ &\implies \exists n.\text{PA} \vdash \psi(\overline{n})\\ &\implies \text{PA} \vdash \exists x\psi(x)\\ &\implies \text{PA} \vdash \varphi\end{aligned}$$

□

Remark. In particular, the equivalence in Theorem 2.10 does not hold for negations of Σ_1-formulas.

Before stating that every primitive recursive function is Σ_1-*represented* in PA, we need the following two definitions (Definition 2.11 & 2.12).

Definition 2.11. Let $R \subset \mathbb{N}^k$ be a k-ary relation. A L_{PA}-formula $\varphi(x_1, \ldots, x_k)$ of k free variables is said to *represent* R (*numeralwise*) if for all $n_1, \ldots, n_k \in \mathbb{N}^k$ we have

$$\begin{aligned}R(n_1, \ldots, n_k) &\implies \text{PA} \vdash \varphi(\overline{n_1}, \ldots, \overline{n_k}) \quad \text{and}\\ \neg R(n_1, \ldots, n_k) &\implies \text{PA} \vdash \neg\varphi(\overline{n_1}, \ldots, \overline{n_k})\end{aligned}$$

Let $F : \mathbb{N}^k \to \mathbb{N}$ be a k-ary function. A L_{PA}-formula $\varphi(x_1, \ldots, x_k, z)$ of $k+1$ free variables represents F (numeralwise) if for all $n_1, \ldots, n_k \in \mathbb{N}$ we have

$$\begin{aligned}&\text{PA} \vdash \varphi(\overline{n_1}, \ldots, \overline{n_k}, \overline{F(n_1, \ldots, n_k)}) \quad \text{and}\\ &\text{PA} \vdash \exists! z\varphi(\overline{n_1}, \ldots, \overline{n_k}, z)\end{aligned}$$

Finally, we say that a relation or function is Σ_1-*represented* if there is a Σ_1-formula representing it.

Definition 2.12. A function $F : \mathbb{N}^k \to \mathbb{N}$ is called *provably recursive* in PA if it is represented by a Σ_1-formula $\varphi(x_1, \ldots, x_k, z)$ for which

$$\text{PA} \vdash \forall x_1 \ldots x_k \exists! z\varphi(x_1, \ldots, x_k, z)$$

These definitions are necessary for the next theorem.

Theorem 2.13. Every primitive recursive function is Σ_1-represented in PA.

Remark. To prove this theorem, one would need Theorem 4.13 of Van Oosten ([6], p. 66), which states that every primitive recursive function is provably recursive. Furthermore, Theorem 2.13 is a slight modification of Theorem 4.14 of Van Oosten ([6], p. 67), for it talks about *total recursive functions* instead of primitive recursive functions. These so-called total recursive functions are a family of more general functions in which primitive recursive functions are contained. Therefore, we can safely use this alternate formulation.

We close this chapter with a result which will be essential in Chapter 3.

Theorem 2.14. For every primitive recursive function $F : \mathbb{N}^k \to \mathbb{N}$, there is a Δ_1-formula $\varphi_F(x_1, \ldots, x_{k+1})$ which represents F such that

$$\mathrm{PA} \vdash \forall x_1 \ldots x_k \exists! x_{k+1} \varphi_F(x_1, \ldots, x_{k+1})$$

Remark. Proving Theorem 2.14 requires some paperwork and would probably detract from the thesis if we were to write it all out. For that reason we will not provide the proof, but know that it requires us to check each rule in Definition 1.2 and that Theorem 2.13 will be helpful in this endeavour.

3 Gödel's Incompleteness Theorems

Now that we have developed the necessary theory for primitive recursion and PA, we can finally discuss Gödel's Incompleteness Theorems.

For reference, we have used Van Oosten [6].

3.1 Gödel's Coding of Formulas and Diagonalization

We already did some initial groundwork regarding coding in Chapter 1. Here we will continue with what we have done so far. In particular, remember the coding of sequences as defined in Definition 1.11, which is primitive recursive. We will use this to encode any L_{PA}-formula φ with a number denoted by $\ulcorner\varphi\urcorner$. This will be done in such a way that all desired operations on formulas are translated into primitive recursive functions on codes.

We assume that in our language L_{PA} we have a countable list of variables $v_0, v_1, \ldots$. Adding $<$ as a primitive symbol of L_{PA}, we consider the following *coding scheme*:

0	1	v	$+$	$\cdot$	$=$	$<$	$\wedge$	$\vee$	$\to$	$\neg$	$\forall$	$\exists$
0	1	2	3	4	5	6	7	8	9	10	11	12

Given any term t, we define its code $\ulcorner t\urcorner$ by recursion on t: $\ulcorner 0\urcorner = \langle 0\rangle$; $\ulcorner 1\urcorner = \langle 1\rangle$; $\ulcorner v_i\urcorner = \langle 2, i\rangle$; $\ulcorner t+s\urcorner = \langle 3, \ulcorner t\urcorner, \ulcorner s\urcorner\rangle$; and $\ulcorner t\cdot s\urcorner = \langle 4, \ulcorner t\urcorner, \ulcorner s\urcorner\rangle$.

Similarly, we can define the codes for formulas: $\ulcorner t = s\urcorner = \langle 5, \ulcorner t\urcorner, \ulcorner s\urcorner\rangle$; $\ulcorner t < s\urcorner = \langle 6, \ulcorner t\urcorner, \ulcorner s\urcorner\rangle$; $\ulcorner \varphi\wedge\psi\urcorner = \langle 7, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner\rangle$; $\ulcorner \varphi\vee\psi\urcorner = \langle 8, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner\rangle$; $\ulcorner \varphi\to\psi\urcorner = \langle 9, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner\rangle$; $\ulcorner\neg\varphi\urcorner = \langle 10, \ulcorner\varphi\urcorner\rangle$; $\ulcorner\forall v_i\varphi\urcorner = \langle 11, i, \ulcorner\varphi\urcorner\rangle$; and $\ulcorner\exists v_i\varphi\urcorner = \langle 12, i, \ulcorner\varphi\urcorner\rangle$.

Since the code of sequences is primitive recursive, it is immediately clear that the above mentioned codes are primitive recursive in their arguments. Having done this work, we continue with the *second* main idea of Gödel – the Diagonalization Lemma. To arrive at this point, we need the following first.

Lemma 3.1. There is a primitive recursive function Sub such that

$$\mathrm{Sub}(x,y,i) = \begin{cases} \ulcorner\varphi[s/v_i]\urcorner & \text{if } y = \ulcorner\varphi\urcorner \text{ and } x = \ulcorner s\urcorner \\ 0 & \text{else} \end{cases}$$

Proof. We first verify that the properties "x codes a term", "y codes a formula", and "v_i is free in formula φ and variables s are not bound in φ when s is substituted for v_i" are primitive recursive. If these are indeed primitive recursive, then the property "y codes a formula φ and x codes a term v_i and v_i is free in formula φ and variables s are not bound in φ when s is substituted for v_i" is primitive recursive in x, y, i via intersection of primitive recursive relations. We then conclude the proof by Definition on Cases to show that Sub is primitive recursive.

To start, we show that "x codes a term" is primitive recursive. Let χ_t be the characteristic function of this property. Using recursion on codes of terms we have $\chi_t(x) = 0$ if and only if

$$\begin{aligned} x = \langle 0\rangle \vee x = \langle 1\rangle & \\ &\vee \exists i < x(x = \langle 2, i\rangle) \\ &\vee \exists ij < x(\chi_t(i) = \chi_t(j) = 0 \wedge x = \langle 3, i, j\rangle) \\ &\vee \exists ij < x(\chi_t(i) = \chi_t(j) = 0 \wedge x = \langle 4, i, j\rangle) \end{aligned}$$

Since each of $\langle 0\rangle, \langle 1\rangle, \langle 3, i, j\rangle, \langle 4, i, j\rangle$ is primitive recursive by the property of codes of sequences, we infer that χ_t is primitive recursive. Hence, the property "x codes a term" is primitive recursive.

Next is "y codes a formula". We denote its characteristic function by χ_f. Via recursion on formulas, we have $\chi_f(y) = 0$ if and only if

$$\begin{aligned} &\exists ij < y(\chi_t(i) = \chi_t(j) = 0 \wedge y = \langle 5, i, j\rangle) \\ \vee&\exists ij < y(\chi_t(i) = \chi_t(j) = 0 \wedge y = \langle 6, i, j\rangle) \\ \vee&\exists ij < y(\chi_f(i) = \chi_f(j) = 0 \wedge y = \langle 7, i, j\rangle) \\ \vee&\exists ij < y(\chi_f(i) = \chi_f(j) = 0 \wedge y = \langle 8, i, j\rangle) \\ \vee&\exists ij < y(\chi_f(i) = \chi_f(j) = 0 \wedge y = \langle 9, i, j\rangle) \\ \vee&\exists i < y(\chi_f(i) = 0 \wedge y = \langle 10, i\rangle) \\ \vee&\exists ij < y(i = \langle 2, i\rangle \wedge \chi_f(j) = 0 \wedge y = \langle 11, i, j\rangle) \\ \vee&\exists ij < y(i = \langle 2, i\rangle \wedge \chi_f(j) = 0 \wedge y = \langle 12, i, j\rangle) \end{aligned}$$

Using a similar argument as before, we see that "y codes a formula" is primitive recursive.

Finally, we demonstrate that "v_i is free in formula φ and variables s are not bound in φ when s is substituted for v_i" is primitive recursive. This is actually the conjunction of two properties, namely "v_i is free in formula φ" and "variables s are not bound in φ when s is substituted for v_i." Showing that both are primitive recursive reveals that "v_i is free in formula φ and variables s are not bound in φ when s is substituted for v_i" is primitive recursive by intersection.

Let χ_b be the characteristic function for "v_i is bound by φ." We then have $\chi_b(v_i, \varphi) = 0$ if and only if φ has the form of one of the following formulas: $\forall v_i\psi$; $\exists v_i\psi$; $\psi \wedge \kappa$ with v_i bound in either ψ or κ; $\psi \vee \kappa$ with v_i bound in either ψ or κ; $\psi \to \kappa$ with v_i bound in either ψ or κ; or $\neg\psi$ with v_i bound in ψ. Each of these can recursively be given by a code of a formula, hence χ_b is primitive recursive by course-of-values recursion. Since "v_i is free by φ" is its negation, it also is primitive recursive.

For "variables s are not bound in φ when s is substituted by v_i," let χ_s be its characteristic function. Then $\chi_s(s, v_i, \varphi) = 0$ if and only if $\chi_b(v_i, s) = 0$ implies $\chi_b(v_i, \varphi) = 0$. This is a composition of primitive recursive functions, hence χ_s is primitive recursive making "variables s are not bound in φ when s is substituted by v_i" primitive recursive. $\square$

In order to prove the first incompleteness theorem, Gödel created a class of sentences that are *independent* of PA, meaning they cannot be proved or disproved in PA. These are famously known as *Gödel sentences*, and they more-or-less say "I am not provable." We use the Diagonalization Lemma to prove that these sentences are indeed independent of PA.

Lemma 3.2. [Diagonalization Lemma] For any L_{PA}-formula φ with free variable v_0, there exists a L_{PA}-formula ψ with the same free variables as φ except for v_0 such that

$$\text{PA} \vdash \psi \leftrightarrow \varphi[\overline{\ulcorner \psi \urcorner}/v_0]$$

Moreover, if $\varphi \in \Pi_1$ then ψ can be chosen to be Π_1 too.

Remark. In literature, the Diagonalization Lemma is sometimes referred to as the "self-reference lemma." However, even though this might heuristically be useful, it can also be quite misleading. To better paint the picture on what we mean, note that the lemma says that there is a provable equivalence between ψ and $\varphi[\overline{\ulcorner \psi \urcorner}/v_0]$, but the lemma does not claim that ψ and $\varphi[\overline{\ulcorner \psi \urcorner}/v_0]$ are the same.

3.2 The First Incompleteness Theorem

Just like the coding of terms and formulas, we can encode proofs in PA via natural numbers. Here is an overview of our coding scheme:

Ass	0	$\vee$I−r	5	$\forall$E	10	$\perp$	15
$\wedge$I	1	$\vee$I−l	6	$\exists$I	11	$\neg\neg$	16
$\wedge$E−r	2	$\to$E	7	$\exists$E	12		
$\wedge$E−l	3	$\to$I	8	$\neg$I	13		
$\vee$E	4	$\forall$I	9	$\neg$E	14		

The coding of these proof trees is done by induction: basic assumption trees with conclusion φ are denoted by the code $\langle 0, \ulcorner \varphi \urcorner \rangle$; suppose we have trees D_1, D_2 with conclusion φ, ψ, respectively; the tree resulting from D_1 and D_2 by $\wedge I$ is $\langle 1, \ulcorner D_1 \urcorner, \ulcorner D_2 \urcorner, \ulcorner \varphi \wedge \psi \urcorner \rangle$, with $\ulcorner D_1 \urcorner$ the code of D_1 and similarly for D_2; If D_2 results from D_1 by $\wedge$E−r, meaning the root of D_1 is labelled by $\varphi \wedge \psi$ and the root of D_2 is labelled as ψ, then $\ulcorner D_2 \urcorner = \langle 2, \ulcorner D_1 \urcorner, \ulcorner \psi \urcorner \rangle$. Something similar is done for $\wedge$E−l but 2 is replaced with 3; If D_4 is the result from trees D_1, D_2, D_3 by $\vee E$, meaning the root of D_1 labelled as $\varphi \vee \psi$, D_2 and D_3 have χ as their labelled root, and D_4 also has this root, whereby in D_2 all the assumptions of φ have been discharged and in D_3 all the assumptions of ψ are discharged, then we say $\ulcorner D_4 \urcorner = \langle 4, \ulcorner D_1 \urcorner . \ulcorner D_2 \urcorner, \ulcorner D_3 \urcorner, \ulcorner \chi \urcorner \rangle$.

The remainder is quite extensive to write out, but we hope the process is clear to the reader: the length of $\ulcorner D \urcorner$ is $n + 2$ where n is the number of "branches" from the root of D. In particular, this n is always $n \leq 3$. Furthermore, the first element of the string of D is always the number that is associated with the operation that is applied to D, and the last element is the label of the root of

D. With this logic, we can – with some time – recover the tree D from its code $\ulcorner D \urcorner$.

Naturally, the coding of proof trees is also primitive recursive in their arguments. With that in mind, we will define some additional functions and explain why these are primitive recursive.

Definition 3.3. Let OA (open assumption), NDT (natural deduction tree), and Ax (axiom) be given by the following:

- $\mathrm{OA}(x)$: gives the code for the set of undischarged assuptions of D given $\ulcorner D \urcorner$.
- $\mathrm{NDT}(x, y)$: y is the code of a formula and x is the code of a correct natural deduction tree with root labelled by the formula coded by y.
- $\mathrm{Ax}(x)$: x is the code of an axiom of PA or the predicate logic (i.e. axioms governing the equality sign $=$: $u = u$; $u = v \wedge v = w \to u = w$; and $t = s \wedge \varphi[t/u] \to \varphi[s/u]$).

Lemma 3.4. The functions OA, NDT, and Ax are primitive recursive.

We give a sketch instead of a proof: writing out OA would involve checking the construction steps for proof trees. Doing so would reveal that OA is primitive recursive by course-of-values recursion and composition of primitive recursive functions. A similar proof would follow for NDT. Finally, Ax is the characteristic function of the set of codes of all axioms of PA and the predicate calculus. Each case is pairwise disjoint and primitive recursive, making Ax primitive recursive by Definition of Cases.

We use these new functions to define a new predicate $\mathrm{Prf}(x, y)$. It can be read as follows: "y is the code of a formula, and x is the code of the correct proof in PA of the formula coded by y." The next definition states this precisely.

Definition 3.5. Define Prf using the following equivalence:

$$\mathrm{Prf}(x, y) \leftrightarrow \mathrm{NDT}(x, y) \wedge \forall z \in \mathrm{OA}(x)\mathrm{Ax}(z)$$

The predicate Prf is crucial for proving the first incompleteness theorem. Note in particular it says that if x is the code of a correct proof of the formula that y codes, that there exists a natural deduction tree where all assumptions that are undischarged are axioms of PA or of the predicate calculus.

Lemma 3.6. Prf is primitive recursive.

Proof. From its definition we can see that Prf is the composition of primitive recursive functions, hence it itself is primitive recursive. □

Since NDT, Ax and Prf are all primitive recursive, we infer via Theorem 2.14 that they can be represented by Δ_1-formulas. Respectively, we denote these by $\overline{\mathrm{NDT}}$, $\overline{\mathrm{Ax}}$, and $\overline{\mathrm{Prf}}$. Then the following lemma is straight-forward.

Lemma 3.7.

i) $\mathrm{PA} \vdash \varphi \implies \mathrm{PA} \vdash \exists x \overline{\mathrm{Prf}}(x, \overline{\ulcorner \varphi \urcorner})$

ii) $\mathrm{PA} \vdash \forall xy (\overline{\mathrm{Prf}}(x, \overline{\ulcorner \varphi \to \psi \urcorner}) \wedge \overline{\mathrm{Prf}}(y, \overline{\ulcorner \varphi \urcorner}) \to \overline{\mathrm{Prf}}(\langle \overline{7}, x, y, \overline{\ulcorner \psi \urcorner} \rangle, \overline{\ulcorner \psi \urcorner}))$

Proof. (i) Let $\mathrm{PA} \vdash \varphi$. Then there exists a proof of φ in PA, hence there exists a legit proof tree D with conclusion φ. Suppose $\ulcorner D \urcorner = x$. Since D is a proof tree, we know $\forall z \in \mathrm{OA}(x)\mathrm{Ax}(x)$ and $\mathrm{NDT}(x, \ulcorner \varphi \urcorner)$. This implies $\mathrm{Prf}(x, \ulcorner \varphi \urcorner)$, which is represented by $\overline{\mathrm{Prf}}(x, \ulcorner \varphi \urcorner)$. Since $\overline{\mathrm{Prf}} \in \Sigma_1$, we infer $\mathrm{PA} \vdash \overline{\mathrm{Prf}}(\overline{n}, \ulcorner \varphi \urcorner)$ by Σ_1-Completeness. Thus, $\mathrm{PA} \vdash \exists x \overline{\mathrm{Prf}}(x, \ulcorner \varphi \urcorner)$.

(ii) Suppose in PA we have $\overline{\mathrm{Prf}}(x, \overline{\ulcorner \varphi \to \psi \urcorner})$ and $\overline{\mathrm{Prf}}(y, \overline{\ulcorner \varphi \urcorner})$ for some arbitrary x and y. Then there exists a legit proof tree for $\varphi \to \psi$ and a legit proof tree for φ. Using implication-elimination (i.e. $\to$E) produces a new correct natural deduction tree with undischarged assumptions consisting only of axioms of PA or the predicate logic. This new tree is represented by $\overline{\mathrm{Prf}}(\langle \overline{7}, x, y, \overline{\ulcorner \psi \urcorner} \rangle, \overline{\ulcorner \psi \urcorner})$. $\square$

Remark. We introduce some abbreviation: let $\Box \varphi$ denote $\exists x \overline{\mathrm{Prf}}(x, \overline{\ulcorner \varphi \urcorner})$. Then Lemma 3.7 can be reformulated as:

D1) $\mathrm{PA} \vdash \varphi \implies \mathrm{PA} \vdash \Box \varphi$

D2) $\mathrm{PA} \vdash \Box(\varphi \to \psi) \wedge \Box \varphi \to \Box \psi$

Using this we can finally state the first incompleteness theorem.

Theorem 3.8. [Gödel's First Incompleteness Theorem] Apply the Diagonalization Lemma to the formula $\neg \exists x \overline{\mathrm{Prf}}(x, v_0)$ to obtain a Π_1-sentence G, such that

$$\mathrm{PA} \vdash G \leftrightarrow \neg \Box G$$

Then G is independent of PA.

Remark. The statement $G \leftrightarrow \neg \Box G$ can be read as "G if and only if G is not provable in PA." Alternatively, it can roughly be viewed as "I am not provable" or "this statement is not provable." Formulations like these can be compared to liar's paradoxes like "I am not true" or "this statement is not true." However, it's important to note that $G \leftrightarrow \neg \Box G$ is formulated in a mathematical language while the liar's paradoxes are found in ordinary languages. Therefore, even though they share some similarities, they are not the same.

As is shown in Van Oosten ([6], p. 78), to prove this theorem we first explain that G can be chosen as Π_1 since $\overline{\mathrm{Prf}} \in \Delta_1$. From that point onwards, we would either assume G or $\neg G$ is provable in PA. If G is provable, we then perceive $\Box G$ by (D1) hence $\neg G$ is provable by our choice of G. But this would mean that $G \wedge \neg G$ is provable in PA, which is absurd. A similar argument occurs when we assume $\neg G$ is provable with one exception: when inferring $\Box G$ in PA we say that $\Box G$ is true in $\mathcal{N}$ by Σ_1-Completeness. Then G would be provable in PA, which again leads to an absurdity. Since G cannot be proved or disproved in PA, we say that G is independent of PA.

3.3 The Second Incompleteness Theorem

The second incompleteness theorem asserts, more or less, that "PA does not prove its own consistency." More formally stated: $\mathrm{PA} \nvdash \neg\Box \perp$. Reading $\neg\Box \perp$ as the sentence expressing the consistency of PA, we will abbreviate it as $\mathrm{Con}_{\mathrm{PA}}$. Using (D2), we can infer $\mathrm{PA} \vdash \Box \perp \to \Box\psi$ for any ψ. Therefore, we will find $\mathrm{PA} \vdash G \to \mathrm{Con}_{\mathrm{PA}}$. For the remainder of this chapter we will see

$$\mathrm{PA} \vdash G \leftrightarrow \mathrm{Con}_{\mathrm{PA}}$$

which will immediately imply the second incompleteness theorem. For this, we need a third rule:

D3) $\mathrm{PA} \vdash \Box\varphi \to \Box\Box\varphi$

But first, let us see that it yields what we want.

Theorem 3.9. For any operation $\Box$ satisfying (D1)-(D3) and any G such that $\mathrm{PA} \vdash G \leftrightarrow \neg\Box G$, we have

$$\mathrm{PA} \vdash G \leftrightarrow \mathrm{Con}_{\mathrm{PA}}$$

Like the first incompleteness theorem, the proof is rather short. The direction $\mathrm{PA} \vdash G \to \mathrm{Con}_{\mathrm{PA}}$ is shown via $\mathrm{PA} \vdash \perp \to G$ and (D1)-(D2) to infer the result $\mathrm{PA} \vdash \Box \perp \to \Box G$. The converse direction requires (D3); by (D2) and the assumption G on PA we find $\mathrm{PA} \vdash \Box G \to \Box(\neg\Box G)$. Combining this with (D3) applied to G on PA gives $\mathrm{PA} \vdash \neg G \to \Box G \to \Box \perp$. By contrapositivity we get $\mathrm{PA} \vdash \mathrm{Con}_{\mathrm{PA}} \to G$ and complete the proof. As was stated before, this immediately proves the second incompleteness theorem.

Corollary 3.9.1. [Gödel's Second Incompleteness Theorem]

$$\mathrm{PA} \nvdash \mathrm{Con}_{\mathrm{PA}}$$

The rule that we want to prove, (D3), is in fact a consequence of a more general theorem. We give this theorem below.

Theorem 3.10. [Formalized Σ_1-Completeness in PA] For every Σ_1-sentence φ in PA we have

$$\mathrm{PA} \vdash \varphi \to \Box\varphi$$

The proof of this theorem is far from trivial. It will require the next and final major result (Lemma 3.11) of this thesis. To formulate it, we assume the language L_{PA} is augmented by the function symbols $\langle\cdot,\ldots,\cdot\rangle, \mathrm{lh}, (\cdot)_i$ for the manipulation of sequences. We will also take the primitive recursive function $n \to \ulcorner\overline{n}\urcorner$ and represent it by the function symbol T; and want function symbols S_f and S_t that satisfy

$$S_f(y,x) = \begin{cases} \ulcorner\varphi[s_0/v_0,\ldots,s_{k-1}/v_{k-1}]\urcorner & \text{if } y = \ulcorner\varphi\urcorner,\ \mathrm{lh}(x) = k, \text{ and } s_i = \ulcorner(x)_i\urcorner \\ & \text{for each } i < k \\ 0 & \text{else} \end{cases}$$

$$S_t(y,x) = \begin{cases} \ulcorner t[s_0/v_0,\ldots,s_{k-1}/v_{k-1}]\urcorner & \text{if } y = \ulcorner t\urcorner,\ \mathrm{lh}(x) = k, \text{ and } s_i = \ulcorner(x)_i\urcorner \\ & \text{for each } i < k \\ 0 & \text{else} \end{cases}$$

Just like before, we are allowed to assume that PA proves the recursions on these functions. With that said, we will state the necessary lemma.

Lemma 3.11. For every Δ_0-formula $\varphi(v_0, \ldots, v_{k-1})$ we have:

$$\mathrm{PA} \vdash \forall x_0 \ldots x_{k-1}(\varphi(\vec{x}) \to \exists y \overline{\mathrm{Prf}}(y, S_f(\overline{\ulcorner \varphi \urcorner}, \langle T(x_0), \ldots, T(x_{k-1})\rangle)))$$

The proof of this lemma is derived by Lemmas 5.9, 5.10, and 5.11 from Van Oosten ([6], p. 83-85). Now we can prove Theorem 3.10.

Proof of Theorem 3.10. Assume $\varphi \in \Sigma_1$. Then there exists a $\psi \in \Delta_0$ such that $\mathrm{PA} \vdash \varphi \leftrightarrow \exists x_0 \ldots x_{k-1}\psi$. By Lemma 3.11 we have

$$\mathrm{PA} \vdash \forall x_0 \ldots x_{k-1}(\psi(\vec{x}) \to \exists y \overline{\mathrm{Prf}}(y, S_f(\overline{\ulcorner \psi \urcorner}, \langle T(x_0), \ldots, T(x_{k-1})\rangle)))$$

Hence

$$\begin{aligned}\mathrm{PA} \vdash \exists x_0 \ldots x_{k-1}\psi(\vec{x}) &\to \exists y \overline{\mathrm{Prf}}(y, S_f(\overline{\ulcorner \psi \urcorner}, \langle T(x_0), \ldots, T(x_{k-1})\rangle))\\ &\to \exists y \overline{\mathrm{Prf}}(y, S_f(\overline{\ulcorner \exists x_0 \ldots x_{k-1}\psi(\vec{x}) \urcorner}, \langle T(x_0), \ldots, T(x_{k-1})\rangle))\end{aligned}$$

This can be rewritten to

$$\mathrm{PA} \vdash \varphi \to \exists y \overline{\mathrm{Prf}}(y, S_f(\overline{\ulcorner \varphi \urcorner}, \langle T(x_0), \ldots, T(x_{k-1})\rangle))$$

By the definition of S_f, this is equivalent to

$$\mathrm{PA} \vdash \varphi \to \Box\varphi$$

□

4 Consequences of the Incompleteness Theorems

Now that we have a better understanding of the incompleteness theorems we consider their use and abuse. First we talk about a concrete application of the theorems; the Paris-Harrington Theorem. Second we state an important consequence to another proposal for the foundation of mathematics, Hilbert's Program. Finally we make a distinction between the mathematical and ordinary usage of "incompleteness" and "consistency" and discuss how one might apply the incompleteness theorems incorrectly.

In order to write this chapter, we have used a Wikipedia article [7], Moerdijk & Van Oosten [4], Zach [8], Hilbert [3], and Franzén [2].

4.1 Paris-Harrington Theorem

First we consider a concrete application of Gödel's Incompleteness Theorems, the Paris-Harrington Theorem. It states that a certain combinatorial principle in *Ramsey Theory* is not provable in PA even though PA can express it. The principle in question is the strengthened finite Ramsey theorem, which is a statement about colorings and natural numbers. It says the following:

> "For any positive integers n, k, m, such that $m \geq n$, one can find N with the following property: if we color each of the n-element subsets of $S = \{1, 2, 3, \ldots, N\}$ with one of the k colors, then we can find a subset Y of S with at least m elements, such that all n-element subsets of Y have the same color, and the number of elements of Y is at least the smallest element of Y."
>
> (Taken from the Wikipedia article *Paris-Harrington theorem* [7].)

To make this more intuitive, let us briefly explain Ramsey's theorem: say we had a large set of objects and wanted to color them differently. No matter how we color these objects, if we have enough of them we will find a group of objects with the same color. To provide an example, suppose we have a group of people at a big party. We ask whether pairs of people have shaken hands or not. We can think of each handshake as being either a "yes" (they have shaken hands) or a "no" (they have not shaken hands). Denote "yes" by green and "no" by red. According to Ramsey's Theorem, if the party is big enough we will find a smaller group of people where either everyone has shaken hands with each other (all "yes", so all green), or no one has shaken hands with each other (all "no", so all red). Now, the strengthened finite Ramsey theorem extends this idea to any finite number of colors.

To bring this back to Gödel's Incompleteness Theorems, the Paris-Harrington theorem roughly says that the strengthened finite Ramsey theorem is unprovable in PA. This was done by showing that if the theorem was in fact provable in PA, then PA would be able to prove its own consistency. But this would contradict the second incompleteness theorem, hence the strengthened finite Ramsey theorem is not provable in PA. This means that the Paris-Harrington theorem describes a true statement about integers that could be stated in the language of arithmetic but is unprovable in PA.

4.2 Hilbert's Program

Remember from the introduction we talked about the *Principia Mathematica* or PM. It was first published in 1910-1913 in three volumes, and tried to reduce mathematics to a single formal system of axioms and rules of reasoning. As we have stated, it has been shown that PM is incomplete if we assume it is consistent. Furthermore, PM cannot prove its own consistency. However, this phenomenon is not unique to PM, for Gödel's Incompleteness Theorems are applicable to other systems; one of which we will discuss here.

In the early 1920s, David Hilbert (1862-1943) had his own proposal for the foundation of mathematics. This proposal became known as Hilbert's Program (HP). In HP, Hilbert attempts to formalize mathematics in axiomatic form, along with a proof that this axiomatization is consistent. The proof in particular was planned to be carried out by something Hilbert referred to as "finitary methods." According to Hilbert, a finitary method considers a restriction on mathematical thought to those objects which are, supposedly, "intuitively present as immediate experience prior to all thought," and methods of reasoning about those objects do not require the introduction of abstract concepts that appeal to, in particular, completed infinite totalities. Simply put, we can view the word "finitary" in this context as simple or constructive.

Using such a method, Hilbert made a distinction between two mathematical worlds, namely the *actual* world of finite things and the *ideal* world of infinitary abstractions. With this he proposed the construction of a logical system capable of describing the actual world, which includes a proof in the ideal world. To make things more explicit, let us denote this logical system by $\mathcal{S}$ along with the theory of PA, since it can be seen as the theory of those truths that can be established by finitary methods. Furthermore, we can safely assume that the ideal world is represented by the system ZF(C), which is the set theory of Zermelo-Fraenkel (along with the axiom of choice). We can then interpret HP with the following formulation from Moerdijk & Van Oosten ([4], p. 119):

- The *weak version* states that PA proves the consistency of ZF(C).
- The *strong version* states that ZF(C) is *conservative*[1] *over* PA, and hence that PA is complete.

Now things start to get interesting. Remember from Chapter 3 that the first incompleteness theorem implies that PA is incomplete. Hence, the strong version of HP is refuted. Moreover, PA is a subsystem of ZF(C), so the consistency of ZF(C) would imply the consistency of PA. But the second incompleteness theorem says that PA cannot prove its own consistency. Thus, the weak version of HP is also refuted by Gödel.

[1]Conservative means that ZF(C) $\vdash \varphi$ implies PA $\vdash \varphi$ for every PA-sentence φ.

It is generally accepted – but not universally – that Gödel's work has shown that HP cannot be carried out. Nevertheless, even if there exists no finitary consistency proof of arithmetic, the question of finding a consistency proof remains. For that reason HP has been revised, and it continues to be influential in the philosophy of mathematics.

4.3 Some Misconceptions about Gödel's theorems

Gödel's work has attracted much attention ever since the incompleteness theorems were first published, both from mathematicians and non-mathematicians alike. And its popularity has not dwindled in the last few decades, as can be seen by the numerous discussion boards and podcasts found on the internet. These discussions can be found in logic, mathematics, physics, computing or philosophy, but also extend to politics, religion, psychology and so much more.

As might be expected, many of these references to the incompleteness theorems outside of the field of formal logic turn out to be nonsensical and based on some big misunderstanding or gross process of free association (for example, "Gödel's Incompleteness Theorems show that a prove of God does not exist," or "all information in science is incomplete and self-referential."). The aim of this section is to high-light a few of these examples and discuss the limits of Gödel's Incompleteness Theorems.

From what we have learned, the incompleteness theorems talk about the consistency and completeness of formal systems. These terms have a technical sense in logic, but they have various senses in ordinary languages. Because of this, many applications outside of mathematics are wrong. We provide a few examples from Franzén ([2], p. 77):

- Religious people claim that all answers are found in the Bible or in whatever text they use. That means the Bible is a complete system, so Gödel seems to indicate it cannot be true. And the same may be said of any religion which claim, as they all do, a final set of answers.

- As Gödel demonstrated, all consistent formal systems are incomplete and all complete formal systems are inconsistent. The U.S. Constitution is a formal system, after a fashion. The Founders made the choice of incompleteness over inconsistency, and the Judicial Branch exists to close that gap of incompleteness.

- Gödel demonstrated that any axiomatic system must be either incomplete or inconsistent, and inasmuch as Ayn Rand's philosophy of Objectivism claims to be a system of axioms and propositions, one of those two conditions must apply.

These examples can be viewed as "incomplete" or "inconsistent" in the ordinary sense. Like, say, the Bible does not state whether or not Moses had a cold on his 18th birthday. Hence, the Bible is "incomplete." And a legal system can be seen as "inconsistent" since there will always be actions and procedures about the law that contradict each other, hence the need for courts and legal decisions.

However, Franzén correctly points out that these misstate the incompleteness theorems. For one, the formal system must be able to formalize a certain amount of arithmetic. Needless to say, the Bible, U.S. Constitution, and Ayn Rand's Objectivism are no sources of arithmetical theorems. Second, are our examples formal systems? In an ordinary sense, perhaps. However, a formal system in mathematical terms is characterized by a formal language, a set of axioms in that language, and a set of formal inference rules which together with the axioms determine a set of theorems of the system. Our examples do not have this capability, hence cannot be applied to Gödel's Incompleteness Theorems.

Conclusion

Gödel's Incompleteness Theorems represent a pivotal moment in the history of mathematics and logic, reshaping our understanding of formal systems and their limitations. As we have seen, the first incompleteness theorem shows that if PA is consistent, then there exists a sentence that cannot be proven within PA itself. The second incompleteness extends this result by demonstrating that PA is a system that cannot prove its own consistency. In particular, these two results hold for any formal system capable of expressing elementary arithmetic.

The publication of these theorems shifted the paradigm of mathematical thought. Personally, before I undertook this thesis, I believed that any statement about mathematics could be proved or disproved given enough time. I was merely concerning myself with the thought if a given statement was true or false. The idea of true unprovable statements seemed alien, just like it did to Hilbert who famously said: "Wir müssen wissen, wir werden wissen" (We must know, we will know). However, Gödel's work challenges the dream of complete formulation for formal systems capable of expressing basic arithmetic, for there exist statements about arithmetic that are true (or false) but unprovable within those systems. Thus, Gödel draws a clear line between truth and provability.

But we do not need to view this as a defeat. Rather, we can allow Gödel's work to humble us in our ongoing quest of mathematical and logical understanding. His theorems remind us that while formal systems are powerful tools, they are ultimately constrained by the very structures they seek to illuminate. In moving forward, instead of viewing mathematics as a monolithic entity with only one correct and true foundation on which it depends, we could study numerous axiomatic systems on a sliding scale of very weak systems to large cardinal axioms in set theory. In doing so, we could raise many more interesting questions.

In closing, this thesis aimed to provide a clear overview of Gödel's Incompleteness Theorems, highlighting their importance and ongoing relevance. The exploration of these ground-breaking results affirms the dynamic and ever-evolving nature of mathematical logic, encouraging future inquiry and discovery.

References

[1] George S. Boolos, John P. Burgess & Richard C. Jeffrey. *Computability and Logic: Fifth Edition*, Cambridge University Press, 2007.

[2] Torkel Franzén. *Gödel's Theorem: An Incomplete Guide to its Use and Abuse*, 2005.

[3] David Hilbert. *Über das Unendliche* (On the Infinite), English translation by Erna Putnam and Gerald J. Massey from *Mathematische Annalen* (Berlin) vol. 95, 1926.

[4] Ieke Moerdijk & Jaap van Oosten. *Sets, Models, and Proofs*, Springer Undergraduate Mathematics Series, 2018.

[5] Craig Smoryński. *Logical Number Theory 1: An Introduction*, Springer Berlin, Heidelberg, 1991.

[6] Jaap van Oosten. Lecture notes for the course *Basic Computability Theory*, 1993, revised 2013. Available at https://webspace.science.uu.nl/~ooste110/syllabi/compthmoeder.pdf.

[7] Wikipedia, the Free Encyclopedia. *Paris-Harrington theorem*, available at https://en.wikipedia.org/wiki/Paris%E2%80%93Harrington_theorem. Consulted on May 20th, 2024.

[8] Richard Zach. *Hilbert's Program*, available at https://plato.stanford.edu/entries/hilbert-program/. Consulted on April 2nd, 2024.