Undecidability in Number Theory Hilbert's Tenth Problem and Extensions of it

Nikolaos Kallergis

Supervisor :
Rizos Sklinos

Master's Thesis

University of Crete
Department of Mathematics and Applied Mathematics

October 2023

The committee of the Thesis consists of:

- **Rizos Sklinos**, my supervisor, Assistant Professor in the Department of Mathematics and Applied Mathematics, University of Crete.

- **Alexandros Kouvidakis**, Professor in the Department of Mathematics and Applied Mathematics, University of Crete.

- **Theodoulos Garefalakis**, Professor in the Department of Mathematics and Applied Mathematics, University of Crete.

Abstract

We are going to examine the negative answer to Hilbert's tenth problem, i.e. the problem of finding an algorithm which, given an arbitrary Diophantine equation with integer coefficients, is able to decide if the equation has integer solutions. The non-existence of such an algorithm will be obtained by combining the original proof of Yuri Matijasevic, Hilary Putnam, Julia Robinson and Martin Davis in 1970, along with Alan Turing's invention, the Turing machine. First we will define formally what a Turing machine is and how it operates. Then we will prove that the Halting problem is undecidable with the aid of the universal Turing machine and then the negative answer to Hilbert's Diophantine problem will be obtained with the aid of the Halting problem. In Chapter 3, two extensions of Hilbert's problem will be examined, one in which the solutions are sought in the polynomial ring $R[T]$, where R is an integral domain of characteristic zero that contains $\mathbb{Z}$, with the coefficients being elements of $\mathbb{Z}[T]$. The other extension will be for the ring of power series $F[[t]]$ where F is an integral domain of characteristic greater than zero with a parameter t.

Contents

Chapter 1

Introduction

In 1900 the German mathematician David Hilbert presented a list of 23 unsolvable, at the time, problems. By the way Hilbert posed the problems, one can see that he expected each of them to have a definite, yes or no, answer. Mathematicians, to this day, are still concerned with some of these problems regarding to whether they have a solution or not. A conspicuous example is the Riemann Hypothesis, the 8th problem of the list. For some problems a definite answer has been given while others are too vague to have such an answer, like the 23rd problem concerning the further development of the calculus of variations. It should be noted that, the fact that all the problems are formulated as if they are going to have a specific answer, has its roots in an entire philosophy of mathematics that Hilbert expressed, namely formalism. Hilbert belonged to a group of mathematicians whose goal was to prove that mathematics, as a theory, is consistent, i.e. one can not derive both a statement and its negation, complete, i.e. any statement can be proved or disproved and decidable, i.e. there exists an algorithm that can decide whether any statement is true or false in a theory. In order to achieve this they tried to construct a universal formal system which would contain the proof to any theorem. It was about to be proved that this is impossible.

First, it was in 1931 with Kurt Gödel's first incompleteness theorem, [9]. It stated that, given any consistent set of axioms that is decidable, i.e. there is an algorithm to decide if a statement is

an axiom, in which elementary arithmetic is used with addition and multiplication, there will always exist true statements that can not be derived from the axioms. If such a system is not consistent, then there is a proof for a statement and its negation. Hence any statement can be derived from the system and therefore it is complete. Another contribution was that of Alan Turing in 1936 when he proved, with the aid of his famous "Turing machines", that the halting problem is undecidable, meaning that there will never exist an algorithm to decide it. The Turing machine and the halting problem will be examined more explicitly in **2.1** and **2.4** respectively.

1.1 Hilbert's Tenth Problem

In this thesis we will deal with Hilbert's tenth problem. The problem seeks an algorithm to decide whether an arbitrary Diophantine equation, i.e. a polynomial equation with finite variables and integer coefficients, has integer solutions or not. We will also examine analogues of Hilbert's tenth problem over rings other than the integers.

In order to understand what Hilbert's tenth problem talks about, we need to give some definitions first (see [7], [6]).

Definition 1.1.1: A language $\mathcal{L}$ is a union of two disjoint sets.

- The set $\mathcal{R}$ of relation symbols , such as $<$, $=$.
- The set $\mathcal{F}$ of function symbols , such as $+$, $\cdot$, 0, 1.

Each of these symbols is associated with an arity m, a function from $\mathcal{R} \cup \mathcal{F}$ to $\mathbb{N}$, where $m(r) = n$ is the number of the arguments each symbol takes. In general a symbol $r \in \mathcal{R} \cup \mathcal{F}$ with $r(a_1, a_2, \ldots, a_n)$ has arity n.

For the needs of the thesis the arity won't be greater than 2.

Example 1.1.1: $m(<) = 2$, $m(+) = 2$, the constants 0 and 1 are function symbols of arity 0. No relation symbol has arity 0.

Example 1.1.2: $L_1 = \{\cdot, +, 0, 1\}$ is the language of rings, such as $\mathbb{Z}$.

Example 1.1.3: $L_2 = \{-, +, 0\}$ is the language of additive Abelian groups.

It should be mentioned that the symbols of the languages do not suffice to explain the way they behave in a particular set. For any set, the interpretation of those symbols must be defined. With this, each symbol is assigned with a meaning and a structure serves exactly that purpose.

Definition 1.1.2: Let L be a language and A a non-empty set. A structure $\mathcal{A}$ for L is the triplet $(A, R^{\mathcal{A}}, F^{\mathcal{A}})$ where $R^{\mathcal{A}}, F^{\mathcal{A}}$ are the interpretations of R, F in A . In particular, $R^{\mathcal{A}}$ is a subset of A^m, where m is the arity of R, and $F^{\mathcal{A}}$ is the operation $F^{\mathcal{A}} : A^n \to A$ where n is the arity of F.

Example 1.1.4: Consider the language $L = \{+, \cdot, -, 0, 1\}$, then $\mathcal{Z} = (\mathbb{Z}, 0^{\mathcal{Z}}, 1^{\mathcal{Z}}, +^{\mathcal{Z}}, -^{\mathcal{Z}}, \cdot^{\mathcal{Z}})$ is a structure for L.
Here we have the function symbols $0^{\mathcal{Z}}, 1^{\mathcal{Z}}, +^{\mathcal{Z}}, -^{\mathcal{Z}}, \cdot^{\mathcal{Z}}$, where:

1. $+^{\mathcal{Z}} : \mathbb{Z}^2 \to \mathbb{Z}$, for $a, b \in \mathbb{Z}$ $+^{\mathcal{Z}}(a, b) = a + b$

2. $\cdot^{\mathcal{Z}} : \mathbb{Z}^2 \to \mathbb{Z}$, for $a, b \in \mathbb{Z}$ $\cdot^{\mathcal{Z}}(a, b) = a \cdot b$

3. $-^{\mathcal{Z}} : \mathbb{Z}^2 \to \mathbb{Z}$, for $a, b \in \mathbb{Z}$ $-^{\mathcal{Z}}(a, b) = a - b$

$0^{\mathcal{Z}}$ and $1^{\mathcal{Z}}$ represent the constant symbols of 0 and 1 in $\mathbb{Z}$ respectively.

Consider a ring R and a subring A of R.

Definition 1.1.3: Consider a polynomial P with n variables. A Diophantine equation is an equation of the form

$$P(x_1, \ldots, x_n) = 0$$

Definition 1.1.4: The Diophantine problem for R with coefficients in A seeks an algorithm to decide if a Diophantine equation with coefficients in A has a solution in R. If such an algorithm exists (does not exist), the problem is solvable (unsolvable).

Definition 1.1.5: Consider the $n-$ary relation $D \in \mathcal{R}$. D is Diophantine over R with coefficients in A if there exists a polynomial $P \in A[X_1, \ldots, X_{n+m}]$ such that for all $x_1, \ldots, x_n \in R$ we have

$$D(x_1, \ldots, x_n) \longleftrightarrow \exists y_1, \ldots, y_m \in R : P(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0$$

Example 1.1.5: The equation $x^2 - 2x + 1 = 0$ is a Diophantine equation in $\mathbb{Z}$ (solution $x = 1 \in \mathbb{Z}$), with coefficients in $\mathbb{Z}$.

Example 1.1.6: For $x, y \in \mathbb{Z}$, $x < y$ is Diophantine over $\mathbb{Z}$ (with coefficients in $\mathbb{Z}$) since

$$x < y \longleftrightarrow \exists z \in \mathbb{Z} : x + z = y$$

Intuitively an algorithm is a finite set of rules which form the guidelines for a certain task to be carried out. This was known in Hilbert's time, but the intuitive notion alone did not suffice to prove the nonexistence of an algorithm to a certain problem. For such an approach, a rigorous and precise definition of what an algorithm is, was needed.
This was achieved in 1936 in the Church-Turing Thesis in which Alonzo Church and Alan Turing, independently, gave two formal definitions for the notion of the algorithm. Eventually, Church's $\lambda-$calculus and Turing's Turing machines were proved to be equivalent definitions.

The original problem that Hilbert expressed, concerned the Diophantine problem for $\mathbb{Z}$ with coefficients in $\mathbb{Z}$. It was proven in 1970 (see [3], [4]), with the contribution of Yuri Matijasevich, Hilary Putnam, Martin Davis and Julia Robinson, that it is unsolvable. This result became known as the DPRM theorem.
In this thesis we will give an overview of the DPRM theorem as well as some results concerning Hilbert's tenth problem for rings other than $\mathbb{Z}$.

Chapter 2

Turing Machines

A Turing machine is an abstract computing model which consists of an infinite tape and a cursor. On the tape one can insert an arbitrarily long string as input. In each step of the computation, the machine is capable of making changes to the initial string using its cursor in order to read, one symbol at a time, and write another in its place, if instructed to do so. The cursor can also move to the left or to the right in the tape and in each step it can only move once. After the computation, the machine either prints an output, accept or reject, or nothing at all and enters in an infinite loop.
We say that the machine halts if it prints an output, otherwise it loops.

Alan Turing invented this model in 1936 for the need to solve the decision problem that was posed in 1928 by Hilbert and Ackermann, known as Entscheidungsproblem. The problem asked for an algorithm to decide whether, any given statement of first-order logic, is universally valid, i.e. valid in any structure in which the axioms are satisfied, or not. The answer to the problem was that there exists no algorithm to carry out the task. To give the answer, Turing reduced the initial problem to that of seeking a Turing machine that decides if any other Turing machine halts or not in an arbitrary input. This is known as the halting problem and Turing proved its undecidability.

2.1 Definition of Turing Machine

We will use [15] as a main reference for the definition of Turing machines.

Definition 2.1.1: A Turing machine is a $7-$tuple $(Q, A, B, \delta, q_0, q_a, q_r)$ where

1. Q is the set of states
2. A is the input alphabet
3. B is the tape alphabet
4. δ is the transition function $\delta : Q \times B \to Q \times B \times \{L, R, -\}$, R =right, $-$ =stay
5. q_0 is the starting state
6. q_a is the accept state
7. q_r is the reject state

A and B are both finite sets. A contains the symbols allowed to use as input and B contains A and the blank symbol, denoted as $\sqcup$. In some cases B contains additional symbols, except for $\sqcup$, not contained in A. Those symbols are only used during the computation in order to make it easier for the machine in cases of long input strings. We will make this more precise later.

Definition 2.1.2: A string produced by the alphabet A is a finite sequence $a_1 a_2 \dots a_n$ where $a_i \in A$ and $i = 1, \dots, n \in \mathbb{N}$. We will denote the set of strings produced by A as A^*.

Definition 2.1.3: Consider an alphabet A. A language L is a subset of A^* and it contains strings produced by A.

To initiate the computation, one must enter a string as input. Once the input string is inserted it occupies the leftmost side of the tape. The rest of the tape contains the blank symbol.

The Turing machine $M = (Q, A, B, \delta, q_0, q_a, q_r)$ makes the computation in the following way:

The steps of the computation are determined by the transition function $\delta : Q \times B \to Q \times B \times \{L, R, -\}$.
Let $q_1 \in Q$, w, $a_1 \in B$ and $\delta(q_0, a_1) = (q_1, w, R)$. From this we understand that the cursor points at the first symbol, a_1, of the word and the current state is the starting state q_0.
In the first step, the cursor will type the symbol w in the place of a_1 and move to the right, as R denotes. At the end of the first step the string will be $a = wa_2a_3 \dots a_n$, the cursor pointing at a_2 and at state q_1.
If in the above step we had L instead of R, since a_1 is in the leftmost side of the tape, the cursor would remain in the same position after the replacement of a_1 with w.
In each state of the computation, the exact position of the cursor, the part of the string on the left of the cursor and the part on the right compose a clear image of the status of the computation.
Let u be the string on the left, v the string on the right and q the current state. The triplet (u, q, v) is called configuration of the Turing machine.
In a configuration the cursor points at the first symbol of the string on the right.

For the configurations $C_1 = (u_1, q_1, v_1)$, $C_2 = (u_2, q_2, v_2)$ we say that C_1 yields C_2 if the machine makes the transition from state q_1 to q_2 in s single step.

We say that the Turing machine accepts a string if there is a sequence $C_1C_2 \dots C_n$ of configurations, in which C_i yields C_{i+1} and the state in C_n is the accepting state.

Let x an input string of a Turing machine M, the output will be denoted as $M(x)$ and if M loops on x we write $M(x) = \nearrow$

Consider an alphabet A and a language L of A.
Consider also a Turing machine $M = (Q, A, B, \delta, q_0, q_a, q_r)$.

Definition 2.1.4: If for every $a \in L$ we have that, M results in an accepting state with input a, and for $a \notin L$, M results in a rejecting state or $M(a) = \nearrow$, then we say that M recognises L.

Definition 2.1.5: A language is Turing-recognisable (or recursively enumerable) if there exists a Turing machine that recognises it.

If a Turing machine M recognises a language L then it is called recogniser of L.
Turing machines that always halt regardless of the input are called deciders.

Definition 2.1.6: A language is decidable (or recursive) if there exists a decider that recognises it.

From the definitions it becomes clear that if a language is decidable then it is also Turing-recognisable.

Example 2.1.1: Below we describe a decider for
$L = \{a * a | a \in \{0,1\}^*\}$.

Consider M_L such a decider, the input alphabet is $\{0, 1, *\}$ and the tape alphabet is $\{0, 1, *, x, \sqcup\}$.
The machine has to accept the input if the string on the left and the string on the right of $*$ are identical and comprised only by 0's and 1's. Otherwise it must reject.

1. First, the cursor will begin reading the word from left to right, if no $*$ is found in the input, the machine rejects, as the word doesn't belong in the language.
   If one $*$ is found, the word splits to two parts, the one on the left of the $*$ and the one on the right.
   If more than one $*$ are found, the machine rejects.

2. The cursor will start reading the first symbol of the left part and then the first symbol of the right part.
   If those symbols are not the same, reject.

3. Else, replace each symbol with x and go on with the procedure. If this procedure is repeated n times, the cursor will have replaced the n first symbols of each word with a x. This will help the machine to keep in memory which of the symbols are identical and how many symbols remain to be examined.
   At the end of the above procedure, if both parts have the same length and all their symbols have been replaced with a x, accept. Otherwise, reject.

   The symbol $x \in B$ is an example of the additional symbols, as mentioned above, contained only in the tape alphabet. In this example the symbol clarifies what part of the string remains to be examined in order to terminate the computation. If A also contained x, it would cause confusion.

   In the following page, a detailed description of the machine is presented, with 9 states in total, including q_a, q_r.

$q \in Q$	$a \in B$	$\delta(q, a)$
q_0	0	(q_1, x, R)
q_0	1	(q_4, x, R)
q_0	$*$	$(q_7, *, R)$
q_1	0	$(q_1, 0, R)$
q_1	1	$(q_1, 1, R)$
q_1	$*$	$(q_2, *, R)$
q_2	0	(q_3, x, L)
q_2	x	(q_2, x, R)
q_2	1	$(q_r, 1, -)$
q_2	$*$	$(q_r, *, -)$
q_3	0	$(q_3, 0, L)$
q_3	1	$(q_3, 1, L)$
q_3	$*$	$(q_6, *, L)$
q_3	x	(q_3, x, L)
q_4	0	$(q_4, 0, R)$
q_4	1	$(q_4, 1, R)$
q_4	$*$	$(q_5, *, R)$
q_5	x	(q_5, x, R)
q_5	1	(q_3, x, L)
q_5	0	$(q_r, 0, -)$
q_6	0	$(q_6, 0, L)$
q_6	1	$(q_6, 1, L)$
q_6	x	(q_0, x, R)
q_6	$*$	$(q_r, *, -)$
q_7	x	(q_7, x, R)
q_7	$\sqcup$	$(q_a, \sqcup, -)$

In the above example, we constructed a Turing machine able to decide, in a finite number of steps, whether any string belongs in L or not. If we want to use different words for our result and ignore the notion of the Turing machine, we can say that an algorithm was given to carry out the task.

The association between Turing machines and algorithms is everything but accidental. Actually in 1936, it was Alan Turing who gave the formal definition of algorithm using the Turing machines. This made possible to prove that there exist algorithmically unsolvable problems, i.e. problems for which a finite sequence of instructions will never be constructed in order to obtain a solution.

2.2 Undecidability

Every Turing machine recognises exactly one language by construction. Even though there is a wide range of computing capabilities concerning Turing machines, there exist many languages that can not be decided or even recognised.
One way to capture the concept of undecidability of some languages is to compare the size of the set of all Turing machines and the size of the set of all languages. We will now make this claim more precise.

The definition of Turing machines makes clear that the transition function δ generates the computation. The domain of δ is the Cartesian product of the set of states Q and the alphabet B, while the image belongs to $Q \times B \times \{L, R, -\}$. It is easy to see that since Q, B are countable sets then both $Q \times B \times \{L, R, -\}$, $Q \times B$ are countable as well. So all the transition functions, therefore all Turing machines, are countably many.

We will prove that the set of all languages is uncountable.

Let $\mathcal{L}$ the set of all languages over a finite alphabet A and
$A^* = \{s_1, s_2, \dots\}$ the set of all strings produced by A. The strings of A^* are enumerated in terms of their length and we keep this enumeration for the rest of the chapter.
For instance, consider the alphabet $A = \{a, b, c\}$ then

$$A^* = \{\sqcup, a, b, c, ab, bc, \dots, abc, \dots, abcb, \dots\}$$

Consider also the set $\mathcal{B}$ of all the binary sequences,
i.e. $\mathcal{B} = \{f : \mathbb{N} \longrightarrow \{0, 1\}\}$.
We will find a, one to one and onto, correspondence $g : \mathcal{L} \to \mathcal{B}$ in order to prove that they have the same size.

For each language $L \in \mathcal{L}$ where $L \subseteq A^*$ produced by A,
$g(L) = b_1 b_2 \dots$ where

$$b_i = \begin{cases} 1, & \text{if } s_i \in L \\ 0, & \text{if } s_i \notin L \end{cases}$$

Example 2.2.1: Consider the alphabet $A = \{a, b, c\}$ and the language $L = \{a^n bc^n | n \in \mathbb{N}\}$.
Every string in L is a concatenation of a string of a's and a string of c's, both of the same length, seperated by a single b. It also includes the empty string $\sqcup$.
We have

$$A^* = \{\sqcup, a, b, c, ab, bc, \dots, abc, acb, cba, \dots, abcb, \dots\}$$

$$L = \{\sqcup, b, abc, aabcc, aaabccc, \dots\}$$

Hence the characteristic function of L is

$$g(L) = 101000 \dots 100 \dots$$

By definition, g is one to one.
g is also onto because for any binary sequence $s \in \mathcal{B}$ there exists $L \in \mathcal{L}$ that contains, in the i-th position according to the enumeration, the i-th string of A^* for which the sequence has 1 in the same position.

Now it is easy to see that $\mathcal{B}$ is an uncountable set.
We will prove it using Cantor's diagonal argument, a technique first used by Georg Cantor to prove that the real numbers are not in a one to one correspondence with the naturals, hence uncountable.

If it was countable one could sort all the binary sequences in a list of infinite length.

$$\begin{array}{ll} 1 & 010101010101111\dots \\ 2 & 101011111000000\dots \\ 3 & 000111000011111\dots \\ \vdots & \qquad\qquad\vdots \end{array}$$

Now we choose a binary sequence that differs from the nth sequence, of the above table, in the nth element.
Meaning that its first element is 1, the second is also 1 and so on.
Therefore we have a binary sequence different from the others so it doesn't belong in the list.
A contradiction since we assumed that all the binary sequences are listed.
Hence $\mathcal{B}$ is an uncountable set.

If we omit the countable set of languages that are recognised from the uncountable set $\mathcal{L}$, what remains is an uncountable set of unrecognisable languages.

2.3 Universal Turing Machine

Turing machines are constructed to receive a string as input. This seems restricting since, there are several mathematical objects, other than strings, of common use. Those objects could be polynomials, sets that contain numbers, graphs and many more that compose a wide variety of problems. In order to work on these problems with the aid of Turing machines, we will have to encode those objects as strings. In other words, we try to turn the language that contains them, into a language that contains their encodings. If this is achieved and the objects are successfully encoded as strings, it becomes feasible to use them as input to the machine. Once we insert the input, the machine first must confirm if the encoding is valid or not. If it is valid, i.e. the encoding of the object is the proper one, then the machine resumes the computation. Let S be an object, then the encoding of S to a string will be denoted as $\langle S \rangle$ and the encoding of a combination of objects $S_1, S_2, \ldots, S_n$ will be denoted as $\langle S_1, S_2, \ldots, S_n \rangle$.

Example 2.3.1: We want to devise a Turing machine that decides whether a polynomial, in one variable with coefficients in $\mathbb{Z}_2 = \{\bar{0}, \bar{1}\}$, the field with 2 elements, has roots in $\mathbb{Z}_2$.
The language of the problem is

$L = \{p(x) \in \mathbb{Z}_2[X] |\ p(x)$ has roots in $\mathbb{Z}_2\}$.
Before we devise the machine we have to represent the polynomials as strings.

Let $p(x) \in \mathbb{Z}_2[X]$, we define $\langle p(x) \rangle = (a_0, a_1, \ldots, a_n)$ where
n is the degree of $p(x)$, $a_i = 0$ or 1 and each of the a_i's is the coefficient of x^i

$$p(x) = 1 + x + x^3 + x^5$$

$$\langle p(x) \rangle = (1, 1, 0, 1, 0, 1)$$

Once $\langle p(x) \rangle$ is inserted, the machine should examine its validity first. $\langle p(x) \rangle$ is the encoding of $p(x)$ if:

1. its first and last symbol is "(" and ")" respectively

2. the symbols between the parentheses should only be "0", "1" and ",".
   More precisely, the first symbol on the right of "(" and on the left of ")" is always 0 or 1. "," comes always after a 0 or 1 and vice versa.

For instance, "$(1, 0, 1)$" is a proper encoding of $1 + x^2$ and therefore the machine goes on with the computation. On the other hand "$(, , 0, 1)$" is not a proper encoding of any polynomial and the machine rejects.

The following fact will be crucial in order to construct the algorithm that will decide L.

Fact: Consider $p \in \mathbb{Z}_2[X]$, 0 is a root of p if and only if the constant term is 0. 1 is a root of p if and only if the sum of the coefficients is a multiple of 2.

L_m is decided in the following way:

1. If the first symbol after the opening parenthesis is 0, then the computation stops and the machine accepts.

2. Else the cursor reads all the symbols of the string.
   If the number of 1's is even then accept, else reject.

Let's consider a Turing machine M and an input string w. We can encode both of these objects to a string $\langle M, w\rangle$ and use it as input in another Turing machine M_1. In [1] the reader will find an example of such an encoding. In this case we say that M_1 simulates M on input w. There is a particular kind of a Turing machine, able to simulate any other Turing machine. It is called universal Turing machine and its computing capabilities are of a much wider variety comparing to those of any Turing machine. Briefly, this machine is able to compute anything that can be computed by a Turing machine.
The use of this machine is crucial for proving the unsolvability of several problems and we will use it to examine one of the most famous problems of this kind, the Halting problem.

2.4 The Halting Problem

A famous undecidable problem, i.e. a problem for which no Turing machine can be constructed to give a solution, is the halting problem. It states that, given a Turing machine M and a string w, can we decide if M halts on w? We ask for the existence of a universal Turing machine H which will decide the language
$HP = \{\langle M, w\rangle |\ M \text{ halts on } w\}$ printing the appropriate output.

H functions as follows:

$$H(\langle M, w\rangle) = \begin{cases} accept, & \text{if M halts on w} \\ reject, & \text{if M loops on w} \end{cases}$$

If such a machine exists, we can construct another Turing machine

D that simulates H to produce an output:

$$D(\langle M \rangle) = \begin{cases} accept, & \text{if H rejects } \langle M, \langle M \rangle \rangle \\ \nearrow, & \text{if H accepts } \langle M, \langle M \rangle \rangle \end{cases}$$

For the input of D we use the encoding of a machine M and for the output D loops if M halts on $\langle M \rangle$ and accepts if M loops on $\langle M \rangle$. Since D is a Turing machine it has an encoding to a string $\langle D \rangle$ so let's use it as it's own input.
We have that, if H accepts $\langle D, \langle D \rangle \rangle$, it means that D halts on $\langle D \rangle$. Then by definition, D loops on $\langle D \rangle$.
If H rejects on $\langle D, \langle D \rangle \rangle$, i.e. D loops on $\langle D \rangle$, then by definition, D halts on $\langle D \rangle$. In both cases we have contradiction.

It must be noted, though, that HP is Turing-recognisable.
The recogniser is the following:

$$R(\langle M, w \rangle) = \begin{cases} accept, & \text{if M halts on w} \\ \nearrow, & \text{if M does not halt on w} \end{cases}$$

So we have an example of a language that is Turing-recognisable, but undecidable.

Theorem 2.4.1
A language L is decidable iff both L, L^c are Turing-recognisable.

Proof. Let M_1, M_2 the recognisers of L and L^c respectively., and w a string.
A new Turing machine M can be obtained which operates as follows:

$$M(w) = \begin{cases} accept, & \text{if } M_1(w) = accept \\ reject, & \text{if } M_2(w) = accept \end{cases}$$

M simply has M_1 and M_2 run in parallel and, regardless of the input, exactly one of them halts.
When that happens it outputs the appropriate word.

Since HP is undecidable but Turing-recognisable it is easy to see that its complement $HP^c = \{\langle M, w \rangle |$ M loops on w$\}$ is not Turing-recognisable. □

2.5 Proof of Hilbert's Tenth Problem

Given an arbitrary polynomial $P \in \mathbb{Z}[X_1, X_2, \ldots, X_n]$, $n \geq 1$.
Is there an algorithm to decide if there exist an $n-$tuple
$(a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$ for which $P(a_1, a_2, \ldots, a_n) = 0$?

As mentioned above this is Hilbert's tenth problem, first presented in 1900. In 1970 it was proven that such an algorithm does not exist. More information about the proof can be found in [3]. The main idea is to reduce the procedure of finding such an algorithm to the procedure of finding a Turing machine that will solve the Halting problem.

The main step is to prove the DPRM theorem which states:

Theorem 2.5.1(DPRM Theorem): A subset X of $\mathbb{Z}^n$ is Diophantine if and only if it is recursively enumerable.

Before we proceed to the proof we should define first what it means for a set to be Diophantine and what it means to be recursively enumerable.

Definition 2.5.1: Let $S \subseteq \mathbb{Z}^n$ and $P \in \mathbb{Z}[X_1, \ldots, X_{n+m}]$. Then S is called Diophantine if

$$(x_1, \ldots, x_n) \in S \Longleftrightarrow \exists (y_1, \ldots, y_m) \in \mathbb{Z}^m : P(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0$$

In short,

$$S = \{\tilde{x} \in \mathbb{Z}^n | \exists \tilde{y} \in \mathbb{Z}^m : P(\tilde{x}, \tilde{y}) = 0\}$$

is a Diophantine subset of $\mathbb{Z}^n$

Example 2.5.1: The set of even numbers $S = \{x | \exists y : x = 2y\} \subset \mathbb{Z}$ is Diophantine with $P(x, y) = x - 2y$

Diophantine sets are closed under the operations of intersection and union. Take D_1 and D_2 to be Diophantine sets defined by the polynomials P_1 and P_2 respectively. Then $D_1 \cup D_2$ and $D_1 \cap D_2$ are Diophantine sets with polynomials $P = P_1 P_2$ and $R = P_1^2 + P_2^2$

respectively.

Definition 2.5.2: A function $f : \mathbb{Z}^n \to \mathbb{Z}$ is called Diophantine if the set of $(n+1)-$tuples of the form $(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$ is a Diophantine set.

Example 2.5.2: $f(x, y) = xy$ is a Diophantine function.
Let $D = \{(x, y, xy) | x, y \in \mathbb{Z}\}$, then

$$(x, y, z) \in D \Longleftrightarrow z - xy = 0$$

Consider a natural number n, the number

$$T(n) = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

is called triangular number.
We have that for any $z \in \mathbb{N}$, where $T(n) < z \leq T(n+1)$ there are unique a, b such that

$$z = T(a + b - 2) + b$$

Theorem 2.5.2(Pairing Function Theorem): There are Diophantine functions

$$P : \mathbb{N}^2 \to \mathbb{N}$$
$$R : \mathbb{N} \to \mathbb{N}$$
$$L : \mathbb{N} \to \mathbb{N}$$

such that

1. for all $a, b \in \mathbb{N}$: $L(P(a, b)) = a$, $R(P(a, b)) = b$.

2. for all $z \in \mathbb{N}$: $P(L(z), R(z)) = z$ and $L(z) \leq z$, $R(z) \leq z$

For the proof see page 5 of [3].

Now we define the function $S : \mathbb{N}^2 \to \mathbb{N}$ as $S(i, u) = w$ where $i, u \in \mathbb{N}$ and w is the least remainder when $L(u)$ is divided by $1 + iR(u)$, i.e.

$$w \equiv L(u) \mod 1 + iR(u)$$
$$w \leq 1 + iR(u)$$

Theorem 2.5.3 (Sequence Number Theorem): There is a Diophantine function S such that:

1. $S(i, u) \leq u$
2. for any finite sequence $a_1, \ldots, a_n$ where $a_i \in \mathbb{N}$ there exists a u such that
$$S(i, u) = a_i$$
for $i = 1, \ldots, n$

Proof. See page 7 of [3]. □

From the above theorem it is clear that every finite sequence of natural numbers can be expressed solely by the Diophantine function S, one need only find the natural number u.

Example 2.5.3: Set $u = 16$, we have $L(16) = 6$, $R(16) = 1$ so

$$S(i, 16) \equiv 6 \mod 1 + i$$

$$S(1, 6) = 2, S(2, 6) = 3, S(3, 6) = 2, S(4, 6) = 1, S(5, 6) = S(6, 6) = \cdots = 6$$

So the sequence $2, 3, 2, 1, 6, 6, \ldots, 6$ is encoded by $u = 16$

This technique was first used by Gödel, [4], in his attempt to find a, one to one and onto, correspondence between the natural numbers and the finite sequences of natural numbers.

Consider the functions

$$c(x) = 1$$
$$S(x) = x + 1$$
$$P_i^n(x_1, \ldots, x_n) = x_i$$
$$S(i, u) = w$$

Definition 2.5.3: A function is called recursive if it is obtained by the above functions by applying composition, primitive recursion and minimalization.

Definition 2.5.4:

1. For the given functions $g_1, \ldots, g_m : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, $f : \mathbb{Z}^m \longrightarrow \mathbb{Z}$. A function, $h : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, of the form

   $$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)))$$

   is the **composition** of $g_1, \ldots, g_m, f$

2. A function $h : \mathbb{Z}^{n+1} \longrightarrow \mathbb{Z}$ is yielded by **primitive recursion** if it satisfies the following conditions:

   $$h(z_1, \ldots, z_n, 1) = f(z_1, \ldots, z_n)$$

   $$h(z_1, \ldots, z_n, t+1) = g(t, h(z_1, \ldots, z_n, t), z_1, \ldots, z_n)$$

   for given functions $f : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, $g : \mathbb{Z}^{n+2} \longrightarrow \mathbb{Z}$

3. For given functions $f, g : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, the function $h : \mathbb{Z}^n \longrightarrow \mathbb{Z}$ of the form

   $$h(x_1, \ldots, x_n) = min_y(f(x_1, \ldots, x_n, y) = g(x_1, \ldots, x_n, y))$$

   is yielded by **minimalization** and it takes, as value, the least natural number y for which $f(x_1, \ldots, x_n, y) = g(x_1, \ldots, x_n, y)$. Provided, of course, that there exists at least one such y that satisfies this equation.

The interested reader can check [7] for more details.

Example 2.5.4: $f(x, y) = x + y$ is recursive since

$$f(x, 1) = s(x) = x + 1$$

$$f(x, t+1) = s(x + t) = s(P_2^3(t, x + t, x))$$

Intuitively a recursive function, for an element $(x_1, \ldots, x_n) \in \mathbb{N}^n$, computes the value $f(x_1, \ldots, x_n)$ in a finite number of steps by referring to itself in every step of the computation.

Theorem 2.5.4: A function is Diophantine if and only if it is recursive.

Proof. Consider a Diophantine function $f : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, we have that

$$y = f(x_1, \ldots, x_n) \longleftrightarrow \exists y_1, \ldots, y_m : P(x_1, \ldots, x_n, y, y_1, \ldots, y_m) = 0$$

it is a fact that for every polynomial P with integer coefficients there exist polynomials Q, R, both with positive integer coefficients, such that $P = Q - R$. Therefore the above expression turns into

$$y = f(x_1, \ldots, x_n) \longleftrightarrow \exists y_1, \ldots, y_m : Q(x_1, \ldots, x_n, y, y_1, \ldots, y_m) = R(x_1, \ldots, x_n, y, y_1, \ldots, y_m)$$

For the sequence $y, y_1, \ldots, y_n$ we have from the **Sequence Number Theorem** that $\exists u \in \mathbb{N}$ such that $y = S(1, u), y_1 = S(2, u), \ldots, y_m = S(m + 1, u)$
Hence $\exists u \in \mathbb{N} : Q(x_1, \ldots, x_n, S(1, u), S(2, u), \ldots, S(m + 1, u)) = R(x_1, \ldots, x_n, S(1, u), S(2, u), \ldots, S(m + 1, u))$

We can take $u = min_u(Q(x_1, \ldots, x_n, S(1, u), S(2, u), \ldots, S(m+1, u)) = R(x_1, \ldots, x_n, S(1, u), S(2, u), \ldots, S(m + 1, u)))$
Hence $f(x_1, \ldots, x_n) = S(1, u)$ where u as above.
Since S, Q, R are recursive, so is f by using composition and minimalization.

Conversely, let f be recursive. Every recursive function is obtained by $c(x)$, $S(x)$, $P_i^n(x_1, \ldots, x_n)$, $S(i, u)$ using composition, minimalization and primitive recursion.
It is already known that the above functions are Diophantine, hence we need only prove that composition, minimalization and primitive recursion of Diophantine functions yield a Diophantine function.

1. **Composition**: For given Diophantine functions $g_1, \ldots, g_m : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, $f : \mathbb{Z}^m \longrightarrow \mathbb{Z}$, composition yields the function

   $$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)))$$

   we have

$$y = h(x_1, \ldots, x_n) \longleftrightarrow \exists t_1, \ldots, t_m : \\ \left(\bigwedge_{i=1}^{m} t_i = g_i(x_1, \ldots, x_n)\right) \wedge y = f(t_1, \ldots, t_m)$$

since $f, g_1, \ldots, g_m$ are Diophantine, the elements $(t_i, x_1, \ldots, x_n)$ for $i = 1, \ldots, m$ and $(y, t_1, \ldots, t_m)$ belong, each, to a Diophantine set. We consider the polynomials of these sets to be $P_1, P_2, \ldots, P_{m+1}$ respectively. Therefore the right hand side of the relation above is a Diophantine set defined by the polynomial $P = P_1^2 + P_2^2 + \cdots + P_{m+1}^2$ and the relation turns into

$$y = h(x_1, \ldots, x_n) \longleftrightarrow \exists t_1, \ldots t_m : P(y, x_1, \ldots x_n, t_1, \ldots, t_m) = 0$$

Hence $y = h(x_1, \ldots, x_n)$ is a Diophantine relation, so h is a Diophantine function.

2. **Primitive recursion**: For given Diophantine functions $g : \mathbb{Z}^{n+2} \longrightarrow \mathbb{Z}$, $f : \mathbb{Z}^n \longrightarrow \mathbb{Z}$, primitive recursion yields the function $h : \mathbb{Z}^{n+1} \longrightarrow \mathbb{Z}$

$$h(x_1, \ldots, x_n, 1) = f(x_1, \ldots, x_n)$$

$$h(x_1, \ldots, x_n, t+1) = g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n)$$

For the finite sequence $h(x_1, \ldots, x_n, t)$, where $t \in \mathbb{N}$, it is known from the Sequence Number Theorem that $\exists u \in \mathbb{N}$ such that

$$S(t, u) = h(x_1, \ldots, x_n, t)$$

we have that

$$y = h(x_1, \ldots, x_n, t) \longleftrightarrow \exists u(\exists v\{v = S(1, u) \wedge \\ \mathrm{v{=}f(x_1, \ldots, x_n)}\} \bigwedge (\forall z)_{\leq t}\{z = t \cup (\exists v) v = S(z+1, u) \\ \wedge v = g(z, S(z, u), x_1, \ldots, x_n)\} \bigwedge y = S(t, u)$$

Note: The predicate $(\forall z)_{\leq t} \ldots$ is equivalent to the expression $(\forall z)(z > t \cup \ldots)$ and in **Theorem 5.1** of [3] is proven to be a Diophantine predicate, like the existential quantifier $\exists$.

Therefore, in the same manner as 1., h is Diophantine since f, g, S are Diophantine and $(\forall z)_{\leq t}$ is a Diophantine predicate.

3. **Minimalization** yields the function
$h(x_1, \ldots, x_n) = min_y(f(x_1, \ldots, x_n, y) = g(x_1, \ldots, x_n, y))$,
where f, g are Diophantine functions. h is Diophantine because

$$y = h(x_1, \ldots, x_n, t) \longleftrightarrow \exists z(z = f(x_1, \ldots, x_n, y) \wedge z = g(x_1, \ldots, x_n, y)) \bigwedge (\forall t)_{\leq y}\{t = y \cup (\exists u, v)$$
$$\big((\mathrm{u{=}f}(\mathrm{x}_1, \ldots, x_n, t) \wedge v = g(x_1, \ldots, x_n, t)) \wedge (u < v \cup v < u)\big)\}$$

□

Definition 2.5.5: A recursively enumerable set S is a subset of $\mathbb{Z}^n$ for which there exist recursive functions f, g such that

$$(x_1, \ldots, x_n) \in S \Longleftrightarrow \exists x : f(x, x_1, \ldots, x_n) = g(x, x_1, \ldots, x_n)$$

Intuitively a recursively enumerable set S is created by an algorithm which, if left running, will eventually list only the elements of S.

Definition 2.5.6 A subset S of $\mathbb{Z}^n$ is recursive if there exists a recursive function $f : \mathbb{Z}^n \to \{0, 1\}$ such that

$$f(\tilde{x}) = \begin{cases} 1, & \text{if } \tilde{x} \in S \\ 0, & \text{if } \tilde{x} \notin S \end{cases}$$

Lemma 2.5.1: If a set S is recursive, then it is recursively enumerable.

Proof. Consider a recursive set S. Then there exists a recursive function f such that

$$\tilde{x} \in S \longleftrightarrow f(\tilde{x}) = 1$$

The constant function 1 is a recursive function as well, therefore we have that there exist recursive functions f, g (where g is the constant function 1) such that

$$\tilde{x} \in S \longleftrightarrow f(\tilde{x}) = g(\tilde{x})$$

By **Definition 2.5.5** S is recursively enumerable. □

The main difference between recursively enumerable sets and recursive sets is that the algorithm for the former does not produce

an output if the input is not in the set, while the algorithm for the latter always produces an output.

Proof of the DPRM Theorem

Proof. Consider a recursively enumerable set S then

$$S = \{(x_1, \ldots, x_n) \in \mathbb{Z}^n | \exists x : f(x, x_1, \ldots, x_n) = g(x, x_1, \ldots, x_n)\}$$

where f, g are recursive functions.
Equivalently

$$S = \{(x_1, \ldots, x_n) \in \mathbb{Z}^n | \exists (x, z) : z = f(x, x_1, \ldots, x_n) \wedge z = g(x, x_1, \ldots, x_n)\}$$

Since f, g are recursive, by **Theorem 2.5.3** they are Diophantine.
So the expression

$$z = f(x, x_1, \ldots, x_n) \wedge z = g(x, x_1, \ldots, x_n)$$

is equivalent to the expression

$$\exists \tilde{y} \in \mathbb{Z}^m : P_1(x, z, x_1, \ldots, x_n, \tilde{y}) = 0 \wedge P_2(x, z, x_1, \ldots, x_n, \tilde{y}) = 0$$

and we have that

$$P_1 = 0 \wedge P_2 = 0 \Longleftrightarrow P_1^2 + P_2^2 = 0$$

so we can find a polynomial P such that

$$S = \{(x_1, \ldots, x_n) \in \mathbb{Z}^n | \exists x, z, \tilde{y} : P(x, z, x_1, \ldots, x_n, \tilde{y}) = 0\}$$

Hence S is Diophantine.

Conversely, consider S to be Diophantine, from **Definition 2.5.1** there exist a polynomial $P \in \mathbb{Z}[X_1, \ldots, X_{n+m}]$ such that

$$(x_1, \ldots, x_n) \in S \longleftrightarrow \exists y_1, \ldots, y_m : P(x_1, x_n, y_1, \ldots, y_m) = 0$$

We have the fact that P can be equal to $R - Q$ for
$R, Q \in \mathbb{Z}[X_1, \ldots, X_{n+m}]$ both with positive integer coefficients. We have

$$(x_1, \ldots, x_n) \in S \longleftrightarrow \exists y_1, \ldots, y_m : R(x_1, \ldots, y_m) = Q(x_1, \ldots, y_m)$$

For the finite sequence $y_1, \ldots, y_m$ and the **Sequence Number Theorem** we have that $\exists u \in \mathbb{Z}$ such that $S(i,u) = y_i$ for $i = 1, \ldots m$. Therefore, we have

$$(x_1, \ldots, x_n) \in S \longleftrightarrow \exists u : R(x_1, \ldots, x_n, S(1,u), \ldots, S(m,u)) = Q(x_1, \ldots, x_n, S(1,u), \ldots, S(m,u))$$

Since R, Q are recursive functions, according to **Definition 2.5.5**, S is recursively enumerable. □

Hence it is proved that all the Diophantine sets are recursively enumerable and vise versa.

Along with the proof of the **DPRM Theorem** came the proof of the negative answer to Hilbert's tenth problem in 1970. In 1949 Martin Davis had conjectured that Diophantine sets and recursively enumerable sets are equivalent notions. A year later, Julia Robinson made the following hypothesis (**J.R Hypothesis**) in her attempt to prove that a set is exponential Diophantine if and only if it is Diophantine:

J.R Hypothesis:

Consider $(a,b) \in \mathbb{Z}^2$, there exists a Diophantine set D such that

1. If $(a,b) \in D$ then $b < a^a$ and
2. For every $k \in \mathbb{N}$ there exists $(a,b) \in D$ such that $b > a^k$.

Consider a polynomial $P \in \mathbb{Z}[X_1, \ldots, X_{n+3m}]$, a set $S \subset \mathbb{Z}^n$ is exponential Diophantine if

$$\tilde{x} \in S \leftrightarrow \exists \tilde{y}, \tilde{u}, \tilde{z} \in \mathbb{Z}^m : P(\tilde{x}, \tilde{y}, \tilde{u}, \tilde{z}) = 0 \bigwedge_{i=1}^{m} y_i = u_i^{z_i}$$

The **J.R. Hypothesis** was an open problem for 20 years and it was a sufficient condition for the negative answer to Hilbert's tenth problem. Later in 1959, Davis and Putnam, assuming that the J.R. hypothesis was true, proved that a set is recursively enumerable if and only if it is exponential Diophantine. Finally in 1970, Yuri Matijasevic proved the hypothesis and therefore the negative answer to the problem.

Consider now, a Diophantine set S and $\langle S \rangle$ to be the language that

contains the encodings of the elements of S as strings.

Theorem 2.5.5: There is a Turing machine $M = (Q, A, B, \delta, q_0, q_a, q_r)$ that recognises $\langle S \rangle$. Furthermore, if S is recursive there exists a Turing machine that decides $\langle S \rangle$.

Proof. Consider a polynomial $P \in \mathbb{Z}[X_1, \ldots, X_{n+m}]$ where $n, m \in \mathbb{N}$, then

$$S = \{\tilde{x} \in \mathbb{Z}^n | \exists \tilde{y} \in \mathbb{Z}^m : P(\tilde{x}, \tilde{y}) = 0\}$$

Consider the strings of the form $\langle \tilde{x}, \tilde{y}, P(\tilde{x}, \tilde{y}) \rangle$ of length $n + m + 1$ where $\langle \tilde{x} \rangle$ of length n, $\langle \tilde{y} \rangle$ of length m and $\langle P(\tilde{x}, \tilde{y}) \rangle$ the $1-$digit string representing the value of P in $\tilde{x}, \tilde{y}$. The language $\langle S \rangle$ contains $\langle \tilde{x}, \tilde{y}, P(\tilde{x}, \tilde{y}) \rangle$ if and only if $P(\tilde{x}, \tilde{y}) = 0$. The input alphabet of the Turing machine is $A = \{0, 1, (,), ;\}$ and the tape alphabet $B = A \cup \{\sqcup\}$. The machine receives as input a string, then uses the cursor to read it from left to right.
For the string to be accepted, it must always start with a "(" (the first digit of the substring $\langle \tilde{x} \rangle$), then it must contain the numbers of $\tilde{x}$ in binary representation (for instance $2 = 01$, $3 = 11$, $4 = 001$, the interested reader can check [14] for more about binary representations of integers) seperated by ";". After the last number of $\tilde{x}$ follows a ")" and then proceeds to substring $\langle \tilde{y} \rangle$, examining it in the same way. After that, follows a single number 0 or 1, whether the value of P in $(\tilde{x}, \tilde{y})$ is zero or not. If the last digit is 0, the machine accepts and we have an element of $\langle S \rangle$. Else, the machine moves the cursor to the right without stopping, thus producing no output. Hence every recursively enumerable set can be translated to a recursively enumerable language.

Consider now a recursive set S. In the same manner as above, we can consider a Turing machine M that accepts input strings of the form $\langle \tilde{x}, f(\tilde{x}) \rangle$ where f is the recursive function from **definition 2.5.6**. M accepts if the last digit of the string is 1, else it rejects. Therefore M is the decider for $\langle S \rangle$ where S is recursive. □

We will use a counter example to prove the following.

Theorem 2.5.6: There is a recursively enumerable set that is not

recursive.

Proof. Consider the set S that contains numbers of the form $2^p 3^x$, where $p, x \in \mathbb{N}$, whenever a Turing machine p halts on input x. Notice that we are allowed to consider natural numbers as Turing machines or input strings of such machines, because, as mentioned in chapter 2.2, there is a one to one and onto correspondence between the set of Turing machines, $\mathbb{N}$ and the set of finite strings.
What we seek is a universal Turing machine U that, given a prime factorization of a number as input, it will be able to tell if the power of 2 is a proper encoding of a Turing machine and the power of 3 an encoding of a string. Then it will examine if p halts on x.

Before we go on with the proof let's first explain how is a Turing machine, along with a string as its input, encoded in order to be used as input to a universal Turing machine.
Since this kind of Turing machine can simulate several other Turing machines, we do not know beforehand how many states it will have. So if we consider an arbitrary Turing machine $M = (Q, A, B, \delta, q_0, q_a, q_r)$ we will treat its states as natural numbers, same for the alphabet, i.e. $B = \{1, 2, \ldots, |A|, \ldots, |B|\}$, $Q = \{|B| + 1, \ldots, |B| + |Q|\}$. The symbols L, R, $-$ are the numbers $|Q| + |B| + 1$, $|Q| + |B| + 2$, $|Q| + |B| + 3$.
All these numbers will be given as input in binary representation. The encoding of M starts with the number $|B|$ followed by "," and then the number $|Q|$. Then follows all the pairs of the form $((q, s), (q, a, S))$, where (q, s) and (q, a, S) are such that
$\delta : (q, s) \to (q, a, S)$ and $S \in \{L, R, -\}$. The symbols "(", ",)", ",", will appear in the input string of U as $|Q| + |B| + 4$, $|Q| + |B| + 5$, $|Q| + |B| + 6$ (in binary). After the last ")" follows the symbol ";", which has the number $|Q| + |B| + 7$ and then the input string of M. Without loss of generality we can consider that the alphabet of M is $\{0, 1\}$, so in every input of U, what follows after ";" is a sequence of 0 and 1.

Let's consider, for instance, the Turing machine with the following properties:
$A = \{0, 1\}$, $B = \{0, 1, \sqcup\}$, $Q = \{q_0, q_1, q_a, q_r\}$ and δ operates as follows

$q \in Q$	$a \in B$	$\delta(q,a)$
q_0	0	$(q_r, 0, -)$
q_0	1	$(q_1, 1, R)$
q_1	0	$(q_a, 0, -)$
q_1	1	$(q_r, 1, -)$

and input string $x = 010101$.

What we want the machine to see is

$$|B|, |Q|, ((q_0, 0), (q_r, 0, -)), ((q_0, 1), (q_1, 1, R)),$$
$$((\mathrm{q}_1, 1), (q_a, 0, -)), ((q_1, 1), (q_r, 1, -)); 010101$$

But since, $|B| = 3$, $|Q| = 4$, we have the following encoding

$$0 = 1, 1 = 2, \sqcup = 3, q_0 = 4, q_1 = 5, q_a = 6, q_r = 7, L = 8, R = 9, - = 10, \text{"("} = 11, \text{")"} = 12, \text{","} = 13, \text{";"} = 14$$

So the machine receives the following string as input (in binary):

$$31341311114131 \ldots 121214121212$$

After the input is received by the machine, the computation begins. The universal machine moves the cursor to find the starting state, then moves to the right of ";" and operates as δ commands on x. This goes on until an accept or reject state occurs.(The procedure on how the universal machine simulates M on input x is explained explicitly in [10])

Proceeding to the proof of **Theorem 2.5.6**, U receives a prime factorization of a number.
The number has the following form

$$2^p 3^x 5^{p_1} 7^{p_2} \ldots$$

if at least one of the powers of $5, 7, \ldots$ is not 0 then U rejects.
Else, the input has the form $2^p 3^x$ and U has to examine if p is a proper representation of a Turing machine $M = (Q, A, B, \delta, q_0, q_a, q_r)$ and if x is a proper representation of an input string of M.
In order to do that, p will be written in the form

$$2^{a_1} 3^{a_2} 5^{a_3} \ldots$$

each of the powers is a natural number and U has to check if those numbers, from left to right, follow the encoding rules as explained above.
For instance, if they follow those rules, then $a_1 = |B|$, $a_2 =$ ",", $a_3 = |Q|$ and so on.
We can again consider, without loss of generality, that $A = \{0, 1\}$.
For the number x, the power of 3, U has to examine the powers of its prime factors. If at least one of the powers is not 1 or 0, U rejects.

If the input is valid, U simulates M on input x.
We can see now that S is recursively enumerable because, if p halts on x, U will halt as soon as p does and accept. If p does not halt on x, then neither will U. Therefore U lists S.
If, for the sake of contradiction, S was recursive, then U should always print an output regardless of the input. In this case U could decide if p halts on x, where p is an arbitrary Turing machine and x its input. Hence U solves the **halting problem**, but it is already proven in **2.4** that it is not possible. □

What we want to prove is:
Theorem 2.5.7: An algorithm to decide whether an arbitrary Diophantine equation with coefficients in $\mathbb{Z}$ has a solution in $\mathbb{Z}$, or not, does not exist.

Proof. Consider a polynomial $P \in \mathbb{Z}[X_1, \ldots, X_{n+m}]$ and the Diophantine set $S \subseteq \mathbb{Z}^n$

$$S = \{\tilde{x} \in \mathbb{Z}^n | \exists \tilde{y} \in \mathbb{Z}^m : P(\tilde{x}, \tilde{y}) = 0\}$$

If there existed an algorithm to decide if a Diophantine equation has a solution, then it could decide if any $(\tilde{x}, \tilde{y}) \in \mathbb{Z}^{n+m}$ where $\tilde{x} \in \mathbb{Z}^n$, $\tilde{y} \in \mathbb{Z}^m$ is or isn't a root of P, equivalently if $\tilde{x}$ is or is not an element of S. This would mean that S is recursive.
Since we assumed that S is an arbitrary Diophantine set, i.e. recursively enumerable, then the existence of this algorithm implies that all the recursively enumerable sets are recursive. This is not possible from **Theorem 2.5.6**. □

Therefore such an algorithm does not exist and Hilbert's tenth problem is unsolvable.

Chapter 3

Extensions of Hilbert's tenth problem

After the negative answer to the Diophantine problem for $\mathbb{Z}$, the question now extends to whether there is an algorithm to decide the same problem for rings other than $\mathbb{Z}$. Such rings are the set of real numbers ($\mathbb{R}$), the set of imaginary numbers ($\mathbb{C}$), the set of rational functions in one variable with coefficients in a finite field ($\mathbb{F}_q(t)$) and many more. For some rings the problem has a positive solution and for others it is still open. The most famous open problem in this area is the Diophantine problem for the rational numbers ($\mathbb{Q}$). To understand a little better the results we need give some definitions.

We have already given the definition of a language and of a structure.
Along with the symbols in a language consider also the logical symbols:

$$A = \{\neg, \vee, \wedge, =, \exists, \forall\}$$

and the set of variables

$$V = \{x_1, x_2, \dots\}$$

The definitions are taken from [7] and [13].

Consider a language L, V and the function symbols of L (denoted L^f) with arity greater than 0.

Definition 3.1.1: A term is an expression produced by the alphabet $V \cup L^f$ and it is defined as follows:

1. the constants and every variable are terms.

2. if $a_1, a_2, \ldots, a_n$ are terms then $f(a_1, \ldots, a_n)$, where $f \in L^f$ of arity n, is a term.

The set of relation symbols in L is denoted by $\mathcal{R}$.

Definition 3.1.2: A formula is an expression produced by the alphabet $L \cup A \cup V$ and is defined as follows:

1. Consider $a_1, a_2, \ldots, a_n$ terms and $r \in \mathcal{R}$ of arity n, then $r(a_1, \ldots, a_n)$ is a formula.

2. if a_1, a_2 are terms then $a_1 = a_2$ is a formula.

3. if ϕ, ψ are formulas then $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$ are also formulas.

4. let x a variable and ϕ a formula, then $\exists x\phi(x)$ and $\forall x\phi(x)$ are also formulas.

$\exists$ is called existential quantifier.
$\forall$ is called universal quantifier.
For a formula ϕ and variables x, y, we say that x, y occur in ϕ if $\phi = \phi(x, y)$.
If a formula is of the form $\exists x\phi(x, y)$ we say that x occurs in ϕ bounded by the existential quantifier and y occurs free in ϕ.

Definition 3.1.3: A sentence ϕ is a formula in which all the variables are bounded by quantifiers.

Example 3.1.1: Consider $L = \{+, \cdot, 0, 1, <\}$ and the $\mathcal{Z}-$structure (integers).
In the formula $\exists x(x < y^2)$, x occurs bounded and y occurs free, while in the formula $\forall x \exists y(x = y)$ all the variables are bounded.

Therefore the second formula is a sentence.

A formula that contains no quantifiers is called quantifier-free.
The sentences that contain the symbols in V, A and L are called first-order sentences.
For the needs of the thesis we will simply call them sentences.

Definition 3.1.4: The sentences of the form $\exists x\phi(x)$, where $x = (x_1, \ldots, x_n)$, x_i are variables and ϕ is a quantifier free formula, are called existential sentences. Existential sentences in which ϕ contains no negations are called positive existential sentences.

Definition 3.1.5: The theory of a structure is the set of sentences that are true in the structure.
Respectively the existential theory is the set of existential sentences that are true in the theory. Same goes for the positive existential theory.

Example 3.1.2: The sentence $\forall x \exists y(y < x)$ belongs to the theory of $\mathbb{Z}$ but not in the theory of $\mathbb{N}$ in the language $L = \{0, 1, +, \cdot, <\}$.

Definition 3.1.6: A theory of a structure is decidable if there is an algorithm to decide if an arbitrary sentence is true or false in the structure.

3.1 Some results

A usual technique for proving the decidability of a theory of a given structure, is to show that the theory admits elimination of quantifiers. That is, if a theory T possesses this property, then any formula in the given language is equivalent, in T, to a quantifier-free formula. It has been proved that the theory of $\mathbb{C}$ in $L = \{+, \cdot, 0, 1\}$ is decidable due to quantifier elimination. Same is true for the Presburger arithmetic, i.e. the arithmetic for natural numbers in $L = \{+, 0, 1, \geq\}$ (see [13]). In [16] is presented an algorithm for quantifier elimination to obtain the decidability for the elementary

algebra. By elementary algebra, we mean the part of the theory of real numbers axiomatized by the following system:

- $\forall x, y \in \mathbb{R}$: $x + y = y + x$ and $x \cdot y = y \cdot x$
- $\forall x, y, z \in \mathbb{R}$: $(x + y) + z = x + (y + z)$ and $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $\exists a \in \mathbb{R} : x + a = a + x = x \ \forall x \in \mathbb{R}$
- $\exists b \in \mathbb{R}$, $b \neq a : x \cdot b = b \cdot x = x \ \forall x \in \mathbb{R}$
- $\forall x \in \mathbb{R}$: $x + (-x) = (-x) + x = 0$ and $x \cdot x^{-1} = x^{-1} \cdot x = 1$
- $\forall x, y \in \mathbb{R}$: $x \cdot (z + y) = x \cdot z + x \cdot y$

consisting of expressions in which the variables represent real numbers and only elementary logic symbols are being used, such as $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$.

For instance, "$\forall a, b \ \exists x : \ ax + b = 0$ is an expression of elementary algebra.
But, "Every polynomial of odd degree has at least one root" is not an expression of elementary algebra.

In general, if a theory of a structure is decidable, then so is its existential theory, therefore a positive answer to Hilbert's tenth problem can be obtained. So for the above structures there exists the procedure required to solve the problem. The converse, though, is obviously not true. The existential theory of $\mathbb{Z}$ in the language $\{+, |, 0, 1\}$ is decidable but the full theory is undecidable (see [2]).

As far as Hildert's tenth problem is concerned, we list below some structures in which it is proven to be unsolvable.

1. In $\mathbb{Z}$ with the language $L = \{+, \cdot, 0, 1\}$. (the original problem)

2. In $\mathbb{F}_q(t)$ (rational functions in variable t with coefficients in a finite field with a positive characteristic other than 2) HTP is unsolvable (see [12]). We also have unsolvability for characteristic 2 (see [17]).

3. In all function fields where the constant field has characteristic greater than 2 in $L = \{0, 1, +, \cdot\}$ (see [8]).

4. In $F[[t]]$, the ring of power series with coefficients in F, which is an integral domain of positive characteristic, in $L = \{+, \cdot, 0, 1, t, P\}$ where $P(x) \leftrightarrow$"x is a power of t"
(see [11]).

5. In all quadratic rings, i.e. integral domains that contain the quadratic integers of quadratic fields (extensions of $\mathbb{Q}$ of degree 2), in the language of rings $\{+, \cdot, 0, 1\}$ (see [5]).

6. In polynomial rings $R[T]$, where R is an integral domain of characteristic zero, we have unsolvability concerning equations with solutions in $R[T]$ and coefficients in $\mathbb{Z}[T]$ (see [6]).

There are numerous rings in which the problem of decidability remains open, but the most famous of them is the ring of rationals. The board below contains results regarding the decidability of the full theory and existential theory of some rings, with "Y" to mean decidable, "N" to mean undecidable and "-" to mean that it is an open problem. The language is $L = \{+, \cdot, 0, 1\}$ and in case of rings of functions with one parameter, it is augmented with the predicate T which expresses those elements of the ring that are not constants.

	exist. theory in L	full theory
$\mathbb{C}$	Y	Y
$\mathbb{Q}$	-	N
$\mathbb{Z}$	N	N
$\mathbb{R}(z)$	-	N
$\mathbb{F}_q(z)$	-	N
$\mathbb{F}_q[z]$, $\mathbb{R}[z]$	N	N
$\mathbb{R}[[z]]$	Y	Y
$\mathbb{F}_q[[z]]$	Y	-

Below we are going to analyze some examples of the Diophantine problem for rings other than the integers, in order to see the techniques being used to prove their undecidability.

3.2 Hilbert's tenth problem for polynomial rings

Consider an integral domain R of characteristic zero that contains $\mathbb{Z}$. We will prove the following theorem (see [6]):

Theorem 3.2.1: The Diophantine problem for $R[T]$ with coefficients in $\mathbb{Z}[T]$ is unsolvable.

To proceed to the solution we will use the Pell equation

$$X^2 - (T^2 - 1)Y^2 = 1$$

where $X, Y \in R[T]$.
Consider U an algebraic element over $R[T]$ for which we have $U^2 = T^2 - 1$. Notice that $U \notin R[T]$ if R is an integral domain but it is the root of $x^2 - (T^2 - 1) \in R[T][X]$.
We define the sequences X_n, Y_n where $n \in \mathbb{Z}$ and $X_n + UY_n = (T + U)^n$.

Example 3.2.1:

$$X_1 + UY_1 = T + U \Leftrightarrow X_1 = T, Y_1 = 1$$

$$X_2 + UY_2 = (T+U)^2 = 2T^2 - 1 + 2TU \Leftrightarrow X_2 = 2T^2 - 1, Y_2 = 2T$$

$$\vdots$$

$$X_i = TX_{i-1} + (T^2-1)Y_{i-1}, Y_i = TY_{i-1} + X_{i-1}$$

Hence X_n, Y_n are defined recursively and, since $X_1, Y_1 \in R[T]$, we have that $X_i, Y_i \in R[T]$ for every $i \in \mathbb{Z}$.

Lemma 3.2.1: Let $X, Y \in R[T]$. Then X, Y form a solution of the Pell equation if and only if $X = \pm X_n$ and $Y = \pm Y_n$ for some $n \in \mathbb{Z}$

Proof. See page 4 of [6]. □

For two polynomials $P, N \in R[T]$ we define the relation $P \sim N$ if $P = N$ when $T = 1$. In other words, for $P, N \in R[T]$ we have

$$P \sim N \longleftrightarrow \exists X \in R[T] : P - N = (T-1)X$$

For instance, if $P, N \in \mathbb{Z}[T]$ and $P = T - 1$, $N = (T-1)^2(T+1)$ then $P = N = 0$ when $T = 1$, so $P \sim N$.
Now if $P = T^2 - 1$ and $N = T + 1$, then for $T = 1$: $P = 0$ and $N = 2$, hence $P \nsim N$

Lemma 3.2.2: We have that $Y_n \sim n$ for $n \in \mathbb{N}$.

Proof. From the relation $X_n + UY_n = (T+U)^n$ we have that

$$Y_n = \sum_{\substack{i=1 \\ i=odd}}^{n} \binom{n}{i} (T^2-1)^{\frac{i-1}{2}} T^{n-i}$$

So when $T = 1$ we have that $Y_n = n$ □

We now define the relation $Imt(Y)$ to be:
$Y \in R[T] \wedge \exists X \in R[T] : X^2 - U^2Y^2 = 1$

Lemma 3.2.3:

1. $Imt(Y)$ is a Diophantine relation over $R[T]$ with coefficients in $\mathbb{Z}[T]$.

2. If $Imt(Y)$ is satisfied for some Y then there exists an integer n such that $Y \sim n$.

3. For every integer n there exists a polynomial $Y \in R[T]$ such that $Y \sim n$ and it satisfies $Imt(Y)$

Proof. 1. By the definition of $Imt(Y)$ we have that $\exists X \in R[T] : P(X,Y) = 0$ with $P(x,y) = x^2 - U^2y^2 - 1$ and $1, U^2 \in \mathbb{Z}[T]$

2. If $Imt(Y)$ is satisfied, for $Y \in R[T]$, then $\exists X \in R[T]$ such that $X^2 - U^2Y^2 = 1$ so from **Lemma 3.2.1** we have $X = X_n, Y = Y_n$ for an integer n and from **Lemma 3.2.2** we have that $Y \sim n$

3. From **Lemma 3.2.2** we have $Y_m \sim m$ where m is a natural number. We have $X_m + UY_m = (T + U)^m$ and $(X_m + UY_m)(X_{-m} + UY_{-m}) = 1 = X_m^2 - U^2Y_m^2$, so $X_{-m} = X_m$ and $Y_{-m} = -Y_m$. So if $Y_m \sim m$ then $Y_{-m} \sim -m$, hence proved for every integer.
Consider an integer m, there exists a polynomial $Y = Y_m \sim m$ that along with another polynomial $X = X_m$ form the solution of $X^2 - (T^2 - 1)Y^2 = 1$, so $Imt(Y)$ is satisfied.

□

Fact: We have that, since R is an integral domain, if two sets S_1, S_2 are Diophantine over $R[T]$ with coefficients in $\mathbb{Z}[T]$, then the sets $S_1 \wedge S_2$ and $S_1 \vee S_2$ are also Diophantine over $R[T]$ with coefficients in $\mathbb{Z}[T]$.
To make this clear, consider P_1, P_2 the polynomials of S_1, S_2 then

$$P_1 = 0 \vee P_2 = 0 \longleftrightarrow P_1P_2 = 0$$

$$P_1 = 0 \wedge P_2 = 0 \longleftrightarrow P_1^2 + P_2^2 = 0$$

If R was not an integral domain the above equivalences would not hold.

From **Lemma 3.2.3** we have that

$$z \in \mathbb{Z} \longleftrightarrow \exists Y \in R[T] : Y \sim z \bigwedge Imt(Y)$$

$$z \in \mathbb{Z} \longleftrightarrow \exists Y, X, X' \in R[T] : Y-(T-1)X' = z \bigwedge X^2-(T^2-1)Y^2 = 1$$

$$z \in \mathbb{Z} \longleftrightarrow \exists Y, X, X' \in R[T] : (Y-(T-1)X'-z)^2+(X^2-(T^2-1)Y^2-1)^2 = 0$$

Hence we proved that the set of the integers is Diophantine over $R[T]$ with coefficients in $\mathbb{Z}[T]$.
This result along with the unsolvability of the Diophantine problem for $\mathbb{Z}$ prove the **Theorem 3.2.1**.
This happens because we can find a procedure which turns the relation

$$\exists z_1, z_2, \ldots, z_n \in \mathbb{Z} : P(z_1, z_2, \ldots, z_n) = 0$$

into the relation

$$\exists Z_1, \ldots, Z_n \in R[T] : P^*(Z_1, \ldots, Z_n) = 0$$

All the struggle is to find a polynomial P^* starting from P. We are going to use the above relations for this purpose.
For each z_i we have that

$$z_i \in \mathbb{Z} \longleftrightarrow \exists Z_i \in R[T] : Z_i \sim z_i \bigwedge Imt(Z_i)$$

And since $\exists X_i \in R[T] : Z_i - (T-1)X_i = z_i$ then the expression

$$P(z_1, \ldots, z_n) = 0$$

turns into

$$P(Z_1 - (T-1)X_1, \ldots, Z_n - (T-1)X_n) = 0$$

Hence for $T = 1 : P(Z_1, \ldots, Z_n) = 0 \longleftrightarrow P(Z_1, \ldots, Z_n) \sim 0$.
So we have that the relation

$$\exists z_1, z_2, \ldots, z_n \in \mathbb{Z} : P(z_1, z_2, \ldots, z_n) = 0$$

is equivalent to

$$\exists Z_1, \ldots, Z_n \in R[T] : Imt(Z_1) \wedge \cdots \wedge Imt(Z_n) \wedge P(Z_1, \ldots, Z_n) \sim 0$$

Since the relations $\sim$ and Imt are Diophantine over $R[T]$ with coefficients in $\mathbb{Z}[T]$ we can obtain P^*.

In the above procedure we started with an arbitrary Diophantine equation in the integers and obtained an equivalent equation in $R[T]$

with coefficients in $\mathbb{Z}[T]$.
Hence if we assume that the Diophantine problem for $R[T]$ with coefficients in $\mathbb{Z}[T]$ is solvable, it will follow that it is solvable for the integers as well. But we proved in chapter 2 that it is not true.
So **Theorem 3.2.1** is proved.

3.3 Hilbert's tenth problem for power series

In this section we will deal with Hilbert's tenth problem concerning the ring of power series over an integral domain F, with
$char(F) = p > 0$ and an indeterminate t, denoted $F[[t]]$

$$a \in F[[t]] \Leftrightarrow a = \sum_{i=1}^{N} a_i t^i$$

where $a_i \in F$. The language we will use is $L = \{0, 1, +, \cdot, P, t\}$ where P is the predicate for the powers of t,
i.e. for $x \in F[[t]]$

$$P(x) \longleftrightarrow \exists n \in \mathbb{N} : x = t^n$$

We also consider K as the quotient field of F and $K((t))$ the ring of Laurent series with coefficients in K. The answer to Hilbert's tenth problem for this ring is again negative and it is proven explicitly in [11], we will examine the proof of the answer briefly below.

First we are going to reduce the existential problem for $F[[t]]$ in L to the Diophantine problem for $\mathbb{N}$ in $L_1 = \{+, |_p, 0, 1\}$ where for $n, m \in \mathbb{N}$

$$n|_p m \longleftrightarrow \exists y \in \mathbb{N} : m = p^y n$$

Then the latter will be proved to be equivalent to the Diophantine problem for $\mathbb{N}$ in $L_2 = \{0, 1, +, \cdot\}$.

For the first part we have the following lemma:
Lemma 3.3.1:

1. For $m, n \in \mathbb{N}$ we have $n|_p m$ if and only if $\exists a \in K((t))$ such that $t^{-m} - t^{-n} = a^p - a$.

2. The Diophantine problem for $F[[t]]$ in L can be reduced to the Diophantine problem for $\mathbb{N}$ in L_1.

Proof. 1. The proof is number theoretical an out of the scope of the thesis. You can find it in [11].

2. For this we need to give a Diophantine definition of the operations $+, |_p$ between natural numbers over $F[[t]]$. Above we have such a definition of $|_p$ over $K((t))$. Observe that for any $a \in K((t))$ there exists $b \in F[[t]]$ such that $ab \in F[[t]]$.
To understand this better consider

$$a = a_{-2}t^{-2} + a_{-1}t^{-1} + a_0 + a_1 t + a_2 t^2 \in K((t))$$

where $a_i \in K$, i.e. $\exists a_{i1}, a_{i2} \in F$ such that $a_i = \frac{a_{i1}}{a_{i2}}$ for any $i \in \{-2, -1, 0, 1, 2\}$. Therefore $a_{i2}a_i = a_{i1} \in F$. At last we see that

$$t^2 a_{-22} a_{-12} a_{02} a_{12} a_{22} a \in F[[t]]$$

Therefore we have a Diophantine definition of $a \in K((t))$ over $F[[t]]$ and that is

$$a \in K((t)) \longleftrightarrow \exists b, c \in F[[t]] : ab = c$$

It follows that we can obtain a Diophantine definition of the operations $+, |_p$ over $F[[t]]$ as well.
We have that for $m, n, l \in \mathbb{N}$

$$n|_p m \longleftrightarrow \exists b, c, d \in F[[t]] : b^p(t^n - t^m) = c^p - cb^{p-1} \wedge c \neq 0 \wedge t^m = dt^n$$

$$m + n = l \longleftrightarrow t^l = t^m t^n$$

□

Therefore if the Diophantine problem for $F[[t]]$ in L is solvable, so is the Diophantine problem for $\mathbb{N}$ in L_1.

Now we proceed to the second step in which we reduce the latter to the Diophantine problem for $\mathbb{N}$ in $L_2 = \{0, 1, +, \cdot\}$.
This is proven by finding a Diophantine definition of multiplication

over $\mathbb{N}$ in the language L_1.
The proofs below are purely from number theory and will be omitted. They can be found in [11].

Lemma 3.3.2:

1. Given any $m, n, s \in \mathbb{N}$ then $m = p^s n$ if and only if $n|_p m$ and $n + 1|_p m + p^s$ and $n + p|_p m + p^{s+1}$.

2. Given any $m, n, p^s \in \mathbb{N}$ the relation $m = p^s n$ is Diophantine over $\mathbb{N}$ in the language L_1.

Proposition 3.3.1:

1. Let $m, n \in \mathbb{N}$ then $n|m$ if and only if $p^n - 1|p^m - 1$

2. Let $m, n \in \mathbb{N}$ and $m \geq 1$ then
$(p^{mn} - 1)/(p^m - 1) \equiv n mod(p^m - 1)$

Lemma 3.3.3: For $n, m \in \mathbb{N}$ then $m = n^2$ if and only if
$\exists r, s \in \mathbb{N} : p^{2s} - 1|p^r - 1 \bigwedge (p^r - 1)/(p^{2s} - 1) \equiv n mod(p^{2s} - 1) \bigwedge$
$n < p^s - 1 \bigwedge ((p^r - 1)/(p^{2s} - 1))^2 \equiv m mod(p^{2s} - 1) \bigwedge m < p^{2s} - 1.$

Therefore we have seen that the relations $m = p^s n$ and $m = n^2$ are Diophantine over $\mathbb{N}$ in the language L_1. Hence we can prove that for $n, m, k \in \mathbb{N}$ the relation $m = nk$ is Diophantine in $\mathbb{N}$ with L_1.

Proof. For $n, m, k \in \mathbb{N}$ we have that $m = nk$ if and only if
$(n + k)^2 = n^2 + 2m + k^2$ and from **Lemma 3.3.3** we have that $m = n^2$ is Diophantine over $\mathbb{N}$ in L_1, hence the relation $m = nk$ is Diophantine over $\mathbb{N}$ in L_1. □

Hence the initial Diophantine problem for $F[[t]]$ with $L = \{0, 1, +, \cdot, t, P\}$ can be reduced to the problem for $\mathbb{N}$ in $L_2 = \{0, 1, +, \cdot\}$.
The latter can be shown to be unsolvable as an immediate consequence of Hilbert's tenth problem for the integers.
That is because, we can prove that the integers are Diophantine over

$\mathbb{N}$ in $L_2 = \{0, 1, +, \cdot\}$.
We have that, if

$$k \in \mathbb{Z} \longleftrightarrow \exists m, l \in \mathbb{N} : k + m = l$$

Hence, the Diophantine for $\mathbb{N}$ with L_2 can be reduced to the Diophantine problem for $\mathbb{Z}$ in the same language. Assuming that the latter has a solution, we come to the conclusion that the former also has a solution, which is known to be proven false. Hence, the Diophantine problem for $F[[t]]$ in L is unsolvable.

Bibliography

[1] David A.Plaisted. *1 Universal Turing Machines*. Mar. 2021.

[2] Anatoly Petrovich Bel'tyukov. "Decidability of the universal theory of natural numbers with addition and divisibility". In: *Zapiski Nauchnykh Seminarov POMI* 60 (1976), pp. 15–28.

[3] Martin Davis. "Hilbert's tenth problem is unsolvable". In: *The American Mathematical Monthly* 80.3 (1973), pp. 233–269.

[4] Martin Davis, Yuri Matijasevič, and Julia Robinson. "Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution". In: *American Math. Soc Providence* 1 (1976).

[5] Jan Denef. "Hilbert's tenth problem for quadratic rings". In: *Proceedings of the American Mathematical Society* 48.1 (1975), pp. 214–220.

[6] Jan Denef. "The Diophantine problem for polynomial rings and fields of rational functions". In: *Transactions of the American Mathematical Society* 242 (1978), pp. 391–399.

[7] Lou van den Dries. "Mathematical Logic". In: *Lecture Notes* (2016).

[8] Kirsten Eisenträger and Alexandra Shlapentokh. "Undecidability in function fields of positive characteristic". In: *International Mathematics Research Notices* 2009.21 (2009), pp. 4051–4086.

[9] Kurt Gödel. *On formally undecidable propositions of Principia Mathematica and related systems*. Courier Corporation, 1992.

[10] Christos H Papadimitriou. "Computational complexity". In: *Encyclopedia of computer science*. 2003, pp. 260–265.

[11] Thanases Pheidas. "An undecidability result for power series rings of positive characteristic. II". In: *Proceedings of the American Mathematical Society* 100.3 (1987), pp. 526–530.

[12] Thanases Pheidas. "Hilbert's tenth problem for fields of rational functions over finite fields". In: *Inventiones mathematicae* 103.1 (1991), pp. 1–8.

[13] Thanases Pheidas and Karim Zahidi. "Decision problems in Algebra and analogues of Hilbert's tenth problem". In: *LONDON MATHEMATICAL SOCIETY LECTURE NOTE SERIES* 350 (2008), p. 207.

[14] Helmut Prodinger. "On binary representations of integers with digits- 1, 0, 1". In: *Integers* (2000), A8–14.

[15] Michael Sipser. "Introduction to the Theory of Computation". In: *ACM Sigact News* 27.1 (1996), pp. 27–29.

[16] Alfred Tarski. "A decision method for elementary algebra and geometry". In: *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 1951, pp. 24–84.

[17] Carlos R Videla. "Hilbert's tenth problem for rational function fields in characteristic 2". In: *Proceedings of the American Mathematical Society* 120.1 (1994), pp. 249–253.