AN EXPOSITION ON HILBERT'S TENTH PROBLEM

EDDY LI

0. Introduction

In 1900, the German mathematician David Hilbert proposed a list of the 23 thorniest and unsolved mathematical problems, each of them varying in degrees of difficulty. In particular, the Tenth Problem is commonly stated as follows.

Question 0.1. *Given a Diophantine equation with any number of unknown quantities and with integral numerical coefficients, is it possible to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in integers?*

The "process" that Hilbert asked for in this above question was equivalent to an algorithm on a Turing machine. The notion that the process must take a finite number of operations means that our Turing machine must halt on all inputs of Diophantine equations. Now consider this theorem, due to Davis, Matiyasevich, Putnam, and Robinson:

Theorem 0.2. *The answer to Question 0.1 is in the negative.*

Proof. This will be the main focus developed for this paper! ■

We can sharpen up Question 0.1 a bit by limiting the scope of our Diophantine equations from accepting integers to only accepting positive integers, motivating Question 0.3.

Question 0.3. *Given a Diophantine equation with any number of unknown quantities and with integral numerical coefficients, is it possible to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in the positive integers?*

Of course, it is not completely obvious why Questions 0.1 and 0.3 are equivalent. The theorem below will fix this issue nicely, and it is motivated by some clever constructions.

Theorem 0.4. *If Question 0.3 is answered in the negative, then so must Question 0.1.*

Proof. Suppose that a Turing machine cannot determine whether some Diophantine equation $D(x_1, x_2, \cdots, x_n) = 0$ contains a solution in the positive integers in a finite amount of time. Then consider Diophantine equation

$$D'(p_1, q_1, r_1, s_1, \cdots, p_n, q_n, r_n, s_n) = D(1 + p_1^2 + q_1^2 + r_1^2 + s_1^2, \cdots, 1 + p_n^2 + q_n^2 + r_n^2 + s_n^2).$$

If it is possible to determine whether D' has integer solutions, then it must also be possible to see if D has positive integer solutions, due to Lagrange's theorem that every positive integer may be written as the sum of four squares. By contradiction, it follows that a Turing machine cannot decide whether D' has integer solutions either. ■

Date: December 12, 2022.

1. Diophantine Sets

From now on, all numbers are assumed to be positive integers and all polynomials are assumed to have integral, possibly negative, coefficients, unless stated otherwise. Under this context, Diophantine equations and polynomials are really the same thing.

Definition 1.1. Let S be a subset of $(\mathbb{Z}^+)^n$ for some n. We call S a *Diophantine set* if there exists a polynomial D in $n+m$ variables, where $m \geq 0$, so that $(x_1, x_2, \cdots, x_n) \in S$ if and only if there exists some $(y_1, y_2, \cdots, y_n) \in (\mathbb{Z}^+)^n$ such that

$$D(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_m) = 0.$$

Definition 1.2. Maintaining the same notation used in the above definition, we consider m to be a *dimension* of S and D to be a *Diophantine representation* of S. In particular, note that D and m might not be unique.

Essentially, the motivation of the above definitions is to rig a multivariable polynomial D so that the set S, which we are given, can be precisely stuffed into the set of zeroes of D. It is helpful to think of the usual process of solving Diophantine equations being done in reverse here, as we are given part of the solution set and asked to construct the original polynomial.

Theorem 1.3. *The set of Diophantine sets is closed under intersection.*

Proof. The proof of this theorem is quite easy. Specifically, suppose that S_1 and S_2 are two Diophantine sets, with corresponding Diophantine representations D_1 and D_2, respectively. Now, consider the polynomial $D = D_1^2 + D_2^2$. Clearly, because D_1^2 and D_2^2 must always be nonnegative, we see that $D_1^2 + D_2^2 = 0$ if and only if $D_1 = D_2 = 0$.

It follows that the multivariable polynomial D is a suitable Diophantine representation of the set $S = S_1 \cap S_2$, so that S is a Diophantine set as well. It becomes immediate that the set of Diophantine sets is closed under intersection, concluding our proof. ■

Essentially, what the above theorem implies is that we can combine large systems of polynomials into one large polynomial. Specifically, we can repeat the logic of the above theorem using any number of Diophantine sets, such as the n sets $S_1, S_2, \cdots, S_n$. Suppose that D_i has Diophantine representation S_i for all integers $1 \leq i \leq n$. Then

$$S = S_1 \cap S_2 \cap \cdots \cap S_n$$

is a Diophantine set, with the corresponding representation

$$D_1^2 + D_2^2 + \cdots + D_n^2 = 0.$$

Now we present the two most classic examples of Diophantine sets.

Example 1.4. *Let* OddDiv *be the set such that* $x \in$ OddDiv *if and only if* x *is not a power of two, which is equivalent to having an odd divisor. We can express this odd divisor as* $2y+1$ *for some* y*, so that* $\frac{x}{2y+1} = z$ *is an arbitrary positive integer.*

As a result, we see that $x \in$ OddDiv *iff there exist* y *and* z *such that* (x, y, z) *is a root of the Diophantine equation*

$$D(x, y_1, y_2) = x - (2y_1 + 1)y_2.$$

We have introduced two free variables y_1 *and* y_2*, so* OddDiv *is Diophantine of dimension* 2 *and Diophantine representation* D*.*

Example 1.5. *Something similar applies to the set of composite numbers* Comp. *It is obvious that* $x \in$ Comp *if and only if there exists two integers,* y *and* z*, such that they are both greater than* 2 *and* $x = yz$*. To represent* y *and* z *using positive integers, we can just define the variables* $y' = y - 1$ *and* $z' = z - 1$*. Then* (y', z') *can be any element of* $(\mathbb{Z}^+)^2$*.*

It follows $x \in$ Comp *is equivalent to the existence of* y' *and* z' *such that* (x, y', z') *is a zero of the polynomial*

$$D(x, y_1, y_2) = x - (y_1 + 1)(y_2 + 1).$$

Again, the above equation has two free variables, hence Comp *is a Diophantine set, having a dimension of* 2 *and a Diophantine representation of* D*.*

Now we extend the above ideas to functions as well.

Definition 1.6. Let f be a function on n parameters, so that it has domain $(\mathbb{Z}^+)^n$. Then, f is a *Diophantine function* if and only if the set S consisting of all $n+1$-tuples $(x_1, x_2, \cdots, x_n, y)$ such that $y = f(x_1, x_2, \cdots, x_n)$ is Diophantine.

Essentially, f is Diophantine if we can make a Diophantine equation relating the x_i to our function $f(x_1, x_2, \cdots, x_n)$. In addition to proving that Hilbert's Tenth Problem is unsolvable, a central theme explored in this paper is the question of determining which sets and functions are Diophantine.

2. Sequence Numbers

With the above groundwork established, we will now consider some more constructions of less trivial Diophantine equations and functions. Specifically, we use the triangular numbers and its properties to create a function S. Perhaps surprisingly, it is this function that provides the groundwork of our main result.

Definition 2.1. More formally, let $T(n)$ be the triangular number function. Specifically, we have $T(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$.

We will give $T(n)$ a more Diophantine flavor using the result of the following theorem.

Theorem 2.2. *For all* z*, there exists a unique* x *and* y *such that* $z = T(x + y - 2) + y$*.*

Proof. Let z be some positive integer. Clearly, the function $T(n)$ is increasing, so there exists some unique nonnegative n such that z is between $T(n) + 1$ and $T(n + 1) = T(n) + n + 1$, inclusive. Hence, there exists some unique $y \leq n + 1$ such that

$$z = T(n) + y.$$

Now, we know that for all x, we must have $x + y \geq n + 2$. It follows that $x = n + 2 - y$ must always be a positive integer, so that we can replace n with $x + y - 2$. It follows that z is uniquely representable in the form

$$z = T(x + y - 2) + y.$$

This completes our proof. ■

Now we will express x and y as a function of z and vice versa. Specifically, consider these two definitions below.

Definition 2.3. For some z, let $L(z)$ and $R(z)$ be the unique positive integers such that

$$z = T(L(z) + R(z) - 2) + R(z).$$

Here, L stands for the *left triangular function* and R stands for the *right triangular function*.

Definition 2.4. For any x and y, write

$$P(x, y) = T(x + y - 2) + y.$$

We call P the *pairing function*.

To link the above with our previous work, consider the lemma below.

Lemma 2.5. *The functions $L(z)$, $R(z)$, and $P(x, y)$ are all Diophantine.*

Proof. Let $x = L(z)$. To show that L is Diophantine, it is both necessary and sufficient to show that the set S_L consisting of the ordered pairs of the form (z, x) is Diophantine. Now this is easy: we have $(z, x) \in S_L$ iff there exists some y such that (x, y, z) is a solution to the Diophantine equation

$$D(x, y, z) = (x + y - 2)(x + y - 1) + 2y - 2z.$$

Note that D is really just $2(T(x + y - 2) + y - z)$.

We can repeat a similar procedure with R and P. Specifically, let S_R and S_P be sets consisting of ordered pairs and triples of the form (z, y) and (x, y, z), respectively. First, note that (z, y) is in S_R if and only if there exists some x such that $D(x, y, z) = 0$. Similarly, the ordered triple $(x, y, z + 1)$ is in S_P iff $D(x, y, z) = 0$. Either way, S_R and S_P are both Diophantine, so their corresponding functions are as well. ■

Our above work can be summarized by the following corollary.

Corollary 2.6. *There exist Diophantine functions $P(x, y)$, $L(z)$, and $R(z)$ such that for any x and y, we must have $L(P(x, y)) = x$ and $R(P(x, y)) = y$. Also, for all z, we must have $P(L(z), R(z)) = z$, and both $L(z)$ and $R(z)$ are less than or equal to z.*

Proof. Considering what we have done above, this result is trivial. It merely serves to summarize our progress so far. ■

In order to clarify any confusions, we present some examples about these three functions below.

Example 2.7. *Let $z = 42$. Notice that the greatest triangular number less than z is 36. Then, we would have $42 = 36 + 6$, giving us*

$$z = T(8) + 6.$$

It follows that $R(z) = 6$. Similarly, we have $L(z) = 8 + 2 - R(z) = 4$, so that

$$(L(42), R(42)) = (4, 6).$$

One can verify that $P(L(42), R(42)) = P(4, 6) = T(8) + 6 = 36 + 6 = 42$.

Example 2.8. *Let $z = 127$. In this case, the greatest triangular number less than z is* 120, *so that*

$$z = T(15) + 7,$$

implying that $R(z) = 7$. Similarly, we have $L(z) = 15 + 2 - R(z) = 10$, giving us

$$(L(127), R(127)) = (10, 7).$$

Here, we have $P(L(127), R(127)) = P(10, 7) = T(15) + 7 = 127$.

Now we define a function $S(i, u)$. The definition is quite unintuitive, but the function S is very important. It would not be an exaggeration to say that Hilbert's Tenth Problem was proved using the method presented here exactly due to the discovery of S.

Definition 2.9. Let $S(i, u)$ be the unique number w such that $w \leq 1 + iR(u)$ and

$$w \equiv L(u) \pmod{1 + iR(u)}.$$

In other words, w is the residue class of $L(u)$ under modulo $1+iR(u)$. We call S the *sequence number function.*

Now consider this theorem, which would make our lives much more convenient.

Theorem 2.10. *The function $S(i, u)$ is a Diophantine function.*

Proof. First we prove that $S(i, u)$ is a Diophantine function. Let $w = S(i, u)$, so that we define the set S_S to consist of all ordered triples of the form (i, u, w). Then, we can notice that $(i, u, w) \in S_S$ if and only if there exists some v and z such that (u, v, w, z, i) is a solution to the Diophantine system

$$\begin{aligned} L(u) &= w + z(1 + iR(u)), \\ 1 + iR(u) &= w + v - 1. \end{aligned}$$

The first equation essentially states that $w \equiv L(u) \pmod{1+iR(u)}$, and the second equation gurantees that $w \leq 1 + iR(u)$; note that the use of $v - 1$ means that equality is allowed.

Possibly a source of objection to the above system is that it is not Diophantine due to the use of L and R. However, this is not true: we can let x and y satisfy the additional equation

$$2u = (x + y - 2)(x + y - 1) + 2y,$$

which, as we have seen before, implies that $x = L(u)$ and $y = R(u)$. Then we would just replace instances of L with x and R with y to get a new system. Specifically, we have a septuple (u, v, w, x, y, z, i) such that

$$\begin{aligned} 2u &= (x + y - 2)(x + y - 1) + 2y, \\ x &= w + z(1 + iy), \\ 1 + iy &= w + v - 1. \end{aligned}$$

It is even possible to combine the above equations into our large one by summing their squares. In full gory detail, this is

$$((x + y - 2)(x + y - 1) + 2y - 2u)^2 + (w + z(1 + iy) - x)^2 + (w + v - 2 - iy)^2 = 0.$$

Either way, we have showed that S_S is a Diophantine set, so $S(i, u)$ is a Diophantine function, so we are done. ∎

Our final theorem about S is quite nontrivial. It is known as the Sequence Number Theorem, hence the name of this section.

Theorem 2.11. *The function $S(i, u)$ has the property that if $a_1, a_2, \cdots, a_N$ is a sequence in the positive integers, there exists a unique number u' such that $S(i, u') = a_i$ for all $1 \leq i \leq N$.*

Proof. We proceed by construction. Suppose y is an integral multiple of $N!$ such that y is greater than any of the a_i. Then consider the sequence $1 + y, 1 + 2y, \cdots, 1 + Ny$. Any two entries in this sequence are relatively prime; to see why this is true, note that

$$\begin{aligned} d &= \gcd(1 + ay, 1 + by) \\ &= \gcd(1 + ay, a + aby) \\ &= \gcd(1 + ay, a + aby - b(1 + ay)) \\ &= \gcd(1 + ay, a - b), \end{aligned}$$

for any $a < b \leq N$. This implies that $d \mid a - b$, so that $d \leq N$; by definition of y it follows that $d \mid y$, so that $d = \gcd(1 + ay, y) = 1$.

Because we have showed that the sequence $1 + y, 1 + 2y, \cdots, 1 + Ny$ has elements that are pairwise relatively prime, we can apply the Chinese Remainder Theorem on this sequence as moduli. Specifically, there must exist some x such that $x \equiv a_i \pmod{1+iy}$ for all $1 \leq i \leq N$. With this, pick $u' = P(x, y)$, implying that $(x, y) = (L_u, R_u)$. Our aforementioned congruence system becomes

$$L(u) \equiv a_i \pmod{1 + iR(u)}$$

for all i on which a_i is defined.

Also, by definition we must have $a_i < y$, and we also have that $y = R(u)$, which is clearly less than $1+iR(u)$. It follows that $a_i < 1+iR(u)$ for all i between 1 and n, and by definition, we see that $a_i = S(i, u)$. This is exactly what we wanted to prove, so we are done. ■

The main reason why Theorem 2.11 is so important is that it allows to encode long but finite number sequences, such as $a_1, a_2, \cdots, a_n$, into one value u. In other words, the finite sequence $S(1, u), S(2, u), \cdots, S(n, u)$ can take on any value we want, so long as we rig u in a satisfactory way. This process will always be legal because S is Diophantine, as shown in Theorem 2.10. Our function S is thus extremely useful, as it has become some sort of sequential everyman that can be represented with only one variable.

3. Recursive Sets

So far, all of the above results have to do with pure number theory. Now, we will present some terminology familiar to readers with a background in theoretical computer science. We begin with recursive languages and enumerability.

Definition 3.1. An *alphabet* is simply a finite set of characters.

Loosely speaking, a Turing machine is equivalent to the intuitive notion of a systematic algorithm, equipped with a large, infinite tape that gurantees infinite memory. There is an extremely technical definition, involving octuples, but it is omitted here as there is no need to include it for the purposes of this paper. Describing the algorithmic steps of a Turing machine is usually enough.

Definition 3.2. A set S of n-tuples in $(\mathbb{Z}^+)^n$ is *recursively enumerable* if it can be decided by a Turing machine M, where the elements of S are interpreted as strings in a sufficiently large alphabet. We call M a *decider* of S.

The rigorous term for this is a recursive language, where a language is just a set of strings with characters from some common alphabet. But we will use stick to sets for clarity.

Definition 3.3. A set S of n-tuples is *recursive* or *decidable* if it can be decided by a Turing machine that halts on all inputs.

Specifically, of S is recursive, there must exist a Turing machine M that either halts and accepts or halts and rejects on any given input. In particular, this might not be the case for recursively enumerable but not recursive languages, as they can either halt and accept, halt and reject, or continue running until eternity; the latter case is considered an implicit rejection. Needless to say, this is not allowed for deciders of recursive languages.

We can rig these definitions to get a more set-theoretic notion of recursive enumerability. Our first step is to extend the definitions of decidability and enumerability to functions.

Definition 3.4. A function $h(x_1, x_2, \cdots, x_n)$ is *recursive* if it is a constant function, the shifted fixed point function $s(x) = x + 1$, the projection function $U_i^n(x_1, x_2, \cdots, x_n) = x_i$, or the sequence number function $S(i, u)$. It is recursive by composition if $g_1, g_2, \cdots, g_m$ and f are recursive functions such that

$$h(x_1, x_2, \cdots, x_n) = f(g_1, g_2, \cdots, g_m).$$

Our function $h(x_1, x_2, \cdots, x_n, z)$ is recursive by primitive recursion if there exist recursive functions f and g such that

$$h(x_1, x_2, \cdots, x_n, 1) = f$$

and

$$h(x_1, x_2, \cdots, x_n, k+1) = g(t, h(x_1, x_2, \cdots, x_n, k), x_1, x_2, \cdots, x_n).$$

Our h is also recursive by minimalization such that

$$h(x_1, x_2, \cdots, x_n) = \min_y (c(x_1, x_2, \cdots, x_n, y) = 0)$$

if $c = f - g$, and both f and g are recursive. No other functions are recursive.

Note that composition can be seen as the nesting of algorithms, primitive recursion as running some algorithms on a modified type of for-loop, and minimalization as running two algorithms on a while-loop until they coincide. Of course, this notion is only superficial, because the f and g in minimalization might contain many variables, all of which can take on any positive integer value, of which there are infinite possibilities. However, this way of thinking about recursive functions is quite intuitive and helpful.

From another perspective, the set of all recursive functions is the smallest set closed under composition, primitive recursion, and minimalization that contains the constant functions, shifted fixed point function, projection functions, or the sequence number function. We shall see all of these types of functions and operations on them in the following section. In particular, the sequence number function $S(i, u)$ will show its true power here.

4. Two Types of Functions

We will prove some very important but technical theorems that essentially solve Hilbert's Tenth Problem for us. First, consider these three results. Also, starting from this section, polynomials with name P has nothing to do with the pairing function from two sections ago.

Lemma 4.1. *Let S be the set consisting of all $n+1$-tuples $(y, x_1, x_2, \cdots, x_n)$ such that for all $z \leq y$ there exists $y_1, y_2, \cdots, y_m$ so that*

$$P(y, z, x_1, \cdots, x_n, y_1, \cdots, y_m) = 0,$$

where P is a polynomial. Then S is Diophantine.

Proof. Davis proved this theorem and used it frequently in the original proof that Hilbert's Tenth Problem is unsolvable, as discovered by Matiyasevich, Robinson, Putnam, and Davis himself. It turns out that this proof is quite long and tedious. ■

It follows that we should be able to incorporate for-all statements into Diophantine expressions, as long as the variable under this statement is bounded.

Example 4.2. *The above lemma allows us to prove that the set* Primes *consisting of all the prime numbers is Diophantine. Specifically, $x \in$ Primes if and only if for all $y, z < x$, either one of y and z is 1, or $yz \neq x$. We have used two for-all conditions here, which can be combined into one if we let $k = yz + x$. Then k can take on any value less than x^2. With this, it follows that there must exist some Diophantine representation of the prime numbers, and this is shown in Example 6.2.*

With this established, consider the theorem below.

Theorem 4.3. *If a function is recursive, then it is also Diophantine.*

Proof. First we prove that all recursive functions are Diophantine. It is obvious that c_k, s, and U_i^n are all Diophantine, and so is S by Theorem 2.8. Hence, it suffices to show that the set of Diophantine functions is closed under the three reduction operations of Diophantine functions: composition, primitive recursion, and minimalization.

First we prove composition. Suppose that the functions $g_1, g_2, \cdots, g_m, f$ are all Diophantine, and let h be their composition; that is,

$$h(x_1, x_2, \cdots, x_n) = f(g_1, g_2, \cdots, g_m).$$

It is now enough to find a Diophantine equation equivalent to having $y = h(x_1, x_2, \cdots, x_n)$. But this is trivial, as we can simply construct h out of the other functions.

Specifically, $y = h(x_1, x_2, \cdots, x_n)$ iff there exists some $z_1, z_2, \cdots, z_m$ such that

$$z_i = g_i(x_1, x_2, \cdots, x_n)$$

for all $1 \leq i \leq m$ and

$$y = f(t_1, t_2, \cdots, t_m).$$

These $m+1$ relationships can all be represented by Diophantine equations, so by summing their squares, we are left with a Diophantine equation equivalent to having $y = h$. It follows that h is Diophantine. Notice that we have invoked Lemma 4.1 as the variable i is under a for-all relationship between 1 and m.

For primitive recursion, recall that this is defined by having

$$h(x_1, x_2, \cdots, x_n, 1) = f$$

and

$$h(x_1, x_2, \cdots, x_n, k+1) = g(t, h(x_1, x_2, \cdots, x_n, k), x_1, x_2, \cdots, x_n)$$

for all positive integers k and recursive functions f and g. Define a sequence

$$a_k = h(x_1, x_2, \cdots, x_n, k).$$

Then Theorem 2.11 tells us that there exists some u such that $S(i, u) = a_i$ for all i.

Hence, we see that $y = a_z = h(x_1, x_2, \cdots, x_n, z)$ if and only if there exists some u such that $S(1, u) = f(x_1, x_2, \cdots, x_n)$ and $y = S(z, u)$. Also, for all $k < z$, we must have

$$S(k+1, u) = g(k, S(k, u), x_1, x_2, \cdots, x_n).$$

Note that we have used Lemma 4.1 here again, this time about a for-all condition on k. Conceptually, we are essentially encoding the recursive relationships using the Diophantine functions S, f, and g, implying that h must be Diophantine too.

Something very similar holds for minimalization, which has a function h be determined by

$$h = \min_y(c(x_1, x_2, \cdots, x_n, y) = 0),$$

so that c is a difference of Diophantine functions. We can write this as

$$h = \min_y(f(x_1, x_2, \cdots, x_n, y) = g(x_1, x_2, \cdots, x_n, y)).$$

We can encode this as a Diophantine equation again. Specifically, if

$$u(k) = f(x_1, x_2, \cdots, x_n, k)$$

and

$$v(k) = g(x_1, x_2, \cdots, x_n, k),$$

then we must have $u(y) = v(y)$ and also that $u(k) - v(k) \neq 0$ for all $k < y$. This is another use of the for-all relationship, and u and v must be Diophantine since f and g are so too; in effect, h is a Diophantine function as well.

We have therefore determined that the set of Diophantine functions is closed under the three basic operations, and it follows that all recursive functions are Diophantine. ■

Now we prove the converse of the above theorem. First we prove two important lemmas.

Lemma 4.4. *The sum function $\Sigma(x, y) = x + y$ is recursive.*

Proof. We notice that Σ is given by

$$\Sigma(x, 1) = s(x)$$

and

$$\Sigma(x, k+1) = g(k, \Sigma(x, k), x),$$

where $g(a, b, c) = b + 1 = s(U_2^3(a, b, c))$. Clearly, our g is a recursive function as it is a composition of s and U_2^3, which are both recursive. By primitive recursion, Σ is recursive. ■

Lemma 4.5. *The product function $\Pi(x, y) = xy$ is recursive.*

Proof. Use a similar strategy to the proof of Lemma 4.4 In this case, we have

$$\Pi(x, 1) = U_1^1(x)$$

and

$$\Pi(x, k+1) = g(k, \Pi(x, k), x),$$

where g is given by $g(a, b, c) = U_2^3(a, b, c) + U_3^3(a, b, c) = b + c$; note that this can be easily verified using direct computation. Because U_i^j is recursive, so is g, and this implies that Π is recursive too by primitive recursion. ■

Lemma 4.6. *If a function is Diophantine and has positive coefficients, then it is recursive.*

Proof. The constant function $c_k(x)$ is always recursive, and with Lemmas 4.4 and 4.5, this means that any multivariable polynomial P with positive integer coefficients can be expressed as an iterative composition of sums, functions, and constants. ■

Example 4.7. *In the spirit of Lemma 4.6, the polynomial $7x_1x_2 + 6$ can be expressed as*

$$\Sigma(\Pi(7, \Pi(x_1, x_2)), 6).$$

Longer polynomials can also be written this way, but it would be much more tedious.

We can put things together to get the theorem below.

Theorem 4.8. *If a function is Diophantine, then it is also recursive.*

Proof. Let f be such a function. Because f is Diophantine, the set consisting of all $n+1$-tuples of the form $(x_1, x_2, \cdots, x_n, y)$, where $y = f(x_1, x_2, \cdots, x_n)$, must be Diophantine. Hence, there exists some Diophantine equation g such that $y = (x_1, x_2, \cdots, x_n)$ if and only if there exists $k_1, k_2, \cdots, k_m$ such that

$$g(x_1, x_2, \cdots, x_n, y, k_1, k_2, \cdots, k_m) = 0.$$

Now, we take all the terms in g with negative terms to form a polynomial Q with all positive coefficients, and we let all the rest of the terms form a polynomial P. For example, if $g = 3x_1^2x_2 - 7x_1x_2 + 31x_1$, we would have $P = 3x_1^2x_2 + 31x_1$ and $Q = 7x_1x_2$.

It follows that $g = P - Q$, implying that if $y = f(x_1, x_2, \cdots, x_n)$, then our $k_1, k_2, \cdots, k_m$ will give us

$$P(x_1, x_2, \cdots, x_n, y, k_1, k_2, \cdots, k_m) = Q(x_1, x_2, \cdots, x_n, y, k_1, k_2, \cdots, k_m).$$

By Theorem 2.11, there must exist some u such that $k_i = S(i, u)$ for all $1 \leq i \leq m$ and also that $y = S(m+1, u)$.

Thus, we can write f in the form

$$S(m+1, \min_u(P(\cdots, S(m+1, u), S(1, u), \cdots) = Q(\cdots, S(m+1, u), S(1, u), \cdots))).$$

Because S, P, and Q are all recursive, it follows from composition and minimalization that our function f is also recursive. Hence, all Diophantine functions are recursive, so we are done. ■

Corollary 4.9. *A function is recursive if and only if it is Diophantine.*

Proof. From Theorem 4.3, we know that all recursive functions are Diophantine, and the converse of this is proven in Theorem 4.8. It clearly follows that recursive and Diophantine functions are the same. ■

5. The Final Attack

We first prove two lemmas that bring recursive enumerability to more theoretical terms.

Lemma 5.1. *A set of n-tuples is recursively enumerable if and only if there there exists a Turing machine that enumerates it.*

Proof. Let this set be S, so that $S \subseteq (\mathbb{Z}^+)^n$. Suppose that M is a Turing machine that halts and accepts if the input is in S and either rejects or does not halt otherwise. It is well-known that $(\mathbb{Z}^+)^n$ is countably infinite, so there exists some bijection $f : \mathbb{Z} \to (\mathbb{Z}^+)^n$. Given this, create another Turing machine M' such that at time t, our M' simulates t copies of M, with inputs $f(0), f(1), \cdots, f(t)$.

In other words, at time $t = 0$, the machine M' first begins to simulate M on $f(0)$; one unit of time later, our M' continues simulating M on $f(0)$ and opens another copy of M that simulates on input $f(1)$. At time $t = 2$, both copies are still simulating, and a copy of M working on $f(2)$ opens. This continues until a copy either accepts, in which the enumerator prints it out, rejects, in which it is ignored, or is still running, in which M' lets it run into oblivion. It is obvious that M' gives an enumeration of S.

Now consider the converse: suppose we know that there exists M' that enumerates S. Then to test whether some input is in S, we run M' and halt iff our input is printed. ■

Lemma 5.2. *A set S consisting of n-tuples is recursively enumerable iff there exist recursive functions $f(x, x_1, x_2, \cdots, x_n)$ and $g(x, x_1, x_2, \cdots, x_n)$ such that $(x_1, x_2, \cdots, x_n) \in S$ if and only if there exists some x' satisfying*

$$f(x', x_1, x_2, \cdots, x_n) = g(x', x_1, x_2, \cdots, x_n).$$

Proof. Let some algorithm fix an x' and insert all possible values of $(x_1, x_2, \cdots, x_n)$. If our n-tuple works, we print out $f(x_1, x_2, \cdots, x_n)$; otherwise, we shift to the next tuple. This algorithm is essentially an enumeration of S, so this implies that the existence of an enumeration is equivalent to the existence of f and g. It follows that that having S be recursively enumerable is also equivalent to the existence existence of f and g, so we are done. ■

We can therefore use the notion of recursive enumerability as stated in Lemma 5.2, giving us another surprising theorem determining when we know a set is Diophantine.

Theorem 5.3. *A set is Diophantine iff it is recursively enumerable.*

Proof. Like before, suppose our set $S \subseteq (\mathbb{Z}^+)^n$ is Diophantine. Then there exists some polynomial f such that $(x_1, x_2, \cdots, x_n) \in S$ if and only if there exists variables $y_1, y_2, \cdots, y_m$ so that

$$f(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_m) = 0.$$

Similar to our argument in proving Theorem 4.8, we can dissect f into a difference $P - Q$ so that both P and Q have only positive coefficients, making them automatically recursive.

Using our above progress as well as Theorem 2.11, it follows that $(x_1, x_2, \cdots, x_n) \in S$ iff there is some integer u and some polynomials P and Q with positive coefficients such that

$$P(x_1, x_2, \cdots, S(1, u), S(2, u), \cdots, S(m, u)) = Q(x_1, x_2, \cdots, S(1, u), S(2, u), \cdots, S(m, u)).$$

It follows we can make two new polynomials, P' and Q', such that

$$P'(u, x_1, x_2, \cdots, x_n) = P(x_1, x_2, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m, u))$$

and

$$Q'(u, x_1, x_2, \cdots, x_n) = Q(x_1, x_2, \cdots, x_n, S(1, u), S(2, u), \cdots, S(m, u)).$$

By composition, P' and Q' are both recursive because S, P, and Q are as well, and this would imply that S is recursively enumerable, by Lemma 5.2.

In the opposite direction, suppose that S is recursively enumerable. Using Lemma 5.2, it follows that there exist two recursive functions f and g such that $(x_1, x_2, \cdots, x_n) \in S$ if and only if there exists some x such that

$$f(x, x_1, x_2, \cdots, x_n) = g(x, x_1, x_2, \cdots, x_n).$$

This is further equivalent to the existence of some x' and z so that

$$z = f(x', x_1, x_2, \cdots, x_n)$$

and

$$z = g(x', x_1, x_2, \cdots, x_n).$$

By definition of Diophantine functions, it follows that the set S_F consisting of all tuples of the form $(x', x_1, x_2, \cdots, x_n, f(x', x_1, x_2, \cdots, x_n))$ is Diophantine. Similarly, the set S_G, defined similarly, must be Diophantine as well. But note that $S = S_F \cap S_G$, implying that our set S is Diophantine using Theorem 1.3, so we are done. ■

Now, we consider this following lemma, which is essentially the final piece in the proof that Hilbert's Tenth Problem is unsolvable.

Lemma 5.4. *There exists at least one set that is recursively enumerable but is not recursive.*

Proof. This makes intuitive sense. There are uncountably many possibilities for recursively enumerable sets, as they must all be subsets of the universal set $\{x_1, x_2, x_3, \cdots\}^*$, but each Turing machine can be described as a finite string such that each character has finitely many possibilities, implying that there must be only countably many Turing machines that always halt. Each halting Turing machine is therefore associated with a recursive language, so it follows that there must exist some set that is not recursive but recursively enumerable. ■

Matiyasevich actually constructed such a set consisting of Fibonacci numbers. Because adjacent Fibonacci numbers grow with ratio that eventually converges to $\phi = \frac{1}{2}(1 + \sqrt{5})$, it follows that Matiyasevich's construction involves some sort of exponentiation. In fact, Davis, Robinson, and Putnam generalized this result to showing that the exponential function n^k is Diophantine, with a corresponding equation using 15 variables. This is a constructive numerical example verifying Lemma 5.4.

Corollary 5.5. *Hilbert's Tenth Problem is unsolvable!!!*

Proof. By contradiction, suppose that there existed some algorithm that would always halt and determine whether some Diophantine equation has a solution in the positive integers. This would imply that all Diophantine sets, which are equivalently recursively enumerable, must be recursive. However, as in Lemma 5.4, there exists some recursively enumerable set that is not recursive, contradicting our initial assumption and implying the result. ■

6. Applications of the Tenth Problem

With Corollary 5.5, we have proven what we wanted to and tied up all the loose ends. The question of determining whether a function or set is Diophantine has now been answered as well. In this section we explore several examples of applications of Diophantine equations. First, we will give the complete Diophantine representations of some well-known sequences.

Example 6.1. *Let* Fibonacci *be the set of Fibonacci numbers. Then* Fibonacci *is equal to the positive range of the Diophantine expression*

$$2y^4x + y^3x^2 - 2y^2x^3 - y^5 - yx^4 + 2y,$$

as discovered by Jones. Note that this implies that Fibonacci $\in$ NP. *Also, this is surprisingly tame compared to the Diophantine equation for a general exponential function.*

Example 6.2. *Recall that the set of primes, denoted by* Primes*, is Diophantine, as shown in Example 4.2. In fact, this set is equal to the positive range of the expression*

$$(k+2)\left(1 - \sum_{\alpha=1}^{14} p_\alpha^2\right),$$

where we have

$$\begin{aligned}
p_1 &= wz + h + j - q,\\
p_2 &= (gk + 2g + k + 1)(h + j) + h - z,\\
p_3 &= 2n + p + q + z - e,\\
p_4 &= 16(k+1)^3(k+2)(n+1)^2 + 1 - f^2,\\
p_5 &= e^3(e+2)(a+1)^2 + 1 - o^2,\\
p_6 &= (a^2 - 1)y^2 + 1 - x^2,\\
p_7 &= 16r^2y^4(a^2 - 1) + 1 - u^2,\\
p_8 &= n + l + v - y,\\
p_9 &= ((a + u^2(u^2 - a))^2 - 1)(n + 4dy)^2 + 1 - (x + cu)^2,\\
p_{10} &= (a^2 - 1)l^2 + 1 - m^2,\\
p_{11} &= ai + k + 1 - l - i,\\
p_{12} &= p + l(a - n - 1) + b(2an + 2a - n^2 - 2n - 2) - m,\\
p_{13} &= q + y(a - p - 1) + s(2ap + 2a - p^2 - 2p - 2) - x,\\
p_{14} &= z + pl(a - p) + t(2ap - p^2 - 1) - pm.
\end{aligned}$$

This equation is in 26 *variables, which is why the dummy variable was forced to be the Greek letter* α*. The above system was discovered by Jones, Sato, Wada and Wiens. Similar to example 6.1, it follows that* Primes $\in$ NP.

Now, we will continue with the Diophantine forms of some famously difficult problems, and see how this resolution of the Tenth Problem has prevented us from many false.

Example 6.3. *Fermat's Last Theorem is false if the Diophantine equation*

$$a^{n+2} + b^{n+2} = c^{n+2}$$

has solutions, where a, b, c, and n are all variables. As shown before, the exponential function is Diophantine, and thus it is possible to write the above expression in the form of a Diophantine equation.

Example 6.4. *Goldbach's Conjecture is true if for all even numbers n, there exist primes p and $n-p$. We can use the monstrous primality expression from Example 6.2 to verify that both p and $n-p$ are prime.*

Example 6.5. *Finally, define the function $\delta(x)$ to be*

$$\delta(x) = \prod_{n=0}^{x-1} \prod_{p \in \text{PRIMES}, p=1}^{\lfloor \log_k n \rfloor} p,$$

where k is a fixed number. Riemann's Hypothesis is false if there exists some integer n such that

$$\left(\sum_{k=1}^{k=\delta(n)} \frac{1}{k} - \frac{n^2}{2} \right)^2 \leq 36n^3.$$

Hence we can reduce Riemann's Hypothesis to a Diophantine equation as well.

If the Tenth Problem were true in an alternate universe, then these three important and difficult theorems could have been solved by some algorithm checking the solvability of their corresponding Diophantine equations. In hindsight, the result of Hilbert's Tenth Problem makes these open problems much more interesting and worthwhile to solve.

Last but not least, a generalized version of Hilbert's Tenth Problem states whether there exists an algorithm for checking whether some Diophantine equation has solutions in some ring R. Our work above has answered in the negatived if R is $\mathbb{Z}$ or $\mathbb{Z}^+$. For the subcases where $R = \mathbb{C}$ or $R = \mathbb{R}$, there in fact is such an algorithm. The problem is still open if we take $R = \mathbb{Q}$. Perhaps surprisingly, Hilbert's Tenth Problem is related to elliptic curves.

Acknowledgements

The author would like to thank Simon Rubinstein-Salzedo for giving tons of opportunities, guidance, and insights, and Aaron Lin for being an awesome and extremely helpful TA.

References

[Dav73] Martin Davis. Hilbert's tenth problem is unsolvable. *American Mathematical Monthly*, 80(3):233–269, 1973.
[Fod12] Brandon Fodden. Hilbert's tenth problem, 2012.
[Ho15] Andrew Ho. Hilbert's tenth problem, 2015.
[Mat96] Yuri Matiyasevich. Hilbert's tenth problem: What can we do with diophantine equations?, 1996.
[Orl19] Ben Orlin. Math with bad drawings, 2019.
[RS21] Simon Rubenstein-Salzedo. Theory of computation week 5: Decidability, 2021.

31415 Riemann Avenue, Gaussian Mountains, CA 92653
(You noticed that this was fake, didn't you?)
Email address: eddylihere@gmail.com