rijksuniversiteit groningen

faculteit Wiskunde en Natuurwetenschappen

Creating a diophantine description of a r.e. set

and on the complexity of such a description

Masteronderzoek Wiskunde

25 februari 2010

Student: L. B. Kuijer

Eerste Begeleider: J. Top

Tweede Begeleider: J. Terlouw

Contents

1. Introduction

A very old class of problems in mathematics is the solving of Diophantine equations. Essentially a Diophantine equation is a polynomial equation that should be solved in the natural numbers. While solutions to specific Diophantine equations have of course been found, a general method to solve any given Diophantine equation has not. In 1900 Hilbert considered this a sufficiently important problem to include the finding of such a method in his famous list of at that time unsolved mathematical problems.
In 1970 Matijasevic [6] showed that such a method does not exist. I will give a proof of the non-existence of such an algorithm that resembles proofs that were developed later. The main idea used in section 3 strongly resembles the method used in 1975 by Matijasevic and Robinson [8], although many details are handled in a different way here. Section 4 uses some of the same methods as presented in 1984 by Jones and Matijasevic [5], but they are applied to a different formalization here.
After giving the proof I will briefly discuss some complexity measures that could have applications when considering the solvability of diophantine equations in a fixed number of unknowns.

2. Definitions

First, we shall need some definitions.

2.1. **Diophantine.**

Definition 1. *A* diophantine equation *is an equation*

$$P(x_1, x_2, \cdots, x_n) = 0$$

where P is a polynomial with integer coefficients, and the variables $x_1, \cdots, x_n$ vary over the natural numbers.

For example,

$$P(b, x_1, x_2) = x_1^2 - bx_2^2 - 1 = 0$$

with $b, x_1, x_2 \in \mathbb{N}$ is a diophantine equation known as the Pell equation.

Note that we can add parameters, or, equivalently, split the variables into parameters and unknowns. With the Pell equation, one usually takes b as parameter and x_1, x_2 as unknowns:

$$P_b(x_1, x_2) = P(b, x_1, x_2) = 0.$$

A diophantine set is defined as the parameters for which a given diophantine equation has solutions.

Definition 2. *A* diophantine set *is a set of the form*

$$\{(a_1, \cdots, a_n) \in \mathbb{N}^n : \exists x_1, \cdots, x_m \in \mathbb{N}\ [P(a_1, \cdots, a_n, x_1, \cdots, x_m) = 0]\}$$

where P is a polynomial with integer coefficients.

So for example the set of all squares in $\mathbb{N}$ is a diophantine set, since it can be given by

$$\{a \in \mathbb{N} : \exists x \in \mathbb{N}[a - x^2 = 0]\},$$

as is the set of all b such that the Pell equation with parameter b has a solution

$$\{b \in \mathbb{N} : \exists x_1, x_2 \in \mathbb{N}[x_1^2 - bx_2^2 - 1 = 0]\}.$$

Also, the set of zeros of any polynomial $P(a_1 \cdots, a_n) = P(\mathbf{a})$ is a diophantine set,

$$\{\mathbf{a} : P(\mathbf{a}) = 0\}$$

with 0 unknowns. Or, alternatively, one can add a new unknown

$$\{\mathbf{a} : P(a) = 0\} = \{\mathbf{a} : \exists x\ [P(\mathbf{a})(x^2 + 1) = 0]\}$$

Every polynomial with at least one parameter describes a diophantine set. It will often be convenient to leave this construction implicit and refer to sets by the corresponding polynomials.

Lemma 1. *If $U, V \subset \mathbb{N}^n$ are diophantine sets corresponding to polynomials P and Q, then*
(i) $U \cup V$ is diophantine, and $U \cup V$ corresponds to PQ.
(ii) $U \cap V$ is diophantine, and $U \cap V$ corresponds to $P^2 + Q^2$.

Proof: (i) U and V are diophantine, so they have the form

$$U = \{\mathbf{a} = (a_1, \cdots, a_n) : \exists \mathbf{x} = (x_1, \cdots x_m)\ [P(\mathbf{a}, \mathbf{x}) = 0]\}$$

and

$$V = \{\mathbf{a} : \exists \mathbf{y} = (y_1, \cdots y_k)\ [Q(\mathbf{a}, \mathbf{y}) = 0]\}$$

so

$$U \cup V = \{\mathbf{a} : \exists \mathbf{x}\ [P(\mathbf{a}, \mathbf{x}) = 0] \vee \exists \mathbf{y}\ [Q(\mathbf{a}, \mathbf{y}) = 0]\}.$$

Then, since $\phi\psi = 0 \Leftrightarrow \phi = 0 \vee \psi = 0$,

$$\begin{aligned} U \cup V &= \{\mathbf{a} : \exists \mathbf{x}, \mathbf{y}\ [P(\mathbf{a}, \mathbf{x}) = 0 \vee Q(\mathbf{a}, \mathbf{y}) = 0]\} \\ &= \{\mathbf{a} : \exists \mathbf{x}, \mathbf{y}\ [PQ(\mathbf{a}, \mathbf{x}, \mathbf{y}) = P(\mathbf{a}, \mathbf{x})Q(\mathbf{a}, \mathbf{y}) = 0]\}. \end{aligned}$$

(ii) Follows from $\phi^2 + \psi^2 = 0 \Leftrightarrow \phi = 0 \wedge \psi = 0$, and the details as presented in (i). □

Due to the way diophantine sets are defined one can easily deal with existential quantifiers:

Lemma 2. *If U is a diophantine set, then so is $V = \{\boldsymbol{a} : \exists \boldsymbol{x}\ [(\boldsymbol{a}, \boldsymbol{x}) \in U]\}$.*

Proof: U is diophantine, so $U = \{\mathbf{b} : \exists \mathbf{y}\ [P(\mathbf{b}, \mathbf{y}) = 0]\}$. Then $V = \{\mathbf{a} : \exists \mathbf{x}, \mathbf{y}\ [P(\mathbf{a}, \mathbf{x}, \mathbf{y}) = 0]\}$. □

There is no such easy way to deal with universal quantifiers however. Likewise, there is no easy way to define the complement of a diophantine set in a diophantine way. Using results I prove later one can quite easily see that $\{\mathbf{a} : \exists x_1 \forall x_2 \leq x_1 [(\mathbf{a}, x_1, x_2) \in S]\}$ is diophantine for diophantine S, but the complement of a diophantine set is not always diophantine.

2.2. **Exponentially diophantine.** We can make the same construction if we also allow exponentiation.

Definition 3. *P is an* exponential polynomial *if it can be obtained in a finite number of steps by applying the following rules:*
(i) Variables are exponential polynomials.
(ii) For $n \in \mathbb{N}$, n is an exponential polynomial.
(iii) If P and Q are exponential polynomials, then so are $P + Q$ and PQ.
(iv) If P and Q are exponential polynomials, then so is P^Q.

It is important that the coefficients, obtained by using (ii), are nonnegative. This will ensure that exponential polynomials will take values in $\mathbb{N}$. While we would like to use coefficients in $\mathbb{Z}$, we cannot simply do this because negative powers of integers are not guaranteed to be integers. An unfortunate side effect of this is that not every polynomial is an exponential polynomial. Since we want every diophantine set to be an exponential diophantine set we cannot define exponential diophantine sets as the parameters for which a given exponential polynomial has a zero. We can however define an exponential diophantine set as the parameters for which a given exponential diophantine equation has a solution by defining exponential diophantine equations in the right way.

Definition 4. *An* exponential diophantine equation *is an equation*

$$P(\boldsymbol{a}, \boldsymbol{x}) - Q(\boldsymbol{a}, \boldsymbol{x}) = 0$$

where P and Q are exponential polynomials, and the parameters and variables vary over in the natural numbers.

Definition 5. *An* exponential diophantine set *is a set of the form*

$$\{\boldsymbol{a} \in \mathbb{N}^n : \exists \boldsymbol{x} \in \mathbb{N}^m\ [P(\boldsymbol{a}, \boldsymbol{x}) - Q(\boldsymbol{a}, \boldsymbol{x}) = 0]\}$$

where P and Q are exponential polynomials.

Since for any polynomial P we can write $P = Q - R$ where Q and R have nonnegative coefficients by expanding P into its monomials, and putting the ones with negative coefficients in $-R$, every diophantine set is an exponential diophantine set.

2.3. **Recursively enumerable.**

Definition 6. *A set $S \subset \mathbb{N}^n$ is* recursive *if there exists an algorithm that for any $\boldsymbol{x} \in \mathbb{N}^n$ returns 0 if $\boldsymbol{x} \in S$, and 1 if $\boldsymbol{x} \notin S$, in a finite amount of time.*

Of course in order to make this definition precise one would have to define what an algorithm is. I will not do that at his point however, in order not to make this section unnecessarily complicated. Any of the usual definitions of an algorithm will do. Note, by the way, that if we would assign the names "true" and "false" to 0 and 1 in this situation, the obvious choice would be to assign "true" to 0, and "false" to 1. While there is no real need to assign "true" and "false", one should be careful not to get confused with the common identification of "true" and 1.

Definition 7. *A set $S \subset \mathbb{N}^n$ is* recursively enumerable*, or* r.e.*, if there exists an algorithm that for any $\boldsymbol{x} \in \mathbb{N}^n$ returns 0 in a finite amount of time if and only if $\boldsymbol{x} \in S$.*

So a set S is recursive if there is an algorithm that for every $x \in S$ proves that $x \in S$, and for every $y \notin S$ proves that $y \notin S$. A set S is recursively enumerable if there is an algorithm that for every $x \in S$ proves that $x \in S$. Recursive sets are also sometimes called *decidable* since there exists an algorithm that for any x decides whether $x \in S$, r.e. sets are sometimes called semi-decidable since there exists an algorithm that decides that $x \in S$ whenever that is the case but does not decide that $x \notin S$.
Every recursive set is therefore recursively enumerable. However,

Theorem 1. *There exists a set S that is recursively enumerable but not recursive.*

Sketch of proof: The binary representation of any $x \in \mathbb{N}$ can be interpreted as a string of bytes which each represent a character. Doing this gives a 1-1 correspondence between strings of characters and natural numbers. Algorithms can be represented as a strings of characters by implementing the algorithms in a programming language. Therefore, we can create a surjective map $x \mapsto A_x$ from integers to algorithms.
Let $S = \{x : A_x \text{ halts in finite time}\}$. Clearly S is r.e., since for any input x one can just determine A_x, run A_x and return 0 if A_x halts. However, if S would be recursive there would be an algorithm that given any algorithm A decides whether or not A halts. This is exactly the halting problem, which was shown to be unsolvable by Church [1] in 1937. I will not prove the unsolvability of the halting problem here, most textbooks on the subject include a proof for those interested in it.

Lemma 3. *Every exponential diophantine set is recursively enumerable.*

Proof: Let S be an exponential diophantine set, so

$$S = \{\mathbf{a} \in \mathbb{N}^n : \exists \mathbf{x} \in \mathbb{N}^m \ [P(\mathbf{a}, \mathbf{x}) - Q(\mathbf{a}, \mathbf{x}) = 0]\}$$

Take any $\mathbf{a} \in \mathbb{N}^n$. Start by checking whether $P(\mathbf{a}, \mathbf{0}) - Q(\mathbf{a}, \mathbf{0}) = 0$. If it is, then output 0. Otherwise, check whether $P(\mathbf{a}, \mathbf{x}) - Q(\mathbf{a}, \mathbf{x}) = 0$ for any $\mathbf{x} = (x_1, \cdots, x_m)$ with $\max(x_1, \cdots, x_m) = 1$. If it is, output 0. Otherwise, check for all $\mathbf{x}$ with $\max(x_1, \cdots, x_m) = 2$, and so on.
If $\mathbf{a} \in S$, then there is an $\mathbf{y} = (y_1, \cdots, y_m)$ such that $P(\mathbf{a}, \mathbf{y}) - Q(\mathbf{a}, \mathbf{y}) = 0$. This $\mathbf{y}$ is finite, so it will be found in a finite amount of time. □

Note that for $\mathbf{a} \notin S$ the algorithm will run indefinitely. The definition of recursively enumerable allows this though, unlike the definition of recursive.

2.4. **Connecting the concepts.** With these concepts we can start to consider Hilbert's tenth problem. Modulo some phrasing, Hilbert's tenth problem is the following:

Given an arbitrary diophantine equation, give a method for determining whether the equation has a solution, in a finite number of steps.

Like with many of Hilbert's problems, there is some debate about how the problem should be interpreted. It seems reasonable to conclude from the fact that a method for an arbitrary diophantine equation is asked for that a single method should be found which determines the solvability of any diophantine equation. After all, that seems the most logical way of giving a method for determining whether an arbitrary diophantine equation has a solution. Furthermore, it seems likely that the method Hilbert asks for should be what is currently known as an algorithm. A common interpretation of Hilbert's tenth problem, and the interpretation I will use, is therefore:

Find an algorithm that, given an arbitrary diophantine equation, determines whether a solution exists.

It turns out, however, that this is not the right question to ask. The more interesting question is:

Is Hilbert's tenth problem solvable?

The answer is "no", as shown by Matijasevich in 1970 when he proved that every exponential diophantine set is diophantine. Together with earlier results by Davis, Putnam and Robinson this proved the unsolvability of Hilbert's tenth problem.

The proof uses part (i) of the following lemma:

Lemma 4. *(i) If Hilbert's tenth problem is solvable, then every diophantine set is recursive.*
(ii) If every diophantine set is recursive, then Hilbert's tenth problem is solvable.

Proof:
(i) Suppose Hilbert's tenth problem is solvable. Take any diophantine set

$$S = \{\mathbf{a} \in \mathbb{N}^n : \exists x \in \mathbb{N}^m \ [P(a, x) = 0]\}.$$

In order to show that S is recursive we have to show there exists an algorithm A that for any $\mathbf{b} \in \mathbb{N}^n$ outputs 0 if $\mathbf{b} \in S$ and outputs 1 if $\mathbf{b} \notin S$.
Take any $\mathbf{b} \in \mathbb{N}^n$. By the assumption that Hilbert's tenth problem is solvable for every diophantine equation, there is an algorithm B that decides whether $P(\mathbf{b}, \mathbf{x})$ has a solution for $\mathbf{x} \in \mathbb{N}^m$. The algortithm A we are looking for can now be taken: "Run B, and output 0 if B decides there is a solution, and 1 if B decided there is no solution".
(ii) Suppose every diophantine set is recursive. Take any diophantine equation

$$P(\mathbf{x}) = 0.$$

Now take the diophantine set

$$S = \{a \in \mathbb{N} : \exists \mathbf{x} \ [(a^2 + 1)P(\mathbf{x}) = 0]\}.$$

Either there is a solution to $P(\mathbf{x}) = 0$ and $S = \mathbb{N}$, or there is no solution and $S = \emptyset$. In order to show that Hilbert's tenth problem is solvable for this diophantine equation we have to find an algorithm A that decides whether it has a solution. By the assumption that every diophantine set is recursive there exists an algortithm B that decides whether $0 \in S$. The algorithm A can now be taken "Run B and output its output". □

However, Matijasevich [6] showed that every r.e. set is diophantine. By theorem 1 there is a r.e. set that is not recursive, so there is a diophantine set that is not recursive. By lemma 4 this implies that Hilbert's tenth problem is not solvable.

In the following section I will give a proof that every exponential diophantine set is also diophantine. In section 4 I will give a proof that ever r.e. set is exponential diophantine.

3. Exponential diophantine to diophantine

While not neseccary for proving that every exponential diophantine set is diophantine, I will first give some examples of diophantine and exponential diophantine sets, how they can be created, and how they are connected in order to develop some intuition.
The methods used in this section are not new, partially going back as far as Lagrange. I will

mostly just present the results here, for a treatment of the results with more context see for example chapter II of Smorynski [9].

3.1. **The Fibonacci numbers.** Most of the techniques used to create diophantine representations of sets can be used for many sets. In order to give some idea of these techniques I will first show how to give a diophantine representation of the Fibonacci numbers, before discussing the Pell equation, which I'll use for a proof that every exponential diophantine set is diophantine.

The Fibonacci numbers are the numbers $\{F_n\} = \{F_i | i \in \mathbb{N}\}$ with $F_0 = 0, F_1 = 1, F_{n+2} = F_{n+1} + F_n$. We would like to have a diophantine representation of $\{F_n\}$. It is quite hard to find a diophantine representation from this recursive definition however, so before we do anything else we'll have to find a more direct description of F_n. There is a common "trick" to do this: we try to find an x such that x^n satisfies the same recurrence. From

$$F_{n+2} = F_{n+1} + F_n,$$

we get

$$x^{n+2} = x^{n+1} + x^n.$$

Dividing by x^n gives

$$x^2 = x + 1,$$

so $x_\pm = \frac{1}{2} \pm \frac{1}{2}\sqrt{5}$ works. Now let's take a closer look at x_+^n.

Take $2x_+^n = g_n + f_n\sqrt{5}$. (Multiplying x_+^n by 2 will ensure that f_n and g_n are integers.) We chose x_+ in such a way that $x_+^{n+2} = x_+^{n+1} + x_+^n$, so

$$g_{n+2} + f_{n+2}\sqrt{5} = (g_{n+1} + f_{n+1}\sqrt{5}) + (g_n + f_n\sqrt{5})$$

It may not be immediately obvious that f_i and g_i are integers, but it is clear that they are rational. Therefore,

$$g_{n+2} = g_{n+1} + g_n, f_{n+2} = f_{n+1} + f_n.$$

Also,

$$g_{n+1} + f_{n+1}\sqrt{5} = (\frac{1}{2} + \frac{1}{2}\sqrt{5})(g_n + f_n\sqrt{5}) = \frac{1}{2}g_n + \frac{5}{2}f_n + (\frac{1}{2}g_n + \frac{1}{2}f_n)\sqrt{5},$$

so $g_{n+1} = \frac{1}{2}g_n + \frac{5}{2}f_n$ and $f_{n+1} = \frac{1}{2}g_n + \frac{1}{2}f_n$.

Now let's take a look at the values of f_n and g_n for low n.

n	x_+^n	g_n	f_n
0	1	2	0
1	$\frac{1}{2} + \frac{1}{2}\sqrt{5}$	1	1
2	$\frac{1}{4} + \frac{5}{4} + \frac{2}{4}\sqrt{5}$	3	1

Obviously I could have stopped after $n = 1$, since both $\{f_n\}$ and $\{g_n\}$ are fully determined by their first two values. We now know that $\{f_n\}$ satisfies the same recurrence relation and has the same initial values as $\{F_n\}$, so $f_n = F_n$.

We were of course "lucky" that the initial values we wanted happened to occur in one of the two sequences we created. However, with the two sequences we can easily create a sequence satisfying the same recurrence relation and arbitrary initial values. Suppose for example we would want to create the sequence S_n satisfying $S_{n+2} = S_{n+1} + S_n$ with initial values $S_0 = s_0, S_1 = s_1$. Then $S_n = \frac{s_0}{2}g_n + (s_1 - \frac{s_0}{2})f_n$.

This still isn't a diophantine description though. For that we will need one more step.

Lemma 5. *Using notation as above,* $g_n^2 - 5f_n^2 = 4 \cdot (-1)^n$.

Proof: First, note that $(\frac{1}{2} - \frac{1}{2}\sqrt{5})^n = \frac{1}{2}g_n - \frac{1}{2}f_n\sqrt{5}$. This follows from the the fact that we didn't actually choose the positive or negative root, so the construction is invariant under conjugation. Therefore,

$$g_n^2 - 5f_n^2 = (2x_+^n) \cdot (2x_-^n) = 4 \cdot (x_+x_-)^n = 4 \cdot (-1)^n.$$

□

This gives us a candidate for a diophantine description of the Fibonacci numbers. Left to show is that only the Fibonacci numbers satisfy the equation.

Lemma 6. *If $x^2 - 5y^2 = \pm 4$ and $x, y \geq 0$ then there is a n such that $x = g_n$ and $y = f_n$.*

Proof: For $y > 1$, the difference between $5y^2$ and $5(y-1)^2$ is greater than 8, so if $\tilde{x}^2 - 5\tilde{y}^2 = \pm 4$ and $\tilde{y} < y$, then also $\tilde{x} < x$. Also, the only squares that differ by 8 are 1 and 9, and if $y > 1$ then $x = 1$ does not give a solution, so for any $y > 1$ there is at most one x such that $x^2 - 5y^2 = \pm 4$. This allows us to use induction on y.
Proof by induction on y:
If $y = 0$, then $x = 2$, so $x = g_0, y + f_0$.
If $y = 1$, then $x^2 = 1$ or $x^2 = 9$, so either $x = g_1 = 1, y = f_1 = 1$ or $x = g_2 = 3, y = f_2 = 1$.
Now we want to prove an induction step. This will be done using the fact that by solving $f_{n+1} = \frac{1}{2}f_n + \frac{1}{2}g_n$ and $g_{n+1} = \frac{5}{2}f_n + \frac{1}{2}g_n$ for f_n and g_n one obtains $f_n = \frac{1}{2}(g_{n+1} - f_{n+1})$ and $g_n = \frac{1}{2}(5f_{n+1} - g_{n+1})$.
Suppose $x^2 - 5y^2 = \pm 4$, $y > 1$ and that for every $\tilde{y} < y$ and $\tilde{x}$ such that $\tilde{x}^2 - 5\tilde{y}^2 = \pm 4$ one has $\tilde{x} = g_m, \tilde{y} = f_m$. Take $\tilde{x} = \frac{1}{2}(5y - x)$ and $\tilde{y} = \frac{1}{2}(x - y)$. By assumption $x^2 - 5y^2 = \pm 4$, so $x^2 - 5y^2$ is even, so either x and y are both even, or x and y are both odd, so $\tilde{x}$ and $\tilde{y}$ are integers. Also, $x^2 \geq 5y^2 - 4$ and $y > 1$ so $x > y$, and $x^2 \leq 5y^2 + 4 \leq 6y^2$ so $x \leq \sqrt{6}y < 3y$. Therefore, $0 < \tilde{y} = \frac{1}{2}(x - y) < \frac{1}{2}(3y - y) = y$.
With these $\tilde{x}$ and $\tilde{y}$ we get

$$\tilde{x}^2 - 5\tilde{y}^2 = \frac{1}{4}(5y - x)^2 - 5\frac{1}{4}(x - y)^2 = \frac{25y^2}{4} - \frac{10xy}{4} + \frac{x^2}{4} - \frac{5x^2}{4} + \frac{10xy}{4} - \frac{5y^2}{4}$$

$$= \frac{20y^2}{4} - \frac{4x^2}{4} = -(x^2 - 5y^2) = \mp 4.$$

Then there is an m such that $\tilde{x} = g_m, \tilde{y} = f_m$, and $x = g_{m+1}, y = f_{m+1}$. This finishes the induction, and with that the proof. □

These two lemma's combined show that the Fibonacci numbers are exactly the $y \in \mathbb{N}$ such that there is an x such that $x^2 - 5y^2 = \pm 4$. This is very close to a diophantine definition of the Fibonacci numbers.

Corollary 1. $\{F_n\} = \{y \in \mathbb{N} : \exists x \in \mathbb{N}\ [(x^2 - 5y^2 - 4)(x^2 - 5y^2 + 4) = 0]\}$

3.2. **The Pell Equation.** Another diophantine set of interest is the set of solutions to the Pell equation

$$x^2 - by^2 = 1$$

for fixed b. In this case we already have a diophantine definition of the set. We can however try to find a little more. The set $Y := \{y : \exists x\ [x^2 - by^2 = 1]\}$ has an obvious enumeration by size. What we want to find is a diophantine description of "the n-th element of Y". In terms of diophantine sets this comes down to finding a diophantine description of the graph of $y(n)$ where $y(m)$ is the m-th element of Y, so the set $\{(n, y(n)) : n \in \mathbb{N}\}$. It is not immediately clear how to find a diophantine description of this set, so in order to find it we will need more information. The obvious thing to try in this case is to find a recursive definition.

We're looking for solutions $(x, y) \in \mathbb{N}^2$ for fixed b. The equation has a trivial solution $(1, 0)$, but we cannot guarantee the existence of nontrivial solutions without placing restrictions on b.

Lemma 7. *If $b = c^2$ for some $c \in \mathbb{N}$ with $c > 0$, then the equation $x^2 - by^2 = 1$ has no nontrivial solutions.*

Proof: Suppose $x^2 - by^2 = 1$. Then x^2 and $by^2 = c^2y^2$ are two squares that differ by one. The only squares that differ by one are 0 and 1, so $x^2 = 1$ and $by^2 = 0$. Since $b \neq 0$ this solution is the trivial solution (1,0). □

The first step towards a recursive definition of the solutions to the Pell equation is to find a candidate sequence. For the Fibonacci sequence our solutions were of the form $\frac{1}{2}g_n + \frac{1}{2}f_n\sqrt{5} = (\frac{1}{2} + \frac{1}{2}\sqrt{5})^n$. Note that this implies that for any m we have $(\frac{1}{2}g_n + \frac{1}{2}f_n\sqrt{5})^m = \frac{1}{2}g_{nm} + \frac{1}{2}f_{nm}\sqrt{5}$. For the Pell equation we will also try to find combined solutions as powers of previous combined solutions.

Lemma 8. *If $(x(n), y(n)), (x(m), y(m)) \in \mathbb{Z}^2$ satisfy the Pell equation, then so do $(x(k), y(k)) \in \mathbb{Z}^2$ given by $x(k) + y(k)\sqrt{b} = (x(n) + y(n)\sqrt{b})(x(m) + y(m)\sqrt{b})$.*

Proof: By elementary calculation.

$$(x(n) + y(n)\sqrt{b})(x(m) + y(m)\sqrt{b}) = x(n)x(m) + by(n)y(m) + (x(n)y(m) + x(m)y(n))\sqrt{b}$$

so $x(k) = x(n)x(m) + by(n)y(m), y(k) = x(n)y(m) + x(m)y(n)$. Then

$$\begin{aligned} x(k)^2 - by(k)^2 &= x(n)^2x(m)^2 + b^2y(n)y(m) + 2x(n)x(m)y(n)y(m) - bx(n)^2y(m)^2 \\ &\quad -bx(m)^2y(n)^2 - 2bx(n)x(m)y(n)y(m) \\ &= x(n)^2x(m)^2 - bx(n)^2y(m)^2 - bx(m)^2y(n)^2 + b^2y(n)y(m) \\ &= (x(n)^2 - by(n)^2)(x(m)^2 - by(m)^2) = 1 \cdot 1 = 1 \end{aligned}$$

since $(x(n), y(n))$ and $(x(m), y(m))$ satisfy the Pell equation. Therefore, $(x(k), y(k))$ satisfies the Pell equation. □

Note that even though $(x(k), y(k))$ satisfies the Pell equation it need not be one of the solutions we're looking for, since $x(k)$ and $y(k)$ may be negative.

Lemma 9. *If $(x(1), y(1))$ is a solution to the Pell equation, then for any $n \in \mathbb{N}$, so is $(x(n), y(n)) \in \mathbb{N}^2$ given by $x(n) + y(n)\sqrt{b} = (x(1) + y(1)\sqrt{b})^n$.*

This follows immediately from lemma 8.

Left to show is that it there is a pair $(x(1), y(1))$ such that every solution is found by taking the powers of $x(1) + y(1)\sqrt{b}$. If there is no non-trivial solution the trivial solution $(1, 0)$ will do. If there is a non-trivial solution, only the smallest one could possibly work. We can unambiguously determine such a smallest solution because $x^2 = by^2 + 1$, so for two solutions $(x(n), y(n))$ and $(x(m), y(m))$ one has $x(n) > x(m)$ if and only if $y(n) > y(m)$.

Lemma 10. *Let $(x(1), y(1))$ be the smallest non-trivial solution to the Pell equation, and $\{(x(n), y(n))\}$ the sequence of solutions given by $x(n) + y(n)\sqrt{b} = (x(1) + y(1)\sqrt{b})^n$. Then for every solution (x, y) to the Pell equation, $(x, y) \in \{(x(n), y(n))\}$.*

Proof: First note that just like with the Fibonacci numbers we have $(x(1) - y(1)\sqrt{b})^n = x(n) - y(n)\sqrt{b}$. In this case we have slightly more though. $(x(1) + y(1)\sqrt{b})(x(1) - y(1)\sqrt{b}) = x(1)^2 - by(1)^2 = 1$, so $x(n) - y(n)\sqrt{b} = (x(n) + y(n)\sqrt{b})^{-1}$.
Now, suppose (x, y) is a solution, with $y(n) \leq y < y(n + 1)$. Then also $x(n) \leq x < x(n + 1)$, so

$$\begin{aligned} (x(1) + y(1)\sqrt{b})^n &= x(n) + y(n)\sqrt{b} \leq x + y\sqrt{b} \\ &< x(n + 1) + y(n + 1)\sqrt{b} = (x(1) + y(1)\sqrt{b})^{n+1}. \end{aligned}$$

Multiplying by $(x(1) - y(1)\sqrt{b})^n = x(n) - y(n)\sqrt{b} = (x(1) + y(1)\sqrt{b})^{-n}$ gives

$$\begin{aligned} 1^n \leq (x + y\sqrt{b})(x(1) - y(1)\sqrt{b})^n &< (x(1) - y(1)\sqrt{b})^n(x(1) + y(1)\sqrt{b})^{n+1} \\ &= x(1) + y(1)\sqrt{b} \end{aligned}$$

By lemma 8 $(\tilde{x}, \tilde{y})$ given by $\tilde{x} + \tilde{y}\sqrt{b} = (x + y\sqrt{b})(x(1) - y(1)\sqrt{b})^n$ also satisfies the Pell equation. Also, $\tilde{x} = xx(n) - yy(n)b \geq 0$. By $\tilde{(x)} - \tilde{y}\sqrt{b} = (\tilde{x} + \tilde{y}\sqrt{b})^{-1}$ and $1 \leq \tilde{x} + \tilde{y}\sqrt{b}$ we also know that

$\tilde{y} \geq 0$, so $(\tilde{x}, \tilde{y})$ is a solution of the type we are looking for.
If this would be a non-trivial solution, there would be a non-trivial solution smaller than $(x(1), y(1))$. This is not possible, since $(x(1), y(1))$ was chosen to be the smallest non-trivial solution. Therefore the solution obtained is the trivial one,

$$1 = \tilde{x} + \tilde{y}\sqrt{b} = (x + y\sqrt{b})(x(1) - y(1)\sqrt{b})^n,$$

so also

$$x(n) + y(n)\sqrt{b} = (x + y\sqrt{b})(x(1) - y(1)\sqrt{b})^n(x(n) + y(n)\sqrt{b}) = x + y\sqrt{b}.$$

Therefore, $(x, y) = (x(n), y(n))$, which was to be shown. □

This gives us a first recursive description.

Corollary 2.

$$x(n+1) = x(1)x(n) + by(1)y(n)$$

and

$$y(n+1) = y(1)x(n) + x(1)y(n).$$

From this we can also find a second recursive description.

Lemma 11.

$$x(n+2) = 2x(1)x(n+1) + (by(1)^2 - x(1)^2)x(n)$$

and

$$y(n+2) = 2x(1)y(n+1) + (by(1)^2 - x(1)^2)y(n).$$

In order to use this we still have to guarantee the existence of a smallest nontrivial solution $(x(1), y(1))$, and preferably some way to find this smallest solution. As noted before this will not be possible without restrictions on b. An easy restriction that will work is $b = a^2 - 1$ for some $a \in \mathbb{N}$ with $a \geq 2$.

Lemma 12. *The smallest non-trivial solution to $x^2 - (a^2 - 1)y^2 = 1$ with $a \geq 2$ is $(a, 1)$.*

Proof: Obviously $(a, 1)$ is a nontrivial solution. Also, any smaller solution $(\tilde{x}, \tilde{y})$ would have to have $\tilde{y} < 1$, so $\tilde{y} = 0$ and $\tilde{x} = 1 + (a^2 - 1)0^2 = 1$, the trivial solution. □

This allows us to simplify the recursion formulas by filling in the vaules for $(x(1), y(1))$.

Corollary 3. *Let $\{(x_a(k), y_a(k))\}$ be the solutions to $x^2 - (a^2 - 1)y^2 = 1$ for some $a \geq 2$. Then:*

i) $(x_a(0), y_a(0)) = (1, 0), (x_a(1), y_a(1)) = (a, 1)$.

ii) $x_a(n+1) = ax_a(n) + (a^2 - 1)y_a(n)$.
$y_a(n+1) = x_a(n) + ay_a(n)$.

iii) $x_a(n+2) = 2ax_a(n+1) - x_a(n)$.
$y_a(n+2) = 2ay_a(n+1) - y_a(n)$.

With these recursions we can find the properties we need.

3.3. **Properties of $(x_a(n), y_a(n))$.** The first important properties are an upper and lower bound for $y_a(n)$.

Lemma 13. *For $a \geq 2$,*

$$(2a-1)^n \leq y_a(n+1) \leq (2a)^n$$

Proof: By induction. For $n = 0$ the lemma holds, since $1 \leq y_a(1) = 1 \leq 1$.
Suppose the lemma holds for $y_a(n+1)$. Then, since $y_a(n+1) > y_a(n)$,

$$(2a)^{n+1} = 2a(2a)^n \geq 2ay_a(n+1) - y_a(n)$$

$$\geq (2a-1)y_a(n+1) \geq (2a-1)(2a-1)^n = (2a-1)^{n+1},$$

so by 3 part iii) the lemma holds for $y_a(n+2)$. □

This is a very important lemma. Not only will it be useful in finding a diophantine description of $\{(n, y_a(n))\}$, it also gives an approximation of $(2a)^n$ once the description of this graph has been found. Our goal is to show that every exponential diophantine set is diophantine, and the most obvious way to do that would be to find a diophantine description of c^d. This approximation will eventually give us exactly that. For now we'll continue finding the diophantine description of the graph of $y_a(n)$ though.

We can also look at relations between $y_a(n)$ and $y_b(n)$.

Lemma 14. ***Congruence lemma:***
For $a, b \geq 2$ with $a \neq b$ and $n \geq 0$,
i) $x_a(n) \equiv x_b(n) \mod (a-b)$
ii) $y_a(n) \equiv y_b(n) \mod (a-b)$
iii) $y_a(n) \equiv n \mod (a-1)$

Proof:
i) By induction. For $n = 0$, $x_a(n) = x_b(n) = 1$, so also $x_a(n) \equiv x_b(n) \mod (a-b)$. For $n = 1$, $x_a(n) = a$, $x_b(n) = b$, so $x_a(n) \equiv x_b(n) \mod (a-b)$.
Now suppose that for some $n \geq 0$, one has $x_a(n) \equiv x_b(n) \mod (a-b)$ and $x_a(n+1) \equiv x_b(n+1) \mod (a-b)$. Then $x_a(n+2) = 2ax_a(n+1) - x_a(n)$ and $x_b(n+2) = 2bx_b(n+1) - x_b(n)$, so $x_a(n+2) - x_b(n+2) \equiv 2ax_a(n+1) - 2bx_b(n+1) - x_a(n) + x_b(n) \equiv (2a-2b)x_a(n+1) \equiv 0 \mod a-b$. This completes the induction, and thereby the proof of part i).
ii) For $n = 0$, $y_a(n) = y_b(n) = 0$, so also $y_a(n) \equiv y_b(n) \mod (a-b)$. For $n = 1$, $y_a(n) = y_b(n) = 1$, so also $y_a(n) \equiv y_b(n) \mod (a-b)$.
Furthermore, $y_a(n)$ satisfies the same recursion rule as $x_a(n)$, so the induction step used for part i) also works here.
iii) Again, by induction. For $n = 0$, $y_a(n) = 0$, so also $y_a(n) \equiv n \mod (a-1)$. For $n = 1$, $y_a(n) = 1$, so also $y_a(n) \equiv n \mod (a-1)$.
Suppose that for some $n \geq 0$ one has $y_a(n) \equiv n \mod (a-1)$ and $y_a(n+1) \equiv n+1 \mod (a-1)$. Then $y_a(n+2) \equiv 2ay_a(n+1) - y_a(n) \equiv 2(n+1) - n \equiv n+2 \mod (a-1)$. This completes the induction, and thereby the proof of part iii). □

Lemma 15. ***First step-down lemma:***
For $a \geq 2$ and $m, n > 0$,
i) $y_a(m) | y_a(n)$ *if and only if* $m | n$
ii) $y_a(m)^2 | y_a(n)$ *if and only if* $m y_a(m) | n$

Proof:
i) Suppose $m|n$. Then $x_a(n) + \sqrt{a^2-1}y_a(n) = (x_a(m) + \sqrt{a^2-1}y_a(m))^k$ for some $k \in \mathbb{N}$. One can write

$$(x_a(m) + \sqrt{a^2-1}y_a(m))^k = \sum_{i=0}^{k} \binom{k}{i} x_a(m)^{k-i} \sqrt{a^2-1}^i y_a(m)^i$$

$$= x_a(m)^k + cy_a(m) + d\sqrt{a^2-1}y_a(m)$$

for some integers c, d, so $y_a(n) = dy_a(m)$, so $y_a(m)|y_a(n)$.
Now suppose $y_a(m)|y_a(n)$, and $m \not| n$. Then there are unique $k, l \in \mathbb{N}$ with $0 < l < m$ such that

$$x_a(n) + \sqrt{a^2-1}y_a(n) = (x_a(mk) + \sqrt{a^2-1}y_a(mk))(x_a(l) + \sqrt{a^2-1}y_a(l))$$

$$= x_a(mk)x_a(l) + (a^2-1)y_a(mk)y_a(l) + \sqrt{a^2-1}(x_a(mk)y_a(l) + x_a(l)y_a(mk)),$$

so $y_a(n) = x_a(mk)y_a(l) + x_a(l)y_a(mk)$. However, $x_a(mk)$ and $y_a(mk)$ are relatively prime. Since $y_a(m)|y_a(mk)$ this also implies that $x_a(mk)$ and $y_a(m)$ are relatively prime. Also, $y_a(l) < y_a(m)$, so $y_a(m) \not| y_a(l)$. Therefore, $y_a(m) \not| x_a(mk)y_a(l)$, so $y_a(m) \not| x_a(mk)y_a(l) + x_a(l)y_a(mk) = y_a(n)$. This contradicts the assumptions, so if $y_a(m)|y_a(n)$ then $m|n$.
ii) First note that $my_a(m)|n$ implies $m|n$, and by part i) so does $y_a(m)^2|y_a(n)$. We can therefore

without loss of generality asume $m|n$ for this proof. Then $n = mk$ for some $k \in \mathbb{N}$, so $x_a(n) + \sqrt{a^2-1}y_a(n) = (x_a(m) + \sqrt{a^2-1}y_a(m))^k$. Writing out the sum gives

$$(x_a(m) + \sqrt{a^2-1}y_a(m))^k = \sum_{i=0}^{k} \binom{k}{i} x_a(m)^{k-i}\sqrt{a^2-1}^i y_a(m)^i$$

$$= \sum_{i \text{ even}} \binom{k}{i} x_a(m)^{k-i}(a^2-1)^{\frac{i}{2}} y_a(m)^i$$

$$+\sqrt{a^2-1}\sum_{i \text{ odd}} \binom{k}{i} x_a(m)^{k-i}(a^2-1)^{\frac{i-1}{2}} y_a(m)^i,$$

so

$$y_a(n) = \sum_{i \text{ odd}} \binom{k}{i} x_a(m)^{k-i}(a^2-1)^{\frac{i-1}{2}} y_a(m)^i = \binom{k}{1} x_a(m)^{k-1} y_a(m) + y_a(m)^3 \cdot c$$

for some $c \in \mathbb{N}$. Therefore, and since $y_a(m)$ and $x_a(m)$ are relatively prime, $y_a(m)^2|y_a(n)$ if and only if $y_a(m)|\binom{k}{1} = k$. Since k was chosen the number such that $n = km$, this is equivalent to $my_a(m)|n$. □

Lemma 16. *Let $a \geq 2$. Then $y_a(n)$ is even if and only if n is even, and $x_a(n)$ is even if and only if n is odd and a is even.*

Proof:
By the recursions $x_a(n+2) = 2ax_a(n+1) - x_a(n)$ and $y_a(n+2) = 2ay_a(n+1) - y_a(n)$, $x_a(n+2)$ is even if and only if $x_a(n)$ is even, and $y_a(n+1)$ is even if and only if $y_a(n)$ is even. For any a one has $y_a(0) = 0$, $y_a(1) = 1$, so $y_a(n)$ is even if and only if n is even. Also, $x_a(0) = 1, x_a(1) = a$, so if a is odd, $x_a(n)$ is odd for all n. If a is even, $x_a(n)$ is even if and only if n is odd. □

Lemma 17. ***Second step-down lemma:***
If $a \geq 2$, $n \geq 1$, $i, j \in \mathbb{N}$ and $y_a(i) \equiv y_a(j) \mod x_a(n)$ then $j \equiv \pm i \mod 2n$.

For the proof of this lemma we will use some other, more simple lemmas.

Lemma 18. *For $a \geq 2, n \geq 0$, one has $x_a(n) > \sum_{i=0}^{n} y_a(i)$.*

Proof: By induction. $x_a(0) = 1 > 0 = y_a(0)$. Suppose $x_a(n) > \sum_{i=0}^{n} y_a(i)$. Then

$$x_a(n+1) = ax_a(n) + (a^2-1)y_a(n) > a(\sum_{i=0}^{n} y_a(i)) + (a^2-1)y_a(n)$$

$$\geq 2(\sum_{i=0}^{n} y_a(i)) + (a^2-1)y_a(n) \geq \sum_{i=0}^{n} y_a(i) + a^2 y_a(n)$$

$$\geq \sum_{i=0}^{n} y_a(i) + 2ay_a(n) \geq \sum_{i=0}^{n} y_a(i) + y_a(n+1) = \sum_{i=0}^{n+1} y_a(i)$$

□

Lemma 19. *For any $a > 1$ and $c, d \in \mathbb{N}$,*
i) $x_a(c+d) = x_a(c)x_a(d) + (a^2-1)y_a(c)y_a(d)$
ii) *If $d \leq c$, then* $x_a(c-d) = x_a(c)x_a(d) - (a^2-1)y_a(c)y_a(d)$
iii) $y_a(c+d) = x_a(c)y_a(d) + y_a(c)x_a(d)$
iv) *If $d \leq c$, then* $y_a(c-d) = -x_a(c)y_a(d) + y_a(c)x_a(d)$
v) $x_a(2c) = 2x_a(c)^2 - 1$
vi) $y_a(2c) = 2x_a(c)y_a(c)$

This lemma follows easily from $x_a(c+d) + y_a(c+d)\sqrt{a^2-1} = (x_a(c) + y_a(c)\sqrt{a^2-1})(x_a(d) + y_a(d)\sqrt{a^2-1})$.

Proof of the second step-down lemma: Let's first deal with the case $a = 2, n = 1$. Then the

statement of the lemma is that if $y_a(i) \equiv y_a(j) \bmod 2$ then $i \equiv j \bmod 2$. This is true since, by lemma 16, $y_a(i) \equiv i \bmod 2$ and $y_a(j) \equiv j \bmod 2$. For the rest of the proof, take $(a, n) \neq (2, 1)$. First, I'll show that $y_a(m) \bmod x_a(n)$ has a period dividing $4n$.

$$y_a(m+4n) = x_a(m)y_a(4n) + y_a(m)x_a(4n)$$

$$= x_a(m)2x_a(2n)y_a(2n) + y_a(m)(2x_a(2n)^2 - 1)$$

$$= x_a(m)2x_a(2n)x_a(n)y_a(n) + y_a(m)(2(2x_a(n)^2 - 1)^2 - 1)$$

$$= x_a(m)2x_a(2n)x_a(n)y_a(n) + y_a(m)(2(4x_a(n)^4 - 4x_a(n) + 1) - 1)$$

so

$$y_a(m+4n) \equiv 0 + y_a(m)(2(4 \cdot 0^4 - 4 \cdot 0 + 1) - 1) \equiv y_a(m) \bmod x_a(n).$$

Since this holds for any m, $y_a(m) \bmod x_a(n)$ has a period dividing $4n$.
Now note that for any $m \leq n$,

$$y_a(m) \equiv y_a(m) \bmod x_a(n). \tag{1}$$

Also,

$$y_a(2n-m) = -x_a(2n)y_a(m) + y_a(2n)x_a(m) =$$

$$-(2x_a(n)^2 - 1)y_a(m) + 2x_a(n)y_a(n)x_a(m),$$

so

$$y_a(2n-m) \equiv y_a(m) \bmod x_a(n), \tag{2}$$

and

$$y_a(2n+m) = x_a(2n)y_a(m) + y_a(2n)x_a(m)$$

$$= (2x_a(n)^2 - 1)y_a(m) + 2x_a(n)y_a(n)x_a(m),$$

so

$$y_a(2n+m) \equiv -y_a(m) \bmod x_a(n). \tag{3}$$

Finally,

$$y_a(4n-m) = -x_a(4n)y_a(m) + y_a(4n)x_a(m)$$

$$= -(2(4x_a(n)^4 - 4x_a(n) + 1) - 1)y_a(m) + y_a(4n)x_a(m),$$

so

$$y_a(4n-m) \equiv -(2(4 \cdot 0^4 - 4 \cdot 0 + 1) - 1)y_a(m) + y_a(0)x_a(m) \equiv -y_a(m) \bmod x_a(n) \tag{4}$$

These four will turn out to be the only ones congruent to $\pm y_a(m)$ modulo $x_a(n)$.

Let $0 \leq k < l \leq n$. Then by lemma 18, $0 < y_a(l) \pm y_a(k) < x_a(n)$, so $y_a(k) \not\equiv \pm y_a(l) \bmod x_a(n)$.
Let $0 \leq k < n$. Then $2y_a(k) < x_a(n)$, so $y_a(k) \not\equiv -y_a(k) \bmod x_a(n)$. Let $k = n$. Then $y_a(k) = y_a(n)$ and $x_a(n)$ are relatively prime, so $y_a(k) \equiv -y_a(k) \bmod x_a(n)$ if and only if $x_a(n) \leq 2$. Since $n \geq 1$, the only case that can happen is if $a = 2, n = 1$. We are in the case that $(a, n) \neq (2, 1)$ though, so $y_a(k) \not\equiv -y_a(k)$

Let $0 \leq i', j' < 4n$, such that $y_a(i') \equiv y_a(j') \bmod x_a(n)$. Then there are $0 \leq \tilde{i}, \tilde{j} <\leq n$ such that $i' \in \{\tilde{i}, 2n - \tilde{i}, 2n + \tilde{i}, 4n - \tilde{i}\}$ and $i' \in \{\tilde{i}, 2n - \tilde{i}, 2n + \tilde{i}, 4n - \tilde{i}\}$. With congruences 1,2,3 and 4 this implies that $y_a(\tilde{i}) \equiv \pm y_a(\tilde{j}) \bmod x_a(n)$. Then $\tilde{i} = \tilde{j}$, so $i' \equiv \pm j' \bmod 2n$. Since $y_a(m) \bmod x_a(n)$ has a period dividing $4n$, this implies that for any i, j, if $y_a(i) \equiv y_a(j) \bmod x_a(n)$, then $i \equiv \pm j \bmod x_a(n)$. □

3.4. **A different proof for the congruence lemma.** Usually, if a proof of a property of the solutions of a Pell equation is given, the proof will use elementary methods. While such proofs are of course sufficient, they do tend to hide some of the structure of the situation. In some sense they show that the lemmas hold, but not why the lemmas hold. In this alternative proof of the congrunece lemma I will try to show some of the structure behind the proof, which might make it more insightful to some people.

The solutions we're looking at are of the form $x_a(n) + y_a(n)\sqrt{a^2-1} \in \mathbb{Z}[\sqrt{a^2-1}]$. In order to get a slightly more convenient description, we can use $\mathbb{Z}[\sqrt{a^2-1}] \cong \mathbb{Z}[X]/_{(X^2-a^2+1)}$, so the solutions to the Pell equation correspond to $x_a(n) + y_a(n)X \in \mathbb{Z}[X]/_{(X^2-a^2+1)}$.
In fact, as we have seen before, $(x_a(n) + y_a(n)X)(x_a(n) - y_a(n)X) = x_a(n)^2 - (a^2-1)y_a(n)^2 = 1$, so $x_a(n) + y_a(n)X$ is a unit, the solutions to the Pell equation correspond to elements of $(\mathbb{Z}[X]/_{(X^2-a^2+1)})^*$. Now, let's start with the last part of the Congruence Lemma.

Lemma 20. *For $a, b \geq 2$ with $a \neq b$ and $n \geq 0$,*
$y_a(n) \equiv n \ mod\ (a-1)$

Proof: We only need equivalence modulo $(a-1)$, so the first thing to do is to use the ring homomorphism

$$\mathbb{Z}[X]/_{(X^2-a^2+1)} \xrightarrow{\text{mod } (a-1)} \mathbb{Z}[X]/_{X^2}$$

which of course induces a group homomorphism

$$(\mathbb{Z}[X]/_{(X^2-a^2+1)})^* \xrightarrow{\text{mod } (a-1)} (\mathbb{Z}[X]/_{X^2})^*.$$

The elements of $(\mathbb{Z}[X]/_{X^2})^*$ are exactly the elements $x + yX$ with $x \in \mathbb{Z}^*$ and $y \in \mathbb{Z}$. In fact, there is an isomorphism

$$f : (\mathbb{Z}[X]/_{X^2})^* \to \mathbb{Z}^* \times \mathbb{Z}$$

given by $f(x + yX) = (x, x^{-1}y)$. Now, $(\mathbb{Z}[X]/_{(X^2-a^2+1)})^* \ni x_a(n) + y_a(n)X = (a+X)^n \mapsto (1+X)^n \in (\mathbb{Z}[X]/_{X^2})^*$, and $(1+X)^n \mapsto f(1+X)^n = (1, 1^{-1}1)^n = (1, n) \in \mathbb{Z}^* \times \mathbb{Z}$. And since $f^{-1}(1, n) = 1 + nX$, this implies that $x_a(n) \equiv 1 \bmod (a-1)$ and $y_a(n) \equiv n \bmod (a-1)$. □

Now let's take a look at the first two statements of the congruence lemma.

Lemma 21. *For $a, b \geq 2$ with $a \neq b$ and $n \geq 0$,*
i) $x_a(n) \equiv x_b(n)\ mod\ (a-b)$
ii) $y_a(n) \equiv y_b(n)\ mod\ (a-b)$

Proof: The solutions $(x_a(n), y_a(n))$ and $(x_b(n), y_b(n))$ correspond to elements of $(\mathbb{Z}[X,Y]/_{(X^2-a^2+1, Y^2-b^2+1)})^*$. Working modulo $(a-b)$ simplifies this,

$$(\mathbb{Z}[X,Y]/_{(X^2-a^2+1, Y^2-b^2+1)})^* \to (\mathbb{Z}/_{(}a-b)[X,Y]/_{(X^2-a^2+1, Y^2-a^2+1)})^*$$

with

$$(x_b(n) + y_b(n)Y) = (b+Y)^n \mapsto (x_b(n) + y_b(n)Y)$$

$$\equiv (b+Y)^n \equiv (a+Y)^n \equiv (x_a(n) + y_a(n)Y),$$

since $a \equiv b$. Therefore, $x_a(n) \equiv x_b(n) \bmod (a-b)$ and $y_a(n) \equiv y_b(n) \bmod (a-b)$. □

I will now return to using the elementary proof methods. Although a bit less insightful, elementary methods are more self contained, and can be used to prove a lot of things that are harder to prove using the group structure.

3.5. **The graph of $y_a(n)$.** Now we have what we need to find a diophantine description of the graph of $y_a(n)$. First, let's consider what we can do with diophantine equations. Firstly, we can require polynomial equations, $P(\mathbf{r}) = Q(\mathbf{s})$, without adding further unknowns. This will for example enable us to make sure that (x, y) is a solution to the Pell equation, by $x^2 - (a^2 - 1)y^2 = 1$. We can also make requirements with added unknowns, which will allow us to create some other useful relations. For example, $r < s$ if and only if $\exists t \in \mathbb{N} : r + t + 1 = s$. Also, $r|s$ if and only if $\exists t \in \mathbb{Z} : rt = s$, which is equivalent to $\exists t \in \mathbb{N} : (rt - s)(rt + s) = 0$. There are many more relations that can be given in such a way, but these are the most common ones, and the ones we'll need here.

I will first give an outline of the method of obtaining the required diophantine description, before proving the method does indeed work. The proof given here is based on the proof given by Matijasevic and Robinson in [8].

Suppose we are given an a and an n. We want to find $y_a(n)$, so we will first have to find a candidate. Consider

$$d^2 - (a^2 - 1)y^2 = 1.$$

This will ensure that $d = x_a(k), y = y_a(k)$ for some k. The goal is to ensure that $k = n$. The most obvious attempt would be to try to use the Congruence Lemma. If $y_a(k) \equiv n \bmod a - 1$, then also $k \equiv n \bmod a - 1$. If we could guarantee $a - 1 > k, n$ we'd be done. However, we want our method to be able to find $y_a(n)$ for any a and n, so we do not have any control over a and n, so we cannot assume $a - 1 > n$.

Another relatively obvious attempt would be to use the Second step-down lemma. If we can get $y_a(k) \equiv y_a(n) \bmod x_a(m)$, then we would have $k \equiv \pm n \bmod 2m$. If we could then take $m \geq k, n$, we would get $k = n$. Since we haven't assumed anything about m yet, this might work.

First, let't take care of $m \geq k, n$. From lemma 13 and the fact that $y_a(0) = 0$, one obtains $y_a(k) \geq k$. We can also take $y_a(k) \geq n$, by simply taking $y \geq n$. If we then use the First step-down lemma, we can take $y_a(k)^2 | y_a(m)$, so we get $k y_a(k) | m$. In particular, this will guarantee $m \geq y_a(k) \geq k, n$.

Left to do is make sure that $y_a(k) \equiv y_a(n) \bmod 2x_a(m)$. This turns out to be hard. What we can do, however, is introduce another solution, and get $y_a(k) \equiv y_a(l) \bmod 2x_a(m)$, and have $l \equiv n \bmod 2y_a(k)$. We once again run into some trouble with lack of control over a at this point, so we simply take $y_b(l)$, with $a \equiv b \bmod x_a(m)$. Now all that is left to do is take $y_b(l) \equiv n \bmod 2x_a(k)$, and $b \equiv 1 \bmod 2x_a(k)$. These two conditions combined give $l \equiv n \bmod 2y_a(k)$, so we're done.

Note that all this only gives sufficient requirements for $k = n$. We will still have to show that they are also neseccary requirements.

Theorem 2. *Suppose $a \geq 2$ and $n, y > 0$. Then $y = y_a(n)$ if and only if the following system can be satisfied in $\mathbb{N}$.*

(A) $d^2 - (a^2 - 1)y^2 = 1$
(B) $e^2 - (a^2 - 1)f^2 = 1$
(C) $g^2 - (b^2 - 1)h^2 = 1$
(D) $n \leq y$
(E) $y^2 | f$
(F) $e | y - h$
(G) $e | a - b$
(H) $2y | h - n$
(I) $2y | b - 1$

Proof:
First, to show that if the system can be satisfied, then $y = y_a(n)$. From (A), (B) and (C) we know

that

$$d = x_a(k), y = y_a(k), e = x_a(m), f = y_a(m), g = x_b(l), h = y_b(l) \tag{5}$$

for some $k, l, m \in \mathbb{N}$. With (F) this implies

$$y_a(k) \equiv y_b(l) \bmod x_a(m). \tag{6}$$

Also, by the Congruence Lemma,

$$y_a(l) \equiv y_b(l) \bmod (a - b). \tag{7}$$

(5) and (6), combined with (G) imply that

$$y_a(k) \equiv y_a(l) \bmod x_a(m). \tag{8}$$

By the second step-down lemma this implies

$$k \equiv \pm l \bmod 2m. \tag{9}$$

Using the first step-down lemma, (E) implies that

$$k y_a(k) | m, \tag{10}$$

so together with (9) we get

$$k \equiv \pm l \bmod 2y_a k. \tag{11}$$

Also, by the Congruence Lemma and (I),

$$y_b(l) \equiv l \bmod 2y_a(k). \tag{12}$$

(H) implies that

$$y_b(l) \equiv n \bmod 2y_a(k). \tag{13}$$

Obviously, from (11), (12) and (13) it follows that

$$k \equiv \pm n \bmod 2y_a(k). \tag{14}$$

Since $k \leq y_a(k)$ and $n \leq y_a(k)$ by (D), this implies

$$k = n, \tag{15}$$

which is what was to be shown.

Now, to show that if $y = y_a(n)$, then the system can be satisfied. Taking $d = x_a(n), y = y_a(n), e = x_a(k), f = y_a(k), g = x_b(l), h = y_b(l)$ for some k, b, l will guarantee (A), (B), (C) and (D). By the second step-down lemma, (E) will be guaranteed if and only if k is chosen a multiple of ny. Later on it will be useful if we choose k a multiple of $2ny$.
Then $y^2 | f$. Also $e^2 = (a^2 - 1)f^2 + 1$, so $y^2 | e^2 - 1$, so e and y are relatively prime. k was chosen even, so $e = x_a(k)$ is odd. Therefore, e and $2y$ are also relatively prime.
Therefore, there is a b that satisfies both $e | a - b$ and $2y | b - 1$. Taking this b guarantees (G) and (I).
For the last two equations, simply take $l = n$. Since $y_b(l) \equiv l \bmod b - 1$ and $2y | b - 1$ this implies $h \equiv n \bmod 2y$, so $2y | h - n$. Also, $y_a(l) \equiv y_b(l) \bmod a - b$, $e | a - b$, $h = y_b(l) = y_b(n)$ and $y = y_a(n) = y_a(l)$, so $y \equiv h \bmod e$, which is equivalent to $e | y - h$. This implies (E) and (H), which were the last two equations that needed to be satisfied. □

This system of equations almost gives us a diophantine description of the graph of $y_a(n)$. All that is left to do is to compile them into a diophantine equation.

Corollary 4.

$$\{(y_a(n), n)\} = \{(y, n) : \exists (d, e, f, g, h, t_1, t_2, t_3, t_4, t_5, t_6) \in \mathbb{N}^{11} \ \ [(d^2 - (a^2 - 1)y^2 - 1)^2$$
$$+ (e^2 - (a^2 - 1)f^2 - 1)^2 + (g^2 - (b^2 - 1)h^2 - 1)^2 + (n - y + t_1)^2 + (t_2 y^2 - f)^2$$
$$+ (t_3 e + y - h)^2 (t_3 e - y + h)^2 + (t_4 e + a - b)^2 (t_4 e - a + b)^2 + (t_5 2y + h - n)^2 (t_5 2y - h + n)^2$$
$$+ (t_6 2y + b - 1)^2 (t_6 y - b + 1)^2 = 0]\}$$

Some optimizations can be done to reduce the amount of unknowns that are needed, but at this point this description will suffice.

3.6. **The graph of exponentiation.** Now let's look back at lemma 13. For any $a \geq 2$ we have

$$(2a-1)^n \leq y_a(n+1) \leq (2a)^n$$

Therefore, $\frac{y_{az}(n+1)}{y_a(n+1)}$ is an approximation of z^n. To be more precise,

$$\frac{(2az-1)^n}{(2a)^n} \leq \frac{y_{az}(n+1)}{y_a(n+1)} \leq \frac{(2az)^n}{(2a-1)^n}$$

If we can get

$$z^n - \frac{1}{2} < \frac{y_{az}(n+1)}{y_a(n+1)} < z^n + \frac{1}{2}$$

we can find a diophantine description of the graph of exponentiation, which would show every exponential diophantine set is diophantine. Let's take a look at what is needed to guarantee the first inequality. First, we'll need that for any real $0 \leq \alpha \leq 1$ and $n \geq 0$ the inequality

$$(1-\alpha)^n \geq 1 - n\alpha$$

holds. This follows by induction, using that since $0 \leq (1-\alpha)^k \leq 1$ one has $(1-\alpha)^k - (1-\alpha)^{k+1} = \alpha(1-\alpha)^k \leq \alpha$.
We have

$$\frac{(2az-1)^n}{(2a)^n} \leq \frac{y_{az}(n+1)}{y_a(n+1)}, \tag{16}$$

and

$$\frac{(2az-1)^n}{(2a)^n} = z^n(1-\frac{1}{2az})^n \geq z^n(1-\frac{n}{2az}) = z^n - \frac{z^{n-1}n}{2a}. \tag{17}$$

Together, this gives

$$z^n - \frac{1}{2}\frac{z^{n-1}n}{a} \leq \frac{y_{az}(n+1)}{y_a(n+1)} \tag{18}$$

so if we take

$$a > z^{n-1}n \tag{19}$$

we will have

$$z^n - \frac{1}{2} < \frac{y_{az}(n+1)}{y_a(n+1)}. \tag{20}$$

For the second inequality we will need another small fact. For $0 \leq \alpha \leq \frac{1}{2}$ the inequality

$$(1-\alpha)^{-1} \leq 1 + 2\alpha$$

holds. This can be seen by multipliying both sides by $1-\alpha$, which shows the given inequality is equivalent to $1 \leq 1 + \alpha - 2\alpha^2$, which holds since $0 \leq \alpha \leq \frac{1}{2}$.
We have

$$\frac{y_{az}(n+1)}{y_a(n+1)} \leq \frac{(2az)^n}{2a-1} = z^n(1-\frac{1}{2a})^{-n} = z^n((1-\frac{1}{2a})^n)^{-1} \leq z^n(1-\frac{n}{2a})^{-1} \leq z^n(1+\frac{n}{a}) \tag{21}$$

for $a \geq n$. So if we take

$$a > nz^n \tag{22}$$

we will have

$$z^n - \frac{1}{2} < \frac{y_{az}(n+1)}{y_a(n+1)} < z^n + \frac{1}{2}, \tag{23}$$

so z^n will be the integer closest to $\frac{y_{az}(n+1)}{y_a(n+1)}$.

Theorem 3. *Let $p, z > 0, n \in \mathbb{N}$. Then $p = z^n$ if and only if the following system can be satisfied.*
(A) $a = y_{nz+2}(n+1)$
(B) $c = y_{az}(n+1)$
(C) $d = y_a(n+1)$
(D) $(2pd - 2c)^2 < d^2$

Proof: First, to show that if the system can be satisfied, then $p = z^n$. From (A) we have

$$a = y_{nz+2}(n+1) \geq (2nz+3)^n \geq (nz)^n + 1 > nz^n, \tag{24}$$

so with (B) and (C) we know that z^n is the closest integer to $\frac{c}{d}$. From (D) we know that

$$(2pd - 2c)^2 < d^2$$

Dividing both sides by $4d^2$ gives

$$(p - \frac{c}{d})^2 < \frac{1}{4},$$

so p is the closest integer to $\frac{c}{d}$, so $p = z^n$.
Then, to show that if $p = z^n$ the system can be satisfied. (A), (B) and (C) can always be satisfied. Then since p is the closest integer to $\frac{c}{d}$, (D) is also satisfied. □

Adding the special case $z = p = 0$ will give a diophantine description of the graph of exponentiation.

Corollary 5. *There is a polynomial $P(p, z, n, \boldsymbol{x})$ such that $p = z^n$ if and only if $\exists \boldsymbol{x} : P(p, z, n, \boldsymbol{x}) = 0$.*

The only difference between polynomials and exponential polynomials is that exponential polynomials can contain exponentiation. The graph of exponentiation is diophantine, so we should have some control over exponentiation using only polynomials. This will of course not be as straightforward as finding a polynomial Q for every exponential polynomial E such that $Q(\mathbf{x}) = E(\mathbf{x})$ for all $\mathbf{x}$, since that is in general not possible. What we can do is the following:

Lemma 22. *For every exponential polynomial $E(\boldsymbol{a})$ there are polynomials $Q(\boldsymbol{b})$ and $R(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x})$ such that for every $\boldsymbol{a}$: (A) for any $\boldsymbol{b}$, if $\exists \boldsymbol{x} : R(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) = 0$, then $E(\boldsymbol{a}) = Q(\boldsymbol{b})$*
(B) $\exists \boldsymbol{b} \exists \boldsymbol{x} : R(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) = 0$.

Proof: The idea of the proof to replace every occurance of z^n by p, which we can do provided that $\exists \mathbf{x} : P(p, z, n, \mathbf{x}) = 0$. Everything else is just the bookkeeping to make it work.
An exponential polynomial can be generated by applying the following rules a finite number of times. (i) Variables are exponential polynomials.
(ii) For $c \in \mathbb{N}$, c is an exponential polynomial.
(iii) If E_1 and E_2 are exponential polynomials, then so are $E_1 + E_2$ and $E_1 E_2$.
(iv) If E_1 and E_2 are exponential polynomials, then so is $E_1^{E_2}$.
For the base cases (i) and (ii) the lemma obviously holds, one can take $R = 0$, $Q = E$.
Suppose the lemma holds for exponential polynomials $E_1(\mathbf{a}_1)$ and $E_2(\mathbf{a}_2)$, so there are polynomials $Q_1(\mathbf{b}_1), Q_2(\mathbf{b}_2), R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1), R_2(\mathbf{a}_2, \mathbf{b}_2, \mathbf{x}_2)$ with the required properties.
Then for every $\mathbf{a}_1, \mathbf{a}_2$:
(1) $E_1(\mathbf{a}_1) + E_2(\mathbf{a}_2) = R_1(\mathbf{b}_1) + R_2(\mathbf{b}_2)$ if $\exists \mathbf{x}_1, \mathbf{x}_2 : R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 + R_2(\mathbf{a}_2, \mathbf{b}_2, \mathbf{x}_2)^2 = 0$
(2) $E_1(\mathbf{a}_1) E_2(\mathbf{a}_2) = R_1(\mathbf{b}_1) R_2(\mathbf{b}_2)$ if $\exists \mathbf{x}_1, \mathbf{x}_2 : R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 + R_2(\mathbf{a}_2, \mathbf{b}_2, \mathbf{x}_2)^2 = 0$
(3) $\exists \mathbf{b}_1 \exists \mathbf{x}_1 : R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) = 0$ and $\exists \mathbf{b}_2 \exists \mathbf{x}_2 : R_2(\mathbf{a}_2, \mathbf{b}_2, \mathbf{x}_2) = 0$, so also $\exists \mathbf{b}_1, \mathbf{b}_2 \exists \mathbf{x}_1, \mathbf{x}_2 : R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 + R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 = 0$.
Therefore, the lemma remains true under application of rule (iii). Let $P(p, z, n, \mathbf{y})$ be a polynomial with the property that $p = z^n$ if and only if $\exists \mathbf{y} : P(p, z, n, \mathbf{y}) = 0$. Then:
(1) $E_1(\mathbf{a}_1)^{E_2(\mathbf{a}_2)} = p$ if $\exists \mathbf{x}_1, \mathbf{x}_2, \mathbf{y} : P(p, Q_1(\mathbf{b}_1), Q_2(\mathbf{b}_2, \mathbf{y})^2 + R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 + R_2(\mathbf{a}_2, \mathbf{b}_2, \mathbf{x}_2)^2 = 0$
(2) If one takes b_1, b_2 such that $\exists \mathbf{x}_1, \mathbf{x}_2 : R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 + R_1(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1)^2 = 0$ and $p = E_1(\mathbf{b}_1)^{E_2(\mathbf{b}_2)}$, then $\exists \mathbf{y} : P(p, z, n, \mathbf{y}) = 0$.
Therefore the lemma remains true under application of rule (iv), so the lemma holds. □

From this lemma one can easily obtain the result we were looking for.

Theorem 4. *Every exponential diophantine set is diophantine.*

Proof: Let S be any exponential diophantine set. Then

$$S = \{\mathbf{a} : \exists \mathbf{x}[E_1(\mathbf{a}, \mathbf{x}) - E_2(\mathbf{a}, \mathbf{x}) = 0]\}$$

where $E_1(\mathbf{a}, \mathbf{x})$ and $E_2(\mathbf{a}, \mathbf{x})$ are exponential polynomials. Then, by lemma 22, there exists polynomials $Q_1(\mathbf{b}_1), Q_2(\mathbf{b}_2), R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)$ and $R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)$ such that if $\exists \mathbf{y}_1 : R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1) = 0$ then $Q_1(\mathbf{b}_1) = E_1(\mathbf{a}, \mathbf{x})$ and if $\exists \mathbf{y}_2 : R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2) = 0$ then $Q_2(\mathbf{b}_2) = E_2(\mathbf{a}, \mathbf{x})$. Also, for any $\mathbf{a}, \mathbf{x}$, $\exists \mathbf{b}_1, \mathbf{b}_2 \exists \mathbf{y}_1, \mathbf{y}_2 : R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1) = R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2) = 0$.
Take

$$\tilde{S} = \{\mathbf{a} : \exists \mathbf{x}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{y}_1, \mathbf{y}_2$$
$$[(Q_1(\mathbf{b}_1) - Q_2(\mathbf{b}_2))^2 + R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = 0]\}.$$

If $(Q_1(\mathbf{b}_1) - Q_2(\mathbf{b}_2))^2 + R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = 0$, then in particular $R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1) = 0$ so $Q_1(\mathbf{b}_1) = E_1(\mathbf{a}, \mathbf{x})$ and $R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2) = 0$ so $Q_2(\mathbf{b}_2) = E_2(\mathbf{a}, \mathbf{x})$. Therefore, $E_1(\mathbf{a}, \mathbf{x}) - E_2(\mathbf{a}, \mathbf{x}) = 0$.

Take any $\mathbf{a} \in \tilde{S}$. Then $\exists \mathbf{x}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{y}_1, \mathbf{y}_2 [(Q_1(\mathbf{b}_1) - Q_2(\mathbf{b}_2))^2 + R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = 0]\}$, so also $\exists \mathbf{x} : E_1(\mathbf{a}, \mathbf{x}) - E_2(\mathbf{a}, \mathbf{x}) = 0$, so $\mathbf{a} \in S$. Therefore, $\tilde{S} \subset S$.

By lemma 22, for any $\mathbf{a}, \mathbf{x}$, $\exists \mathbf{b}_1, \mathbf{b}_2, \mathbf{y}_1, \mathbf{y}_2 : R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = 0$. This also implies that $Q_1(\mathbf{b}_1) = E_1(\mathbf{a}, \mathbf{x})$ and $Q_2(\mathbf{b}_2) = E_2(\mathbf{a}, \mathbf{x})$.

Take any $\mathbf{a} \in S$. Then $\exists \tilde{x} : E_1(\mathbf{a}, \mathbf{x}) - E_2(\mathbf{a}, \mathbf{x}) = 0$, so also $\exists \mathbf{x}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{y}_1, \mathbf{y}_2 : (E_1(\mathbf{a}, \mathbf{x} - E_2(\mathbf{a}, \mathbf{x})^2 + R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = (Q_1(\mathbf{b}_1) - Q_2(\mathbf{b}_2))^2 + R_1(\mathbf{a}, \mathbf{x}, \mathbf{b}_1, \mathbf{y}_1)^2 + R_2(\mathbf{a}, \mathbf{x}, \mathbf{b}_2, \mathbf{y}_2)^2 = 0$, so $\mathbf{a} \in \tilde{S}$. Therefore $S \subset \tilde{S}$.

Together with $\tilde{S} \subset S$ which was concluded earleir, this implies $S = \tilde{S}$. Since $\tilde{S}$ is diophantine by construction, S is also diophantine. □

4. From r.e. to exponential diophantine

Recall that a recursively enumerable or r.e. set $S \subset \mathbb{N}^n$ is a set for which there is an algorithm that for any $x \in S$ determines that x is in S in a finite number of steps, and for any $x \notin S$ does not determine that x is in S. The goal is now to show that any such set S is exponential diophantine, and therefore also diophantine.
In order to do this we will first need to be more precise about what is meant by an algorithm. Here we will consider algorithms to be the sets of instructions of a Turing machine, as first proposed in 1937 by Turing [10].

4.1. **Turing machines.**

Usually, Turing machines are described as (idealized) physical machines. There are many equivalent descriptions of a Turing machine, one of them is the following.

A Turing machine consists of a tape which extends infinitely into both directions and a read/write head. At evenly spaced intervals the tape has a position where a symbol can be written. The read/write head can read a symbol from such a position, or write a symbol to it.
An algorithm for such a Turing machine consists of a number of states $Q = \{q_1, \cdots, q_m\}$ with q_1 the initial state, a number of accepting states $R = \{r_1, \cdots r_l\}$ with $R \cap Q = \emptyset$, a number of symbols $A = \{\alpha_1, \cdots \alpha_s\}$, a special symbol $\beta \in A$ that is the only symbol that can occur on the tape infinitely often, and a set of instructions that can best be represented by a function $G : Q \times A \to (Q \cup R) \times A \times \{L, R\}$.
Given input in the form of an initial position of the read/write head and initial contents of the tape, the Turing machine can execute the algorithm in the following way:
In each time step where the system is not in an accepting state, it is in a state q_i. The head reads the symbol α in the position it is currently in. Write $G(q_i, \alpha) = (s, \gamma, m)$. The head then writes symbol γ and moves to the left if $m = L$ or to the right if $m = R$, and the state becomes s. This

continues either forever, or until the system halts by reaching an accepting state. If the system halts, the contents of the tape and/or the accepting state the system is in can be considered as output.

One way of making this more precise is the following definition.

Definition 8. *A* Turing machine algorithm *consists of*

- *States* $Q = \{q_1, \cdots, q_m\}$, $m \in \mathbb{N}_{>0}$.
- *Accepting states* $R = \{r_1, \cdots r_l\}$, $l \in \mathbb{N}_{>0}$ *with* $R \cap Q = \emptyset$.
- *An alphabet* $A = \{\alpha_1, \cdots \alpha_s\}, s \in \mathbb{N}, s \geq 2$.
- *A special symbol* $\beta \in A$.
- *A function* $G : Q \times A \to (Q \cup R) \times A \times \{L, R\}$.

A run of such an algorithm can be characterized by three functions. One function T to describe the contents of the tape on every time step and every location, one funtion P to describe the location of the read/write head on every time step and one function S to describe the state on every time step. Likewise, any three such functions can be interpreted as the run of an algorithm if they satisfy a number of extra properties.

Definition 9. *A* run of an algorithm *consists of*

- *A function* $T : \mathbb{N}_{>0} \times \mathbb{Z} \to A$.
- *A function* $P : \mathbb{N}_{>0} \to \mathbb{Z} \cup \{NULL\}$.
- *A function* $S : \mathbb{N}_{>0} \to Q \cup R \cup \{NULL\}$.

with the following properties:

(1) *For every* $i \in \mathbb{N}_{>0}$ *and* $\alpha \in A$ *with* $\alpha \neq \beta$ *there are only finitely many* j *such that* $T(i, j) = \alpha$.
(2) $S(1) = q_1$.
(3) $P(i) = NULL$ *if and only if* $\exists j < i : S(j) \in R$.
(4) $S(i) = NULL$ *if and only if* $\exists j < i : S(j) \in R$.
(5) *For* $i > 1, j \in \mathbb{Z}, \alpha \in A$, $T(i, j) = \alpha$ *if and only if either* $T(i-1, j) = \alpha$ *and* $P(i-1) \neq j$ *or* $\exists s \in Q \cup R \exists m \in \{L, R\} : G(S(i-1), T(i-1, j)) = (s, \alpha, m)$ *and* $P(i-1) = j$.
(6) *For* $i > 1, j \in \mathbb{Z}$, $P(i) = j$ *if and only if either* $P(i-1) = j+1$ *and* $\exists s \in Q \cup R \exists \alpha \in A : G(S(i-1), T(i-1, j+1)) = (s, \alpha, R)$ *or* $P(i-1) = j-1$ *and* $\exists s \in Q \cup R \exists \alpha \in A : G(S(i-1), T(i-1, j-1)) = (s, \alpha, L)$.
(7) *For* $i > 1, s \in Q \cup R$, $S(i) = s$ *if and only if* $\exists \alpha \in A \exists m \in \{L, R\} : G(S(i-1), T(i-1, P(i-1)) = (s, \alpha, m)$.

Here $T(i, j)$ is the contents of the j-th position on the i-th time step, $P(i)$ is the position of the tape on the i-th time step, and $S(i)$ is the state the system is in on the i-th time step. Note that the $i+1$-th position is considered to be to the left of the i-th position.
Property 5 represents the symbols on the tape only changing at the position of the read/write head, and changing into the correct symbol at the position of the read/write head.
Property 6 represents the read/write head moving to the left or right.
Property 7 represents the state changing in the correct way.

4.2. **Bitwise operators.** Out goal will be to find exponential diophantine conditions on a number of variables that hold if and only if a given Turing machine algorithm with a given input eventually halts. In order to do this, we will need a number of techniques using the binary representation of integers.

Definition 10. $A \in \mathbb{N}$ *is a* k-bit integer *if* $A < 2^k$.

We will mostly consider k-bit integers as strings of bits with length k. So while a k-bit integer A is also a $k+1$-bit integer, A considered as a k-bit integer and A considered as $k+1$-bit integer will represent different strings. In the rest of this section, whenever an integer is specified to be k-bit, it will always be considered as a k-bit integer.

It will sometimes be convenient to denote the i-th bit of A as A_i, so if $A = \sum_{n=0}^{N-1} a_n 2^n$ with $a_n \in \{0,1\}$ then $A_i = a_{i-1}$. If a variable name already contains a subscript a comma will be used to separate the subscripts, $A_{i,j}$ is the j-th bit of A_i. Note that we start counting at the 1st bit, and that if one writes the binary representation of A in the usual way, the $i+1$-th bit is to the left of the i-th bit.
When considering integers as strings of bits it makes sense to define AND, OR and NOT operators

Definition 11. *Let A, B and C be k-bit integers. Then*

- ***Bitwise NOT.*** *$B = !_k A$ if for all $0 \leq i < k$, $B_i \neq A_i$.*
- ***Bitwise OR.*** *$C = A||B$ if for all $0 \leq i < k$, $C_i = 1$ if and only if $A_i = 1$ or $B_i = 1$.*
- ***Bitwise AND.*** *$C = A\&\&B$ if for all $0 \leq i < k$, $C_i = 1$ if and only if $A_i = 1$ and $B_i = 1$.*

Note that unlike the bitwise AND and OR, the bitwise NOT depends on the amount of bits that are considered, necessitating the subscript.

These binary operators will be an important part in the binary description of a Turing machine algorithm. So since the goal in this section is to provide an exponential diophantine description of a Turing machine algorithm, we should find an exponential diophantine description of the bitwise AND, OR and NOT. The usual method to do this uses the so-called masking operator $\preceq$, see for example Matijasevic [5] or chapter II of Smorynski [9]. I will do the same here.

Definition 12. *Let $A = \sum_{n=0}^{N} a_n 2^n, B = \sum_{n=0}^{N} b_n 2^n$ with $a_n, b_n \in \{0,1\}$. Then $A \preceq B$ if for all $n \leq N$, $a_n \leq b_n$.*

In order to use the masking operator we need to link it to something we can easily find a diophantine description for. We will use binomial coefficients for this. In order to do this we first need two theorems. They both hold more generally, but this is the form in which we will use them. The first theorem is due to Legendre.

Theorem 5. *Let a be a positive integer, $a = \sum_{n=0}^{N} a_n 2^n$ with $a_n \in \{0,1\}$. Furthermore, let $s = \sum_{n=0}^{N} a_n$ and let μ be the highest power of 2 that divides $a!$, so $2^\mu | a!$ and $2^{\mu+1} \not| a!$. Then*

$$\mu = a - s.$$

Proof: Take $b \leq a$, and let ν be the highest power of 2 dividing b. Then the factor b in $a!$ provides ν powers of 2. So if we take $d_i =$"the amount of integers $b \leq a$ with 2^i the highest power of 2 dividing b", then $\mu = \sum_{i=1}^{\infty} i d_i = \sum_{i=1}^{N} i d_i$. If we use $[x]$ to denote the largest integer smaller or equal to x, then $[\frac{a}{2^i}]$ is the amount of integers $b \leq a$ such that $2^i | b$, so $d_i = [\frac{a}{2^i}] - [\frac{a}{2^{i+1}}]$. Therefore,

$$\mu = \sum_{i=1}^{N} i(\left[\frac{a}{2^i}\right] - \left[\frac{a}{2^{i+1}}\right]) = \sum_{i=1}^{N} \left[\frac{a}{2^i}\right].$$

Writing a in binary gives

$$\mu = \sum_{i=1}^{N} \left[\frac{a_N 2^N + \cdots + a_0}{2^i}\right] = \sum_{i=1}^{N} a_N 2^{N-i} + \cdots + + a_{i+1} 2 + a_i,$$

so

$$\mu = a_n(2^0 + \cdots + 2^{N-1}) + a_{n-1}(2^0 + \cdots + 2^{N-2}) + \cdots a_2(2^0 + 2^1) + a_1$$
$$= a_N(2^N - 1) + \cdots + a_1(2^1 - 1) = a_N 2^n + \cdots a_1 2^1 + a_0 - a_N - \cdots - a_0 = a - s.$$

□

The next theorem is due to Kummer.

Theorem 6. *Let a, b be positive integers, $a = \sum_{n=0}^{N} a_n 2^n$, $b = \sum_{n=0}^{N} b_n 2^n$ with $a_n, b_n \in \{0,1\}$ for all n and let μ be the highest power of 2 dividing $\binom{a+b}{b}$. Then μ is the amount of carries when adding a to b in base 2.*

Proof: By padding the smaller of a and b with extra zeros, we can write $a = \sum_{n=0}^{N} a_n 2^n$, $b = \sum_{n=0}^{N} b_n 2^n$, $c = a+b = \sum n = 0^{N+1} c_n 2^n$. We have $\binom{c}{b} = \frac{c!}{a!b!}$. If we write $s_a = \sum_{n=0}^{N} a_n, s_b = \sum_{n=0}^{N} b_n, s_c = \sum_{n=0}^{N+1} c_n$, we can use theorem 5. Then $\mu = c - s_c - a + s_a - b + s_b = s_a + s_b - s_c$. That is, μ is the amount of non-zero digits of a plus the amount of non-zero digits of b, minus the amount of non-zero digits of c. Since the amount of non-zero digits of c is exactly the amouint of non-zero digits of a plus the amount of non-zero digits of b minus the amount of carries, this implies that μ is the amount of carries when adding a and b. □

Corollary 6. *Let a, b be positive integers with $a < b$. Then $a \preceq b$ if and only if $\binom{b}{a}$ is odd.*

Proof: $a \preceq b$ if and only if one can subtract a from b without carries, or equivalently, add a to $b - a$ without carries. This in turn is equivalent to the highest power of 2 dividing $\binom{b-a+a}{a}$ being 2^0, so $\binom{b}{a}$ being odd. □

Now we just have to find an exponential diophantine description of $\binom{b}{a}$.

Lemma 23. *Let a, b, y, x be positive integers $x > b!$. Then $y = \binom{b}{a}$ if and only if $\exists c, d \in \mathbb{N}$: $(x+1)^b = c + yx^a + dx^{a+1}$ and $c < x^a$.*

Proof: $(x+1)^b = \sum_{n=0}^{b} \binom{b}{n} x^n$. Also, for all n, $\binom{b}{n} < x$. Therefore, $\binom{b}{a}$ is the $a+1$-th digit in the x-adic expansion of $(x+1)^b$. □

Corollary 7. *The relation $a \preceq b$ is exponential diophantine.*

Proof: Taking $x = b^b + 1$, we can use theorem 4.2 and lemma 23 for the case where $0 < a < b$. We then only have to take care of some special cases. We have $a \preceq b$ if and only if

- $a = 0$
- $a = b$

or

- $0 < a < b$ and $\binom{b}{a}$ is odd.

□

Using the masking relation, we can quite easily find exponential diophantine descriptions of the bitwise operators.

Lemma 24. *Let A be a k-bit integer. Then $B =!_k A$ if and only if $A + B = 2^k - 1$.*

Lemma 25. *Let A and B be up to k-bit integers. Then $A||B = C$ if and only if $A \preceq C$,$B \preceq C$ and $C - A \preceq B$.*

Proof: $A \preceq C$ and $B \preceq C$ hold if and only if $A||B \preceq C$, while $C - A \preceq B$ holds if and only if any "1" in C that does not come from A comes from B, so $C \preceq A||B$. □

Lemma 26. *Let A and B be up to k-bit integers. Then $A\&\&B = C$ if and only if $C \preceq A$, $C \preceq B$ and $(!_k C) - (!_k A) \preceq (!_k B)$.*

Proof: $C \preceq A$ and $C \preceq B$ hold if and only if $C \preceq A\&\&B$, while $(!_k C) - (!_k A) \preceq (!_k B)$ holds if and only if any "0" in C that does not come from A comes from B, so $A\&\&B \preceq C$. □

One more thing that will come in useful is the following.

Lemma 27. *Let $k, m \geq 1, q = 2^k$. Then $A = \sum_{i=0}^{m-1} q^i$ if and only if $q^{m-1} \preceq A$ and $A = 1 + qA - q^m$.*

The condition $q^{m-1} \preceq A$ is not actually necessary, as it is already implied by $A = 1 + qA - q^m$, but it simplifies the proof.

Proof: Suppose $A = \sum_{i=0}^{m-1} q^i$. Then

$$A = \underbrace{\underbrace{0\cdots0}_{k-1}1\underbrace{0\cdots0}_{k-1}\cdots1\underbrace{0\cdots0}_{k-1}1}_{m \text{ copies}},$$

so obviously $q^{m-1} \preceq A$. Also, multiplying by q is the same as moving k spots to the left, so

$$qA = \underbrace{\underbrace{0\cdots0}_{k-1}1\underbrace{0\cdots0}_{k-1}\cdots1\underbrace{0\cdots0}_{k-1}}_{m \text{ copies}}1\underbrace{0\cdots0}_{k},$$

so $A = 1 + qA - q^m$.
Now suppose $q^{m-1} \preceq A$ and $A = 1 + qA - q^m$. The fact that $q^{m-1} \preceq A$ guarantees that we can subtract q^m from qA by changing only one bit. Therefore, the first k bits of $qA - q^m$ are all zero. Therefore, $1 + qA - q^m$ ends in $\underbrace{0\cdots0}_{k-1}1$, so A also end in $\underbrace{0\cdots0}_{k-1}1$. However, if $m \geq 2$ this implies that $1 + qA - q^m$ ends in $\underbrace{0\cdots0}_{k-1}1\underbrace{0\cdots0}_{k-1}1$, so A also ends in $\underbrace{0\cdots0}_{k-1}1\underbrace{0\cdots0}_{k-1}1$. This continues until q^m is reached, so

$$A = \underbrace{\underbrace{0\cdots0}_{k-1}1\underbrace{0\cdots0}_{k-1}\cdots1\underbrace{0\cdots0}_{k-1}1}_{m \text{ copies}},$$

so $A = \sum_{i=0}^{m-1} q^i$. □

Having $A = \sum_{i=0}^{m-1} q^i$ will allow us to make m copies of an up to k-bit integer, since for any N, $q^i N$ is N, shifted ik bits to the left. If, for example, $k = 5$ and $N = 10011$, then $AN = \underbrace{10011\ 10011\ \cdots\ 10011}_{m \text{ copies}}$

4.3. **Well-formed and consistent.** Suppose $S \subset \mathbb{N}^n$ is an r.e. set. Then there is an algorithm such that for any x, one has $x \in S$ if and only if the algorithm halts in a finite amount of steps when starting with the input corresponding to x. We want to simulate such a halting run of the algorithm.
Let s be the amount of different symbols.
Let k be the amount of time-steps the algorithm needs.
Let h be the amount of tape locations used. The algorithm uses only a finite amount of time-steps, so it cannot use an infinite amount of tape locations. As such the symbol β is not very relevant. We consider only a finite part of the tape, all other tape locations can be assumed to contain β.
It is convenient for the description of the algorithm to consider the rightmost position used to be position 1. We will have to account for this when considering the input of the algorithm.
Let T_i, for $1 \leq i \leq s$ describe the tape, in the following way: for any $1 \leq j \leq k$, $T_{i,pk+j} = 1$ if and only if the tape contains symbol i at position p at time j.
Let P describe the position of the read/write head, such that for any $1 \leq i \leq k, P_{pk+i} = 1$ if and only if the head is at position p at time i.
Let $Q_1, \cdots Q_m$ be k-bit integers that describe the non-accepting states, $Q_{p,i} = 1$ if and only if the system is in state p at time i.
Let $R_1, \cdots R_l$ describe the accepting states in the same way.
Let U_i, for $1 \leq i \leq s$ be k-bit integers that describe the content of the tape in another way, such that $U_{i,j} = 1$ if and only if the tape contains symbol i at the position the head is currently at, at time j.

It will be convenient to use some more variables.
Let $q = 2^k$.
Let $G_{i,j}$, for $1 \leq i \leq s, 1 \leq j \leq m$ be the sum of all states where the next state is j if the tape contains symbol i, $G_{i,j} = \sum_{e \in E} Q_e$ where $E = \{e | \exists x, y : G(q_e, \alpha_i) = (q_j, x, y)\}$.
Let $F_{i,j}$, for $1 \leq i \leq s, 1 \leq j \leq l$ be the sum of all states where the next state is accepting state j if the tape contains symbol i,$F_{i,j} = \sum_{e \in E} Q_e$ where $E = \{e | \exists x, y : G(q_e, \alpha_i) = (r_j, x, y)\}$.
Let LS_i, for $1 \leq i \leq s$ be the sum of all states where the tape head is shifted to the left if the tape contains symbol i at the current position,$LS_i = \sum_{e \in E} Q_e$ where $E = \{e | \exists x, y : G(q_e, \alpha_i) = (x, y, L)\}$.

Let RS_i, for $1 \leq i \leq s$ be the sum of all states where the tape head is shifted to the right if the tape contains symbol i at the current position, $RS_i = \sum_{e \in E} Q_e$ where $E = \{e | \exists x, y : G(q_e, \alpha_i) = (x, y, R)\}$.
Let $W_{i,j}$, for $1 \leq i, j \leq s$ be the sum of all the states where symbol j is written if the tape contains symbol i, $W_{i,j} = \sum_{e \in E} Q_e$ where $E = \{e | \exists x, y : G(q_e, \alpha_i) = (x, \alpha_j, y\}$.
Let $M = \sum_{i=0}^{h-1} q^i$.

In order to show that these variables code the run of the algorithm, the following things need to be shown.

- The variables are well-formed.
  In order for an integer to fill a role, say the position of the read/write head, it will have to satisfy certain basic properties. The read/write head for example, can be in only one position at a time, and the integer will have to reflect this. If all variables are well-formed, it should be possible to interpret them as a machine with a tape, a read/write head and a number of states the machine can be in. In terms of the formal description of a Turing machine given before, the variables should describe functions T, P and S, but they need not satisfy the required properties for those functions.
  Of course an integer that is well-formed to code one role will generally not be well-formed for other roles. It will be clear from the context for which role a variable is well-formed though, so for ease of notation I will not explicitly mention the role for the variables.
- The variables are consistent with each other.
  Obviously, the variables do not only need to be well-formed, they also need to fit together in some way. Like with well-formedness, consistency will depend on the role of a variable. It will be clear from context for which role a variable should be consistent.
  If the variables are well-formed and consistent, they should describe functions T, P and S that satisfy the required properties.

Definition 13. *k-bit integers $Q_1, \cdots Q_m, R_1, \cdots R_l$ are well-formed if they code for the system being in exactly one state at each time step, and the system is in an accepting state in the last and only the last time step.*

For example, in a system using three normal states, two accepting states, three symbols, 5 time steps and 4 locations, one could have

	Well-formed	Ill-formed
Q_1	00001	10000
Q_2	00110	00110
Q_3	01000	01010
R_1	00000	01000
R_2	10000	00000

The rightmost values are ill-formed because they have a normal state at the last time step, they have an accepting state at the 4th time step which isn't the last one, they have two states at the second and fourth time steps, and no states at all at the first time step.

Definition 14. *k-bit integers $U_1, \cdots U_s$ are well-formed if they code for there always being exactly one symbol under the read/write head.*

Once again using an example with three normal states, two accepting states, three symbols, 5 time steps and 4 locations, a well-formed U_1, U_2, U_3 could be

U_1	00001
U_2	11100
U_3	00010

Definition 15. *kh-bit integers $T_1, \cdots T_s$ are well-formed if they code for exactly one symbol being at every position at every time step.*

An example of well-formed T_1, T_2, T_3 could be

T_1	00000	00000	00000	00001
T_2	00000	00111	11100	11110
T_3	11111	11000	00011	00000

This example could be consistent with some other well-formed variables. It is however possible to have well-formed T_1, T_2, T_3 that cannot be consistent with well-formed other variables. For example,

T_1	10001	10001	10001	10001
T_2	01010	01010	01010	01010
T_3	00100	00100	00100	00100

is well-formed, but since it changes value at multiple positions in a single time step it cannot be consistent with other well-formed variables.

Definition 16. *kh-bit integer P is well-formed if it codes for the read/write head being in exactly one position at each time step.*

An example of well-formed P could be

P	00000	00100	01010	10001

Definition 17. *Interpret $T_1, \cdots, T_s, P, Q_1, \cdots Q_m, R_1, \cdots, R_l, U_1, \cdots U_s$ as a run of a given algorithm. Then*

(1) *$T_1, \cdots, T_s$ are consistent with the other variables if:*
- *If the tape contains symbol i at time j at position p and the head is not at position p, then the tape contains symbol i at time $j+1$ at position p.*
- *If the head is at position p at time j and the head writes symbol i, then the tape contains symbol i at time $j+1$ at position p.*

(2) *P is consistent with the other variables if:*
- *If the head is in position p at time j and the head should move to the left, then the head is in position $p+1$ at time $j+1$.*
- *If the head is in position p at time j and the head should move to the right, then the head is in position $p-1$ at time $j+1$.*

(3) *$Q_1, \cdots Q_m, R_1, \cdots R_l$ are consistent with the other variables if:*
- *The system is initially in state q_1.*
- *If the system should go to state r at time j, then the system is in state r at time $j+1$.*

(4) *$U_1, \cdots U_s$ are consistent with the other variables if:*
- *If the head is in position p at time j and the tape contains symbol i at position p at time j, then the symbol under the head at time j is i according to $U_1, \cdots U_s$.*

Consistency doesn't require the variables to be well-formed. If according to P the head is in two places at time j, then it can still be consistent with he other variables as long as the head moves in the right direction from both positions. However, it does not seem to be very useful to consider consistent but ill-formed variables.

Lemma 28. *The variables describe a halting run of the Turing machine algorithm if and only if they are well-formed and consistent.*

Giving an actual proof of this lemma would be more trouble than it is worth, since it would require a more formal approach to well-formedness and consistency, which would in the end only amount to checking a lot of statements about indices. The lemma should be pretty obvious though, except for the consistency of $U_1, \cdots, U_s$ the consistency requirements clearly correspond to the requirements on the functions T, P and S from the definition.

4.4. **Conditions for well-formedness and consistency.** The following lemmas give conditions under which variables code for a well-formed and consistent run of an algorithm and are unfortunately rather technical. In order to give a better idea of why the lemmas hold I will apply the lemmas to an example before proving them. I will use the following example: Consider a Turing machine algorithm with three normal states, two accepting states, three symbols and the following instructions.

	Symbol		
	1	2	3
q_1	$(2, q_2, L)$	$(3, q_2, L)$	$(3, r_1, L)$
q_2	$(2, q_3, R)$	$(3, q_3, R)$	$(2, q_2, L)$
q_3	$(1, q_1, L)$	$(2, r_2, R)$	$(3, q_1, R)$

I will use the following run of this algorithm.

Time	Tape	State	Position
1	323**1**	q_1	1
2	32**3**2	q_2	2
3	3**2**22	q_2	3
4	33**2**2	q_3	2
5	332**2**	r_2	1

So $s = 3, k = 5, h = 4$.
The binary encoding of this algorithm run is

$Q_1 =$	00001			
$Q_2 =$	00110			
$Q_3 =$	01000			
$R_1 =$	00000			
$R_2 =$	10000			
$U_1 =$	00001			
$U_2 =$	11100			
$U_3 =$	00010			
$T_1 =$	00000	00000	00000	00001
$T_2 =$	00000	00111	11100	11110
$T_3 =$	11111	11000	00011	00000
$P =$	00000	00100	01010	10001

Note that these are exactly the examples of well-formed variables given before. Also, we have

$$q = 2^k = 1\ 00000$$

and

$$M = \sum_{i=0}^{h-1} q^i = 00001\ 00001\ 00001\ 00001.$$

Lemma 29. *The Q_i are well-formed if and only if*

1. $\forall 1 \leq i < j \leq m : Q_i \&\& Q_j = 0$
2. $\forall 1 \leq i \leq m, 1 \leq j \leq l : Q_i \&\& R_j = 0$
3. $\forall 1 \leq i < j \leq l : R_i \&\& R_j = 0$
4. $Q_1 + \cdots + Q_m = 2^{k-1} - 1$
5. $R_1 + \cdots + R_l = 2^{k-1}$

Proof: 1, 2 and 3 hold if and only if there is no time at which the system would be in multiple states. Given 1, 2 and 3, equation 4 holds if and only if for every time step except the final one, the system is in one of the states $q_1, \cdots q_m$. Given 1,2 and 3, equation 5 holds if and only if the system is in one of the states $r_1, \cdots r_l$ at the final time step. Taken together, 1 through 5 hold if and only if the system is in exactly one state at each time step, and in an accepting state at the final time step and only at the final time step. □

The next two lemma's hold for the same reason, I will give them here without proof.

Lemma 30. *The U_i are well-formed if and only if*

$$\forall 1 \leq i < j \leq s : U_i \&\& U_j = 0$$

and

$$U_1 + \cdots U_s = 2^k - 1.$$

Lemma 31. *The T_i are well-formed if and only if*

$$\forall 1 \leq i < j \leq s : T_i \&\& T_j = 0$$

and

$$T_1 + \cdots T_s = 2^{hk} - 1.$$

A diophantine description of P being well-formed is harder to formulate. However, it can easily be added to the diophantine description of P being consistent with the other variables.

Lemma 32. *If the other variables are well-formed, then P is well-formed and consistent with the other variables if and only if $\exists d \in \mathbb{N}$ such that*

$$q^d + 2q^2(M(\sum_{i=1}^{s} LS_i \&\& U_i) \&\& P) || 2(M(\sum_{i=1}^{s} RS_i \&\& U_i) \&\& P) = qP$$

Example: The states that shift the head to the left if the tape contains symbol 1 are states 1 and 3, so $LS_1 = Q_1 + Q_3 = 01001$. The only state that shifts the head to the left if the tape contains symbol 2 is state 1, so $LS_2 = Q_1 = 00001$. Also, $LS_3 = Q_1 + Q_2 = 00111$, $RS_1 = 2^4 - 1 - LS_1 = 00110$, $RS_2 = 2^4 - 1 - LS_2 = 01110$, $RS_3 = 2^4 - 1 - LS_3 = 01000$.

According to the lemma there should be a d such that

$$q^d + 2q^2(M(\sum_{i=1}^{s} LS_i \&\& U_i) \&\& P) || 2(M(\sum_{i=1}^{s} RS_i \&\& U_i) \&\& P) = qP.$$

First, let's calculate most of the left-hand side.

$$\sum_{i=1}^{s} LS_i \&\& U_i = 01001 \&\& 00001 + 00001 \&\& 11100 + 00111 \&\& 00010 = 00011.$$

This gives exactly the time steps where the head moves to the left. Likewise,

$$\sum_{i=1}^{s} RS_i \&\& U_i = 00110 \&\& 00001 + 01110 \&\& 11100 + 01000 \&\& 00010 = 01100,$$

exactly the time steps where the head moves to the right. Note that $00011 + 01100 = 01111$, the head always moves to either the right or the left except in the last time step.
Multiplying by M creates a copy for every tape location.

$$M(\sum_{i=1}^{s} LS_i \&\& U_i) = 00011\ 00011\ 00011\ 00011$$

$$M(\sum_{i=1}^{s} RS_i \&\& U_i) = 01100\ 01100\ 01100\ 01100$$

We then take a bitwise AND with P.

$$M(\sum_{i=1}^{s} LS_i \&\& U_i) \&\& P$$

$$\begin{array}{ll} = & 00011\ 00011\ 00011\ 00011 \\ \&\& & 00000\ 00100\ 01010\ 10001 \\ = & 00000\ 00000\ 00010\ 00001, \end{array}$$

exactly the positions and times the head moves to the left.

$$M(\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P = 00000\ 00100\ 01000\ 00000,$$

exactly the positions and times the head moves to the right.
Moving one step forward in time corresponds to multiplying by 2. Moving one posision to the left corresponds to multiplying by q. Moving one position to the right would correspond to dividing by q. However, to ensure we only need integers, we will instead multiply everything else by q.

$$2q^2M(\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P) = 00000\ 00000\ 00100\ 00010\ 00000\ 00000.$$

$$2(M(\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P) = 00000\ 01000\ 10000\ 00000.$$

So

$$\begin{aligned} qP - 2q^2M(\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)||2(M(\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P) \\ = \quad & 00000\ 00100\ 01010\ 10001\ 00000 \\ - \quad & 00000\ 00100\ 01010\ 10000\ 00000 \\ = \quad & 00000\ 00000\ 00000\ 00001\ 00000 \quad = q^1. \end{aligned}$$

Taking $d = 1$ will therefore work. In fact, knowing that the system started in position 1, we could immediately have taken $d = 1$, since q^d is the only 1 at the first bit of a block.
End of example.

Proof of lemma 32: Let's first analyze the left-hand side of the equation.
For any i, $LS_{i,j} = 1$ if and only if the system is in a state where the head should move to the left if the symbol under the head is a i at time j. Also, $U_{i,j} = 1$ if and only if the symbol under the head is a i at time j, so $(LS_i\&\&U_i)_j = 1$ if and only if the head should move to the left and the symbol under the head is an i at time j. Since $U_1, \cdots, U_s$ are well-formed, there is one symbol under the head at each time step, so $(\sum_{i=1}^{s} LS_i\&\&U_i)_j = 1$ if and only if the head should move to the left at time j. Multiplying with M creates one copy for each possible location of the head, so $(M\sum_{i=1}^{s} LS_i\&\&U_i)_{pk+j} = 1$ with $j < k$ if and only if the head should move to the left at time j. $P_{pk+j} = 1$ if and only if the head is in position p at time j. Then $((M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)_{pk+j} = 1$ if and only if the head should move to the left from position p at time j. Since moving one position to the left corresponds to multiplying by q and moving one time step forward corresponds to multiplying by 2, the condition

$$2q((M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P) \preceq P$$

holds if and only if the head is in position $p + 1$ at time $j + 1$ whenever the head should move to the left from position p at time j.
Likewise, $((M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P)_{pk+j} = 1$ with $j < k$ if and only if the head should move to the right from position p at time j, so the condition

$$2((M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P) \preceq qP$$

holds if and only if the head is in position $p - 1$ at time $j + 1$ whenever the head should move to the right from position p at time j.
P is therefore consistent with the other variables if and only if

$$2q^2((M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)||2((M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P) \preceq qP.$$

In order to guarantee that P is also well-formed, slightly more is needed. There are only three ways in which the head can get to position p at time j. The head was in position $p-1$ at time $j-1$ and the head moved to the left, the head was in position $p+1$ at time $j-1$ and the head moved to the right, or $j=0$ and p is the initial position of the head. The masking condition

$$qP \preceq q^d + 2q^2((M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)||2((M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P) \preceq qP$$

therefore holds for some $d \in \mathbb{N}$ if and only if every position of the head was reached in a correct way.
(Note that both $M\sum_{i=1}^{s} LS_i\&\&U_i$ and $M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P$ cannot have a 1 in the final position, since the Q_i are well-formed, so the system is in none of the states that require the head to move in the last time step. $(M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P$ and $(M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P$ therefore have no 1 at the end of a block of k bits, so $2q^2((M\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)||2((M\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P)$ has no 1 at the start of a block. We can therefore add q^d, which is a 1 at the start of a block without worrying about carries.)
The other variables are well-formed, so in each time step the head moves either to the left or to the right, but never both. Together with the above masking condition, this implies that P is well-formed. Therefore (and since $q^d \preceq qP$ if and only if the head is in a position at the first time step),

$$\exists d \in \mathbb{N} : q^d + 2q^2(M(\sum_{i=1}^{s} LS_i\&\&U_i)\&\&P)||2(M(\sum_{i=1}^{s} RS_i\&\&U_i)\&\&P) = qP$$

if and only if P is well-formed and consistent with the other variables. □

Lemma 33. *Suppose all variables are well-formed. Then: Q_j is consistent with the other variables if and only if*

$$1 \preceq Q_1$$

and

$$2(\sum_{i=1}^{s} G_{i,j}\&\&U_i) \preceq Q_j.$$

R_j is consistent with $Q_1, \cdots Q_m$ and $R_1, \cdots R_l, U_1, \cdots U_s, T_1, \cdots T_S$ and P if and only if

$$2(\sum_{i=1}^{s} F_{i,j}\&\&U_i) \preceq R_j.$$

Example: I will give the example for Q_2. The only state that goes to state 2 if the tape contains symbol 1 is state 1, so $G_{1,2} = Q_1 = 00001$. The only state that goes to state 2 if the tape contains symbol 2 is again state 1, so $G_{2,2} = Q_1 = 00001$. The only state that goes to state 2 if the tape contains symbol 3 is state 2, so $G_{3,2} = Q_2 = 00110$.
$G_{1,2}\&\&U_1 = 00001\&\&00001 = 00001$, every time step where the state should go to 2 because of a 1 under the head.
$G_{2,2}\&\&U_1 = 00001\&\&11100 = 00000$, every time step where the state should go to 2 because of a 2 under the head.
$G_{3,2}\&\&U_1 = 00110\&\&00010 = 00010$, every time step where the state should go to 2 because of a 3 under the head.
Therefore, $\sum_{i=1}^{3} G_{i,2}\&\&U_i = 00011$, every time step where the state should go to 2. Going one time step forward corresponds to multiplying by 2, so the system goes to state 2 every time it should if and only if $2\sum_{i=1}^{3} G_{i,2}\&\&U_i = 00110 \preceq Q_j = 00110$.
In our example Q_2 is therefore consistent with the other variables.
End of example.

Proof of lemma 33: $1 \preceq Q_1$ holds if and only if the system is initially in state q_1.
$G_{i,j,c} = 1$ if and only if the system is in a state where it should go to state j if the symbol under the head is a i at time c. $U_{i,c} = 1$ if and only if the symbol under the head at time c is a i, so $(G_{i,j}\&\&U_i)_c = 1$ if and only if the system should go to state j and the symbol under the head is

a i at time c. There can be only one symbol under the head at any one time due to the variables being well-formed, so $(\sum_{i=1}^{s} G_{i,j}\&\&U_i)_c = 1$ if and only if the system should go to state j at time c. Going one time step forward corresponds to multiplying by 2, so Q_j is consistent with the other variables if and only if $2(\sum_{i=1}^{s} G_{i,j}\&\&U_i) \preceq Q_j$.
Likewise, $(\sum_{i=1}^{s} F_{i,j}\&\&U_i)_c = 1$ if and only if the system should go to accepting state j at time c, so R_j is consistent with the other variables if and only if $2(\sum_{i=1}^{s} F_{i,j}\&\&U_i) \preceq R_j$. □

Lemma 34. *Suppose all variables are well-formed. Then U_i is consistent with the other variables if and only if*

$$P\&\&T_i \preceq M \cdot U_i$$

and

$$P\&\&!_{hk}T_i \preceq M\cdot!_k U_i$$

Example: Let's look at U_2.
$P\&\&T_2 = 00000\ 00100\ 01000\ 10000$, the times and positions where the symbol under the head is symbol 2. If there is a 1 on $P\&\&T_2$ at the j-th bit of a block, then the symbol under the head is a 2 at time j. On the other hand, $M \cdot U_2 = 11100\ 11100\ 11100\ 11100$ contains a 1 at every j-th bit of a block if the symbol under the head is a 2 at time j. Therefore, we should have

$$\begin{aligned} P\&\&T_2 = &\quad 00000\ 00100\ 01000\ 10000 \\ \preceq M \cdot U_i = &\quad 11100\ 11100\ 11100\ 11100 \end{aligned}$$

On the other hand, $P\&\&!_{hk}T_2 = 00000\ 00000\ 00010\ 00001$ corresponds to the times and positions where the symbol under the head is not symbol 2. If there is a 1 on $P\&\&T_2$ at the j-th bit of a block, then the symbol under the head is not a 2 at time j. $M\cdot!_k U_2 = 00011\ 00011\ 00011\ 00011$ contains a 1 at every j-th bit of a block if the symbol under the head is not a 2 at time j. Therefore, we should also have

$$\begin{aligned} P\&\&!_{hk}T_2 = &\quad 00000\ 00000\ 00010\ 10001 \\ \preceq M\cdot!_k U_2 = &\quad 00011\ 00011\ 00011\ 00011 \end{aligned}$$

There is only one 5-bit integer U_2 that satisfies both these properties, so U_2 is consistent with the other variables if and only if the two conditions hold.
End of example.

Proof of lemma 34: $P_{pk+j} = 1$ with $j < k$ if and only if the head is in position p at time j and $T_{i,pk+j} = 1$ if and only if the tape contains a i at position p at time j. Then $(P\&\&T_i)_{pk+j} = 1$ if and only if the tape contains a i at position p and the head is at position p at time j. $U_{i,j}$ should be 1 if and only if the symbol under the head is a i at time j, so $(M \cdot U_i)_{pk+j}$ should be 1 if and only if the symbol under the head is a i at time j. Therefore, U_i contains a 1 at every position it should contain a 1 if and only if

$$P\&\&T_i \preceq M \cdot U_i.$$

On the other hand, $(P\&\&!_{hk}T_i)_{pk+j} = 1$ with $j < k$ if and only if the tape does not contain symbol i at position p and the head is at position p at time j. $(!_k U_i)_j$ should be 1 if and only if the symbol under the head is not symbol i at time j, so $(M\cdot!_k U_i)_{pk+j}$ should be 1 if and only if the symbol under the head is not symbol i at time j. Therefore, $!_k U_i$ contains a 1 at every positions it should contain a 1 if and only if

$$P\&\&!_{hk}T_i \preceq M\cdot!_k U_i.$$

Together these two conditions hold if and only if U_i contains a 1 everywhere it should contain a 1 and a 0 everywhere it should contain a 0, so if and only if U_i is consistent with the other variables. □

Lemma 35. *Suppose all variables are well-formed. Then T_j is consistent with the other variables if and only if*

$$2(P\&\&M(\sum_{i=1}^{s} W_{i,j}\&\&U_i)) \preceq T_j,$$
$$2(!_{hk}P\&\&T_j\&\&M(2^{k-1}-1)) \preceq T_j,$$

$$2(!_{hk}P\&\&!_{hk}T_j\&\&M(2^{k-1}-1)) \preceq !_{hk}T_j,$$

and

$$2(P\&\&M((\sum_{c=1}^{s}\sum_{i=1}^{s}W_{i,c}\&\&U_i)-(\sum_{i=1}^{s}W_{i,j}\&\&U_i))) \preceq !_{hk}T_j.$$

Example: There are four masking conditions in the lemma, and they each guarantee a different part of the consistency.

- If the head is at position p at time j and the system writes symbol i, then the tape should contain symbol i at position p at time $j+1$.
- If the head is not at position p at time j and the tape contains symbol i at position p at time j, then the tape should still contain symbol i at position p at time $j+1$.
- If the head is not at position p at time j and the tape does not contain symbol i at position p at time j, then the tape should still not contain symbol i at position p at time $j+1$.
- If the head is at position p at time j and the system writes a symbol other than i, then the tape should not contain symbol i at position p at time $j+1$.

Looking at T_2, we first need $W_{i,c}$ for all i and c.

$W_{i,c}$	$i=1$	$i=2$	$i=3$
$c=1$	01000	00000	00000
$c=2$	00111	01000	00110
$c=3$	00000	00111	01001

Calculating the first masking condition then gives

$$\sum_{i=1}^{s}W_{i,2}\&\&U_i = 00111\&\&00001 + 01000\&\&11100 + 00110\&\&00010 = 01011,$$

the times when symbol 2 is written. Multiplying by M and taking the bitwise AND with P gives

$$\begin{aligned}P\&\&M(\sum_{i=1}^{s}W_{i,2}\&\&U_i) &= 00000\ 00100\ 01010\ 10001\&\&01011\ 01011\ 01011\ 01011\\ &= 00000\ 00000\ 01010\ 00001,\end{aligned}$$

the times and places where symbol 2 is written. In the time steps following this the tape should therefore contain symbol 2 at those places,

$$\begin{aligned}2(P\&\&M(\sum_{i=1}^{s}W_{i,2}\&\&U_i)) &= 00000\ 00000\ 10100\ 00010\\ &\preceq 00000\ 00111\ 11100\ 11110 = T_2\end{aligned}$$

For the second masking condition,

$$\begin{aligned}!_{hk}P\&\&T_2 &= 11111\ 11011\ 10101\ 01110\&\&00000\ 00111\ 11100\ 11110\\ &= 00000\ 00011\ 10100\ 01110,\end{aligned}$$

all places and times the tape contains a 2 and the head is in another position, so the 2 should stay there. However, this also includes the last time step. Since we do not code for the time step after the last one, we remove the bits corresponding to the last time step.

$$\begin{aligned}!_{hk}P\&\&T_2\&\&M(2^4-1) &= 00000\ 00011\ 10100\ 01110\&\&01111\ 01111\ 01111\ 01111\\ &= 00000\ 00011\ 00100\ 01110.\end{aligned}$$

For each of these places and times where the tape contains a 2, it should still contain a 2 in the next time step, so

$$\begin{aligned}2(!_{hk}P\&\&T_j\&\&M(2^{k-1}-1)) &= 00000\ 00110\ 01000\ 11100\\ &\preceq 00000\ 00111\ 11100\ 11110 = T_j.\end{aligned}$$

Likewise, for the third masking condition,

$$!_{hk}P\&\&!_{hk}T_2\&\&M(2^{k-1}-1) = 01111\ 01000\ 00001\ 00000,$$

all the places and times the symbol on the tape should stay at a symbol other than 2, so

$$2(!_{hk}P\&\&T_2\&\&M(2^4-1)) = 11110\ 10000\ 00010\ 00000$$
$$\preceq 11111\ 11000\ 00011\ 00001 = !_{hk}T_2.$$

The last masking condition should guarantee that if a symbol other than 2 is written, then the tape will contain a symbol other than 2 in the next time step.

$$P\&\&M((\sum_{c=1}^{3}\sum_{i=1}^{3} W_{i,c}\&\&U_i) - (\sum_{i=1}^{3} W_{i,2}\&\&U_i)) = 00000\ 00100\ 00000\ 00000,$$

every time and place the head writes something other than a 2. We should therefore have

$$2(P\&\&M((\sum_{c=1}^{3}\sum_{i=1}^{3} W_{i,c}\&\&U_i) - (\sum_{i=1}^{3} W_{i,2}\&\&U_i))) = 00000\ 01000\ 00000\ 00000$$
$$\preceq 11111\ 11000\ 00011\ 00001 = !_{hk}T_2$$

End of example.

Proof of lemma 35: $(\sum_{i=1}^{s} W_{i,j}\&\&U_i)_d = 1$ if and only if the system writes symbol j at time d, so $(P\&\&M(\sum_{i=1}^{s} W_{i,j}\&\&U_i))_{pk+d} = 1$ if and only if the system writes symbol j at time d and position p. Therefore, T_j is consistent with symbol j being written if and only if

$$2(P\&\&M(\sum_{i=1}^{s} W_{i,j}\&\&U_i)) \preceq T_j.$$

$(!_{hk}P\&\&T_j\&\&M(2^{k-1}-1))_{pk+d} = 1$ if and only if the tape contains symbol j at position p at time d, the head is not in position p at time d and $d \neq k$. If the head is not in position p at time $d \neq k$, then the symbol at position p should be the same at time $d+1$. Therefore, T_j is consistent with symbol j not changing to another symbol if the head is in another position if and only if

$$2(!_{hk}P\&\&T_j\&\&M(2^{k-1}-1)) \preceq T_j.$$

Likewise, $(!_{hk}P\&\&!_{hk}T_j\&\&M(2^{k-1}-1))_{pk+d} = 1$ if and only if the tape does not contain symbol j at position p at time d, the head is not in position p at time d and $d \neq k$. Therefore, T_j is consistent with the symbol not changing to symbol j if the head is in another position if and only if

$$2(!_{hk}P\&\&!_{hk}T_j\&\&M(2^{k-1}-1)) \preceq !_{hk}T_j.$$

Finally, $(P\&\&M((\sum_{c=1}^{s}\sum_{i=1}^{s} W_{i,c}\&\&U_i)_{pk+d} = 1$ if and only if the system writes a symbol other than symbol j at time d and position p. Therefore, T_j is consistent with a symbol other than j being written if and only if

$$2(P\&\&M((\sum_{c=1}^{s}\sum_{i=1}^{s} W_{i,c}\&\&U_i) - (\sum_{i=1}^{s} W_{i,j}\&\&U_i))) \preceq !_{hk}T_j.$$

These four parts of consistency together are equivalent with consistency. □

4.5. **Exponential diophantine description of input.** This gives an exponential diophantine description of well-formed and consistent. We still need a way to get a grip on the input for the algorithm though. A problem here is that while we would like to take $x \in \mathbb{N}^n$ as input, a Turing machine can only take a string of characters from it's alphabet as input. We will therefore need to transform x into some string $\tilde{x}$ which is then used as input. Obiously there need to be some restrictions on what transformations are considered acceptable. The transformation "$\tilde{x} = 0$ if $x \in S$ and $\tilde{x} = 1$ if $x \notin S$" for example is not acceptable since it would make all sets trivially recursive.

We could restrict ourselves to a certain type of reasonable input. A good example would be unary input, since it works for any alphabet with at least two symbols. If $x = (x_1, \cdots, x_n)$, then

$$\cdots \alpha_2\alpha_2 \underbrace{\alpha_1 \cdots \alpha_1}_{x_n} \alpha_2 \underbrace{\alpha_1 \cdots \alpha_1}_{x_{n-1}} \alpha_2 \cdots \alpha_2 \underbrace{\alpha_1 \cdots \alpha_1}_{x_1} \alpha_2\alpha_2 \cdots = \tilde{x}.$$

This is more restrictive than we would like, but it does not matter. While there can certainly be acceptable inputs other than unary, it would be reasonable to define acceptable in such a way that an algorithm - so a Turing machine - can reach it from the unary input. That way, any algorithm that takes an acceptable input can be modified to take unary input.
We also need to specify the starting position of the head, which can for example be taken the α_2 immediately to the right of the string of α_1 representing x_1.
For well-formed and consistent variables, we can then guarantee that the input was $\tilde{x}$ in the following way:

Lemma 36. *Suppose $T_1, \cdots T_s, P, Q_1, \cdots, Q_m, R_1, \cdots R_l, U_1, \cdots U_s$ are well-formed and consistent. Then the input of the algorithm is the unary representation of $x = (x_1, \cdots, x_n)$ if and only if there exists $d \in \mathbb{N}$ such that:*

$$q^d \preceq P,$$

$$(\sum_{i=0}^{h-1} (\tilde{x}(2^{k-1})^i \&\& q^i(q-1))) \&\& M \preceq T_1$$

and

$$(!_{hk}(\sum_{i=0}^{h-1} (\tilde{x}(2^{k-1})^i \&\& q^i(q-1)))) \&\& M \preceq T_2,$$

where $\tilde{x}$ is the h-bit integer given by

$$\tilde{x} = 2^{d+1}(2^{x_1} - 1 + 2^{x_1+1}(2^{x_2} - 1) + \cdots + 2^{\sum_{j=1}^{i-1} x_j + 1}(2^{x_1} - 1) + \cdots + 2^{\sum_{j=1}^{n-1} x_j + 1}(2^{x_n} - 1))$$

Proof: The input specifications do not state at what position the rightmost bit of the input should be located. This makes sense, since the tape extends infinitely in both directions, making locations arbitrary. The factors 2^{d+1} and q^d allow this arbitrary placement of the input on the tape. The condition

$$q^d \preceq P$$

holds if and only if the head is initially in position d.

$$\tilde{x} = 2^{d+1}(2^{x_1} - 1 + 2^{x_1+1}(2^{x_2} - 1) + \cdots + 2^{\sum_{j=1}^{i-1} x_j + 1}(2^{x_1} - 1) + \cdots + 2^{\sum_{j=1}^{n-1} x_j + 1}(2^{x_n} - 1))$$

is merely a diophantine description of $\tilde{x}$, (with 0 as α_2 and 1 as α_1) where the representation of x_1 starts at position $d+1$.
The other two masking conditions are more interesting. We have

$$\begin{aligned}
q^i q - 1 = &\quad 0 \cdots 0 \underbrace{1 \cdots 1}_{k} \underbrace{0 \cdots 0}_{ik} \\
\tilde{x}(2^{k-1})^i = &\quad 0 \cdots 0 \tilde{x}_h \cdots \tilde{x}_1 \underbrace{0 \cdots 0}_{i(k-1)} \\
\tilde{x}(2^{k-1})^i \&\& q^i q - 1 = &\quad 0 \cdots 0 \tilde{x}_{\max(h,(i+k))} \cdots \tilde{x}_{1+i} \underbrace{0 \cdots 0}_{ik}
\end{aligned}$$

so

$$(\sum_{i=0}^{h-1} (\tilde{x}(2^{k-1})^i \&\& q^i(q-1))) \&\& M = 0 \cdots 0 \tilde{x}_h \underbrace{0 \cdots 0}_{k-1} \tilde{x}_{h-1} \cdots \tilde{x}_2 \underbrace{0 \cdots 0}_{k-1} \tilde{x}_1.$$

Therefore, $(\sum_{i=0}^{h-1} (\tilde{x}(2^{k-1})^i \&\& q^i(q-1))) \&\& M \preceq T_1$ if and only if the tape initially contains symbol 1 everywhere it should contain symbol 1. On the other hand,

$$(!_{hk}(\sum_{i=0}^{h-1} (\tilde{x}(2^{k-1})^i \&\& q^i(q-1)))) \&\& M = 0 \cdots 0 !_1 \tilde{x}_h \underbrace{0 \cdots 0}_{k-1} !_1 \tilde{x}_{h-1} \cdots !_1 \tilde{x}_2 \underbrace{0 \cdots 0}_{k-1} !_1 \tilde{x}_1,$$

so the tape initially contains symbol 2 everywhere it should contain symbol 2 if and only if $(!_{hk}(\sum_{i=0}^{h-1}(\tilde{x}(2^{k-1})^i \&\& q^i(q-1)))) \&\& M \preceq T_2$.
Due to well-formedness of $T_1, \cdots T_s$ this is sufficient to show that the tape initially contains the required string.

4.6. **Exponential diophantine description of an algorithm run.**

Lemma 37. *Let $S \subset \mathbb{N}^n$ be an r.e. set. Take a Turing machine algorithm that recognizes members of S, taking input in the unary form described above. Let $q_1, \cdots q_m$ be the states of this Turing machine, $r_1, \cdots r_l$ the accepting states, $\alpha_1, \cdots, \alpha_s$ the alphabet for the Turing machine and G the instructions function. Then for any $x \in \mathbb{N}^n$, one has $x \in S$ if and only if*

$$\exists k, h, T_1, \cdots, T_s, P, Q_1, \cdots, Q_m, R_1, \cdots R_l, U_1, \cdots, U_s, q, LS_1, \cdots, LS_s,$$

$$RS_1, \cdots RS_s, M, d, \tilde{x} \in \mathbb{N},$$

for every $1 \leq i \leq s, 1 \leq j \leq m$

$$\exists G_{i,j} \in \mathbb{N},$$

for every $1 \leq i \leq s, 1 \leq j \leq l$

$$\exists H_{i,j} \in \mathbb{N},$$

and for every $1 \leq i, j \leq s$

$$\exists W_{i,j} \in \mathbb{N}$$

such that:

(1) $T_1, \cdots, T_s, P < 2^{hk}$.
(2) $Q_1, \cdots, Q_m, R_1, \cdots, R_l, U_1, \cdots, U_s < 2^k$.
(3) $\forall 1 \leq i \leq s, 1 \leq j \leq m : G_{i,j} = \sum_{e \in E} Q_e$ *where* $E = \{e | \exists y_1, y_2 : G(q_e, \alpha_i) = (q_j, y_1, y_2)\}$.
(4) $\forall 1 \leq i \leq s, 1 \leq j \leq l : F_{i,j} = \sum_{e \in E} Q_e$ *where* $E = \{e | \exists y_1, y_2 : G(q_e, \alpha_i) = (r_j, y_1, y_2)\}$.
(5) $\forall 1 \leq i \leq s : LS_i = \sum_{e \in E} Q_e$ *where* $E = \{e | \exists y_1, y_2 : G(q_e, \alpha_i) = (y_1, y_2, L)\}$.
(6) $\forall 1 \leq i \leq s : RS_i = \sum_{e \in E} Q_e$ *where* $E = \{e | \exists y_1, y_2 : G(q_e, \alpha_i) = (y_1, y_2, R)\}$.
(7) $\forall 1 \leq i, j \leq s : W_{i,j} = \sum_{e \in E} Q_e$ *where* $E = \{e | \exists y_1, y_2 : G(q_e, \alpha_i) = (y_1, \alpha_j, y_2\}$.
(8) $M = \sum_{i=0}^{h-1} q^i$.
(9) $\forall 1 \leq i < j \leq m : Q_i \&\& Q_j = 0$.
(10) $\forall 1 \leq i \leq m, 1 \leq j \leq l : Q_i \&\& R_j = 0$.
(11) $\forall 1 \leq i < j \leq l : R_i \&\& R_j = 0$.
(12) $Q_1 + \cdots + Q_m = 2^{k-1} - 1$.
(13) $R_1 + \cdots + R_l = 2^{k-1}$.
(14) $\forall 1 \leq i < j \leq s : U_i \&\& U_j = 0$.
(15) $U_1 + \cdots U_s = 2^k - 1$.
(16) $\forall 1 \leq i < j \leq s : T_i \&\& T_j = 0$.
(17) $T_1 + \cdots T_s = 2^{hk} - 1$.
(18) $q^d + 2q^2(M(\sum_{i=1}^{s} LS_i \&\& U_i) \&\& P) || 2(M(\sum_{i=1}^{s} RS_i \&\& U_i) \&\& P) = qP$.
(19) $1 \preceq Q_1$
(20) $\forall 1 \leq j \leq m : 2(\sum_{i=1}^{s} G_{i,j} \&\& U_i) \preceq Q_j$.
(21) $\forall 1 \leq j \leq l : 2(\sum_{i=1}^{s} F_{i,j} \&\& U_i) \preceq R_j$.
(22) $\forall 1 \leq i \leq s : P \&\& T_i \preceq M \cdot U_i$.
(23) $\forall 1 \leq i \leq s : P \&\& !_{hk} T_i \preceq M \cdot !_k U_i$.
(24) $\forall 1 \leq j \leq s : 2(P \&\& M(\sum_{i=1}^{s} W_{i,j} \&\& U_i)) \preceq T_j$.
(25) $\forall 1 \leq j \leq s : 2(!_{hk} P \&\& T_j \&\& M(2^{k-1} - 1)) \preceq T_j$.
(26) $\forall 1 \leq j \leq s : 2(!_{hk} P \&\& !_{hk} T_j \&\& M(2^{k-1} - 1)) \preceq !_{hk} T_j$.
(27) $\forall 1 \leq j \leq s : 2(P \&\& M((\sum_{c=1}^{s} \sum_{i=1}^{s} W_{i,c} \&\& U_i) - (\sum_{i=1}^{s} W_{i,j} \&\& U_i))) \preceq !_{hk} T_j$.
(28) $q^d \preceq P$.
(29) $(\sum_{i=0}^{h-1}(\tilde{x}(2^{k-1})^i \&\& q^i(q-1))) \&\& M \preceq T_1$.
(30) $(!_{hk}(\sum_{i=0}^{h-1}(\tilde{x}(2^{k-1})^i \&\& q^i(q-1)))) \&\& M \preceq T_2$.
(31) $\tilde{x} = 2^{d+1}(2^{x_1} - 1 + 2^{x_1+1}(2^{x_2} - 1) + \cdots + 2^{\sum_{j=1}^{i-1} x_j + 1}(2^{x_1} - 1) + \cdots + 2^{\sum_{j=1}^{n-1} x_j + 1}(2^{x_n} - 1))$.

Note that the first two sets of inequalities are needed to make sure the main variables can be considered as strings of the required length. All other variables are defined directly from these main variables, and as such it is not necessary to add a condition that specifies their amount of bits.

Corollary 8. *Every r.e. set is diophantine.*

Proof: Although the list of equations equivalent to $x \in S$ given in lemma 37 contains some sums and bounded universal quantifiers, what they sum over and are bounded by is determined by the algorithm. As such, for every algorithm there is a fixed number of exponential diophantine equations such that the algorithm halts if and only if the equations have solutions in the natural numbers. Therefore, every r.e. set is exponential diophantine. And since it was already shown that every exponential diophantine set is diophantine, this implies that every r.e. set is diophantine. □

5. Hilbert's tenth problem is unsolvable

We can now quite easily show that Hilbert's tenth problem is unsolvable.

Corollary 9. *There is no algorithm that, given any polynomial P determines whether P has a zero in the natural numbers.*

Proof: Suppose such an algorithm would exist. Take any r.e. set S that is not recursive. By the previous results there is a polynomial P such that $\mathbf{a} \in S$ if and only if $\exists \mathbf{x} \in \mathbb{N}^n : P(\mathbf{a}, \mathbf{x}) = 0$. The algorithm could be used to determine whether there is such an $\mathbf{x}$, and could therefore decide whether $\mathbf{a} \in S$. This contradicts S not being recursive, so such an algorithm cannot exist. □

5.1. **Other proofs.** Before Matijasevic showed that every r.e. set is diophantine it was shown by Davis [2] in 1950 that every r.e. set S can be written in what is now known as the Davis normal form

$$S = \{\mathbf{a} : \exists x_1 \forall x_2 \leq x_1 \exists x_3, \cdots, x_n [P(\mathbf{a}, \mathbf{x}) = 0]\}.$$

In 1961 Davis, Putnam and Robinson [3] showed that any set that can be put in Davis normal form is exponential diophantine. Thus, Matijasevic only had to show that any exponential diophantine set is diophantine to show that every r.e. set is diophantine.
Newer proofs tend to be more direct and show that algorithms can be represented in an exponential diophantine way. Some notable examples include a proof using Turing machines by Matijasevic [7] and a proof using so-called register machines, a different formalization of algorithms, by Davis and Matijasevic [5].
The proof presented above is quite similar to the proof by Davis and Matijasevic except for the choice of formalization. The proof using Turing machines by Matijasevic on the other hand is significantly less explicit.

6. Discussion

6.1. **Practicality of the diophantine description.** While we now have a way to find a diophantine description of any recursively enumerable set, this description is clearly unsuitable for practical use in most computations. To give some idea of how unsuitable the diophantine description is, I will give an example here. We will consider the set of powers of 2, and try to give a diophantine proof that $2 = 2^1$ is a member of this set. We already have a convenient exponential diophantine description of this set, so we will only need to use the step from exponential diophantine to diophantine.

Recall that $p = 2^n$ if and only if the following system can be satisfied in the natural numbers:
(A) $a = y_{2n+2}(n+1)$
(B) $c = y_{2a}(n+1)$
(C) $d = y_a(n+1)$
(D) $(2pd - 2c)^2 < d^2$
We want to look at the case $n = 1$, so (A) becomes $a = y_4(2) = 8$. This gives $c = y_{16}(2) = 32$, $d = y_8(2) = 16$, so taking $p = 2$ one gets $(2pd - 2c)^2 = (4 * 16 - 2 * 32)^2 = 0 < 16^2$. Up to now

all the numbers required are quite managable. However, in order to give a diophantine proof that $2 \in \{x \in \mathbb{N} : \exists n \in \mathbb{N}[x = 2^n]\}$, a diophantine proof that $y_4(2) = 8, y_{16}(2) = 32$ and $y_8(2) = 16$ is still required. For this we can use the fact that if $a \geq 2$ and $n, y > 0$, then $y = y_a(n)$ if and only if the following system can be satisfied in the natural numbers:

(A) $d^2 - (a^2 - 1)y^2 = 1$
(B) $e^2 - (a^2 - 1)f^2 = 1$
(C) $g^2 - (b^2 - 1)h^2 = 1$
(D) $n \leq y$
(E) $y^2|f$
(F) $e|y - h$
(G) $e|a - b$
(H) $2y|h - n$
(I) $2y|b - 1$

Let us first look at the proof of $y_{16}(2) = 32$, so we want to show that $a = 16, n = 2, y = 32$ works. (A) states nothing other than that (d, y) is a solution to the Pell equation with parameter a. The appropriate d is $d = 127$. Next, we need another solution (e, f) to the same Pell equation, with the property that $y^2|f$. Following the proof that if the system can be satisfied then $y = y_a(n)$, we can take $(e, f) = (x_{16}(k), y_{16}(k))$ with k any multiple of $yn = 64$. Since this is already even, we do not have to take k a multiple of $2yn$ to guarantee e and $2y$ to be relatively prime. We are interested in finding a small solution, so we could take $k = 64$. Obiously e and f will be a bit larger than d and y. In fact, $f = y_{16}(64)$ is approximately 32^{63}. Or, to be more exact,

$$e = x_{16}(128) = 10031950124735540791749555078054269642123871406009588136\backslash 1645353459410376114312309625382673055744 1,$$

$$f = y_{16}(128) = 62822508550640910332428858385102791097105202345836082485\backslash 8663969274684506733003969947916309616 64.$$

Then we have to find a b such that $e|a - b$ and $2y|b - 1$. The smallest b for which this holds is

$$b = 49156555611204149879572819882465921246406969889446981867206223195111 0\backslash 842960130317164375097973146 25,$$

approximately 49 times e. Finally, we have to find a l such that if we take $h = y_b(l)$, then $2y|h - n$ and $e|y - h$. This could be quite hard to compute, since h will be approximately $(2b)^{l-1}$. We are "lucky", however, since $l = 2$ works. The remaining two numbers can therefore be taken

$$g = x_b(2) = 48327339191148123862748844482527493599668356906462189367784609 8\backslash 8563354993709159764514450649680895613611218203210483416198382152462240 20\backslash 4699886161536788466817485306560602116668078568497722477781249,$$

$$h = y_b(2) = 98313111222408299759145639764931842492813939778893963734412446 3\backslash 9022216859202606343287501959462925 0.$$

This g is the largest of the unknowns that have to be computed in order to show that $2 = 2^1$, with 196 digits. Essentially, the problem is that the size of the unknowns needed to show that 2^n is a power of 2 in this way grows at least about as fast as $n^{(n^{(n^2)})}$.

While it should be noted that the diophantine description of the powers of two given here has not been optimized for small unknowns, the set being considered is one of the simplest sets to describe in an exponential diophantine way. As such, the growth of the unknowns in and the size of the diophantine description do suggest that there might be a rather high "complexity cost" of exponentiation in some sense.

6.2. **Complexity.** While corollary 9 shows that Hilbert's tenth problem is unsolvable, one could ask a more fine-grained question.

For which $n \in \mathbb{N}$ is there an algorithm that determines whether any given diophantine equation with n unknowns has a solution?

Of course this question is only interesting if there are n for which such an algorithm exists, and n for which such an algorithm does not exist, but this is the case.

After all, if one takes any r.e. but not recursive set S, one can construct a diophantine representation of S by using the techniques from the previous section. If one takes n to be the number of unknowns in this diophantine representation, there is no algorithm that determines whether any given diophantine equation in n unknowns has a solution.
On the other hand, there are algorithms that determine whether any given diophantine equation in one unknown has a solution. For example, one could use the fact that for large values of the unknown the highest order term will dominate to find a bound on the size of possible solutions and do a brute-force search of all values under this bound.

Since unsolvability of diophantine equations in n unknowns implies unsolvability of diophantine equations in more than n unknowns there must be N such that there is an algorithm that determines whether any diophantine equation in up to N unknowns has a solution, but there is no such algorithm for diophantine equations in more than N unknowns.
There has not been much progress on finding a lower bound greater than 1 for N, but significant progress towards finding and improving an upper bound has been made since Matijasevic first showed that there must be such an N. For example, in 1975 Matijasevic and Robinson [8] showed that for any polynomial P in any number of unknowns there is a polynomial Q in 13 unknowns such that P has a solution if and only if Q has a solution, so $N \leq 12$. In 1982 Jones [4] published a proof, based on a proof by Matijasevic, that the amount of unknowns can be further reduced to 9, so $N \leq 8$.

So far all improvements on the upper bound on N are based on reducing any diophantine equation to one in fewer unknowns. I will consider a method that could perhaps be used to show the limitation of finding upper bounds for N in this way.

The idea is to find an appropriate measure of complexity for diophantine sets and polynomials, and show that any polynomial must have at least the same complexity as the diophantine set it represents.

6.3. **Algorithm size.** Using the method shown before one can create a diophantine description for the r.e. set corresponding to any algorithm.
The usual measure for the complexity of an algorithm, the amount of time it needs to terminate, only influences the values the unknowns have at the zero of the polynomial, not the amount of unknowns that are needed or the degree of the polynomial, so it is not very useful here.
The amount of unknowns used does however grow with the amount of states the algorithm has. While the dipohantine description obtained uses more unknowns than needed, this does suggest that the appropriate measure for the complexity of an algorithm is the length of the algorithm.
We can then define the complexity of a diophantine set S as the length of the shortest algorithm that semi-decides S.

For any polynomial $P(\mathbf{a}, \mathbf{x})$ we can find an algorithm that semi-decides the diophantine set determined by P by performing a brute-force search of all possible values for $\mathbf{x}$. The length of this algorithm depends on the number of unknowns, but also on the degree of P and the size of the coefficients in P.

As such, while this complexity for an algorithm can put a lower bound on the complexity of the corresponding polynomial, this will only bound the combined amount of unknowns, degree and size of coefficients.

6.4. **Davis Normal Form.** As mentioned before, Davis showed that any r.e. set S can be written in the Davis normal form,

$$S = \{\mathbf{a} : \exists x_1 \forall x_2 \leq x_1 \exists x_3, \cdots, x_n [P(\mathbf{a}, \mathbf{x}) = 0]\}$$

for some polynomial P.
Obviously, if S cannot be written in Davis normal form with less than n unknowns there is no diophantine description of S with less than $n-1$ unknowns. Likewise, if S has a diophantine description with n unknowns than S can be written in Davis normal form with $n+1$ unknowns. However, finding a minimal Davis normal form does not seem to be easier than finding a minimal diophantine description.
At first glance the sets that have a Davis normal form with two unknowns seem like an interesting special case since it might have some relation to the diophantine sets that need at least two unknowns to describe. Slightly closer inspection shows that there is no such relation though.

Lemma 38. *Let* $S = \{\mathbf{a} \in \mathbb{N}^k : \exists x_1 \forall x_2 \leq x_1 [P(\mathbf{a}, x_1, x_2) = 0]\}$ *for some polynomial* P. *Then there is a polynomial* Q *such that* $S = \{\mathbf{a} \in \mathbb{N}^k : \exists x [Q(\mathbf{a}, x) = 0]\}$.

References

[1] Alonzo Church. A note on the entscheidungsproblem. *The Journal of Symbolic Logic*, 1(1):40–41, 1936.
[2] Martin Davis. Arithmetical problems and recursively enumerable predicates. *The Journal of Symbolic Logic*, 18(1):33–41, 1953.
[3] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential diophantine equations. *Annals of Mathematics*, 74(3):425–436, 1961.
[4] James Jones. Universal diophantine equation. *The Journal of Symbolic Logic*, 47(3):549–571, 1982.
[5] James Jones and Yuri Matijasevic. Register machine proof of the theorem on exponential diophantine representation of enumerable sets. *The Journal of Symbolic Logic*, 49(3):818–829, 1984.
[6] Yuri Matijasevic. Enumerable sets are diophantine. *Soviet Mathematics: Doklady*, 11:354–357, 1970.
[7] Yuri Matijasevic. A new proof of the theorem on exponential diophantine representation of enumerable sets. *Journal of Mathematical Sciences*, 14(5):1475–1486, 1980.
[8] Yuri Matijasevic and Julia Robinson. Reduction of an arbitrary diophantine equation to one in 13 unknowns. *Acta Arithmetica*, 27(3):521–553, 1975.
[9] Craig Smorynski. *Logical Number Theory I*. Springer-Verlag, 1991.
[10] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.