Hilbert's Tenth Problem

Krishna Praneeth Sidde, Saee Patil

December 12, 2022

Abstract

In this paper, we provide an exposition on Hilbert's Tenth Problem with an intent to make it easier for a non-expert to understand. We will show that the Hilbert's Tenth Problem is unsolvable as proved by the *MRDP (Matiyasevich-Robinson-Davis-Putnam) Theorem.* Along the way, we introduce concepts from Number Theory such as Diophantine Sets, and Computability Theory as it provides the nomenclature and setting for proving that The Hilbert's Tenth Problem is unsolvable.l

1 Introduction

In the year 1900 at a conference in Paris, German mathematician David Hilbert presented 23 problems that to some extent set the research agenda for 20^{th} century mathematics. Some of the problems like the Riemann Hypothesis, still remain unsolved. We investigate the tenth problem in this paper, the original statement of which is as follows:

"Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers."

Let us break down the statement and see what each part means.

Definition 1.1 (Diophantine equation)**.** A *Diophantine equation* is a polynomial equation with (possibly) multiple variables where we are only interested in the integral solutions.

For degree one equations (i.e. equations of the form $\sum a_i x_i = b$, $a_i, b \in \mathbb{Z}$), there is a simple process which can tell us whether it is solvable or not which involves the greatest common divisor of a_i.

Lemma 1.2. *$\sum a_i x_i = b$ has integral solutions x_i if and only if $gcd(a_i) \mid b$.*

Further, the term *rational integer* simply refers to the set of integers (positive, negative and zero). Hilbert did not rigorously define what he meant by 'process', which was done by Turing in his influential paper in '36. The problem was finally proved to be unsolvable in 1970, using the combined work of 4 mathematicians which spanned a duration of 21 years - Yuri Matiyasevich, Julia Robinson, Martin Davis, and Hilary Putnam.

The proof involves certain sets known as the Diophantine sets and *languages* which can be computed/recognized by a computer.

Definition 1.3 (Language)**.** A language $\mathcal{L}$ is a set of finite strings with entries in a given set of alphabet.

Example. The language of even integers with alphabet $\{0, 1\}$: $\{10, 100, 110, 1000, \cdots\}$

Definition 1.4 (Diophantine Set)**.** A set $S \subseteq \mathbb{N}^n$ is Diophantine if there is a polynomial $P(x_1, x_2, \cdots, x_{m+n}) \in \mathbb{Z}[x_1, \cdots, x_{m+n}]$, $m \geq n$ such that $(s_1, s_2, \cdots, s_n) \in S$ if and only if $\exists\, t_1, t_2, \cdots, t_m \in \mathbb{N}$ such that $P(s_1, \cdots, s_n, t_1, \cdots, t_m) = 0$.

Example (1). The set of even numbers: $s_1 \in S \iff \exists t_1 : P(s_1, t_1) := s_1 - 2t_1 = 0$.

Example (2). The set of non-powers of 2: $s_1 \in S \iff \exists (t_1, t_2) : P(s_1, t_1, t_2) := s_1 - t_1(2t_2 + 1) = 0$.

Example (3). As we will see, the set of all primes and the set of all 3-tuples $\langle a, b, c\rangle$ such that $c = a^b$ are Diophantine. These are obviously harder to prove.

1.1 History

The story which started off with Hilbert was carried on by Martin Davis. He proved a significant result in 1940s, and conjectured the MRDP theorem (proved in section 4) as we know it now. In the meanwhile, Julia Robinson - unaware of Davis's work, attempts to prove that the exponential function is Diophantine. Unable to do so, she comes up with the Julia Robinson hypothesis about something slightly less general. Together with Davis and Putnam, it is shown that the hypothesis, if true, will imply the unsolvability of Hilbert's tenth problem. However, the hypothesis remains unproven for a long time. In 1970, inspired by the work of Nikolai Vorob'ev on Fibonacci numbers, Matiyasevich finally proves the hypothesis, completing the proof.

2 Everything Diophantine

Throughout this paper, we consider variables to be positive integers unless mentioned otherwise. We try to give proofs whenever possible and leave the reader with an outline where the algebra is tedious.

Lemma 2.1. *If there is no algorithm for finding positive integer solutions of a Diophantine equation, the same holds true for integers. i.e. it is enough to show the unsolvability of Diophantine equations in* $\mathbb{Z}^+$.

Proof. By Lagrange's 4-square theorem, we know that every positive integer x can be written as a sum of 4 squares $a^2 + b^2 + c^2 + d^2$. Thus, the equation

$$P(x_1, x_2, \cdots, x_n)$$

has solutions in $\mathbb{Z}^+$ if and only if

$$P(a_1^2 + b_1^2 + c_1^2 + d_1^2, \cdots, a_n^2 + b_n^2 + c_n^2 + d_n^2)$$

has solutions in $\mathbb{Z}$. ■

Let A and B be Diophantine sets with corresponding polynomials P and Q. Observe that $P^2 + Q^2 = 0$ iff $P = 0$ and $Q = 0$. Thus, we conclude that $A \cap B$ is also Diophantine with the corresponding polynomial $P^2 + Q^2$. This can further be generalized to more than 2 sets.

Definition 2.2. A function f with n inputs is Diophantine if $\{\langle a_1, \cdots, a_n, b\rangle \mid b = f(a_1, \cdots, a_n)\}$ is a Diophantine set. i.e. a function if Diophantine if it's graph is a Diophantine set.

Theorem 2.3 (Pairing Function Theorem)**.** *There are Diophantine functions P, L and R such that $L(P(x, y)) = x, R(P(x, y)) = y, P(L(z), R(z)) = z$. Further, $L(z), R(z) \leq z$.*

Proof. We can simply consider the Cantor Pairing function. ■

Corollary 2.4. $\mathbb{N}^2$ *is countable.*

Theorem 2.5 (Sequence Number Theorem)**.** *There is a Diophantine function $S(i, u)$ such that $S(i, u) \leq u$ and for every sequence $a_1, \cdots, a_N$, $\exists u : S(i, u) = a_i \ \forall 1 \leq i \leq N$. i.e. every sequence can be encoded as a single number $u \in \mathbb{N}$.*

Proof. Define $S(i, u)$ to be the remainder when $L(u)$ is divided by $1 + i \cdot R(u)$. This function is Diophantine since $w = S(i, u)$ iff there exist solutions to x, y which satisfy $u = P(x, y)$, $x = w + z(1 + iy)$, and $1 + iy = w + v - 1$. Let $y > a_i \ \ \forall i \in \{1, 2, \cdots, N\}$ such that $i \mid y$. Notice that the numbers $1 + y, 1 + 2y, \cdots, 1 + Ny$ are mutually co-prime. Using the Chinese Remainder theorem, we find an x so that $x = a_j \bmod 1 + jy$ for $j \in \{1, 2, \cdots, n\}$. Finally, set $u = P(x, y)$ to have the desired result. ■

Theorem 2.6. *If P is a polynomial, then the sets with bounded quantifiers are Diophantine. i.e.*

$$\{\langle y, x_1, x_2, \cdots, x_n\rangle \mid (\exists z)_{\leq y}(\exists y_1, \cdots, y_m)[P(y, z, x_1, \cdots, x_n, y_1, \cdots, y_m) = 0]\}$$

and

$$\{\langle y, x_1, x_2, \cdots, x_n\rangle \mid (\forall z)_{\leq y}(\exists y_1, \cdots, y_m)[P(y, z, x_1, \cdots, x_n, y_1, \cdots, y_m) = 0]\}$$

are Diophantine.

Proof. The first part is trivial - we can add an extra condition which forces z to be $\leq y$. The second part is slightly more complicated which we will skip entirely. ■

This result along with the Sequence Number theorem is a powerful tool and allows us to show the Diophantine-ness of a lot of of sets and functions. For example,

Corollary 2.7. *The set $\{p \mid p$ is a prime $\in \mathbb{N}\}$ is Diophantine.*

Proof. x is a prime $\iff x > 1$, and $\forall(y, z)_{\leq x}[yz < x \vee yz > x \vee y = 1 \vee z = 1]$ ■

Corollary 2.8. *The function $g(y) = \prod_{k=1}^{y}(1 + k^2)$ is Diophantine.*

Proof. Use the Sequence Number theorem to 'encode' $g(1), \cdots, g(y)$ into a single number u and use it to come up with the polynomial. ■

3 Computability Theory

While not going into too much detail about Computability Theory, we will go through a generalized idea of Turing Machines in this section and establish a connection between them and Diophantine Equations. Eventually, we will look at Hilbert's Tenth Problem from the point of view of Computability Theory.

A Turing Machine M has memory in the form of a *tape* divided into *cells*. The tape has a single end, on the left, and is potentially infinite to the right. Each cell will either be empty or will contain a single *symbol* from a finite set of symbols Σ called an *alphabet*. An empty cell on this tape is denoted using the symbol Λ. Symbols on the tape are read and written by a *head*, which scans each one of the cells as it moves along the tape from left to the right. At each moment, M is in one of the finitely many states $Q = \{q_1, q_2, \cdots, q_v\}$ where q_1 is declared to be the *initial state*. $q_f \in Q$ is said to be the *accepting state* and $q_r \in Q \backslash \{q_f\}$ is said to be the *rejecting state*. There are one or more such states in a Turing Machine declared to be *final*. In a single step, the current state and the symbol is scanned by the head. Then, the machine can either change the symbol in the cell, move the head one cell to the left, or to the right. The head then passes onto another state. These actions performed by the head are defined by a set of *instructions*. While performing a computation, the work will stop if a *final state* is achieved.

Definition 3.1. [Mat] A *Turing machine* is an octuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \Lambda, q_f, q_r),$$

where

- Q is the set of states,
- Σ is the input alphabet,
- Γ is a finite set of symbols called the *tape alphabet*,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $\Lambda \in \Gamma$ is a special symbol called the *blank*,
- $q_f \in Q$ is the accepting state, and
- $q_r \in Q \setminus \{q_f\}$ is the rejecting state.

Definition 3.2. A Language L is said to be *recursively enumerable* if there exists a Turing machine M which accepts the language L. For a recursively enumerable language, a Turing Machine may or may not enter into rejecting state for string which are not part of L.

If there exists a Turing machine that **halts** on all of the inputs for L, then L is said to be *recursive*. For recursive languages, the Turing Machine will always halt since it rejects strings which are not a part of L.

For the purposes of the present paper however, we will restrict our definition of recursive languages only to a subset $S \subseteq \mathbb{N}$. We can say that S is recursively enumerable if a Turing Machine M accepts $n \in S$ and will terminate after a finite amount of time. If the Turing Machine M accepts $n \in S$, and will halt for all n irrespective of whether they are in S, S is recursive.

Examples of recursively enumerable sets:

- Any finite set: $S = \{s_1, s_2, \cdots, s_n\} = \{s | s = s_1 \vee s = s_1 \cdots \vee s = s_n\}$.
- The set of even numbers: $S = \{s | \exists y : s = 2y\}$.
- The set of prime numbers: $S = \{s | \neg(\exists y)_{1<y<s}(\exists z)_{1<z<s} : s = yz\}$.

A variant of Turing machines that helps us determine recursively enumerable languages are *enumerators*. An enumerator is a Turing machine that is attached to a printer, and it can output one or more characters to the printer at any time during the computation. It starts with a blank input tape, and it will print a (possibly infinite) list of strings on the printer. The language of an enumerator E is the collection of strings that it eventually prints. The strings it outputs are allowed to occur in any order, and strings printed out may be repeated. We will now understand what an algorithm exactly means. An informal description of an algorithm is that it is a collection of basic instruction for carrying out a task, however, using Turing Machines, we can define an algorithm to be a process that can be implemented on a Turing Machine.

Theorem 3.3. *[Ho15] A set S is* recursively enumerable *if there exists an algorithm that enumerates S.*

Proof. Take a set $S \subset \mathbb{N}$. First suppose that there exists an algorithm that is guaranteed to terminate on inputs contained in S and run infinitely for inputs not contained in S. Let this algorithm be denoted $A(n)$ where n is its input.

To show that there exists an algorithm for enumerating the members of S, consider the following construction: Run $A(0)$ for one time step, then $A(0), A(1)$ for one time step, then run $A(0), A(1), A(2)$ for one time step, and so on and so forth. This described algorithm will "eventually" reach arbitrarily large timestamps for $A(n)$ given any choice of n, and so for all $n \in S$ it is guaranteed to confirm that $n \in S$ in a finite span of time. Modify $A(n)$ to print n if it halts, and we have the desired enumeration.

Conversely we shall suppose that there exists an algorithm that enumerates S; call it A. To construct an algorithm $B(n)$ that halts only if $n \in S$, simply run A and halt if n is printed. ■

From the above theorem, we've precisely proved that *A language is recursively enumerable if and only if some enumerator enumerates it.* We will now introduce certain operations that can be performed on languages to produce new languages.

Definition 3.4. Let Σ be an alphabet, and let A and B be languages on Σ.

- The *union* of A and B is the language $A \cup B = \{x : x \in A \text{ or } x \in B\}$.

- The *intersection* of A and B is the language $A \cap B = \{x : x \in A \text{ and } x \in B\}$.
- The *concatenation* of A and B is the language $A \circ B = \{xy : x \in A \text{ and } y \in B\}$ where xy is the string that starts with x and is followed by y.
- The *complement* of A is the language $\bar{A} = \{x : x \notin A\}$.

We will not prove this but it is quite straightforward to see that unions and intersections of recursively enumerable sets are also recursively enumerable. We now have the nomenclature set up to state and understand the MRDP Theorem, which considers the set of Diophantine sets with the set of recursively enumerable sets.

4 The MRDP Theorem

Suppose that there does exist an algorithm capable of deciding the solvability of arbitrary Diophantine equations. This algorithm would be capable of deciding, in a finite amount of time, whether or not there exists a exists in the Diophantine set S such that $D(a, x_1, \cdots, x_m) = 0$ for some m-tuple $(x_n, \cdots, x_m)$. The existence of such an algorithm would mean that every Diophantine set is recursive. However, the MRDP theorem asserts that every set is Diophantine if and only if it is recursively enumerable, so the algorithm would imply that all recursively enumerable sets are also recursive, which is untrue. This contradiction, as a result of the MRDP Theorem, yields a negative answer to Hilbert's Tenth Problem and shows that Hilbert's Tenth Problem is unsolvable. In this section, we will therefore present the celebrated Matiyasevich-Robinson-Davis-Putnam (MRDP) Theorem.

Theorem 4.1. *(Matiyasevich-Robinson-Davis-Putnam) A set is Diophantine if and only if it is recursively enumerable.*

4.1 Proof of the MRDP Theorem

We will prove the MRDP Theorem through the following lemmas. The initial direction of the MRDP theorem would be to show that Diophantine Sets are recursively enumerable.

Lemma 4.2. *[Bow14] If a set is Diophantine, then it is recursively enumerable.*

Proof. Let $S \subset \mathbb{Z}^n$ be a Diophantine set, with the corresponding polynomial $p \in \mathbb{Z}[X_1, X_2, \cdots, X_m]$ and $n \leq m$. Let $s \in \mathbb{Z}^n$ be arbitrary. Then an algorithm that systematically checks all elements of $\mathbb{Z}^{m-n}$ (for instance by ordering on the absolute value of the sum of the coordinates) will suffice. If s is in S, then, by definition of S, $p(s, t) = 0$ for some $t \in \mathbb{Z}^{m-n}$. We know that this t will be found by our algorithm after a finite amount of time, and therefore the algorithm will halt. On the other hand, when s is not in S, no such $t \in \mathbb{Z}^{m-n}$ can be found, and thus our algorithm will run forever. As this algorithm halts precisely for elements in S, we conclude that S is recursively enumerable. Moreover, S was chosen arbitrarily, so we conclude that every Diophantine set is recursively enumerable. ■

The other way round of MRDP Theorem, to show that all recursively enumerable sets are Diophantine is quite complicated. It is followed from two major results: First, Davis, Putnam, and Robinson showed in 1961 that every recursively enumerable set is exponential Diophantine. Second, Matiyasevisch showed in 1970 that sets which are exponential Diophantine are also Diophantine (proved in §2).

Lemma 4.3. *[DPR61] (The DPR-theorem) Every recursively enumerable set is exponential Diophantine.*

The proof of this theorem is intricate and contains quite a lot of analysis of algorithms and the Turing Machine. An outline of this proof can be found in [DPR61].

Lemma 4.4. *[Dav73] The exponential function $f(n,k) = n^k$ is Diophantine.*

The entire proof of this is tedious and uses a lot of algebra. However, we give an outline here which provides sufficient intuition.

Proof. The original proof by Matiyasevich describes the Fibonacci numbers using Diophantine equations and uses their exponential like growth to complete this proof. We will take a slightly different approach; consider a special case of the Pell's equation $x^2 - Dy^2 = 1$ where $D = a^2 - 1$ for some $a \in \mathbb{N}$. All the solutions of this can be given by Lucas sequences x_a and y_a:

$$x_a(0) = 1, x_a(1) = a, x_a(n+2) = 2ax_a(n+1) - x_a(n)$$
$$y_a(0) = 0, y_a(1) = 1, y_a(n+2) = 2ay_a(n+1) - y_a(n)$$

We come up with a sequence of Diophantine equations in a, n which is solvable iff $g(a,n) = x_a(n)$ or $h(a,n) = y_a(n)$. Notice that the roots exhibit almost exponential like growth. We now use the idea if two positive integers m and n^k are less than and congruent modulo a third integer, they have to be equal. ■

These two results show that all recursively enumerable sets are Diophantine and the MRDP Theorem follows as a directly corollary from 4.2, 4.3, and 4.4.

4.2 Consequences of the MRDP Theorem

We list (but do not prove) some interesting results relevant to the MRDP Theorem.

Theorem 4.5. *For a given axiom system, there exists some Diophantine equation which is unsolvable in integers; and that the it is unprovable within the system.*

Theorem 4.6. *There is a polynomial whose range is the set of primes.*

Proof Idea. The set of primes is recursively enumerable, and hence Diophantine. ■

Definition 4.7 (Hilbert's 10th problem over a ring R)**.** This problem asks if a given equation with integer coefficients has solutions in a ring R.

Setting $R = \mathbb{Q}$ in the above definition remains to be an open problem.

Theorem 4.8. *The problem for $R = \mathbb{Z}[i]$ is the same as it is for $\mathbb{Z}$. i.e., a Diophantine equation has solutions in the Gaussian integers iff it has solutions in $\mathbb{Z}$.*

References

[Bow14] Nicole Bowen. *Hilbert's Tenth Problem*. PhD thesis, University of Connecticut, 2014.

[Dav73] Martin Davis. Hilbert's tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, 1973.

[DPR61] Martin D. Davis, Hilary Putnam, and Julia Jean Robinson. The decision problem for exponential diophantine equations. *Annals of Mathematics*, 74:425, 1961.

[Ho15] Andrew J Ho. Hilbert's tenth problem, 2015.

[Mat] IUriĭ Matiiasevich. *Hilbert's tenth problem*.