UNIVERSITY OF COPENHAGEN
DEPARTMENT OF MATHEMATICAL SCIENCES

Project Outside the Course Scope

Thorvald Demuth Jørgensen

Gödel's Incompleteness Theorems

-

Date: June 19, 2022

Advisor: Asger Dag Törnquist

Contents

1 Introduction

This project will be about Gödel's two incompleteness theorems. These theorems play an important role in the *crisis of mathematical foundtation* in the beginning of the 20th century. In this introduction I will only briefly explain Hilberts Program and how Gödel's incompleteness theorems undermined this position.

1.1 Hilbert's program and incompleteness

Hilbert saw the infinite and other abstract ideas as *ideal element* that should only be added to the finite and meaningful mathematics only if they were consistence and made the mathematical theory more simple. For example $i^2 = -1$ dose not have any meaning, but it makes the theory of roots of polynomials more simple and it is consistent with the theory of real numbers.

To explain Hilbert's Program we will introduce two formal theories: S that consist of the finite meaningful statement and proof methods and S that consist of the transfinite, idealised of these. The goal of the Hilbert Program was then to show that if φ is any meaningful statement, then we want to show in S that if $T \vdash \varphi$ then $S \vdash \varphi$. So the extension T should not be able to conclude things that are meaningful that could not be conclude in S; this is called Hilbert's Conversation Program. Hilbert himself had remarked that this results follows from a proof in S that T is consistent if we define "meaningful" narrowly enough. Hilbert had hoped such a result would be proven; sadly Gödel's incompleteness theorems show that such a proof is not possible if the theory S is strong enough.

Gödels first incompleteness theorem show that there is a true sentence φ whcih is meaningful even in a very narrow definition of meaning and hence a teorem of T, but is not a theorem of S; i.e a direct counter example to Hilbert's conversation program. Further Gödel's second incompelteness theorem showed that S could not even prove its own consistency.

1.2 What to come

I will start of by introducing and showing some results about a class of functions called primitive recursive functions. After that I will introduce the system of primitive recursive arithmetics, which is powerful enough to compute the primitive recursive functions, i.e basic arithmetics. Next I will show how to encode the formulas of primitive recursive arithmetics and how this enable self-reference, and define the important $Pr(\cdot)$ predicate that can state if a given sentence is provable in our system. Then the next section will contain the two incompleteness theorems and the last bit of groundwork that is needed for the proof of these. The project will end with with some comments on the relation of the $Pr(\cdot)$ predicate to provability logic and some general philosophical remarks. This project will not contain every proof of results used. Further this project will follow Craig Smoryński's book *Self-Reference and Modal Logic*. Further I will use notation in a rather relax way, so I will sometimes use "=" in the meta-theory as well as in the syntax of a given language.

2 Recursive functions

We will start of by defining what a primitive recursive function is. These functions are the functions of natural numbers, that are obtained by recursion. In this project these function are used for recursively encoding of syntactical object in the theory we will define in the next section. The definition will now follow.

Definition 1. A function $f : \omega^n \to \omega$ is called primitive recursive if it can be defined by finitely many steps of combining the following rules:

1. $Z(x) = 0$

2. $S(x) = x + 1$
3. $P_i^n(x_1, \ldots, x_n) = x_i$ for $1 \leq i \leq n$
4. $f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n))$
5. $f(0, x_1, \ldots, x_n) = g(x_1, \ldots, x_n)$
   $f(x + 1, x_1, \ldots, x_n) = h(f(x, x_1, \ldots, x_n), x, x_1, \ldots, x_n)$

⊣

The function Z is called the zero function, S is called the successor function and P_i^n is called the projection. These three functions are called the initial functions and all other primitive recursive functions can be generated by these by rule 4 (called composition) and 5 (called primitive recursion).

Beside primitive recursive functions, we can also define what it means for a relation to primitive recursive; these will also play a crucial role in our encoding.

Definition 2. A relation $R \subseteq \omega^n$ is primitive recursive if its representing function:

$$\chi_R(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } R(x_1, \ldots, x_n) \\ 1 & \text{if } R(x_1, \ldots, x_n) \end{cases}$$

is a primitive recursive function. ⊣

We will now prove that we can add "dummy" variables to our primitive recursive functions.

Proposition 1. Let $g(y_1, \ldots, y_k)$ be primitive recursive. Let $x_1, \ldots, x_n$ be distinct variable and for $i \leq i \leq k$ let z_1 be one of these x_i's. Then the function $f(x_1, \ldots, x_n) = g(z_1, \ldots, z_k)$ is primitive recursive.

Proof. Let $z_i = x_{j_i}$ where $1 \leq j_i \leq n$ Then we have that $z_i = P_{j_i}^n(x_1, \ldots, x_n)$ and thus:

$$f(x_1, \ldots, x_n) = g(P_{j_1}^n(x_1, \ldots, x_n), P_{j_2}^n(x_1, \ldots, x_n), \ldots, P_{j_k}^n(x_1, \ldots, x_n))$$

and thus f is primitive recursive since it is construced of g and the $P_{j_i}^n$'s by *rule 4*. ⊣

We will use this result to prove the following:

Proposition 2. The zero function $Z_n(x_1, \ldots, x_n) = 0$ and the constant function $K_k^n(x_1, \ldots, x_n) = k$ where k is some fixed integer is primitive recursive.

Proof. We will start by showing that Z_n is primitive recursive. We will use proposition 1 where g is Z with $k = 1$ and set z_1 to be x_1.
For the function K_k^n we have that when $k = 0$ it is just the function Z_n. Assume that it is true for k, then $K_{k+1}^n(x_1, \ldots, x_n) = S(K_k^n(x_1, \ldots, x_n))$ gives the right function. ⊣

2.1 Some recursive functions and relations

Having defined the notion of primitive recursive functions and relation, we can now show that some functions is primitive recursive. Not all the results in this section will be proved. We will start of by listing some primitive recursive function:

Proposition 3. The following list of functions are all primitive recursive:

After having show that 2-4 are primitive recursive, we will sometimes use the notation $x + y, x \cdot y$ and x^y for them.

1.	$K_k^n(x_1, \ldots, x_n) = k$	Constant
2.	$A(x, y) = x + y$	Addition
3.	$M(x, y) = x \cdots y$	Multiplication
4.	$E(x, y) = x^y$	Exponentiation
5.	$pd(x) = \begin{cases} x - 1, & x > 0 \\ 0, & x = 0 \end{cases}$	Predecessor
6.	$x \dot{-} y = \begin{cases} x - y & x, \geq y \\ 0, & x < y \end{cases}$	Cut-off subtraction
7.	$sg(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases}$	Signum
8.	$\overline{sg}(x) = \begin{cases} 1, & x = 0 \\ 0, & x > 0 \end{cases}$	Signum complement
9.	$\lvert x - y \rvert = \begin{cases} x - y, & x \geq y \\ y - x, & x < y \end{cases}$	absolute value

Proof. We will show these one at a time and we will use rule 4 and 5 a lot, without stating where we use them:
1. This is prop 2
2. We can define addition in the following way by using rule 5:

$$A(x, 0) = P_1^1(x)$$
$$A(x, S(y)) = S(A(x, y))$$

3. We will again use rule 5:

$$M(x, 0) = Z(x)$$
$$M(x, y + 1) = A(M(x, y), x)$$

4. We will again use rule 5. Here we will use that $K_1^2(x_1, x_2) = 1$ is primitive recursive, and thus we get:

$$E(x, 0) = K_1^2(x, 0) = 1$$
$$E(x, y + 1) = M(E(x, y), x)$$

5. In this cases we have:

$$pd(0) = 0$$
$$pd(x + 1) = x$$

Which is clearly is primitive recursive. 6. Here we have:

$$x \dot{-} 0 = x$$
$$x \dot{-} (y + 1) = pd(x \dot{-} y)$$

and this is primitive recursive since pd is. 7.We have:

$$sg(0) = 0$$
$$sg(x + 1) = 1$$

8. Here we just use $sg(x)$ and define $\overline{sg}(x) = 1 \dot{-} sg(x)$
9. We can define this by composition:

$$\lvert x - y \rvert = (x \dot{-} y) + (y \dot{-} x)$$

⊣

We can now show the following useful lemma:

Lemma 1. Let g_1, g_2 and h be primitive recursive functions and define f as:

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n), & h(x_1, \ldots, x_n) = 0 \\ g_2(x_1, \ldots, x_n), & h(x_1, \ldots, x_n) \neq 0 \end{cases}$$

Then f is also primitive recursive.

Proof.

$$f(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_n) \cdot \overline{sg}(h(x_1, \ldots, x_n)) + g_2(x_1, \ldots, x_n) \cdot sg(h(x_1, \ldots, x_n))$$

And so f is a primitive recursive funtion. ⊣

We will now turn to primitive recursive relations and show a few results about them. We will start of by showing the following result:

Proposition 4. The equality relation is primitive recursive.

Proof. Let $h(x, y)$ be the relation of equality. Then we have:

$$h(x, y) = |x - y| = (x \dot{-} y) + (y \dot{-} x)$$

⊣

We also have the two following propositions that will be stated without proof

Proposition 5. Relations obtained from primitive recursive relations by means of the propositional connectives are also primitive recursive.

Proposition 6. The class of primitive recursive relations are closed under complement, intersection and union. Moreover disjunction corresponds to multiplication, bounded existential quantification corresponds to bounded iterated multiplication. Let $R(y, x_1, \ldots, x_n)$ be given then:

$$\chi_{\exists y \leq x R(y x_1, \ldots, x_n)}(x, x_1, \ldots, x_n) = \prod_{y \leq x} \chi_R(y, x_1, \ldots, x_n)$$

Where $\exists y \leq x \varphi$ means $\exists y (y \leq x \wedge \varphi$. We also have that:

$$\chi_{\forall y \leq x (R(y, x_1, \ldots, x_n)}(x, x_1, \ldots, x_n) = \chi_{\neg \exists y \leq x \neg R(x, x_1, \ldots, x_n)}$$

Where $\forall x \leq x \varphi$ means $\forall y (y \leq x \to \varphi)$

We can now show the following lemma:

Lemma 2.

1. Let $g(y, x_1, \ldots, x_n)$ be primitive recursive. Then f_1 and f_2 are primitive recursive where:
   $f_1(x, x_1, \ldots, x_n) = \sum_{y \leq x} g(y, x_1, \ldots, x_n)$
   $f_2(x, x_1, \ldots, x_n) = \prod_{y \leq x} g(y, x_1, \ldots, x_n)$
2. If $R(y, x_1, \ldots, x_n)$ is primitive recursive, then the relations S and T are also primitive recursive where:
   $S(x, x_1, \ldots, x_n) \leftrightarrow \exists y \leq x (R(y, x_1, \ldots, x_n)$
   $T(x, x_1, \ldots, x_n) \leftrightarrow \forall y \leq x (R(y, x_1, \ldots, x_n)$

Proof. We will only show 1. We can define the function f_1 by recursion in the following way:

$$f_1(x, x_1, \ldots, x_n) = \begin{cases} f_1(0, x_1, \ldots, x_n) = 0 \\ f_1(x+1, x_1, \ldots, x_n) = f_1(x, x_1, \ldots, x_n) = +g(x, x_1, \ldots, x_n + 1) \end{cases}$$

and this is primitive recursive. We show that the bounded product is primitive recursive in similar way:

$$f_2(x, x_1, \ldots, x_n) = \begin{cases} f_2(0, x_1, \ldots, x_n) = 0 \\ f_2(x+1, x_1, \ldots, x_n) = f_2(x, x_1, \ldots, x_n) \cdot g(x+1, x_1, \ldots, x_n) \end{cases}$$

And this is also primitive recursive. ⊣

We will also use the following lemma that holds since the class of primitive recursive functions are closed under composition:

Lemma 3. If $R, f_1, \ldots, f_m$ are primitive recursive, then we also have that

$$R(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$$

is primitive recursive.

We can now show that some relations are primitive recursive:

Proposition 7. The following list of relations are all primitive recursive:

1.	$x \mid y$	x divides y
2.	$x < y$ and $x \leq y$	Inequalities
3.	$Res(x, y) = z$	The residue of x after division by y is z
4.	$z = [x/y]$	z is the greatest integer in x/y
5.	$x \in exp(y)$	x occurs in as an exponent in the dyadic expansion of y

Proof. We will show that each of these relations are primitive recursive.
1. We have that $x|y \leftrightarrow \exists z \leq y(x \cdot y)$ and this is primitive recursive by the lemma.
2. $x < y \leftrightarrow \exists z \leq y(z = x + 1)$ and this is clearly primitive recursive, and we also have $x \leq y \leftrightarrow \exists z \leq y(z = x)$
3.Here we have:

$$Rex(x, y) = z \leftrightarrow \exists w \leq x(x = y \cdot w + z \wedge z < y)$$

And this is also primitive recursive since...
4. $z = [x/y] =\leftrightarrow y \cdot z \leq x < y(z + 1)$
5. The last proof is a bit more tricky:

$$\begin{aligned} x \in exp(y) &\leftrightarrow y = \cdots + 2^x + \cdots \\ &\leftrightarrow Res(y, 2^{x+1}) = 2^x + \cdots \\ &\leftrightarrow [Res(y, 2^{x+}/2^x] = 1 \end{aligned}$$

⊣

The last one of these relations $x \in exp(y)$ is very important, since it gives a bijection between natural numbers and finite sets of natural numbers. This becomes important since we can make a bijection between finite sets of natural numbers and sequences of natural numbers. This property will be used when we have to arithmetize our syntax. We will start of with a definition:

Definition 3. We define the finite set D_x in the following way:

$$D_0 = \{\ \},\ \text{if}, x = 0$$
$$D_x = \{x_0, \ldots, x_{n-1}\},\ \text{if}\ x = 2^{x_0} + \cdots + 2^{x_{n-1}},\ \text{where}\ x_0 < \cdots < x_{n-1}$$

Further we call x the canonical index. ⊣

We can see the sets D_x as a finite strictly increasing sequences if we list its entries in order. A sequence where each $a_i \in \omega$:

$$(a_0, \ldots, a_{n-1})$$

Can be made into the following strictly increasing sequence:

$$(a_0, a_0 + a_1, a_0 + a_1 + a_2 + 2, \ldots, a_0 + \cdots a_{n-1} + n - 1)$$

Since these sequences can be seen as sets, we have an encoding S_x of finite sequences of natural numbers:

$$S_0 = (\ \ )$$
$$S_x = (x_0, \ldots, x_{n-1})$$

Where $D_x = \{x_0, x_0 + x_1 + 1, \ldots, x_0 + \cdots x_{n-1} + n - 1\}$ for $x > 0$. Conversely if $x = 2^{x_0} + \cdots 2^{x_{n-1}}$ where $x_0 < x_1 < \cdots < x_{n-1}$ then we have:

$$S_x = (x_0, x_1 - x_0, \ldots, x_{n-1} - x_{n-2} - 1)$$

The most important ability this gives us, is that we can primitive recursive simulate the following three operations on sequences in the system we will introduce in the next section: finding the length of a sequences, projecting onto a coordinate and concatenating sequences.

Finding the length of a given sequence is intuitively easy. But we will still need a little trick in the proof

Proposition 8. The function $lh(x)$ that gives the length of a sequence x is primitive recursive.

Proof. It is clear that the length of S_0 is 0 and that the length of $S_x = (x_0, \ldots, x_{n-1})$ is n. But need the following trick, to see that the operation that gives the length of a sequences is primitive recursive. We have that:

$$x = 2^{x_0} + 2^{x_0 + 2_1 + 1} + \cdots 2^{x_0 + \cdots 2_{n-1} + n - 1}$$

This means that n is the number of 1's in the binary expansion of x . So we have that the function $lh(x)$ that gives the length of a sequence can be defined as follows:

$$lh(x) = \sum_{y \leq x} \overline{sg}(\chi_{exp}(y, x))$$

Where χ_{exp} is the representing function of $y \in exp(x)$ and thus we have defined $lh(x)$ as a primitive recursive function. ⊣

We will now show that the projection and concatenation are primitive recursive functions. For this we need a lemma and a definition:

Definition 4. Let $g(y, x_1, \ldots, x_n)$ be a function. We then define

$$f(x, x_1, \ldots, x_n) = \mu y < x(g(y, x_1, \ldots, x_n) = 0)$$

as $f(x, x_1, \ldots, x_n)$ being the least $y < x$ such that $g(y, x_1, \ldots, x_n) =$ if such a y exists and $f(x, x_1, \ldots, x_n) = 0$ otherwise. ⊣

Lemma 4. If $g(y, x_1, \ldots, x_n)$ is primitive recursive then so is:

$$f(x, x_1, \ldots, x_n) = \mu y < x(g(y, x_1, \ldots, x_n) = 0)$$

Proof. We will first define the following function by cases, and see that is clearly primitive recursive:

$$f_1(x, x_1, \ldots, x_n) = \begin{cases} 0, & \exists y \leq x(g(y, x_1, \ldots, x_n) = 0) \\ 1, & \neg\exists y \leq x(g(y, x_1, \ldots, x_n) = 0) \end{cases}$$

We now see that $f(x, x_1, \ldots, x_n) = \sum_{y<x} f_1(y, x_1, \ldots, x_n)$ and thus f is primitive recursive. ⊣

We can now show the following proposition.

Proposition 9. The projection function $(x)_i$ for $0 \leq i \leq ln(x) - 1$ that outputs the i entry of a sequences and the function $x * y$ that concatenate two sequences x and y are both primitive recursive.

Proof. We will start of by showing that the projection is primitive recursive. This follows fairly easily from the lemma since we have that:

$$\langle x \rangle_0 = \mu y \leq x(y \in \exp(x))$$
$$\langle x \rangle_{i+1} = \mu y \leq x(y \in \exp(x) \wedge y > ((x)_i) \dot{-} ((x)_i + 1)$$

Defines the function that we want, and this is clearly primitive recursive.

It takes a bit more work to see that concatenation is primitive recursive. We start of by letting:

$$S_x = (x_0, \ldots, x_n - 1),\ S_y = (y_0, \ldots, y_m - 1)$$

We then have that the code of $S_x * S_y = (x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1})$ is the following ugly expression:

$$2^{x_0} + 2^{x_0 + x_1 + 1} + \cdots + 2^{x_0 + x_1 + \cdots x_{n-1} + n - 1} + 2^{x_0 + \cdots x_{n-1} + y_0 + n} + \cdots + 2^{x_0 + \cdots x_{n-1} + y_0 + \cdots + y_{m-1} + n + m - 1}$$

This expression is the same as the less ugly expression:

$$x + 2^{x_0 + \cdots x_{n-1} + x} \cdot y$$

Thus we can define the following primitive recursive function:

$$E(x) = x_0 + \cdots x_{n-1} + n = \sum_{z < lh(x)} ((x)_z + 1)$$

And then we have:

$$x * y = \begin{cases} 0, & x = 0 \wedge y = 0 \\ y, & x = 0 \wedge y \neq 0 \\ x, & x \neq 0 \wedge y = 0 \\ x + 2^{E(x)} \cdot y, & x \neq 0 \wedge y \neq 0 \end{cases}$$

And this is primitive recursive. ⊣

We have now created a theory of finite sequences that is primitive recursive. This was the overall goal of this section, since we will need this ability for our Gödel numbering process. We will need one more result, that we will be important, for this encoding, since it will make it possible for us to encode the more advance syntax of the system of the next section.

Lemma 5. Let g and h be primitive recursive and define f in the following way:

$$f(0, x_1, \ldots, x_n) = g(x_1, \ldots, x_n)$$
$$f(x+1, x_1, \ldots, x_n) = h(\tilde{f}(x, x_1, \ldots, x_n), x, x_1, \ldots, x_n)$$

Where $\tilde{f} = (f(0, x_1, \ldots, x_n), \ldots, f(x, x_1, \ldots, x_n))$ is the so called course of values function for f. Then we have that f is primitive recursive.

Proof. We will define f as $f(x, x_1, \ldots, x_n) = (\tilde{f}(x, x_1, \ldots, x_n))_x$, the result will then clearly follow if we show that $\tilde{f}$ is primitive recursive. But that is the cases since we have:

$$\tilde{f}(0, x_1, \ldots, x_n) = (g(x_1, \ldots, x_n))$$
$$\tilde{f}(x+1, x_1, \ldots, x_n) = \tilde{f}(x, x_1, \ldots, x_n) * (h(\tilde{f}(x, x_1, \ldots, x_n), x, x_1, \ldots, x_n))$$

⊣

We have now introduced the notion a primitive recursive function. In the next section, we will introduce a system that is strong enough to compute these function, which in turn will make it powerful enough to encode its own syntax.

3 Primitive recursive arithmetics

The goal of this section will be to introduce **PRA** and show a few results about it. This is the system that we will want to show is incomplete. A crucial part of the proof to show that **PRA** is incomplete, is to show that it is strong enough to compute the primitive recursive function, and thus the ability to show things about finite sequences of numbers, which in turn we enable us to encode the syntax of **PRA**. This will be the main theorem of this section. When we have shown this theorem, we will know that **PRA** can compute every function from the previous section.

3.1 The formal theory PRA

We will specify the language and rules of **PRA**. We will start of by defining the language of it.

Definition 5. The langauge of PRA consists of the following symbols:

$$\text{Variables} : v_0, v_1, \ldots$$
$$\text{Constant} : \overline{0}$$
$$\text{Function symbol} : \overline{f} \text{ for each recursive function } f$$
$$\text{Relation symbol} :=$$
$$\text{propositional connective} : \neg, \wedge, \vee \rightarrow$$
$$\text{Quantifiers} : \forall, \exists$$

⊣

The set of terms and formulas is defined in the obvious way and is left out. Further the notion of a variable v occurring free in a formula φ and the relation of t being substitutable for v in φ are also defined in the obvious way and are left out. It should also be noted that parentheses are not part of the language of **PRA**, but we will still use them and the connectives is defined via Polish notation, so we do not need them in our language.

We will now list the axioms of **PRA**. There will be four kinds of axioms: propositional axioms, quantifier axioms, equality axioms and non-logical axioms. The first three of these axioms should be known from the logic course; they are stated here for easier reference and for recalling them. The non-logical axioms are now and concerns the primitive recursive functions and induction.

Definition 6. The axioms **PRA** are the following:

1. Propositional axioms
   (a) $\varphi \to (\psi \to \varphi)$
   (b) $(\varphi \to (\psi \to \vartheta)) \to ((\varphi \to \psi) \to (\varphi \to \vartheta))$
   (c) $\varphi \wedge \psi \to \varphi$
   (d) $\varphi \wedge \psi \to \psi$
   (e) $\varphi \to (\psi \to \varphi \wedge \psi)$
   (f) $\varphi \to \varphi \vee \psi$
   (g) $\psi \to \varphi \vee \psi$
   (h) $(\varphi \to \vartheta) \to ((\psi \to \vartheta) \to (\varphi \vee \psi \to \vartheta))$
   (i) $(\varphi \to \psi) \to ((\varphi \to \neg\psi) \to \neg\varphi)$
   (j) $\neg\neg\varphi \to \varphi$
2. Quantifier axioms
   (a) $\forall v \varphi v \to \varphi t$
   (b) $\varphi t \to \exists v \varphi v$

   Where t is substitutable for v in φv in both cases.
3. Equality axioms
   (a) $v_0 = v_0$
   (b) $v_0 = v_0 \to v_1 = v_0$
   (c) $v_0 = v_1 \wedge v_1 = v_2 \to v_0 = v_2$
   (d) $v_i = w \to \bar{f}(v_1, \ldots, v_i, \ldots, v_n) = \bar{f}(v_1, \ldots, w, \ldots, v_n)$

   Where $1 \leq i \leq n$ and $\overline{f}$ is an n.ary function symbol.
4. Non-logical axioms
   (a) Initial functions
      i. $\overline{Z}(v_0) = \overline{0}$
      ii. $\neg(\overline{0} = \overline{S}(v_0))$
      iii. $\overline{S}(v_0) = \overline{S}(v_1) \to v_0 = v_1$
      iv. $\overline{P}_i^n(v_1, \ldots, v_n) = v_i$, for $1 \leq i \leq n$
   (b) Derived functions
      i. $\overline{f}(v_1, \ldots, v_n) = \overline{g}(\overline{h}_1(v_1, \ldots, v_n), \ldots, \overline{h}_m(v_1, \ldots, v_n)$. Here f is defined by composition of $g, h_1, \ldots, h_m$
      ii. Let f be defined by primitive recursion from the primitive functions g and h, then the following two things holds:
      $\overline{f}(\overline{0}, v_1 1, \ldots, v_n) = \overline{g}(v_1, \ldots, v,)$ and
      $\overline{f}(\overline{S}v_0, v_1, \ldots, v_n) = \overline{h}(\overline{f}(v_0, v_1, \ldots, v_n), v_0, v_1, \ldots, v_n)$
   (c) (Induction)
      $\varphi\overline{0} \wedge \forall v(\varphi v \to \varphi(\overline{S}v)) \to \forall v \varphi v$, where φv is $\exists v_n(\overline{f}(v, v_0, v_1, \ldots v_n) = \overline{0})$

⊣

These axioms results in a theory that is a version of Peano Arithmetics; it is just not as "strong". The point of **PRA** is to be able to compute the primitive recursive functions, which play the crucial part of the proof of the incompleteness theorems. The induction axiom is also not a full induction axiom, but it is enough for the goal of this project.

Having written down all the axioms of our system, we will now define the inference rules of **PRA**.

Definition 7. The rules of inference of the system **PRA** are the following:

1. From $\varphi, \varphi \to \psi$ derive ψ (Modus ponens)
2. From $\varphi v \to \psi$ derive $\exists v \varphi v \to \psi$, under the assumption that no v occurs free in ψ (1. Schemata of generalisation)
3. From $\psi \to \varphi v$ derive $\psi \to \forall v \varphi v$, under the assumption that no v occurs free in ψ (2. Schemata of generalisation)

⊣

We can now finally define what a formal derivation on **PRA** is.

Definition 8. A formal derivation in the system **PRA** is a sequence of formulas of formulas of **PRA**: $\varphi_0, \varphi_1, \ldots, \varphi_k$ such that for each φ_i we have that either φ_i is an axiom of **PRA** or it follows either from two other formulas φ_j, φ_l where $j, l < i$ by *modus ponens* or if it follow from one formula φ_j where $j < i$ by one of the two generalisation rules. ⊣

In the rest of this section we will show that **PRA** is strong enough to compute the primitive recursive functions. First we will need a definition and a lemma:

Definition 9. Let $n \in \omega$. The numeral $\overline{n}$ is the term $\overline{S^n 0}$, where S^n is n S's in a row. ⊣

So this definition gives us that $\overline{0}$ is the constant $\overline{0}$ and that $\overline{n+1}$ is $\overline{Sn}$. We now state and prove the following lemma:

Lemma 6. Let φv be given and let t be substitutable for v in φ. If $\mathbf{PRA} \vdash \varphi v$ then $\mathbf{PRA} \vdash \varphi t$

Proof. We start of by letting ψ be the sentence $\overline{0} = \overline{0} \wedge \overline{0} = \overline{0} \to \overline{0} = \overline{0}$. We will now show the derivation:

(1)	φv	Assumption
(2)	$\varphi v \to (\psi \to \varphi v)$	Tautology
(3)	$\psi \to \varphi v$	1,2 Modus ponens
(4)	$\psi \to \forall v \varphi v$	2, generalization
(5)	ψ	Tautology
(6)	$\forall v \varphi v$	4,5, Modus Ponens
(7)	$\forall v \varphi v \to \varphi s$	Quantifier axiom
(8)	φs	6,7 Modus ponens

⊣

The last result of this section, and the first main theorem of this project, will be the following:

Theorem 1. Let f be an n-ary primitive recursive function and let $\overline{f}$ be the function symbol representing it in **PRA**. Then for all $n_1, \dots n_m, n \in \omega$ we have:

$$f((n_1 \dots, n_m) = m \Rightarrow \mathbf{PRA} \vdash \overline{f}(\overline{n_1}, \dots \overline{n_m}) = \overline{n}$$

Proof. We will prove this by induction on the generation of f by the five generating rules for primitive recursive functions of definition 1.
Rule 1: From the axioms we have that $\overline{Z}(v_0) = \overline{0}$ and thus by the lemma we have for all $n_1 \in \omega$ that $\mathbf{PRA} \vdash \overline{Z}(\overline{n_1}) = \overline{0}$ since each $n_1 \in \omega$ is substitutable for v_0
Rule 2: We have that $\overline{n_1 + 1}$ is the same as $\overline{S}(\overline{n_1})$ for all $n_1 \in \omega$ so by axiom 3.a and the lemma we get that $\mathbf{PRA} \vdash \overline{S}(\overline{n_1}) = \overline{n_1 + 1}$
Rule 3: Here we get the result from axiom 4.a.iii which gives us that

$$\mathbf{PRA} \vdash \overline{P}_i^k(\overline{n}_1, \dots, \overline{n}_k) = \overline{n}_i$$

for all $n_1, \dots, n_k \in \omega$.
Rule 4: Assume that f is defined as $f(n_1, \dots, n_m) = n$ by composition of

$$h_i(n_1, \dots, n_m) = l_i$$

for $1 \leq i \leq k$ and

$$g(l_n, \dots, l_m) = n$$

. By the induction hypothesis we have that the following holds:

$$\mathbf{PRA} \vdash \overline{h_i}(\overline{n_1}, \dots, \overline{n_k}) = \overline{l_i} \tag{1}$$

for $1 \leq i \leq k$ and also that

$$\mathbf{PRA} \vdash \overline{g}(\overline{l_1}, \dots, \overline{l_k}) = \overline{k} \tag{2}$$

By axiom 4.b.i) and by using the lemma k times we have that:

$$\mathbf{PRA} \vdash \overline{f}(\overline{n}_1, \dots \overline{n}_m) = \overline{g}(\overline{h}_1(\overline{n}_1, \dots, \overline{n}_m), \dots, \overline{h}_m(\overline{n}_1, \dots, \overline{n}_m)) \tag{3}$$

Further by the use of the lemma on axiom 3.d) we also get the following:

$$\begin{aligned} \mathbf{PRA} \vdash \bigwedge_{i=1}^{k} \overline{h}_i(\overline{n}_1, \dots, \overline{n}_m) = \overline{l}_i \rightarrow \\ \overline{g}(\overline{h}_1(\overline{n}_1, \dots, \overline{n}_m), \dots, \overline{h}_m(\overline{n}_1, \dots, \overline{n}_m)) = \overline{g}(\overline{l}_i, \dots, \overline{l}_k) \end{aligned} \tag{4}$$

Now we can use modus ponens on line (1) and (4) and get that:

$$\mathbf{PRA} \vdash \overline{g}(\overline{h}_1(\overline{n}_1, \dots, \overline{n}_m), \dots, \overline{h}_m(\overline{n}_1, \dots, \overline{n}_m)) = \overline{g}(\overline{l}_i, \dots, \overline{l}_k) \tag{5}$$

Then since equality is transitive and line (3) we get the following:

$$\mathbf{PRA} \vdash \overline{f}(\overline{n}_1, \dots, \overline{n}_m) = g(\overline{l}_1, \dots, \overline{l}_m)$$

By line (2) and again since equality is transitive we get that

$$\mathbf{PRA} \vdash \overline{f}(\overline{n}_1, \dots, \overline{n}_m) = \overline{n}$$

and thus this part is over.
Rule 5 Now assume that f is defined by primitive recursion from g and h in the following way:

$$\begin{aligned} f(0, x_1, \dots, x_k) &= g(x_1, \dots, x_k) \\ f(x+1), x_1, \dots, x_k) &= h(f(x, \dots, x_k), x, x_1, \dots, x_k) \end{aligned}$$

We will prove the following by using induction on m:

$$f(m, n_1, \ldots, n_k) = n \Rightarrow \mathbf{PRA} \vdash \overline{f}(\overline{m}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{n}$$

Base case. Let $m = 0$. We have by axiom 4.b.ii and 3.d that the following holds:

$$\mathbf{PRA} \vdash \overline{f}(\overline{0}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{g}(\overline{n}_1, \ldots, \overline{n}_k)$$
$$\mathbf{PRA} \vdash \overline{f}(\overline{0}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{g}(\overline{n}_1, \ldots, \overline{n}_k) \wedge \overline{g}(\overline{n}_1, \ldots, \overline{n}_k) = \overline{n} \rightarrow \overline{f}(\overline{0}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{n}$$

Since we have $\mathbf{PRA} \vdash \overline{g}(\overline{n}_1, \ldots, \overline{n}_k) = \overline{n}$ by the induction hypothesis we easily see that we get:

$$\mathbf{PRA} \vdash \overline{g}(\overline{0}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{n}$$

Induction step Again by axiom 4.b.ii and the induction hypothesis we have that the following three things holds:

$$\mathbf{PRA} \vdash \overline{f}(\overline{Sm}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{hf}(\overline{m}, \overline{n}_1, \ldots, \overline{n}_k), \overline{m}, \overline{n}_1, \ldots, \overline{n}_k)$$
$$\mathbf{PRA} \vdash \overline{f}(\overline{m}, \overline{n}_1, \ldots, \overline{n}_k = \overline{l}$$
$$\mathbf{PRA} \vdash \overline{h}(\overline{l}, \overline{m}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{n}$$

From these three things we easily see that we get the following:

$$\mathbf{PRA} \vdash \overline{f}(\overline{Sm}, \overline{n}_1, \ldots, \overline{n}_k) = \overline{k}$$

And our result follows from our induction axiom. ⊣

This section will end with an example of how we can formalize our primitive recursive functions within **PRA**. For example our addition function can be defined in the following way.

Let $\overline{g}$ and $\overline{h}$ be terms built up from the projection and successor functions such that:

$$\mathbf{PRA} \vdash \overline{g}(x) = x$$
$$\mathbf{PRA} \vdash \overline{h}(z, y, x) = \overline{S}z$$

We can introduce $\overline{f}$ by using the axioms of **PRA** and get:

$$\overline{f}(\overline{0}, x) = \overline{g}(x)$$
$$\overline{f}(\overline{S}y, x) = \overline{h}(\overline{f}(y, x), y, x)$$

We can then se that the function symbol $\overline{f}_1$ defined by:

$$\overline{f}_1(x, y) = \overline{f}(y, x)$$

Satisfies our definition of *Add* . We also have that:

$$m + n = k \Rightarrow \mathbf{PRA} \vdash \overline{f}(\overline{m}, \overline{n})) = \overline{k}$$

Further in the rest of this project, it will not be shown that the functions we have proven are primitive recursive functions in **PRA**fulfills the standard facts about these functions. Like addition is commutative and associative; we will however use these facts without proof.[1]

In the next section we will encode the syntax of **PRA** and to this end we will use the fact that **PRA** can compute the primitive recursive functions. That is why we did go through all this work to show that that **PRA** can compute the primitive recursive functions.

[1]The proofs of these properties can be found in Mendelson [1964]

4 Gödel numbering and the arithmetics of syntax

We can now finally begin encoding our syntax of **PRA**. This is done by assigning every symbol of our language with a numerical code. So if a is a symbol in our language, then its numerical code will be denoted by $\ulcorner a \urcorner$ called the Gödel number of that symbol. Later on we will assign a Gödel number to each string of symbols in our language. For this end we will use the fundamental theorem of arithmetic, that states that every natural number $a \geq 2$ has a unique representation:

$$a = p_{i_0}^{n_0} \cdots p_{i_k}^{n_k}$$

Where each p is a distinct prime and all n_i are positive. So if we have sequence $(j_0, \ldots, j_t)$ we can coded it with a unique code of powers of prime numbers as follows:

$$c = 2^{j_0+1} \cdots p^{j_t+1}$$

If we for example have a sequences $(3, 4, 1)$ then then this sequences is coded as:

$$2^{3+1} \cdot 3^{4+1} \cdot 5^{1+1} = 16 \cdot 243 \cdot 25 = 97200$$

This means that if we make our encoding in the right way, that each formula gets an unique code. So give a number we can find the formula that has that number as a code. The first goal of this section is to give each finite sequence of syntactical symbols of **PRA** a unique numerical code We can now list some of the codes for our symbols.

1. $\ulcorner \overline{0} \urcorner$ is (0).
2. $\ulcorner = \urcorner$ is (1).
3. $\ulcorner \neg \urcorner$ is (2); $\ulcorner \wedge \urcorner$ is (3); $\ulcorner \vee \urcorner$ is (4) and $\ulcorner \to \urcorner$ is (5).
4. $\ulcorner \forall \urcorner$ is (6) and $\ulcorner \exists \urcorner$ is (7).
5. $\ulcorner v_i \urcorner$ is $(8, i)$.

These are the easy one to gives. The hard ones are the codes for the functions symbols. We will define the codes for these inductively.

Function	Index
$Z(x) = 0$	$(9, 1, 1)$
$S(x) = x + 1$	$(9, 2, 1)$
$P_i^n(x_1, \ldots, x_n) = x_i$	$(9, 3, n, i)$
$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n))$	$(9, 4, n, m, (g^*, h_1^*, \ldots, h_m^*)$
$f(0, x_1, \ldots, x_n) = g(x_1, \ldots, x_n)$ $f(x+1, , x_1, \ldots, x_n) = h(f(x, x_1, \ldots, x_n), x, x_1, \ldots, x_n)$	$(9, 5, n+1, g^*, h^*)$

Here g^* and the h_i^*'s in the last two entries of the table, are the codes assigned to g and h_i.

Having defined this encoding, we must now check that it is primitive recursive. It is easy to check that the encoding for our non-function symbols are primitive recursive. For example the codes for variables are, since we have that:

$$x \in \ Var \ \leftrightarrow \ lh(x) = 2 \wedge (x)_0 = 8$$

Where $(x)_0$ is the first index of the tuple of x. That the rest of the encoding of the non-function symbols are primitive recursive is seen in a similarly way.

Showing that the encoding for the function symbols is primitive recursive is a bit more tedious. We will show that a relation stating that x is the code of a function symbols is primitive recursive, by showing

that its representing function is primitive recursive. We will start of by defined the code for our initial functions:

$$Init(x) \leftrightarrow x = (9,1,1) \vee x = (9,2,1) \vee \exists n \leq x \exists i \leq n(1 \leq i \wedge x = (9,3,n,i))$$

$Init(x)$ is clearly seen to state that a x is the code of one of our initial functions.
We will now define the codes of the functions construed by *rule 4*. Here we have :

$$\begin{aligned} F4(x) \leftrightarrow & lh(x) = 5 \wedge (x)_0 = 9 \wedge (x)_1 = 4 \wedge (x)_2 > 0 \wedge (x)_3 > 0 \\ & lh((x)_4) = (x)_3 + 1 \wedge (((x)_4)_0)_2 = (x)_3 \wedge \\ & \forall i < (x)_3((((x)_4)_{i+1})_2 = (x)_2) \end{aligned}$$

$F4$ says that x has the form $(9,4,n,m,(a,b_1,\ldots,b_m))$ and if $a,b_1,\ldots,b_m$ are functions codes then a codes a m-ary function and the b_i's codes n-ary functions.

Lastly we will also make assert that the functions generated by *rule 5* have the right form:

$$\begin{aligned} F5(x) \leftrightarrow & lh(x) = 5 \wedge (x)_0 = 9 \wedge (x)_1 = 5 \wedge (x)_2 > 1 \wedge \\ & ((x)_3)_2 = (x)_2 \dot{-} 1 \wedge ((x)_4)_2 = (x)_2 + 1 \end{aligned}$$

We have now done the hard work, and we can see that the following function defined by course-of-value recursion is primitive recursive:

$$f(x) = \begin{cases} 0, & Init(x) \\ 0, & F4(x) \wedge \forall i \leq (x)_3(f(((x)_4)_i) = 0) \\ 0, & F5(x) \wedge f((x)_3) = 0 \wedge f((x)_4) = 0 \\ 1, & \text{else} \end{cases}$$

Thus we have that $x \in FuncSym$ iff $f(x) = 0$ is a primitive recursive relation. The next step will be to generate the codes for the terms and formulas of our language. This will be done inductively. We will look at a term, f.ex $\overline{f}v_0v_1\overline{g}v_2$. We will view terms like this a "tree" so the code for this would be:

$$(\ulcorner \overline{f} \urcorner, \ulcorner v_0 \urcorner, \ulcorner v_1 \urcorner, (\ulcorner \overline{g} \urcorner, \ulcorner v_2 \urcorner))$$

So we will define the codes of terms in the following way:

1. $\ulcorner \overline{0} \urcorner, \ulcorner v_i \urcorner$ are codes for $\overline{0}$ and v_i.
2. if $\overline{f}$ is an n-ary function symbol and $t_1,\ldots,t_n$ are terms with codes $\ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner$ then the following holds: $\ulcorner \overline{f} t_1 \ldots t_n \urcorner = (\ulcorner \overline{f} \urcorner, \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner)$

We can now define the representing function for the set of codes of terms as follows:

$$f(x) = \begin{cases} 0, & x = \ulcorner \overline{0} \urcorner \\ 0, & x \in Var \\ 0, & (x)_0 \in FuncSym \wedge (lh(x) = ((x)_0)_1 + 1) \wedge \forall i < lh(x)(f(x)_{i+1} = 0) \\ 1, & otherwise \end{cases}$$

Where the third clauses makes sure that x is of the correct form and that $\overline{x}$ is n-ary. Thus $x \in Term$ iff $f(x) =$ is a primitive recursive relation.

The formulas are also given codes in a similar inductively way:

1. If t_1 and t_2 are terms then $\ulcorner = t_1t_2 \urcorner$ is $(\ulcorner = \urcorner, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$
2. Let φ and ψ be formulas then we have the following codes:

(a) $\ulcorner\neg\varphi\urcorner = (\ulcorner\neg\urcorner, \ulcorner\varphi\urcorner)$
(b) $\ulcorner\varphi \wedge \psi\urcorner = (\ulcorner\wedge\urcorner, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner)$
(c) $\ulcorner\varphi \vee \psi\urcorner = (\ulcorner\vee\urcorner, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner)$
(d) $\ulcorner\varphi \to \psi\urcorner = (\ulcorner\to\urcorner, \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner)$

3. Let φ be a formula and v a variable then:

(a) $\ulcorner\forall v\varphi\urcorner = (\ulcorner\forall\urcorner, \ulcorner v\urcorner, \ulcorner\varphi\urcorner)$
(b) $\ulcorner\exists v\varphi\urcorner = (\ulcorner\exists\urcorner, \ulcorner v\urcorner, \ulcorner\varphi\urcorner)$

Again we will also define the representing function almost directly to live up to our goals by course-of-values recursion:

$$f(x) = \begin{cases} 0, & \exists y \leq x \exists z \leq x(y \in Term \wedge z \in Term \wedge x = (\ulcorner=\urcorner, y, z)) \\ 0, & \exists y < x(f(y) = 0 \wedge x = (\ulcorner\neg\urcorner, y)) \\ 0, & \exists y < x \exists z < x(f(y) = 0 \wedge f(z) = 0 \wedge \\ & (x = (\ulcorner\wedge\urcorner, y, z) \vee x = (\ulcorner\vee\urcorner, y, z) \vee x = (\ulcorner\to\urcorner, y, z))) \\ 0, & \exists y < x \exists z < x(f(z) = 0 \wedge y \in Var \wedge (x = (\ulcorner\forall\urcorner, y, z) \vee x = (\ulcorner\exists\urcorner, y, z))) \\ 1, & otherwise \end{cases}$$

This function looks complex at first look, but it just make sure that x is a formula, and it is clearly primitive recursive. So this shows that the relation $x \in Fmla$ iff $f(x) =$ is a primitive recursive relation. So the formula $v_0 \to v_1$ can be translated to the following sequence::

$$(\ulcorner\to\urcorner, \ulcorner v_0\urcorner, \ulcorner v_1\urcorner) = (5, (8, 0), (8, 1))$$

Whcih then have the following code:

$$2^6 \cdot 3^{2^9 \cdot 3^1} \cdot 5^{2^9 \cdot 3^2}$$

This number is extremely big even for such a simple formula.

Since the task of showing these things are very tedious (and not very advance), I will state the following list of relation that are primitive recursive without proving the most of them:

Proposition 10. The following list of relations are all primitive recursive

1.	$x \in Var$	x is the code of a variable
2.	$x \in FncSym$	x is the code of a function symbol
3.	$x \in Term$	x is the code of a term
4.	$x \in Fmla$	x is the coe of a formula
5.	$x \in Var_T(y)$	x is the code of a variable occurring in the term with code y
6.	$x \in Var_F(y)$	x is the code of a variable occurring in the formula with code y
7.	$x \in FV(t)$	x is the code of a free variable of the formula with code y
8.	$x \in BV(y)$	x is the code of a free bound variable of the formula with code y
9.	$x \in Sent$	x is the code of a sentence
10.	x sub for y in z	x is the code of a term substitution for the variable with code y in the formula with code z

Proof. We will only prove 9. 1-4 have already been proven and 5-8 and 9 are proven in a similar way to 1-4. The relation $x \in Sent$ can be written as:

$$x \in Sent \leftrightarrow x \in Fmla \wedge \forall y < x(y \notin FV(x))$$

And this is clearly primitive recursive. ⊣

Our next goal will be to primitive recursively define the logical axioms. For this goal the last relation of the list above will be crucial, since we need it to define two substitution function function (one for substitution inside terms and one for substitution inside formulas) that will give us the code $\ulcorner \varphi t \urcorner \ulcorner \varphi v \urcorner, \ulcorner v \urcorner$ and $\ulcorner t \urcorner$. We will start of be defining the one for substitution inside terms. And we start of by defining:

$$subst^T(x; y, z) = (\ulcorner t_1(v) \urcorner; \ulcorner v \urcorner, \ulcorner t_2 \urcorner) = t_1(t_2)$$

That is about substituting t_2 for v in $t_1(v)$. We can now define the more general relation:

$$subst^T(x; y, z) = \begin{cases} x, & x \in Var \wedge y \in Var \wedge z \in Term \wedge x \neq y \\ z, & x \in Var \wedge y \in Var \wedge z \in Term \wedge x = y \\ (x)_0 * \bullet\limits_{i<lh(x)} (subst^T((x)_i; y, z)), & x \in Term \wedge y \in Var \wedge z \in Term \wedge x \notin Var \\ 0, & \text{else} \end{cases}$$

Where $\bullet\limits_{i<y} f(i)$ is defined as follows:

$$\bullet\limits_{i<0} f(i) = (\ )$$
$$\bullet\limits_{i<y+1} f(i) = (\bullet\limits_{i<y} f(i)) * f(y)$$

Having defined this function, we can now define the one for formulas:

$$subst(\ulcorner \varphi v \urcorner; \ulcorner v \urcorner, \ulcorner t \urcorner) = \ulcorner \varphi t \urcorner$$

This function will be defined as by the following big piecewise function:

$$subst(x;y,z) = \begin{cases} ((x)_0, subst^T((x)_1;y,z), subst^T((x)_2;y,z)), & \text{if } x \in FmLa \wedge \exists w_i, w_2 \leq x(x = (\ulcorner = \urcorner, w_1, w_2)) \wedge y \in Var \wedge z \in Term \\ (\ulcorner \neg \urcorner, subst((x)_1;y,z)), & \text{if } x \in FmLa \wedge y \in var \wedge z \in Term \wedge (x)_0 = \ulcorner \neg \urcorner \\ (\ulcorner \rightarrow \urcorner, subst((x)_0,y,z), subst((x)_2,y,z)), & \text{if } x \in FmLa \wedge y \in Var \wedge z \in Term \wedge (x)_1 = \ulcorner \rightarrow \urcorner \\ (\ulcorner \forall yz \urcorner & \text{if } x \in FmLa \wedge y \in var \wedge ((x)_1)_1 = (y)_1 \wedge z \in Term \wedge (x)_0 = \ulcorner \forall \urcorner \\ (\ulcorner \forall \urcorner, \ulcorner (x)_1 \urcorner, subst(\ulcorner (x)_2 \urcorner; \ulcorner y \urcorner, \ulcorner z \urcorner)), & \text{if } x \in FmLa \wedge y \in var \wedge z \in Term \wedge (((x)_1)_1 \neq (y)_1 \wedge (x)_0 = \ulcorner \forall \urcorner \\ \vdots & \\ 0, & \text{else} \end{cases}$$

This function is not fully defined here. But it is somewhat clear how to fill out the rest of it; the task to do so is just very tedious. With this function we can now primitive recursively define our logical axioms. For this end we will make use of the following abbreviations:

1. $\dot{\neg} x = (\ulcorner \neg \urcorner, x)$
2. $x \dot{\wedge} y = (\ulcorner \wedge \urcorner, x, y)$
3. $x \dot{\vee} y = (\ulcorner \vee \urcorner, x, y$
4. $x \dot{\rightarrow} y = (\ulcorner \rightarrow \urcorner, x, y)$

We can now define "x codes a propositional axiom" by the following relation:

$$\begin{aligned} PropAx(x) \leftrightarrow \exists yzw \leq x(x = y \dot{\rightarrow} (z \dot{\rightarrow} y) \vee \\ x = (y \dot{\rightarrow} (z \dot{\rightarrow} w)) \dot{\rightarrow} ((y \dot{\rightarrow} w) \dot{\rightarrow} (z \dot{\rightarrow} w)) \\ \vee \ldots \vee x = (\dot{\neg} \dot{\neg} y \dot{\rightarrow} y) \end{aligned}$$

Where ... is an aberration for the 7 missing axioms, and it is clear how these are written up. We can also define "x codes a quantifier axiom" in a similar way:

$$\begin{aligned} QuantAx(x) \leftrightarrow \exists yzw(y \in var \wedge z \in FmLa \wedge w \in Term \wedge w \text{ sub for } y \text{ in } z \\ \wedge (x = (\ulcorner \forall \urcorner, y, z) \dot{\rightarrow} subst(z;y,w) \\ \vee x = subst(z;y,w) \dot{\rightarrow} (\ulcorner \exists \urcorner, y, z))) \end{aligned}$$

The equality axioms consist of the three axioms that make sure that equality is a transitive relation. They have specific codes that we will call e_1, e_2 and e_3. For the last axiom we will define the following relation that says that $x = \ulcorner f v_1 \ldots v_n \urcorner$ for a n-ary function symbol:

$$\begin{aligned} SimpTerm(x) \leftrightarrow \exists fn \leq x(f \in FnSym \wedge (f)_1 = n \wedge lh(x) = n+1 \wedge (x)_0 = f \wedge \\ \forall i < n((x)_{i+1} = (8, i+1))) \end{aligned}$$

We also define the following abbreviation:

$$x \dot{=} y = (\ulcorner = \urcorner, x, y)$$

We can now define the set of equality axioms:

$$\begin{aligned} EqAx(x) \leftrightarrow & x = e_1 \vee x = e_2 \vee x = e_3 \vee \exists iyz \leq x(x = y \dot{\rightarrow} z \wedge \\ & y = (8,i) \dot{=} (8,0) \wedge \exists w \leq z(SimpTerm(w) \wedge \\ & z = w \dot{=} substs^T(w;(8,1),(8,0)))) \end{aligned}$$

I will not show that the set of non-logical axioms are primitive recursive. But we will call the relation that states that they are for $NonLogAx(x)$ and thus we the following relation states that the set of axioms can be primitive recursively defined

$$Ax(x) \leftrightarrow PropAx(x) \vee QuantAx(x) \vee EqAx(x) \vee NonLogAx(x)$$

The last thing we need to do, before we can define provability, is to be able to express our three rules of inference. We will start of by defining "x is follows form y and z by modus ponens" by the following formual:

$$MP(x;y,z) \leftrightarrow z = y \rightarrow x$$

and now our two generalisation rules. We will start with the first one:

$$\begin{aligned} Gen_1(x;y) \leftrightarrow & \exists vwz \leq x(v \in var \wedge w \in Fmla \wedge z \in Fmla \wedge \notin FV(z) \\ & \wedge y = w \rightarrow z \wedge x = (\ulcorner \exists \urcorner, v, w) \rightarrow z) \end{aligned}$$

and now the second one:

$$\begin{aligned} Gen_2(x;y) \leftrightarrow & \exists vwz \leq x(v \in var \wedge w \in Fmla \wedge z \in Fmla \wedge \notin FV(z) \\ & \wedge y = w \rightarrow z \wedge x = w \rightarrow (\ulcorner \forall \urcorner, v, z) \end{aligned}$$

We will combine these and get $Gen(X;y) \leftrightarrow Gen_1(x;y) \vee Gen_2(x;y)$. A formal derivation is a sequence of formulas $\varphi_0, \ldots, \varphi_n$ each of which is either an axiom or follows from an earlier entry in the sequence by one of our inference rules. To such a sequence we assign the code $(\ulcorner \varphi_0 \urcorner, \ldots, \ulcorner \varphi_n \urcorner$. We can now define the primitive recursive relation "y codes a derivation of the formula with code x in the following way:

$$\begin{aligned} Prov(x,y) \leftrightarrow & (y)_{lh(y)-1} = x \wedge \forall i < lh(y)(Ax((y)_i) \vee \\ & \exists jk < i \; MP((y)_i;(y)_j,(y)_k) \vee \exists j < i \; Gen((y)_i;(y)_j)) \end{aligned}$$

With this we can define the non primitive recursive relation "x codes a provable formula" with:

$$Pr(x) \leftrightarrow \exists y(Prov(y,x))$$

We will end this section by showing that the primitive recursive relations behave as they should in **PRA**. By Theorem 1 we have for a relation $R \subset \omega^n$ that for all $k_1, \ldots, k_n \in \omega$ that:

$$\begin{aligned} R(k_1,\ldots,k_n) \text{ holds} & \Rightarrow \chi_R(k_1,\ldots,k_n) = 0 \\ & \Rightarrow \mathbf{PRA} \vdash \overline{\chi}_R(\overline{k}_1,\ldots,\overline{k}_n) = \overline{0} \\ & \Rightarrow \mathbf{PRA} \vdash \overline{R}(\overline{k}_1,\ldots,\overline{k}_n) \end{aligned}$$

Where we have assumed that that $\overline{\chi}(v_1,\ldots,v_n) = 0$ represent $\overline{R}(v_1,\ldots,v_n)$. Further we also have that:

$$\begin{aligned} R(k_1,\ldots,k_n) \text{ fails} & \Rightarrow \chi_R(k_1,\ldots,k_n) = 1 \\ & \Rightarrow \mathbf{PRA} \vdash \overline{\chi}_R(\overline{k}_1,\ldots,\overline{k}_n) = \overline{1} \\ & \Rightarrow \mathbf{PRA} \vdash \neg\overline{R}(\overline{k}_1,\ldots,\overline{k}_n) \end{aligned}$$

When will use a existentially quantifier on such a relation (like we do in $Pr(x)$) we only have:

$$\begin{aligned}\exists k(R(k,k_1,\dots k_n) \text{ holds} &\Rightarrow \exists k(\chi_R(k,k_1,\dots k_n)=0)\\ &\Rightarrow \exists k(\mathbf{PRA}\vdash \overline{\chi}_R(\overline{k},\overline{k}_1,\dots,\overline{k}_n)=\overline{0})\\ &\Rightarrow \mathbf{PRA}\vdash \exists v(\overline{\chi}_R(v,\overline{k}_1,\dots,\overline{k}_n)=\overline{0})\\ &\Rightarrow \mathbf{PRA}\vdash \exists v\overline{R}(v,\overline{k}_1,\dots,\overline{k}_n)\end{aligned}$$

We do not get anything about the relation $\neg\exists vR$.

5 The incompleteness theorems

In this section we will state and prove the two incompleteness theorems.

For this we will need a digaonalisation lemma and a three formalisations of some of the basic properties of our $Pr(\cdot)$ predicate called derivabilitiy conditions. We will start of with stating and proving that $Pr(\cdot)$ have fulfills these conditions in **PRA**. These conditions will be used in proving the second incompleteness theorem.

5.1 Derivability conditions

The conditions that we will state and prove here is called the Löb derivability conditions. They are the following three:

1. $\vdash \varphi \to\vdash Pr(\ulcorner\varphi\urcorner)$
2. $\vdash Pr(\ulcorner\varphi\urcorner)\wedge Pr(\ulcorner\varphi\to\psi\urcorner\to Pr(\ulcorner\psi\urcorner)$
3. $\vdash Pr(\ulcorner\varphi\urcorner)\to Pr(\ulcorner(Pr\ulcorner\varphi\urcorner)\urcorner)$

We will start of by proving the first of these.

Lemma 7. For any formula φ we have that:

$$\mathbf{PRA}\vdash\varphi\Rightarrow\mathbf{PRA}\vdash Pr(\ulcorner\varphi\urcorner)$$

Proof. If $\mathbf{PRA}\vdash\varphi$ holds, then there is a proof $\varphi_1,\dots,\varphi_n=\varphi$ of φ, hence we have that the following is true:

$$Prov((\ulcorner\varphi_1\urcorner,\dots,\ulcorner\varphi_n\urcorner),\ulcorner\varphi\urcorner)$$

Hence we have that:

$$\mathbf{PRA}\vdash\overline{Prov}((\ulcorner\varphi_1\urcorner,\dots,\ulcorner\varphi_n\urcorner),\ulcorner\varphi\urcorner)$$

But this means that we have:

$$\mathbf{PRA}\vdash\exists v\overline{Prov}(v,\ulcorner\varphi\urcorner)$$

Hence:

$$\mathbf{PRA}\vdash\overline{Pr}(\ulcorner\varphi\urcorner)$$

⊣

The second condition is also not to hard to show:

Lemma 8. For any formulas φ and ψ we have:

$$\mathbf{PRA}\vdash Pr(\ulcorner\varphi\urcorner)\wedge Pr(\ulcorner\varphi\to\psi\urcorner)\to Pr(\ulcorner\psi\urcorner)$$

Proof. This follows since the following can be verified for any formulas φ, ψ:

$$Prov(v, \ulcorner\varphi\urcorner) \wedge Prov(v_1, \ulcorner\varphi \to \psi\urcorner) \to Prov(v_0 * v_1 * (\ulcorner psi\urcorner), \ulcorner\varphi\urcorner)$$

Since we have if we assume that we have a proof of φ and one of $\varphi \to \psi$. Then if we concatenate these two proofs with ψ it is clear that this inference proves ψ. ⊣

The last condition will take a bit more work to prove. We will start of by introducing a new primitive recursive function. The function will send a numeral $\overline{n}$ to $\ulcorner\overline{n}\urcorner$ and is defined by recursion:

$$num(0) = \ulcorner\overline{0}\urcorner = (0) = 2^0 = 1$$
$$num(x+1) = (\ulcorner\overline{S}\urcorner, num(x))$$

We will introduce one last notation before showing the lemma where the third conditions follows from. We will introduce the notation $\ulcorner\varphi\dot{v}_0 \dots \dot{v}_{n-1}\urcorner$. This will be defined recursively on n as follows:

$$\ulcorner\varphi\dot{v}_0\urcorner = subst(\ulcorner\varphi v_0\urcorner; (8,0), num(v_0))$$
$$\ulcorner\varphi\dot{v}_0 \dots \dot{v}_k\urcorner = subst(\ulcorner\varphi\dot{v}_0 \dots \dot{v}_{k-1}\dot{v}_k\urcorner; (8,k), num(v_k)))$$

We can now state the following lemma without proof:

Lemma 9. Let $\overline{f}$ be an n-ary primitive recursive function symbol. Then there is a function g, depending on f such that:

$$\mathbf{PRA} \vdash \overline{f}v_0, \dots v_{n-1} = v_n \to Prov(\overline{g}v_0 \dots v_{n-1}, \ulcorner\overline{f}\dot{v}_0 \dots \dot{v}_{n-1} = \dot{v}_n\urcorner)$$

This gives us the following colloray which show that the third condition holds:

Corollary 1. Let φ be any formula; then:

$$\mathbf{PRA} \vdash Pr(\ulcorner\varphi\urcorner) \to Pr(\ulcorner Pr(\ulcorner\varphi\urcorner)\urcorner)$$

Having shown that **PRA** and $Pr(\cdot)$ forfills these conditions the only thing left before we will state and prove the incompleteness theorems will be our diagonalisation lemma.

5.2 Diagonalisation lemma

In this subsection we will show the diagonalisation lemma. For this end will will define the following substitution function with the help of $subst(x; y, z)$ and $num(x)$:

$$sub(y, x) = subst(y; (8,0), x)$$

Thus if φv_0 has v_0 as a free variable we have that:

$$sub(\ulcorner\varphi v_0\urcorner, v_1) = \ulcorner\varphi\dot{v}_1\urcorner$$

and

$$sub(\ulcorner\varphi v_0\urcorner, n) = \ulcorner\varphi\overline{n}\urcorner$$

We can now prove the following lemma that will is crucial for both the incompleteness theorems.

Lemma 10. Given any formula ψ where the only free variable is v, we can find a sentence φ such that:

$$\mathbf{PRA} \vdash \varphi \leftrightarrow \psi(\ulcorner\varphi\urcorner)$$

Proof. Let ψv be given and let θv_0 be $\psi(sub(v_0, v_0))$, $m = \ulcorner \theta v_0 \urcorner$ and $\varphi = \theta\overline{m}$. Then it is clear that: $\varphi = \theta\overline{m} = \psi(sub(\overline{m}, \overline{m}))$ so we have that:

$$\mathbf{PRA} \vdash \overline{sub}(\ulcorner \theta v_0 \urcorner, \ulcorner \theta v_0 \urcorner) = \ulcorner \varphi \urcorner$$

But this is the same as having:

$$\mathbf{PRA} \vdash \psi(\overline{sub}(\ulcorner \theta v_0 \urcorner, \ulcorner \theta v_0 \urcorner) \leftrightarrow \psi(\ulcorner \varphi \urcorner)$$

Thus:

$$\mathbf{PRA} \vdash \theta(\ulcorner \theta v_0 \urcorner) \leftrightarrow \psi(\ulcorner \varphi \urcorner)$$

And since $\ulcorner \theta v_0 \urcorner$ is m we get

$$\mathbf{PRA} \vdash \theta\overline{m} \leftrightarrow \psi(\ulcorner \varphi \urcorner)$$

and since φ is $\theta\overline{m}$ we get that:

$$\mathbf{PRA} \vdash \varphi \leftrightarrow \psi(\ulcorner \varphi \urcorner)$$

⊣

Having done all the groundwork, we can now finally begin to state and prove the two main theorems of this project.

5.3 The first incompleteness theorem

We can now prove our next main theorem of this project: The first incompleteness theorem:

Theorem 2. The predicate $Pr()$ have the following properties for all formulas ψ:

1. $\vdash \psi \Rightarrow \vdash Pr(\ulcorner \psi \urcorner)$
2. $\vdash Pr(\ulcorner \psi \urcorner) \Rightarrow \vdash \psi$

And further we have that if we let $\vdash \varphi \leftrightarrow \neg Pr(\ulcorner \varphi \urcorner)$ then we have:

1. $\nvdash \varphi$
2. $\nvdash \neg\varphi$

i.e a sentence where neither it or its negation can be proved.

It is important to note, that we have two different kinds of completeness here.

Proof. Completeness and soundness is clear. So we will pick φ such that $\mathbf{PRA} \vdash \varphi \leftrightarrow \neg Pr(\ulcorner \varphi \urcorner)$ by the diagonalisation lemma. For the first we will suppose the $\mathbf{PRA} \vdash \varphi$. By completeness we then get $\mathbf{PRA} \vdash Pr(\ulcorner \varphi \urcorner$ but we also have by the definition of φ that $\vdash \neg\varphi$ and thus get a contradiction so we get that $\mathbf{PRA} \nvdash \varphi$.
Now for the second. Assume that $\mathbf{PRA} \vdash \neg\varphi$ and hence by the choice of φ we get that $\mathbf{PRA} \vdash Pr(\ulcorner \varphi \urcorner)$. By soundness we then get $\mathbf{PRA} \vdash \varphi$ and thus another contradiction and therefor $\mathbf{PRA} \nvdash \neg\varphi$. ⊣

5.4 The second incompleteness theorem

Having done all the hard work previously, that was showing that the $Pr(\cdot)$ fulfills the derivability conditions in **PRA**, we can "quickly" show the second incompleteness theorem:

Theorem 3. Let Con be the statement $\neg Pr(\bot)$, where $\bot$ is false, ie. $\bot$ is $\overline{0} = \overline{S}0$ (If we cannot prove a false statement in our theory, then the theory is consistence.). Then we have $\mathbf{PRA}\neg \vdash Con$.

So the first incompleteness theorem tells us that there are true sentences that we can not prove in **PRA**. The second then "shows" that the consistency of **PRA**is one of these sentences.

Proof. We will choose φ such that $\mathbf{PRA} \vdash \varphi \leftrightarrow \neg Pr(\ulcorner \varphi \urcorner)$ and further we will let Con be $\neg Pr(\bot)$. The goal in this prove is to show that $\vdash Con \leftrightarrow \varphi$, because by the first incompleteness theorem φ is underivable, and thus Con will be the same.

We start of by seeing that:

$$\begin{aligned} \vdash \varphi \leftrightarrow \neg Pr(\ulcorner \varphi \urcorner) &\Rightarrow \vdash \varphi \leftrightarrow Pr(\ulcorner \varphi \urcorner) \\ &\Rightarrow \vdash Pr(\ulcorner \neg \varphi \urcorner \leftrightarrow Pr(\ulcorner Pr \ulcorner \varphi \urcorner \urcorner) \end{aligned}$$

By a few uses of the first and the second derivability condition and by a similarly argument we get:

$$\vdash \varphi \wedge \neg\varphi \Rightarrow \vdash Pr(\ulcorner \varphi \urcorner) \wedge Pr(\ulcorner \neg \varphi \urcorner) \rightarrow Pr(\ulcorner \bot \urcorner)$$

We also have by the third derivability condition that:

$$\vdash Pr(\ulcorner \varphi \urcorner) \rightarrow Pr(\ulcorner Pr(\ulcorner \varphi \urcorner) \urcorner)$$

All in all this gives us:

$$\vdash Pr(\ulcorner \varphi \urcorner) \rightarrow Pr(\ulcorner \varphi \urcorner) \wedge Pr(\ulcorner \neg \varphi \urcorner)$$

And this gives us that we have:

$$\vdash Pr(\ulcorner \varphi \urcorner) \rightarrow Pr(\ulcorner \bot \urcorner)$$

and thus by contraposition:

$$\vdash \neg Pr(\ulcorner \bot \urcorner) \rightarrow \neg Pr(\ulcorner \varphi \urcorner)$$

and by definition this is exactly:

$$\vdash Con \rightarrow \varphi$$

And thus **PRA** cannot prove its own consistency ⊣

Thus we have proven the two incompleteness theorems for **PRA**. The first incompleteness undermines Hilbert's Programme, since in **PRA**we have that we have sentence φ in **PRA** that is meaningful and unprovable. Further since φ "says" that it cannot be proven and is unprovable thus it is true. This means that it is a theorem of our transfinite system T from the introduction and thus T is not conservative over **PRA**. The second incompleteness theorem gives us such a meaningful statement; the system cannot prove its own consistency.

We will end this section with the statement and proof of Löbs Theorem:

Theorem 4. Let φ be any sentence then:

$$\vdash Pr(\ulcorner \varphi \urcorner \rightarrow \varphi \Leftrightarrow \vdash \varphi$$

Proof. The right to left implication is trivial. We will prove the other way by contraposition. Let $\nvdash \varphi$. Then we can add $\neg\varphi$ as an axiom to **PRA**; call this extension S and let $Con_S = \neg Pr(\ulcorner \neg\varphi \to \bot \urcorner)$, i.e it denotes the consistency of this theory. Since this extension fulfills the Betingelser for the two incompleteness theorems we get by the second incompleteness theorem that

$$S \nvdash \neg\varphi \to Con_S$$

.
But it is clear that Con_S is equivalent to $\neg Pr(\ulcorner \varphi \urcorner)$ and we can thus contrapose the above formula and get: $\nvdash Pr(\ulcorner \varphi \urcorner) \to \varphi$ ⊣

From this results we can get the following corollary. The proof of this corollary will be omitted here.

Corollary 2. Let φ be any sentence then:

$$\vdash Pr(\ulcorner Pr(\ulcorner \varphi \urcorner) \urcorner) \to Pr(\ulcorner \varphi \urcorner)$$

6 Provability logic

The result that $\vdash Pr(\ulcorner Pr(\ulcorner \varphi \urcorner) \to \varphi \urcorner) \to Pr(\ulcorner \varphi \urcorner)$, is very similarly in structure to the axiom schema of provability $\Box(\Box\varphi \to \varphi) \to \Box\varphi$, where $\Box\varphi$ is read as it is provably that φ. This hints at a deep connection between arithmetics and provability logic called GL. Robert Solovay showed in his article *Provability interpretations of modal logic* from 1976 that there is such a deep connection between these two seemingly different parts of mathematics. A realization of GL in **PRA** is function that assign for each formula φ of GL a sentence φ^* of **PRA**which satisfies the following requirements:

1. $(\bot)^* = \text{"}0 = S(0)\text{"}$
2. $(\varphi \to \psi)^* = \text{"}\varphi^* \to \psi^*\text{"}$
3. $(\Box\varphi)^* = \text{"}Pr(\ulcorner \varphi^* \urcorner)\text{"}$

Solovay then shows that for the following holds:

$$GL \vdash \varphi \Leftrightarrow \text{ for all realizations } {}^*, \ \mathbf{PRA} \vdash \varphi^*$$

Solovay originally proved it for Peanos axioms for arithmetics, but the proof also holds for **PRA**. The proof of this statement is way beyond the scope of this project.

References

P. Cohen. *Set Theory and the Continuum Hypothesis*. W. A. Benjamin, Inc., 1966.

H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2001.

E. Mendelson. *Introduction to Mathematical Logic*. Van Norstrand, 1964.

P. Raatikainen. Gödel's Incompleteness Theorems. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2022 edition, 2022.

C. Smorynski. The incompleteness theorems. In J. Barwise, editor, *The Handbook of Mathematical logic*. North-Holland Publishing Company, 1977.

C. Smorynski. *Self-Rference and modal logic*. Springer-verlag, 1985.

R. M. Solovay. Provability interpretations of modal logic. *Israel journal of mathematics*, 25(3-4), 1976.