Hilbert's 10th Problem

Brian Wu

Abstract

A solution to Hilbert's 10th problem is described here, largely following [1]. This is done through the use of Diophantine sets, Diophantine functions, and recursive(μ-recursive) functions. Hilbert's 10th problem asks for an algorithm to check whether a given Diophantine equation is solvable. However, Matiyasevich proved that this was impossible, that no such algorithm could exist. A solution, not the original, is discussed here.

Contents

1 Introduction

The ancient Greeks were really into math. This is a pretty supported statement, and it helps to remember that before the advent of science, math was studied for its own sake. In the time of the ancient Greeks this was certainly the case, there was no science as we think of it. No dynamics or electromagnetism or ideal gas or anything of the sort. Math was studied for the sake of math, without all the applications it's more like an elaborate game and this was certainly one of the perspectives taken. Now in this time there was this mathematician called Diophantus who we know very little about. In fact, one of the only things we know about him is that he lived until 84 years old, thanks to a riddle given about his age. Now aside from the fact that one of the only things we know of him comes from a math problem, something interesting about him is that he is the name sake of Diophantine equations, which continues to be an object of fascination well into the modern day.

Diophantine equations are just polynomial equations with integer coefficients(we only consider integer solutions). So for example an equation like $5x^3 + 7y^2 - 6z = 0$ is Diophantine but $5.3x^3 + 7y^2 - 6z = 0$ isn't since 5.3 is not an integer. Diophantine equations turns up everywhere, including in many puzzles, and some can be shockingly difficult to solve. In the 20th century, Hilbert created a list of 23 problems and among them was the question of whether there was some universal way to check whether a Diophantine equation has a solution. In 1970, the problem was solved in the negative in the doctoral paper of one Yuri Matiyasevich. One implication here being that some Diophantine equations are indeed really, really tough to solve. However, this also

guides some mathematical research. In particular, Diophantine equations are known for sometimes looking deceptively simple. For example, the equation $x^3 + y^3 + z^3 = 42$ was without a known (integer)solution for 65 years, and when the solution was found it still took over a million hours of computer time.
With that being said, the topic discussed here will be a very summarized version of the proof. It is probably important to first talk about what Hilbert's 10th problem actually is. So for Diophantine equations we only consider integer solutions, and what Hilbert's 10th problem asks is whether there exists some algorithm, so in other words a finite set of rules, where we can determine if any given Diophantine equation is solvable. This does not sound so impossible since it's still just another class of equations and people can be very clever with algorithms. However, as we now know, such an algorithm does not exist.

Let's think about one specific way to approach this problem. So the problem asks if there exists an algorithm to determine, for any given Diophantine equation, whether it has integer solutions. If we wanted to prove it in the negative, we just need to find a single counterexample. In other words, a single solvable Diophantine equation where there exists no such algorithm. In fact we can broaden this a bit more, we just need to find a single set of coefficients that is not recursive where there exists a solvable Diophantine equation with those coefficients. This idea of finding a set of coefficients that are not decidable but also the coefficients of a solvable Diophantine equation is that of Diophantine sets.

A Diophantine set is a set of integers $(a_1, a_2, \cdots, a_n)$ such that there exists some set of integers $(x_1, \cdots, x_n)$ such that there exists a polynomial $P(a_1, \cdots, a_n, x_1, \cdots, x_n)$ where $P(a_1, \cdots, a_n, x_1, \cdots, x_n) = 0$. In less formal terms, A Diophantine set is the set of coefficients of some solvable Diophantine equation. Now the way that Matiyasevich did it was he showed that the solutions to some Diophantine equations can grow exponentially, and earlier work in the field by Davis, Robinson, and Putnam showed that if it could grow exponentially, then every recursively enumerable set would be Diophantine. This is not however the avenue that we will take. Notice that there exists recursively enumerable sets that are not decidable, so you have a Diophantine set that is not recursive which implies that there cannot exist a universal algorithm, more details than this are of course supplied.

Here's a rough description of what we will be doing. We will define something called Diophantine functions whose graph has Diophantine sets in it(this will make more sense later when we define it). Then we will show that a function is Diophantine if and only if it is a recursive function. This is then used to prove that a set is Diophantine if and only if it is recursively enumerable. The solution to Hilbert's 10th problem is achieved from proof by contradiction using the result that a set is Diophantine if and only if it is recursively enumerable.

This discussion will be split into two halves. In the first half we will start building up the results and in the second we will actually prove the impossibility of Hilbert's 10th problem.

2 Preparation

In proofs it is often more easy to describe functions as the intersection of functions and talk about the properties of the functions over the properties of the set itself and that is indeed true here. This idea is what created Diophantine functions, functions whose "graph" is a Diophantine set.

Definition 1. A function f is Diophantine if $\{x_1, \cdots, x_n, y | y = f(x_1, \cdots, x_n\}$ is a Diophantine set.

Now due to the abstract nature of functions, this describes Diophantine sets well, since we can basically just define functions however we want. However, it is due to the abstract nature that this does not seem useful. However, as we will show later, We can exhaustively build Diophantine functions from simpler ones using a strict set of rules(Recursive functions). It should be mentioned that the domain of Diophantine functions is taken to be $\mathbb{Z}^+$.

We need one more result about Diophantine functions called the Sequence number theorem first. What this basically says is that Diophantine functions are indeed as abstract as we need them to be.

Theorem 2. There exists a Diophantine function $S(i, u)$ such that $S(i, u) \leq u$ and for each sequence $a_1, \cdots, a_n$, there exists u such that $S(i, u) = a_i$ for $1 \leq i \leq n$

Now we can talk about recursive functions, which are functions that are built up from 4 fundamental functions, which are

- $c(x) = 1$
- $s(x) = x + 1$
- $U_i^n(x_1, \cdots, x_n) = x_i \qquad 1 \leq i \leq n$
- $S(i, u)$

Using the operations of

- Composition:
  $h(x_1, \cdots, x_n) = F(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n))$
- Primitive Recursion:
  $h(x_1, \cdots, x_n, 1) = f(x_1, \cdots, x_n)$
  $h(x_1, \cdots, x_n, t + 1) = g(t, h(x_1, \cdots, x_n, t), x_1, \cdots x_n)$
- Minimalization:
  $h(x_1, \cdots, x_n) = min_y[f(x_1, \cdots, x_n, y) = g(x_1, \cdots, x_n, y)]$

Composition is just generally useful, allows combinations of functions, I can't really tell you what primitive recursion is for since I don't know either, minimalization is finding the the smallest y

that satisifies the given equation.

We can now prove one of our major results

Theorem 3. A function is Diophantine if and only if it is recursive.

Proof:
We first show that all Diophantine functions are recursive. The idea here is that all Diophantine functions can be described through Diophantine equations(this will become more clear in a bit) and that every Diophantine polynomial can be given through a few operations, all of which we can show to be recursive.

1. $x + y$ is recursive since $x + 1 = s(x)$ and $x + (t + 1) = s(x + t) = s(U_2^3(t, x + t, x))$. Here we have used primitive recursion.
2. $x \cdot y$ is recursive since $x \cdot 1 = U_1^1(x)$ and $x \cdot (t + 1) = (x \cdot t) + x = g(t, x \cdot t, x)$ where $g(u, v, w) = U_2^3(u, v, w) + U_3^3(u, v, w)$. Here we have used composition and primitive recursion.
3. For all fixed k, $c_k(x) = k$ is recursive since $c_1(x) = c(x)$ and $c_{k+1}(x) = c_k(x) + c(x)$. Here we have used composition.

With these operations, we can form any polynomial with positive coefficients and thus any polynomial with positive coefficients is recursive. Now, earlier it was mentioned that all Diophantine functions can be described through Diophantine equations and here's what is meant by that. If f is a Diophantine function then we can write
$y = f(x_1, \cdots, x_n) \iff (\exists t_1, \cdots, t_m)[P(x_1, \cdots, x_n, y, t_1, \cdots, t_m) = Q(x_1, \cdots, x_n, y, t_1, \cdots, t_m)]$
where P and Q are Diophantine polynomials. Using the Sequence Number Theorem, we can write
$f(x_1, \cdots, x_n) = S(1, min_u[P(x_1, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m + 1, u)) =$
$Q(x_1, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m + 1, u))])$.
Now this is a pretty massive leap so let's break it down piece by piece. First, we are using the recursive function $S(i, u)$ to represent $(y, t_1, \cdots, t_m)$ since any sequence can be represented by it(by the Sequence Number Theorem). Now notice that the minimalization operation gives the smallest u for any given $x_1, \cdots, x_n$, and by the sequence number theorem, we represented y with $S(1, u)$. Note that this is the same u that the minimalization operation gives. Thus we have
$y = S(1, min_u[P(x_1, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m + 1, u)) =$
$Q(x_1, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m + 1, u))])$. And so the graph of this function is $(x_1, \cdots, x_n, y)$, which is the same graph as $f(x_1, \cdots, x_n)$. Thus they are equal. Notice that the way we wrote it used only recursive functions, since as we've shown earlier that Diophantine polynomials are recursive. Thus every Diophantine function is recursive.

We now prove the other direction, that is that every recursive function is Diophantine. We know that $S(i, u)$ is a Diophantine function. $c(x) = 1$, $s(x) = x + 1$ and $U_i^n(x_1, \cdots, x_n) = x_i$ are Diophantine functions(you can come up with a formulaic way to find a Diophantine equation for each to convince yourself further if needed). So then all we need to show is that Diophantine functions are closed under composition, primitive recursion, and minimalization. To do so we need

to introduce the idea of Diophantine predicates.

Given Diophantine functions, we can use the predicates $\wedge$(and), $\vee$(or), $\forall$(for all), and $\exists$(there exists) to form other Diophantine functions. You will have to take me at my word for this one.

1. Composition: If $h(x_1, \cdots, x_n) = f(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots x_n))$, and all the functions involved to create it are Diophantine functions, we write h in the following way.
$y = h(x_1, \cdots, x_n) \iff (\exists t_1, \cdots, t_m)[t_1 = g_1(x_1, \cdots, x_n) \wedge \cdots \wedge t_m = g_m(x_1, \cdots, x_n) \wedge y = f(t_1, \cdots, t_m)]$ Since we used predicates to form this new function, h is a Diophantine function by the closure of Diophantine functions under Diophantine predicates.

2. Primitive Recursion: If $h(x_1, \cdots, x_n, 1) = f(x_1, \cdots, x_n)$ and
$h(x_1, \cdots, x_n, t+1) = g(t, h(x_1, \cdots, x_n, t), x_1, \cdots, x_n)$, where f and g are Diophantine, then by the Sequence number theorem(using it to represent $h(x_1, \cdots x_n, 1)$, $h(x_1, \cdots x_n, 2)$, $\cdots$, $h(x_1, \cdots x_n, z)$)
$y = h(x_1, \cdots, x_n, z) \iff (\exists u)\{(\exists v[v = S(1, u) \wedge v = f(x_1, \cdots, x_n)] \wedge (\forall t)_{\leq z}[(t = z) \vee (\exists v)(v = S(t+1, u) \wedge v = g(t, S(t, u), x_1, \cdots, x_n))] \wedge y = S(z, u)\}$

3. Minimalization: If $h(x_1, \cdots, x_n) = min_y[f(x_1, \cdots, x_n, y) = g(x_1, \cdots, x_n, y)]$ where f and g are Diophantine, then we can write h as a Diophantine function like the following.
$y = h(x_1, \cdots, x_n) \iff (\exists z)[z = f(x_1, \cdots, x_n, y) \wedge z = g(x_1, \cdots, x_n, y)] \wedge (\forall t)_{\leq y}[(t = y) \vee (\exists u, v)(u = f(x_1, \cdots, x_n, t) \wedge v = g(x_1, \cdots, x_n, t) \wedge (u < v \vee v < u)]$

■

Now that we are done with the proof that Diophantine functions are equivalent to Recursive functions, we move on to the next big result, which is the last piece of the puzzle before finishing this solution to Hilbert's 10th problem.
We will be using an equivalent definition of recursively enumerable.

Definition 4. A set S of n-tuples of positive integers is called recursively enumerable if there are recursive functions $f(x, x_1, \cdots, x_n), g(x, x_1, \cdots, x_n)$ such that $S = \{(x_1, \cdots, x_n) | (\exists x)[f(x, x_1, \cdots, x_n) = g(x, x_1, \cdots, x_n)]\}$

Now we can prove our result.

Theorem 5. A set S is Diophantine if and only if it is recursively enumerable.

Proof: Suppose S is Diophantine. Then there are polynomials P and Q such that $(x_1, \cdots, x_n) \in S$ if and only if $(\exists y_1, \cdots, y_m)$ where $P(x_1, \cdots, x_n, y_1, \cdots, y_m) = Q(x_1, \cdots, x_n, y_1, \cdots, y_m)$.
However, we can rewrite this using the sequence number theorem to represent $y_1, \cdots, y_m$ and get the form $(\exists u)[P(x_1, \cdots, x_n, S(1, u), \cdots, S(m, u)) = Q(x_1, \cdots, x_n, S(1, u), \cdots, S(m, u))]$. Notice then that this set must be recursively enumerable by the definition given above.
We now prove the converse. Suppose S is recursively enumerable. By definition, there are recursive functions $f(x, x_1, \cdots, x_n), g(x, x_1, \cdots, x_n)$ such that
$(x_1, \cdots, x_n) \in S \iff (\exists x)[f(x, x_1, \cdots, x_n) = g(x, x_1, \cdots, x_n)]$. We can rewrite this with

Diophantine predicates to get the form $(\exists x, z)[z = f(x, x_1, \cdots, x_n) \wedge z = g(x, x_1, \cdots, x_n)]$. Recall that we have shown that recursive functions are Diophantine, and so Diophantine Predicates on recursive functions must give another Diophantine function. Thus, S is Diophantine.
■

3 The Proof

Our next step is not to describe something called a universal Diophantine set. It's a bit annoying to refer to each Diophantine set individually, it would be so much better to talk about a set that includes all Diophantine sets. As it turns out, an explicit enumeration of the Diophantine sets is possible. This is not as surprising, since the coefficients of Diophantine equations are all integers and $\mathbb{Z}^n$ is enumerable. We will skip over the actual enumeration here.

There's a certain question we can ask about this Universal Diophantine set, and it's whether or not it itself is a Diophantine set. The answer is yes, and it is called the Universality Theorem. Let D_n represent the nth Diophantine set.

Theorem 6. $\{(n, x)|x \in D_n\}$ is Diophantine.

The actual proof involves the use of the explicit enumeration of the Diophantine sets. However, it does not provide much insight so it is not included.

Here's an outline of the path of proof we take from here to a solution of Hilbert's 10th problem. We have been referencing recursive functions a lot but another name for them are μ-recursive functions, which are equivalent to turing-computable functions, which is the formalized notion of algorithm that we will be using. So we define a set $V = \{n|n \notin D_n\}$ and show that it is not Diophantine. Then, we define g which tests if a number is in a Diophantine set and show that it is not recursive. Using the fact that recursive functions are equivalent to algorithms, we can proceed by proof by contradiction and suppose that an algorithm did exist, which would imply g is recursive, which is a contradiction.
Now we continue down this path.

Theorem 7. $V = \{n|n \notin D_n\}$ is not Diophantine

Proof: Suppose for contradiction V is Diophantine, then $V = D_i$ for some fixed i. Since V is Diophantine, then we have $i \in V \iff i \in D_i$ but by definition of V, $i \in V \iff i \notin D_i$, which is a contradiction. Thus V is not Diophantine
■
Now we get to the g that was somewhat unclearly described earlier

Theorem 8. Define a function $g(n, x)$ where $g(n, x) = 1$ if $x \notin D_n$ and $g(n, x) = 2$ if $x \in D_n$. Then g is not recursive.

Proof: Suppose for contradiction that g is recursive, recall that recursive and Diophantine are equivalent and so g mus also be Diopahntine. Then we can write g in the form of a Diophantine function, that is of the form $y = g(n, x) \iff (\exists y_1, \cdots, y_m)[P(n, x, y, y_1, \cdots, y_m) = 0]$. We can then write V as $V = \{x|(\exists y_1, \cdots, y_m)[P(x, x, 1, y_1, \cdots, y_m) = 0]\}$ which implies that V is Diophantine. This is a contradiction, thus g is not recursive.
■
Now we get to the conclusion.

Theorem 9. Hilbert's 10th problem is unsolvable.

Proof: Since the universal Diophantine set $\{(n, x)|x \in D_n\}$ is Diophantine, we can write $x \in D_n \iff (\exists z_1, \cdots, z_k)[P(n, x, z_1, \cdots, z_k) = 0]$. Now, suppose for contradiction that there were an algorithm to test if Diophantine equations had positive integer solutions(i.e. suppose Hilbert's 10th problem was solvable). Then, for all n, x we could use this algorithm to test if $P(n, x, z_1, \cdots, z_k) = 0$ had a solution, in other words whether $x \in D_n$. Then $g(n, x)$ would be computable and thus recursive(μ-recursive), which is a contradiction.
■
Even though Hilbert's 10th problem has been solved, many questions still remain. For example, we still cannot exactly characterize which Diophantine equations are undecidable, and although we have gotten closer to an answer we are still very far. What the solution to Hilbert's 10th problem does is it tells us that it is an undecidable problem but this just adds more mystery to it, we still know very little of each type of Diophantine equation. In fact, we still do not know how many unknowns is needed exactly to make the problem undecidable and when it does become decidable.

References

[1] Martin Davis. "Hilbert's 10th Problem is Unsolvable". In: *The American Mathematical Monthly* (1973).

[2] Walter Dean. "Recursive Functions". In: *The Stanford Encyclopedia of Philosophy* (2021).

[3] Yuri Matiyasevich. *Matiyasevich's Theorem*. 2008.

[4] Andrew Misner. *Hilbert's 10th Problem*. 2013.