CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Register Machines as Diophantine Equations

AUTHOR: Nick Murphy

SUPERVISOR: Dr. Kevin Cheung

DATE: May 4th 2020

Contents

1 Introduction

Hilbert's 10th problem is the 10th of 23 problems proposed by David Hilbert in 1900 [4] [3]. Titled "Determination of the Solvability of a Diophantine Equation", it asks:

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

Much work was done on this problem from 1949 onwards by many mathematicians including Martin Davis, Julia Robinson and Hilary Putnam. This culminated in 1970 with the work of Yuri Matiyasevich [10], who finished off the proof that it is unsolvable by showing the existence of an Ordinary Diophantine Equation that exhibits exponential growth. Since then, there have been efforts to find an upper bound on the degree and dimension for which the problem is unsolvable. Here degree simply means the highest power present in a polynomial, whereas dimension is the number of indeterminates. In 1938, Thoralf Skolem proved that any Ordinary Diophnatine Equation could be reduced to an equivalent equation of degree at most 4 [11]. He did this by repeatedly replacing indeterminates with a new indeterminate equal to their product. For example, any instance of X^{16} in a polynomial could be replaced by a new indeterminate defined as follows: $Y = Z^2, Z = W^2, W = X^2$ then Y^2 could replace X^{16} and there would be no instance of a power greater than 2. With this new system of quadratic Diophantine Equations, one could combine them into a single equation by summing the square of the difference of each equation and equating it to 0, resulting in an equation with degree at most 4. For the dimension, Matiyasevich has reduced the upper bound to 9 indeterminates [9] [5].

In this write-up, I will go about showing the unsolvability of Hilbert's

10th problem by demonstrating a surprising equivalence: one between Register Machines and Exponential Diophantine Equations. The distinction between Ordinary and Exponential Diophantine Equations will be made clear in the next section. I will be using modified versions of techniques used by Gregory J. Chaitin [1] in arithmetising a lisp-interpreting register machine and the work of Jones and Matiyasevich [6]. The former formulates a register machine that operates on natural numbers which are implicitly interpreted as 8-bit character strings. Operations on registers consist of concatenating and deleting characters from said strings as well as comparing the last character in one string to the last character in another. Chaitin considers a Diophantine Equation to be equivalent to a given register machine if the equation has a solution if and only if the register machine halts on a given input. Jones and Matiyasevich formulate a more basic machine in which operations consist of addition and subtraction of the number 1. They consider a Diophantine Equation to be equivalent to a register machine if the parameter values that yield a single solution to the equation are precisely the input values which the register machine "accepts". They say a register machine "accepts" an input if it eventually halts with a 0 in every register when given that input. The formulation in this write-up is more similar to the latter, but perhaps even more basic. It consists of only three types of instructions: an increment instruction, a decrement instruction, and a conditional jump instruction.

In the cited works, implications of conditions and equations generated by the conversion procedure are simply stated as fact. This write-up seeks to provide a concrete procedure for turning an arbitrary register machine into an "equivalent" Exponential Diophantine Equation and to prove the validity of such a procedure. Equivalent here simply means the resulting Diophantine Equation has a solution if and only if the Register Machine it was constructed from halts. Furthermore, within the solution lies the output of said Register Machine. For certain basic functionality relevant to

all Register Machine formulations such as forcing one instruction to directly follow another, equations from the cited works are reproduced here. In addition, the "trick" of summing indeterminates which represent when the value in a register changes is taken from Chaitin [1]. Other equations and conditions are of my own creation.

2 Definitions

For the purposes of this construction we use the following definitions.

2.1 Register Machine

A Register Machine (RM) consists of an unbounded number of registers — labelled $R_0, R_1, R_2, ...$ — each of which may contain a natural number (including 0) of any size. A RM program consists of a finite sequence of instructions — labelled $I_0, I_1, I_2, ..., I_{n-1}$ — which determine the process taken by the RM. Unless otherwise stated, each instruction I_i is followed immediately by instruction I_{i+1}. Each instruction is of one of the following 3 forms:

1. $I(k)$ — Add one to the number in register R_k.
2. $D(k)$ — Subtract one from the number in register R_k. If R_k contains 0 already, then the number remains unchanged.
3. $G(k, i)$ — If the number in register R_k is not equal to 0, then "goto" instruction I_i. i.e. Execute instruction I_i as the next instruction in the sequence.

A RM halts when it tries to execute an instruction that does not exist. This can happen when it reaches the end of the list or when it attempts to "goto" an index i such that instruction I_i does not exist. In either case, we call this the implicit "halting" instruction and label it I_n. A particular RM program

may take a fixed number k of inputs, which will be the initial contents of the first k registers $(R_0, R_1, ..., R_{k-1})$. All registers that are not inputs to the RM initially contain the number 0. Note that since a given RM program only has finitely many instructions and finitely many inputs, only a finite amount of registers are actually used. Let M be 1 + the maximum index of either an input or a register referred to by an instruction. Then the output of the RM consists of the contents of the first M registers $R_0, R_1, ..., R_{M-1}$ when it halts.

2.2 Example

Take for example the following Register Machine that takes two inputs:

I_0 : $G(1, 3)$
I_1 : $I(2)$
I_2 : $G(2, 6)$
I_3 : $D(1)$
I_4 : $I(0)$
I_5 : $G(1, 3)$

The two inputs for this register machine are the initial contents of registers R_0 and R_1. Suppose R_0 contains x and R_1 contains y. Then the computation would proceed as follows:

1. The first instruction is $G(1, 3)$, in other words, if R_1 does not contain zero then goto instruction I_3. Suppose R_1 does contain zero. Then the next instruction should be I_1

   (a) Instruction I_1 is $I(2)$ which means register R_2 will be incremented by 1. Since R_2 is not an input to this machine, it should now contain the number 1.

   (b) The next instruction is I_2, or $G(2, 6)$. Since R_2 does not contain zero, the next instruction should be I_6, but I_6 does not exist, so we halt. In this case, the final contents of R_0 is x, the same as the initial contents.

2. Suppose R_1 contains $y > 0$, then the next instruction is I_3.

3. I_3 is $D(1)$ so R_1 is replaced with $y - 1$ and the next instruction is I_4

4. I_4 is $I(0)$ so R_0 is replaced with $x + 1$ and the next instruction is I_5

5. I_5 is $G(1, 3)$. If R_1 contains zero at this point, the Machine halts with $x + 1$ in register R_0. If not, the process repeats from instruction I_3 and continues until we have $y - y = 0$ in R_1 and $x + y$ in R_0

From this we can conclude that this Register Machine takes 2 inputs and results in their sum being placed in register R_0.

2.3 Diophantine Equation

A Diophantine Equation is an equation of the form $L = R$ where L and R are Diophantine Expressions. Diophantine Expressions consist of finitely many additions, multiplications, and exponentiations of natural number constants and indeterminates $X_0, X_1, ..., X_n$ which take on natural number values. In particular these are usually referred to as "Exponential Diophantine Equations". A Diophantine Equation without any exponentiation is referred to as an "Ordinary Diophantine Equation". A solution to a Diophantine Equation is a tuple $(x_0, x_1, ..., x_n)$ of natural numbers which satisfies the condition $L = R$.

2.4 Examples

The equation

$$X_0^2 + X_1 X_2 = 2^{X_0} + X_1^{X_2}$$

is Exponential Diophantine. Note, however, that it is not an Ordinary Diophantine Equation because it makes use of an indeterminate as an exponent. On the other hand, the equation

$$X_0^2 = 4X_0$$

is an Ordinary Diophantine Equation.

This write-up will make use of Exponential Diophantine Equations because it makes much of the construction involved less cumbersome and allows the result to yield a unique solution. Using techniques by Davis, Matiyasevich, Robinson, and Putnam [2], any Exponential Diophantine Equation can be converted into an Ordinary Diophantine Equation. This is done by introducing a system of Ordinary Diophantine Equations with parameters a, b, and c which has at least one solution if and only if $a = b^c$. See the Appendix for a more detailed explanation. The result is a system of strictly Ordinary Diophantine Equations. However, this system may have more than one solution, so we cannot guarantee uniqueness when using this method.

2.5 Combining Diophantine Equations

Any number of Diophantine Equations can be combined into a single equation by using the following method. A tuple $(x_0, x_1, ..., x_n)$ is a solution to the system of equations

$$L_0 = R_0,$$

$$L_1 = R_1,$$

$$...$$

$$L_m = R_m$$

if and only if it is a solution to the single equation

$$L_0^2 + R_0^2 + L_1^2 + R_1^2 + ... + L_m^2 + R_m^2 = 2L_0R_0 + 2L_1R_1 + ... + 2L_mR_m$$

This is proven in Lemma 1.

For example, the famous "Fermat's last theorem" states that the equation

$$x^n + y^n = z^n$$

has no natural number solutions for $n > 2$, $x, y, z > 0$. This would be equivalent to claiming the system of Diophantine Equations

$$x^n + y^n = z^n,$$

$$a + 3 = n,$$

$$b + 1 = x,$$

$$c + 1 = y,$$

$$d + 1 = z$$

has no solutions. Using the technique defined above this is equivalent to claiming that the single equation

$$a^2+6a+b^2+2b+c^2+2c+d^2+2d+n^2+x^{2n}+2x^ny^n+y^{2n}+z^{2n}+x^2+y^2+z^2+12$$

$$= 2an + 2bx + 2cx + 2dx + 2x^nz^n + 2y^nz^n + 6n + 2x + 2y + 2z$$

has no solutions.

2.6 Notation

L ⇒ R

The notation $L \Rightarrow R$ should be understood as follows: let L and R be Diophantine expressions which resolve to natural numbers and let $L = \sum_k l_k 2^k$ and $R = \sum_k r_k 2^k$ be the binary expansions of L and R respectively. This notation asserts that each bit l_k is less than or equal to the corresponding bit r_k, including leading zeros if necessary. In other words, each bit in the binary expansion of L "implies" the corresponding bit in the binary expansion of R. For example, $9 \Rightarrow 27$ would be true since each bit in 01001_2 is less than or equal to the corresponding bit in 11011_2. However, $13 \Rightarrow 27$ would be false since the 2^2 bit in 01101_2 is greater than the 2^2 bit in 11011_2. See Figure 1 for an illustration.

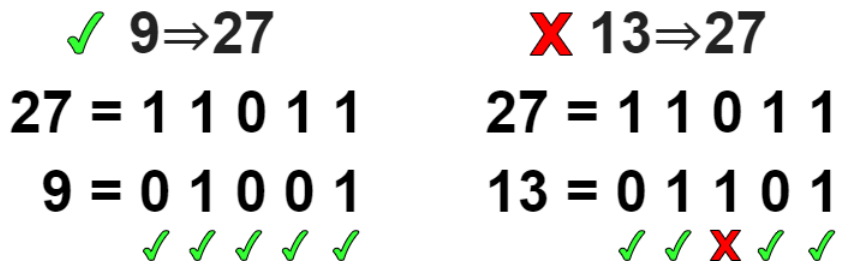

Figure 1: Example of $L \Rightarrow R$

Base-q digit notation

The notation $X[j]$ is meant to refer to the q^j coefficient of the base q expansion of X. In other words, $X = \sum_{j=0}^{n} X[j]q^j$ for some n.

2.7 Lucas' theorem

Lucas' theorem states the following: let $n = \sum_i n_i p^i$ and $k = \sum_i k_i p^i$ be the base-p expansions of n and k, where p is prime. Then

$$\binom{n}{k} \equiv \prod_i \binom{n_i}{k_i} (mod\ p).$$

We use the convention $\binom{n}{k} = 0$ if $k > n$. In particular, $\binom{n}{k}$ is divisible by p if and only if at least one base-p digit of k is larger than the corresponding base-p digit of n. [8]

In the special case of the above theorem where $p = 2$, it can be restated as $\binom{n}{k}$ is even if and only if at least one digit of the base-2 expansion of k is greater than the corresponding digit of the base-2 expansion of n. Conversely, $\binom{n}{k}$ is odd if and only if each base-2 digit of k is less than or equal to the corresponding base-2 digit of n. i.e. if and only if $k \Rightarrow n$.

3 The construction

With these definitions out of the way we can begin outlining the construction of an equivalent Diophantine Equation given an arbitrary Register Machine.

The entire construction will be laid out in this section and the following section will prove that each piece of the construction is valid. Suppose we have a Register Machine RM. Let RM consist of n instructions $\{I_0, I_1, ..., I_{n-1}\}$, and m inputs located in registers $R_0, R_1, ..., R_{m-1}$. Let M be the maximum between m and 1 + the largest index of a register referred to by an instruction. If $M > m$, we say the registers $R_m, R_{m+1}, ..., R_{M-1}$ are "implicit inputs", all equal to zero.

Inputs to a Register Machine can affect the computation path taken by said machine and thus, affect its output or even whether it halts or not. For this reason, we want the inputs to affect the solution of the Diophantine Equation as well. Define parameters for each input of RM as follows:

$$\mathsf{N}_0$$

$$\mathsf{N}_1$$

$$...$$

$$\mathsf{N}_{M-1}$$

which should be equal to the initial contents of registers $R_0, R_1, ..., R_{M-1}$.

We will want to encode the contents of each register at each step of the computation. To do this we can use indeterminates X_k encoded in a sufficiently high base which we will call q and have the j^{th} digit in this base represent the contents of a register following the j^{th} step of the computation. So the q^0 digit would be the initial contents of the register, the q^1 digit would be the content of the register following step 1 and so on. This base needs to be large enough so that no register can ever contain a number greater than the base itself. We can define this base q as follows:

$$\mathsf{E} = \mathsf{T} + \mathsf{n} + 1 + \sum_{k=0}^{M-1} \mathsf{N}_k \tag{1}$$

$$\mathsf{q} = 2 \times 2^{\mathsf{E}} \tag{2}$$

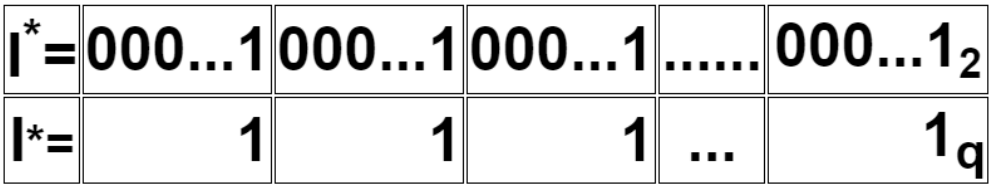

Figure 2: Illustration of l^* in base 2 (top) and base q (bottom)

where T is an indeterminate representing the number of steps taken by RM and n is the number of instructions. If a Register Machine halts on a given input, it must halt in a finite number of steps. In addition, since the only instructions are $I(k), D(k)$, and $G(k,i)$, a register can only increase its contents by 1 per step and so it is not possible for the contents to be greater than the initial contents plus the number of steps. Thus our choice of base is sufficient for encoding the contents of a register over time.

We will need a way to assert which instruction is being executed at a given time. Define an indeterminate l^* as follows

$$(\mathsf{q}-1)\mathsf{l}^* = \mathsf{q}^{\mathsf{T}} - 1. \tag{3}$$

Note that this forces l^* to be T 1's in base q i.e. $\mathsf{l}^* = 11111...1_q$. See Figure 2 for an illustration. Let l_i be an indeterminate which represents whether or not an instruction I_i is executed following the j^{th} step. We want $\mathsf{l}_i[j]$ to be a 1 if it is being executed and a 0 if it is not. Conditions of the form

$$\mathsf{l}_i \Rightarrow \mathsf{l}^* \tag{4}$$

force each base-q digit of l_i to be a zero or a one. To ensure that the first instruction executed is I_0 simply use the condition

$$1 \Rightarrow \mathsf{l}_0. \tag{5}$$

The final step taken by RM is the implicit halting instruction. Call this instruction I_n and use the equation

$$\mathsf{q}^{\mathsf{T}-1} = \mathsf{l}_n. \tag{6}$$

This assures us that $\mathsf{I}_n[\mathsf{T}-1]=1$ and the rest of the base-q coefficients are 0's.

Since a register machine only executes one instruction at a time, we need an additional equation to make sure only one indeterminate has a 1 in a particular coefficient. The following equation accomplishes this:

$$\mathsf{I}^* = \sum_{i=0}^{n} \mathsf{I}_i \tag{7}$$

(see Lemma 7). This only works because we made sure the base q was larger than the number of instructions, so the only way for the equality to hold is for one and only one indeterminate I_i to have a 1 in a particular base-q coefficient.

In order to encode the order of instructions, we need a way to encode whether or not each register contains a 0 over time. We will use indeterminates of the form X_k to encode the contents of register R_k over time. The following condition is necessary to make sure that each base-q digit of X_k is less than half of q and that X_k consists of no more than T base-q digits:

$$\mathsf{X}_k \Rightarrow (2^{\mathsf{E}}-1)\mathsf{I}^*. \tag{8}$$

With this condition asserted, we will define a new indeterminate Z_k for each register such that $\mathsf{Z}_k[j]=1$ if $\mathsf{X}_k[j]=0$ and $\mathsf{Z}_k[j]=0$ otherwise. These conditions force this to happen:

$$\mathsf{Z}_k \Rightarrow \mathsf{I}^* \tag{9}$$

$$2^{\mathsf{E}}\mathsf{Z}_k \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k \tag{10}$$

$$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_k + (2^{\mathsf{E}}-1)\mathsf{I}^*. \tag{11}$$

See Lemma 8 for a proof and Figure 3 for an example.

We need a way to encode when the contents of a register are about to change, and what they are about to change to. For each register R_k, define a set $S_k \subseteq \{I_0, I_1, ..., I_{n-1}\}$ where $I_i \in S_k \iff I_i$ is $I(k)$ or I_i

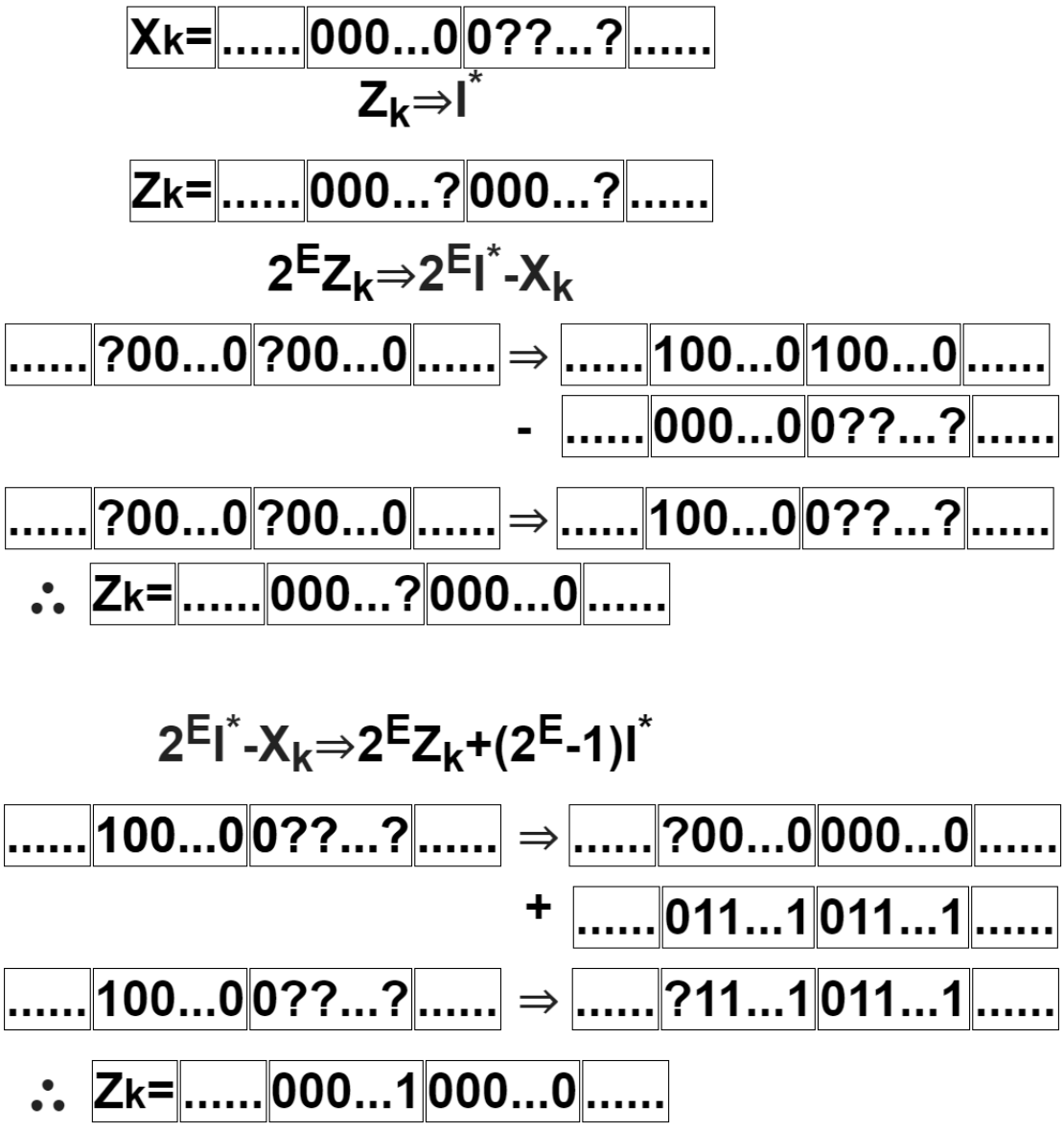

Figure 3: Example solution to Z_k where '?' is a wild-card (either 1 or 0)

is $D(k)$. For each $I_i \in S_k$ we will make an indeterminate $\mathsf{S}(k,i)$. If I_i is the next instruction to be executed after j steps, the q^j coefficient should be the number it is about to change to. If I_i is not the next instruction, this coefficient should be 0. If I_i is the increment instruction $I(k)$ which increments register R_k we would use

$$\mathsf{S}(k,i) \Rightarrow (\mathsf{q}-1)\mathsf{I}_i \tag{12}$$

$$\mathsf{S}(k,i) \Rightarrow \mathsf{X}_k + \mathsf{I}^* \tag{13}$$

$$\mathsf{X}_k + \mathsf{I}^* \Rightarrow \mathsf{S}(k,i) + (\mathsf{q}-1)(\mathsf{I}^* - \mathsf{I}_i). \tag{14}$$

A proof of this is given in Lemma 9. Note the first condition forces $\mathsf{S}(k,i)[j]$ to be 0 whenever $\mathsf{I}_i[j]$ is 0. In other words, when instruction I_i is not about to be executed. Similarly, for instructions of the form $D(k)$ we would use

$$\mathsf{S}(k,i) \Rightarrow (\mathsf{q}-1)\mathsf{I}_i \tag{15}$$

$$\mathsf{S}(k,i) \Rightarrow \mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k \tag{16}$$

$$\mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k \Rightarrow \mathsf{S}(k,i) + (\mathsf{q}-1)(\mathsf{I}^* - \mathsf{I}_i). \tag{17}$$

We also need to encode when register R_k is not being changed by any instruction over time. For this we will use indeterminates of the form $\mathsf{C}(k)$. In this case, if none of the instructions that would change R_k is next to be executed following the j^{th} step, $\mathsf{C}(k)[j]$ should be the same as $\mathsf{X}_k[j]$. Otherwise, it should be 0. The following conditions accomplish this:

$$\mathsf{C}(k) \Rightarrow (\mathsf{q}-1)(\mathsf{I}^* - \sum_{I_{i'} \in S_k} \mathsf{I}_{i'}) \tag{18}$$

$$\mathsf{C}(k) \Rightarrow \mathsf{X}_k \tag{19}$$

$$\mathsf{X}_k \Rightarrow \mathsf{C}(k) + (\mathsf{q}-1)(\sum_{I_{i'} \in S_k} \mathsf{I}_{i'}). \tag{20}$$

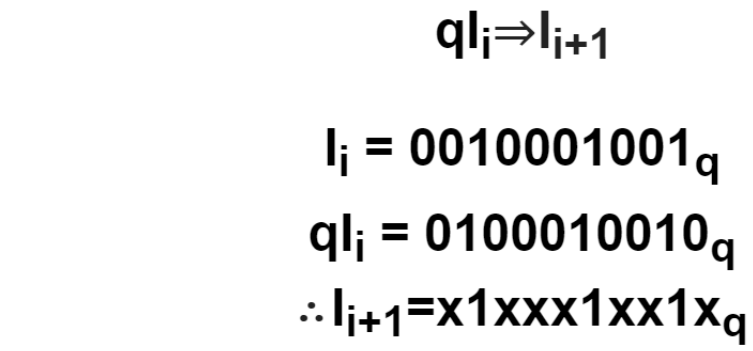

Figure 4: $q\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1}$ Example

Note that the q^j digit of the right hand side of the third condition is precisely $\mathsf{C}(k)[j]$ if none of the instructions that change R_k is the next to be executed following the j^{th} step. See Lemma 11 for a proof.

Before we get to the final encoding of X_k, we need a way to encode the order of instructions. If an instruction I_i is of the form $I(k)$, $D(k)$, or $G(k, i+1)$ then the next instruction is definitely instruction I_{i+1}. Therefore, every time we have $\mathsf{I}_i[j] = 1$ we want $\mathsf{I}_{i+1}[j+1] = 1$. The following condition accomplishes this:

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1} \tag{21}$$

See Figure 4 for an illustration. If instruction I_i is of the form $G(k, i')$ where $i' \neq (i+1)$, it should be read as "if register R_k does not contain 0, goto instruction $I_{i'}$". Therefore the next instruction will be I_{i+1} if register R_k contains 0 and $I_{i'}$ if it does not contain 0. We can make use of the Z_k indeterminate from earlier since it encodes whether register R_k contains 0 over time. The following conditions assure us that the right instruction comes next:

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1} + \mathsf{I}_{i'} \tag{22}$$

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i'} + \mathsf{q}\mathsf{Z}_k. \tag{23}$$

Note that the first condition assures us that one (and only one) of the indeterminates I_{i+1} and $\mathsf{I}_{i'}$ will have a 1 in the coefficient immediately following

a coefficient in which the indeterminate I_i has a 1. The second condition forces said coefficient of $\mathsf{I}_{i'}$ to be 0 if the coefficient of Z_k is 1. In other words, if register R_k contains 0.

Note: if there are any instructions of the form $G(k, i')$ where $i' > n$, then the result is always the same: RM halts. Because of this all instructions of this form should be treated as if the instruction were $G(k, n)$. In other words, replace $\mathsf{I}_{i'}$ with I_n in equations 22 and 23.

Finally, we need a way to tie all of this together and make sure X_k does in fact encode the contents of register R_k over time. We know that each digit of one of $\mathsf{C}(k)$, $\mathsf{S}(k, i)$ for some i correspond to what the contents of register R_k should be after executing the next step in the computation, and the rest are equal to 0, so their sum should be precisely what the next coefficient of X_k is equal to. We also know that N_k is the initial contents of register R_k, so that should be $\mathsf{X}_k[0]$. Lastly, we know that $\mathsf{C}(k)[\mathsf{T} - 1]$ corresponds to the value of register R_k just before executing the final step, the implicit halting instruction. In other words, it corresponds to the output of register R_k. Let the indeterminate O_k represent the output of register R_k. Then the equation

$$\mathsf{O}_k\mathsf{q}^{\mathsf{T}} + \mathsf{X}_k = \mathsf{N}_k + \mathsf{q}(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i')) \tag{24}$$

uniquely determines X_k as the contents of register R_k over time and O_k as the final contents of register R_k. This is proven in Lemma 13 and Lemma 14.

To write conditions of the form $L \Rightarrow R$ in Diophantine Equation form, we simply need to expand them into the following 7 equations:

$$l = L$$

$$r = R$$

$$t = 2^r$$

$$(1+t)^r = at^{l+1} + bt^l + c$$
$$c + d + 1 = t^l$$
$$b + e + 1 = t$$
$$b = 2f + 1,$$

relabelling indeterminates so that there are no collisions. This system asserts that the binomial coefficient $\binom{R}{L}$ is odd (see Lemma 2). By using Lucas' theorem with $p = 2$, we have that this system has a solution if and only if the condition $L \Rightarrow R$ is satisfied (by Lemma 2). Once we have all these equations and have converted all the conditions into Diophantine Equation systems, simply use the sum of squared differences technique to combine them all into one single equation.

Tables 1-4 offer a summary of the procedure.

Table 1: Register Content Encoding Summary

X_k	Z_k
$\mathsf{X}_k \Rightarrow (2^{\mathsf{E}} - 1)\mathsf{I}^*$	$\mathsf{Z}_k \Rightarrow \mathsf{I}^*$
-	$2^{\mathsf{E}}\mathsf{Z}_k \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k$
$\mathsf{O}_k\mathsf{q}^{\mathsf{T}} + \mathsf{X}_k = \mathsf{N}_k + \mathsf{q}(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i'))$	$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_k + (2^{\mathsf{E}} - 1)\mathsf{I}^*$

Table 2: Instruction Encoding Summary

I_i	Increment $(\mathsf{S}(k, i))$
$\mathsf{I}_i \Rightarrow \mathsf{I}^*$	$\mathsf{S}(k, i) \Rightarrow (\mathsf{q} - 1)\mathsf{I}^*$
$\mathsf{I}^* = \sum_{i=0}^{n} \mathsf{I}_i$	$\mathsf{S}(k, i) \Rightarrow \mathsf{X}_k + \mathsf{I}^*$
$1 \Rightarrow \mathsf{I}_0, \mathsf{q}^{\mathsf{T}-1} = \mathsf{I}_n$	$\mathsf{X}_k + \mathsf{I}^* \Rightarrow \mathsf{S}(k, i) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_i)$
Decrement $(\mathsf{S}(k, i))$	R_k unchanged $(\mathsf{C}(k))$
$\mathsf{S}(k, i) \Rightarrow (\mathsf{q} - 1)\mathsf{I}^*$	$\mathsf{C}(k) \Rightarrow (\mathsf{q} - 1)(\mathsf{I}^* - \sum \mathsf{I}_{i'})$
$\mathsf{S}(k, i) \Rightarrow \mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k$	$\mathsf{C}(k) \Rightarrow \mathsf{X}_k$
$\mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k \Rightarrow \mathsf{S}(k, i) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_i)$	$\mathsf{X}_k \Rightarrow \mathsf{C}(k) + (\mathsf{q} - 1)(\sum_{I_{i'} \in S_k} \mathsf{I}_{i'})$

Table 3: Instruction Sequence Encoding Summary

Increment/Decrement/$G(k, i+1)$	$G(k, i')$
$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1}$	$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_i + \mathsf{I}_{i'}$
-	$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i'} + \mathsf{q}\mathsf{Z}_k$

Table 4: Misc. Encoding Summary

q	I^*
$\mathsf{E} = \mathsf{T} + \mathsf{n} + 1 + \sum_{i=0}^{M-1} \mathsf{N}_i$	-
$\mathsf{q} = 2 \times 2^{\mathsf{E}}$	$(\mathsf{q} - 1)\mathsf{I}^* = \mathsf{q}^{\mathsf{T}} - 1$

3.1 Example construction

Recall the example of a register machine RM given earlier which adds the numbers in the first two registers and ends with the result placed in register R_0.

I_0 : $G(1, 3)$

I_1 : $I(2)$

I_2 : $G(2, 6)$

I_3 : $D(1)$

I_4 : $I(0)$

I_5 : $G(1, 3)$

This example will use the procedure outlined in Section 3 to convert this register machine — which we will call RM — into a Diophantine Equation that has a solution if and only if RM halts, and whose O_k indeterminates correspond to the output of register R_k for all k.

First note that RM takes 2 inputs, but the highest register referred to in the instructions is R_2. Since R_2 is not an explicit input, we call it an implicit input and set it equal to 0. With that being said we define the parameters

$$\mathsf{N}_0 = initial\ contents\ of\ R_0$$

$$\mathsf{N}_1 = initial\ contents\ of\ R_1$$

$$\mathsf{N}_2 = 0$$

.

Now note the number of instructions (including the implicit halting instruction) is 7. We determine E and q as follows:

$$\mathsf{E} = \mathsf{T} + 8 + \mathsf{N}_0 + \mathsf{N}_1 + \mathsf{N}_2$$

$$\mathsf{q} = 2 \times 2^{\mathsf{E}}.$$

We then define I^* as follows. Note that this portion of the construction is the same regardless of RM

$$(\mathsf{q} - 1)\mathsf{I}^* = \mathsf{q}^{\mathsf{T}} - 1.$$

For each instruction I_i we then define:

$$\mathsf{I}_0 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_1 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_2 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_3 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_4 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_5 \Rightarrow \mathsf{I}^*$$

$$\mathsf{I}_6 \Rightarrow \mathsf{I}^*$$

Note that here I_6 refers to the implicit halting instruction I_6. Now we make sure execution starts at I_0.

$$1 \Rightarrow \mathsf{I}_0$$

Similarly, we make sure the final instruction executed is the implicit halting instruction I_6:

$$\mathsf{q}^{\mathsf{T}-1} = \mathsf{I}_6.$$

We now ensure that one and only one instruction is being executed at any given step of the computation:

$$\mathsf{I}^* = \mathsf{I}_0 + \mathsf{I}_1 + \mathsf{I}_2 + \mathsf{I}_3 + \mathsf{I}_4 + \mathsf{I}_5 + \mathsf{I}_6.$$

We then make sure that each indeterminate X_k consists of precisely T base-q digits, with each digit being less than half of q.

$$\mathsf{X}_0 \Rightarrow (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

$$\mathsf{X}_1 \Rightarrow (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

$$\mathsf{X}_2 \Rightarrow (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

Now we set up the conditions for encoding Z_k, i.e. whether or not register R_k contains 0 over time.

$$\mathsf{Z}_0 \Rightarrow \mathsf{I}^*$$

$$2^{\mathsf{E}}\mathsf{Z}_0 \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_0$$

$$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_0 \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_0 + (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

$$\mathsf{Z}_1 \Rightarrow \mathsf{I}^*$$

$$2^{\mathsf{E}}\mathsf{Z}_1 \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_1$$

$$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_1 \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_1 + (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

$$\mathsf{Z}_2 \Rightarrow \mathsf{I}^*$$

$$2^{\mathsf{E}}\mathsf{Z}_2 \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_2$$

$$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_2 \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_2 + (2^{\mathsf{E}} - 1)\mathsf{I}^*$$

We will now set up conditions for indeterminates corresponding to when register R_k changes and does not change. Consider register R_0. The only instruction that changes the contents of this register is instruction I_4 which is $I(0)$, so $S_0 = \{I_4\}$. We define the indeterminate $\mathsf{S}(0,4)$ using the conditions

$$\mathsf{S}(0,4) \Rightarrow (\mathsf{q} - 1)\mathsf{I}_4$$

$$\mathsf{S}(0,4) \Rightarrow \mathsf{X}_0 + \mathsf{I}^*$$

$$\mathsf{X}_0 + \mathsf{I}^* \Rightarrow \mathsf{S}(0,4) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_4)$$

and the indeterminate $\mathsf{C}(0)$ using the conditions

$$\mathsf{C}(0) \Rightarrow (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_4)$$

$$\mathsf{C}(0) \Rightarrow \mathsf{X}_0$$

$$\mathsf{X}_0 \Rightarrow \mathsf{C}(0) + (\mathsf{q} - 1)\mathsf{I}_4.$$

Similarly, for register R_1, the only instruction that changes it is I_3 which is $D(1)$, so $S_1 = \{I_3\}$ and we define the indeterminate $\mathsf{S}(1,3)$ by using the conditions

$$\mathsf{S}(1,3) \Rightarrow (\mathsf{q} - 1)\mathsf{I}_3$$

$$\mathsf{S}(1,3) \Rightarrow \mathsf{X}_1 - \mathsf{I}^* + \mathsf{Z}_1$$

$$\mathsf{X}_1 - \mathsf{I}^* + \mathsf{Z}_1 \Rightarrow \mathsf{S}(1,3) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_3)$$

and the indeterminate $\mathsf{C}(1)$ using the conditions

$$\mathsf{C}(1) \Rightarrow (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_3)$$

$$\mathsf{C}(1) \Rightarrow \mathsf{X}_1$$

$$\mathsf{X}_1 \Rightarrow \mathsf{C}(1) + (\mathsf{q} - 1)\mathsf{I}_3.$$

Finally, register R_2 is only changed by instruction I_1 which is $I(2)$, so $S_2 = \{I_1\}$. Define $\mathsf{S}(2,1)$ by using

$$\mathsf{S}(2,1) \Rightarrow (\mathsf{q}-1)\mathsf{I}_1$$

$$\mathsf{S}(2,1) \Rightarrow \mathsf{X}_2 + \mathsf{I}^*$$

$$\mathsf{X}_2 + \mathsf{I}^* \Rightarrow \mathsf{S}(2,1) + (\mathsf{q}-1)(\mathsf{I}^* - \mathsf{I}_1)$$

and $\mathsf{C}(2)$ by using

$$\mathsf{C}(2) \Rightarrow (\mathsf{q}-1)(\mathsf{I}^* - \mathsf{I}_1)$$

$$\mathsf{C}(2) \Rightarrow \mathsf{X}_2$$

$$\mathsf{X}_2 \Rightarrow \mathsf{C}(2) + (\mathsf{q}-1)\mathsf{I}_1.$$

No other instructions modify the contents of any register.

We now encode the order of instructions. Instructions I_1, I_3, and I_4 are always followed by instructions I_2, I_4, and I_5, respectively, so we use the following conditions:

$$\mathsf{q}\mathsf{I}_1 \Rightarrow \mathsf{I}_2$$

$$\mathsf{q}\mathsf{I}_3 \Rightarrow \mathsf{I}_4$$

$$\mathsf{q}\mathsf{I}_4 \Rightarrow \mathsf{I}_5.$$

Instruction I_0 is followed by instruction I_3 if register R_1 does not contain 0, and is followed by I_1 otherwise, so we use the following conditions:

$$\mathsf{q}\mathsf{I}_0 \Rightarrow \mathsf{I}_1 + \mathsf{I}_3$$

$$\mathsf{q}\mathsf{I}_0 \Rightarrow \mathsf{I}_3 + \mathsf{q}\mathsf{Z}_1.$$

Similarly, instruction I_2 is followed by the implicit halting instruction I_6 if register R_2 does not contain 0, and is followed by I_3 otherwise, so we use the following conditions:

$$\mathsf{q}\mathsf{I}_2 \Rightarrow \mathsf{I}_3 + \mathsf{I}_6$$

$$\mathsf{q}\mathsf{I}_2 \Rightarrow \mathsf{I}_6 + \mathsf{q}\mathsf{Z}_2.$$

Finally, instruction I_5 is followed by instruction I_3 if register R_1 does not contain 0, and is followed by the implicit halting instruction I_6 otherwise, so we use the following conditions:

$$\mathsf{q}\mathsf{I}_5 \Rightarrow \mathsf{I}_6 + \mathsf{I}_3$$

$$\mathsf{q}\mathsf{I}_5 \Rightarrow \mathsf{I}_3 + \mathsf{q}\mathsf{Z}_1.$$

We finally determine the indeterminates O_0, O_1, and O_2 corresponding to the final contents of registers R_0, R_1, and R_2, respectively, by using the following equations:

$$\mathsf{O}_0\mathsf{q}^\mathsf{T} + \mathsf{X}_0 = \mathsf{N}_0 + \mathsf{q}(\mathsf{C}(0) + \mathsf{S}(0,4))$$

$$\mathsf{O}_1\mathsf{q}^\mathsf{T} + \mathsf{X}_1 = \mathsf{N}_1 + \mathsf{q}(\mathsf{C}(1) + \mathsf{S}(1,3))$$

$$\mathsf{O}_2\mathsf{q}^\mathsf{T} + \mathsf{X}_2 = \mathsf{N}_2 + \mathsf{q}(\mathsf{C}(2) + \mathsf{S}(2,1)).$$

Now all that is left to do is expand each condition into a system of Diophantine Equations and to combine all the equations into one. In the interest of saving space, only one condition will be expanded here. Consider the following condition:

$$\mathsf{X}_1 - \mathsf{I}^* + \mathsf{Z}_1 \Rightarrow \mathsf{S}(1,3) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_3).$$

This condition expands into the following 7 equations:

$$l = \mathsf{X}_1 - \mathsf{I}^* + \mathsf{Z}_1$$

$$r = \mathsf{S}(1,3) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_3)$$

$$t = 2^r$$

$$(1+t)^r = at^{l+1} + bt^l + c$$

$$c + d + 1 = t^l$$

$$b + e + 1 = t$$

$$b = 2f + 1.$$

3.2 Corollary: Hilbert's 10th problem is unsolvable

Hilbert's 10^{th} problem asks for an algorithm that can decide, in a finite amount of time, whether or not an arbitrary Diophantine Equation has solutions. In contrast, the halting problem asks for an algorithm that can decide, in a finite number of steps, whether or not an arbitrary Turing Machine will halt on a given input. Since Register Machines and Turing Machines are equivalent [7], the halting problem can be equivalently posed as a question about arbitrary register machines. These two seemingly unrelated problems are more closely related than one might expect. In fact, the halting problem is solvable if and only if Hilbert's 10^{th} problem is solvable.

If the halting problem were solvable, then the algorithm in question could be used to decide whether an arbitrary Diophantine Equation had solutions. One way to do this would be to write a program that searches, in some arbitrary order, for solutions to the Diophantine Equation and to halt if it finds one. If the ordering is such that every n-tuple is guaranteed to be checked after a finite number of steps, then the program halts if and only if at least one solution exists. Simply asking the algorithm that solves the halting problem whether or not this program halts would tell you whether or not the Diophantine Equation had solutions.

Conversely, if Hilbert's 10^{th} problem were solvable, then the algorithm could be used to decide if an arbitrary Register Machine halts. Simply construct a Diophantine Equation from the Register Machine you are concerned with as outlined in Section 3, and ask the algorithm whether it has any solutions. Since the equation has solutions if and only if the Register Machine it was made from halts, the algorithm could be used to solve the halting problem.

In 1936 [12], Turing showed that the halting problem was unsolvable. Informally, suppose we have an algorithm that can decide in a finite amount of time whether an arbitrary algorithm will halt given a particular input. Call this algorithm $H(k, n)$ where k is the number of the algorithm in ques-

tion and n is the input to this algorithm. $H(k,n)$ returns "true" if the algorithm with number k halts when given input n, and returns "false" otherwise. Now create a new algorithm $H^*(k)$ which simply runs $H(k,k)$, and if it returns "true" then $H^*(k)$ loops forever. If $H(k,k)$ returns "false", then $H^*(k)$ halts. Let P be the number of the algorithm $H^*(k)$. What happens when we run $H^*(P)$? If $H(P,P)$ returns "true" then $H^*(P)$ loops forever, meaning it should have returned "false". If $H(P,P)$ returns "false" then $H^*(P)$ will halt, meaning it should have returned "true". Since $H(k,n)$ determines whether algorithm k halts on input n, we have that $H^*(P)$ halts iff it does not halt, which is a contradiction. Hence $H(k,n)$ cannot exist.'

Since the halting problem is unsolvable, we must conclude that Hilbert's 10^{th} problem is also unsolvable.

4 Proofs for the validity of the construction

In this section I will prove the validity of the aforementioned procedure for turning an arbitrary Register Machine into a Diophantine Equation. Starting from Lemma 3 onwards, results and equations from previous lemmas are assumed. In addition, any recurrence of indeterminates from previous lemmas are taken to be the same unless otherwise specified.

Theorem 1. *Any Diophantine Equation constructed following the procedure outlined in Section 3 has a solution if and only if the Register Machine it was derived from halts. Furthermore, the solution of* O_k *is precisely the output of register* R_k *for each* k*.*

Before proving this theorem at the end of this section (see Proof of Theorem 1), several lemmas need to be taken care of first.

4.1 Lemmas

Lemma 1 (Combining Diophantine Equations)**.** *Let* L_i *and* R_i *be Diophantine Expressions,* $0 \leq i \leq m$*. Then a tuple* $(x_0, x_1, ..., x_n)$ *is a solution to*

the equation

$$L_0^2 + R_0^2 + L_1^2 + R_1^2 + ... + L_m^2 + R_m^2 = 2L_0R_0 + 2L_1R_1 + ... + 2L_mR_m$$

if and only if it is a solution to the system of equations

$$L_0 = R_0$$

$$L_1 = R_1$$

$$...$$

$$L_m = R_m$$

Proof. Suppose $(x_0, x_1, ..., x_n)$ is a solution to the system of equations above. Then we have

$$L_i = R_i,\ 0 \leq i \leq m$$

$$\iff L_i - R_i = 0,\ 0 \leq i \leq m$$

$$\iff (L_i - R_i)^2 = 0,\ 0 \leq i \leq m$$

$$\iff \sum_{i=0}^{m} (L_i - R_i)^2 = 0$$

$$\iff L_0^2 + R_0^2 + L_1^2 + R_1^2 + ... + L_m^2 + R_m^2 - 2L_0R_0 - 2L_1R_1 + ... - 2L_mR_m = 0$$

$$\iff L_0^2 + R_0^2 + L_1^2 + R_1^2 + ... + L_m^2 + R_m^2 = 2L_0R_0 + 2L_1R_1 + ... + 2L_mR_m$$

□

Lemma 2. *The binomial coefficient* $\binom{R}{L}$ *is odd if and only if the following system has a solution*

$$l = L$$

$$r = R$$

$$t = 2^r$$

$$(1 + t)^r = at^{l+1} + bt^l + c$$

$$c + d + 1 = t^l$$

$$b + e + 1 = t$$

$$b = 2f + 1$$

Proof. By the Binomial Theorem $(1 + x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i$. If we let $x = 1$ we have $(1 + 1)^n = \sum_{i=0}^{n} \binom{n}{i} 1^i$, and so $\sum_{i=0}^{n} \binom{n}{i} = 2^n$, meaning $\binom{n}{i} < 2^n$ for all i. In particular, we have $\binom{r}{l} < t$ in the above Diophantine system. Since $c < t^l$ and $b < t$, we must have that b is precisely the t^l coefficient in the base-t expansion of $(1 + t)^r$. That is, $b = \binom{r}{l}$. The final equation asserts that b is odd. Therefore, the above system of equations has a solution if and only if the binomial coefficient $\binom{R}{L}$ is odd. □

Note that due to Lucas' theorem with $p = 2$, we conclude that $\binom{R}{L}$ is odd if and only if each bit in the binary expansion of L is less than or equal to the corresponding bit in the binary expansion of R. In other words, if and only if $L \Rightarrow R$. Each instance of conditions of the form $L \Rightarrow R$ can be converted into a similar Diophantine system as long as we make sure there are no naming conflicts. That is, that each indeterminate is only present in one of these systems (other than the indeterminates that appear in the L or R expressions).

Lemma 3. *The equations*

$$\mathsf{E} = \mathsf{T} + \mathsf{n} + 1 + \sum_{k=0}^{M-1} \mathsf{N}_k$$

$$\mathsf{q} = 2 \times 2^{\mathsf{E}}$$

assert that $\mathsf{q} > \mathsf{T}$, $\mathsf{q} > \mathsf{N}_k$ *for all* k, *and* $\mathsf{q} > (\mathsf{n} + 1)$.

Proof. $2^k > k \ \forall k \in \mathbb{N}$ □

Lemma 4. *Let* $\mathsf{q} = 2^k$ *for some* k, *and let* $L = L[0] + L[1]\mathsf{q} + L[2]\mathsf{q}^2 + ... + L[n]\mathsf{q}^n$ *and* $R = R[0] + R[1]\mathsf{q} + R[2]\mathsf{q}^2 + ... + R[n]\mathsf{q}^n$ *be the base-*q *expansions of* L *and* R, *respectively. Then* $L \Rightarrow R$ *if and only if* $L[j] \Rightarrow R[j]$ *for all* j.

Proof. Let $l_{i,j}$ and $r_{i,j}$ be the 2^i digit of $L[j]$ and $R[j]$, respectively.

We have that

$$\begin{aligned} L &= \sum_{j=0}^{n} L[j]\mathsf{q}^j \\ &= \sum_{j=0}^{n} \left(\left(\sum_{i=0}^{k-1} l_{i,j} 2^i \right) \mathsf{q}^j \right) \\ &= \sum_{j=0}^{n} \left(\left(\sum_{i=0}^{k-1} l_{i,j} 2^i \right) 2^{jk} \right) \\ &= \sum_{j=0}^{n} \left(\sum_{i=0}^{k-1} l_{i,j} 2^{jk+i} \right). \end{aligned}$$

The same is true for R. Since the binary digits of L and R are the same as the binary expansions of their base-q digits, the statements in the lemma are equivalent. □

Lemma 5. *The equation*

$$(\mathsf{q}-1)\mathsf{I}^* = \mathsf{q}^{\mathsf{T}} - 1$$

determines I^* *to be* $\sum_{j=0}^{\mathsf{T}-1} \mathsf{q}^j$*, i.e.* I^*=$1111...11_q$.

Proof. Consider the base-q expansion of the above equation:

$$(\mathsf{q}-1)\mathsf{I}^* = (\mathsf{q}-1)\mathsf{q}^{\mathsf{T}-1} + (\mathsf{q}-1)\mathsf{q}^{\mathsf{T}-2} + ... + (\mathsf{q}-1)\mathsf{q} + (\mathsf{q}-1)$$

$$(\mathsf{q}-1)\mathsf{I}^* = (\mathsf{q}-1)(\mathsf{q}^{\mathsf{T}-1} + \mathsf{q}^{\mathsf{T}-2} + ... + \mathsf{q} + 1)$$

$$\mathsf{I}^* = \mathsf{q}^{\mathsf{T}-1} + \mathsf{q}^{\mathsf{T}-2} + ... + \mathsf{q} + 1.$$

Hence

$$\mathsf{I}^* = \sum_{j=0}^{\mathsf{T}-1} \mathsf{q}^j.$$

□

Lemma 6. *Conditions of the form*

$$\mathsf{l}_i \Rightarrow \mathsf{l}^*$$

assert that each base-q *digit of* l_i *is either 0 or 1.*

Proof. Since the condition must hold for each base-q digit, if any digit were greater than 1 then l_i would have a 1 bit somewhere to the left of the 2^0 bit. Since all base-q digits of l^* are 1's, the condition fails to hold, so the condition $\mathsf{l}_i \Rightarrow \mathsf{l}^*$ fails to hold as well. Therefore each base-q digit of l_i must be either 0 or 1. □

Lemma 7. *The equation*

$$\mathsf{l}^* = \sum_{i=0}^{n} \mathsf{l}_i$$

asserts that for each j*, one, and only one, of* $\{\mathsf{l}_0, \mathsf{l}_1, ..., \mathsf{l}_n\}$ *has a 1 in the* q^j *place of their base-*q *expansion, and the rest have a 0.*

Proof. The left hand side of the equation has a 1 for every base-q digit, so the right hand side must have all 1's as well. Since each digit of l_i is either a 1 or a 0 and since $\mathsf{q} > (n+1)$, the only way for this to happen is if one and only one l_i has a 1 in each digit place. □

Corollary 1. *The condition*

$$1 \Rightarrow \mathsf{l}_0$$

asserts that $\mathsf{l}_0[0]$ *is a 1. Additionally, the equation*

$$\mathsf{q}^{\mathsf{T}-1} = \mathsf{l}_n$$

asserts that $\mathsf{l}_n[\mathsf{T}-1]$ *is a 1. Since this is a strict equality, we can also conclude that* $\mathsf{l}_n[j] = 0$ *whenever* $j \neq (\mathsf{T}-1)$*.*

Observation 1. *The condition*

$$\mathsf{X}_k \Rightarrow (2^{\mathsf{E}} - 1)\mathsf{l}^*$$

asserts that each base-q *digit of* X_k *is less than* 2^{E}

Lemma 8. *Suppose the following three conditions hold:*

$$\mathsf{Z}_k \Rightarrow \mathsf{I}^*$$

$$2^{\mathsf{E}}\mathsf{Z}_k \Rightarrow 2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k$$

$$2^{\mathsf{E}}\mathsf{I}^* - \mathsf{X}_k \Rightarrow 2^{\mathsf{E}}\mathsf{Z}_k + (2^{\mathsf{E}} - 1)\mathsf{I}^*.$$

Then we have that $\mathsf{Z}_k[j] = 1$ *if* $\mathsf{X}_k[j] = 0$*, and otherwise* $\mathsf{Z}_k[j] = 0$*.*

Proof. First, note that the first condition forces each digit of Z_k to be 1 or 0 (by Lemma 6). Suppose $\mathsf{X}_k[j] = 0$. Then the q^j digit of the left hand side of the third condition has a 1-bit in the 2^{E} place, so there must be a 1 in the 2^{E} place of the right hand side as well. The only way for this condition to be fulfilled is if $\mathsf{Z}_k[j] = 1$. Now suppose $\mathsf{X}_k[j] > 0$. Recall that $2^{\mathsf{E}} = \frac{1}{2}\mathsf{q}$. By Observation 1, each base-q digit $\mathsf{X}_k[j]$ is less than 2^{E}, and so $(2^{\mathsf{E}}\mathsf{I}^*)[j] - \mathsf{X}_k[j] > 0$ for all j. Additionally, the q^j digit of the right hand side of the second condition is less than 2^{E} meaning its 2^{E} bit is a 0. Therefore, the 2^{E} bit of the q^j digit of the left hand side must be a 0 as well, forcing $\mathsf{Z}_k[j]$ to be 0. □

Observation 2. *Given a condition of the form*

$$L \Rightarrow (\mathsf{q} - 1)R$$

such that $R \Rightarrow \mathsf{I}^*$*, it follows that if* $R[j] = 0$ *then* $L[j] = 0$*, and if* $L[j] > 0$ *then* $R[j] = 1$*.*

Lemma 9. *Consider the following conditions*

$$\mathsf{S}(k, i) \Rightarrow (\mathsf{q} - 1)\mathsf{I}_i$$

$$\mathsf{S}(k, i) \Rightarrow \mathsf{X}_k + \mathsf{I}^*$$

$$\mathsf{X}_k + \mathsf{I}^* \Rightarrow \mathsf{S}(k, i) + (\mathsf{q} - 1)(\mathsf{I}^* - \mathsf{I}_i).$$

They imply that if $\mathsf{I}_i[j] = 0$ *then* $\mathsf{S}(k, i)[j] = 0$*. Furthermore, if* $\mathsf{I}_i[j] = 1$ *then* $\mathsf{S}(k, i)[j] = \mathsf{X}_k[j] + 1$*.*

Proof. For the first claim, simply note that if $\mathsf{I}_i[j] = 0$ then the q^j digit of the left hand side must be 0 as well, by Observation 2. For the second claim, suppose $\mathsf{I}_i[j] = 1$. This will make the q^j digit of the right hand side of the third condition equal to $\mathsf{S}(k,i)[j]$. Conditions 2 and 3 taken together now state $L[j] \Rightarrow R[j]$ and $R[j] \Rightarrow L[j]$ so we must have $L[j] = R[j]$. In other words, $\mathsf{S}(k,i)[j] = \mathsf{X}_k[j] + 1$ (recall $\mathsf{I}^*[j] = 1$ for each j). □

Lemma 10. *Suppose the following conditions hold:*

$$\mathsf{S}(k,i) \Rightarrow (\mathsf{q}-1)\mathsf{I}_i$$

$$\mathsf{S}(k,i) \Rightarrow \mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k$$

$$\mathsf{X}_k - \mathsf{I}^* + \mathsf{Z}_k \Rightarrow \mathsf{S}(k,i) + (\mathsf{q}-1)(\mathsf{I}^* - \mathsf{I}_i).$$

Then we have that if $\mathsf{I}_i[j] = 0$ *then* $\mathsf{S}(k,i)[j] = 0$. *Furthermore, if* $\mathsf{I}_i[j] = 1$, *then* $\mathsf{S}(k,i)[j] = \mathsf{X}_k[j] - 1$, *unless* $\mathsf{X}_k[j] = 0$, *in which case* $\mathsf{S}(k,i)[j] = 0$.

Proof. For a proof of the first claim see Lemma 9. For the second claim suppose $\mathsf{I}_i[j] = 1$ and $\mathsf{X}_k[j] = 0$. We know from Lemma 8 that $\mathsf{Z}_k[j] = 1$. Using the same argument in Lemma 9 on conditions 2 and 3 we have that $\mathsf{S}(k,i)[j] = 0 - 1 + 1 = 0$. Now suppose $\mathsf{X}_k[j] > 0$. Then we know $\mathsf{Z}_k[j] = 0$, so $\mathsf{S}(k,i)[j] = \mathsf{X}_k[j] - 1$, as required. □

Lemma 11. *Consider the following conditions:*

$$\mathsf{C}(k) \Rightarrow (\mathsf{q}-1)(\mathsf{I}^* - \sum_{I_{i'} \in S_k} \mathsf{I}_{i'})$$

$$\mathsf{C}(k) \Rightarrow \mathsf{X}_k$$

$$\mathsf{X}_k \Rightarrow \mathsf{C}(k) + (\mathsf{q}-1)(\sum_{I_{i'} \in S_k} \mathsf{I}_{i'})$$

where $S_k \subseteq \{I_0, I_1, ..., I_{n-1}\}$. *They imply have that* $\mathsf{C}(k)[j] = 0$ *if* $\mathsf{I}_{i'}[j] = 1$ *for any* i'. *Furthermore, if no* $\mathsf{I}_{i'}[j] = 1$, *then* $\mathsf{C}(k)[j] = \mathsf{X}_k[j]$.

Proof. As proven in Lemma 7, one and only one of $\mathsf{I}_i[j] = 1$ for a particular j. If that I_i is one of the $\mathsf{I}_{i'}$ included in the sum, then the q^j digit of the right hand side of the first condition is 0, forcing the q^j digit of the left hand side to be 0 as well. Since the left hand side is precisely $\mathsf{C}(k)$ we must have $\mathsf{C}(k)[j] = 0$. Now suppose that none of $\mathsf{I}_{i'}[j] = 0$. Then the q^j digit of the right hand side of the third equation is simply $\mathsf{C}(k)[j]$. Once again we have $\mathsf{X}_k[j] \Rightarrow \mathsf{C}(k)[j]$ and $\mathsf{C}(k)[j] \Rightarrow \mathsf{X}_k[j]$, so we must conclude $\mathsf{C}(k)[j] = \mathsf{X}_k[j]$. □

Lemma 12. *Conditions of the form*

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i'}$$

assert that if $\mathsf{I}_i[j] = 1$*, then* $\mathsf{I}_{i'}[j+1] = 1$.

Proof. Suppose $\mathsf{I}_i[j] = 1$. Then the q^{j+1} digit on the left hand side of the condition is a 1, forcing the q^{j+1} digit of the right hand side to be a 1 as well. □

Observation 3. *The condition*

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1} + \mathsf{I}_{i'}$$

asserts that one (and only one) of $\mathsf{I}_{i+1}[j+1]$ *and* $\mathsf{I}_{i'}[j+1]$ *will be 1 whenever* $\mathsf{I}_i[j] = 1$*. Similarly, the condition*

$$\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i'} + \mathsf{q}\mathsf{Z}_k$$

asserts that if $\mathsf{I}_i[j] = 1$ *and* $\mathsf{X}_k[j] = 0$ *(meaning* $\mathsf{Z}_k[j] = 1$*), then* $\mathsf{I}_{i'}[j+1]$ *is a 0.*

Lemma 13. *Given the equation*

$$\mathsf{q}^\mathsf{T}\mathsf{O}_k + \mathsf{X}_k = \mathsf{N}_k + \mathsf{q}(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i')),$$

the following hold:

1. $\mathsf{X}_k[0] = \mathsf{N}_k$.

2. $\mathsf{O}_k = \left(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i')\right)[\mathsf{T} - 1]$.

3. *For* $j < (\mathsf{T} - 1)$, $\mathsf{X}_k[j+1] = \left(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i')\right)[j]$.

Proof.

1. Since $\mathsf{N}_k < \mathsf{q}$ (as in Lemma 3), the only q^0 digit on the right hand side is N_k itself, so it must be equal to $\mathsf{X}_k[0]$.

2. Note that $\mathsf{X}_k[\mathsf{T}] = 0$, so the only q^{T} digit on the left hand side is O_k itself. Similarly, the q^{T} digit on the right hand side is

$$\left(\mathsf{D}(\mathsf{k}) + \sum_{\mathsf{I}_{\mathsf{i}'} \in \mathsf{S}_\mathsf{k}} \mathsf{S}(\mathsf{k}, \mathsf{i}')\right)[\mathsf{T} - 1]$$

(since it is multiplied by q), so they must be equal.

3. First, note that at most one of $\mathsf{C}(k)[j]$ or $\mathsf{S}(k, i')[j]$ for some i' can be non-zero. This is because at most one $\mathsf{I}_i[j]$ can be 1 and if the $\mathsf{I}_i[j] = 0$ then $\mathsf{S}(k, i)[j] = 0$ (as in Lemma 9 and Lemma 10). For $\mathsf{C}(k)$, we have that $\mathsf{C}(k)[j] = 0$ unless every one of $\mathsf{I}_{i'}[j] = 0$ such that $I_{i'} \in S_k$ (as in Lemma 11). Since all but at most one of $\mathsf{C}(k)[j]$ or $\mathsf{S}(\mathsf{k}, \mathsf{i}')[j]$ for some i' is 0, their sum must be equal to the q^{j+1} digit of the right hand side. Therefore, for $j < (\mathsf{T} - 1)$, this is equal to $\mathsf{X}_k[j+1]$. □

Lemma 14. *For any Diophantine Equation generated from a Register Machine RM using the procedure outlined in Section 3, the following statements hold:*

1. *For any i and j, $\mathsf{I}_i[j] = 1$ if the next step to be executed by RM after the j^{th} step is instruction I_i and 0 otherwise.*

2. *For any j and k, $X_k[j]$ is equal to the content of register R_k following the j^{th} step executed by RM.*

Proof. For statement 1, first, consider a special case where only instructions I_i of the form $I(k), D(k)$, and $G(k, i+1)$ are considered. For $j = 0$ RM is about to execute instruction I_0 and by $1 \Rightarrow \mathsf{I}_0$ we are assured that $\mathsf{I}_0[0]$ is indeed 1. Additionally, $\mathsf{I}^* = \sum_{i=0}^{n} \mathsf{I}_i$ and Lemma 7 assure us that all other $\mathsf{I}_{i'}$, $i' \neq 0$ have a 0 in the q^0 digit, as required.

Suppose the statement holds until j. Then RM is about to execute instruction I_i and $\mathsf{I}_i[j] = 1$. Since I_i is of the form $I(k), D(k)$, or $G(k, i+1)$ for some k, the next instruction to be executed by RM is instruction I_{i+1}. $\mathsf{q}\mathsf{I}_i \Rightarrow \mathsf{I}_{i+1}$ and Lemma 12 assure us that the q^{j+1} digit of the right hand side is 1 whenever the q^j digit of the left hand side is 1. Thus $\mathsf{I}_{i+1}[j+1]$ is 1 and RM is about to execute instruction I_{i+1} following the $(j+1)^{th}$ step, as required.

Now consider a special case of statement 2. For an arbitrary register R_k and variable X_k and for instructions I_i of the form $I(k), D(k), G(k, i+1), I(k'), D(k'), G(k', i+1)$ for $k' \neq k$. For $j = 0$, RM has executed 0 steps and therefore the contents of each register are precisely the initial contents i.e. the inputs. Indeed, the equation

$$\mathsf{q}^{\mathsf{T}}\mathsf{O}_k + \mathsf{X}_k = \mathsf{N}_k + \mathsf{q}(\mathsf{C}(k) + \sum_{I_{i'} \in S_k} \mathsf{S}(k, i'))$$

and Lemma 13 assure us that $\mathsf{X}_k[0]$ is equal to N_k, the initial contents of register R_k, as required.

Suppose the statement holds until j. We know that $\mathsf{I}_i[j] = 1$ if and only if RM is about to execute instruction I_i following the j^{th} step. Suppose I_i is $I(k)$, then the content of register R_k following the $(j+1)^{th}$ step will be 1 more than the content of R_k following the j^{th} step. Since $\mathsf{I}_i[j] = 1$, the conditions from Lemma 9 assure us that $\mathsf{S}(k, i)[j] = \mathsf{X}_k[j] + 1$ (recall that $\mathsf{I}^*[j] = 1 \ \forall j$). Since all but one of $\mathsf{C}(k)[j], \mathsf{S}(k, 0)[j], ...$ must be zero, the equation from Lemma 13 forces $\mathsf{X}_k[j+1] = \mathsf{S}(k, i)[j]$. In other words, equal to 1 more than the content of register R_k following the j^{th} step executed by RM, which is precisely the content of R_k following the $(j+1)^{th}$ step

executed by RM, as required.

Similarly, suppose I_i is $D(k)$. Then the systems in Lemma 10 assure us that $\mathsf{S}(k,i)[j]$ is $\mathsf{X}_k[j] - 1 + \mathsf{Z}_k[j]$. Suppose the content of register R_k is 0 following the j^{th} step. Then the content should still be 0 following the $(j+1)^{th}$ step. Since the statement holds up to j, $\mathsf{X}_k[j] = 0$. The conditions from Lemma 8 then assure us that $\mathsf{Z}_k[j] = 1$, so $\mathsf{S}(k,i)[j] = 0 - 1 + 1 = 0$. Again, this fact taken with the equation in Lemma 13 asserts that $\mathsf{X}_k[j+1] = 0$, which is precisely the content of register R_k following the $(j+1)^{th}$ step, as required. Similarly, suppose the content of register R_k is greater than 0. Then the content following the $(j+1)^{th}$ step should be 1 less than it was following the j^{th} step. We have that $\mathsf{Z}_k[j] = 0$ so the conditions from Lemma 10 assure us that $\mathsf{S}(k,i)[j]$ is 1 less than $\mathsf{X}_k[j]$. Again, the equation in Lemma 13 forces $\mathsf{X}_k[j+1]$ to be 1 less than the $\mathsf{X}_k[j]$, which is precisely the content of register R_k following the $(j+1)^{th}$ step, as required.

Finally, suppose I_i is one of $I(k'), D(k'), G(k', i+1)$, or $G(k, i+1)$ where $k' \neq k$. Then the content of register R_k should be the same following the j^{th} step and the $(j+1)^{th}$ step. Since I_i is not $I(k)$ or $D(k)$, we know that $I_i \notin S_k$, so $\mathsf{S}(k,i)$ is not an indeterminate in the Diophantine Equation. We also have that $\mathsf{S}(k,i)[j] = 0$ for all $\mathsf{S}(k,i)$ that are defined. Therefore, the conditions in Lemma 11 assure us that $\mathsf{C}(k)[j] = \mathsf{X}_k[j]$. Once again, the equation in Lemma 13 asserts that $\mathsf{X}_k[j+1]$ is equal to $\mathsf{X}_k[j]$, which is precisely the content of register R_k following the $(j+1)^{th}$ step.

Returning to statement 1. If only instructions of the form $I(k), D(k)$, or $G(k, i+1)$ are considered, then $I_i[j] = 1$ if and only if I_i is the next instruction to execute following the j^{th} step taken by RM, and we know that $\mathsf{X}_k[j]$ is equal to the content of register R_k following the j^{th} step for each k. Consider the lowest j' such that RM is about to execute an instruction I_i of the form $G(k, i')$, $i' \neq i$ following the j'^{th} step. We know that $\mathsf{I}_i[j'] = 1$. Suppose the content of register R_k is 0 then the next instruction to execute should be I_{i+1}. We know that $\mathsf{X}_k[j'] = 0$ thus $\mathsf{Z}_k[j'] = 1$. This taken with

the systems in Observation 3 assure us that $\mathsf{I}_{i'}[j'+1] = 0$, which in turn forces $\mathsf{I}_{i+1}[j'+1] = 1$. So the statement holds when register R_k contains 0. Now suppose it does not contain 0, then the next instruction should be instruction $I_{i'}$. Indeed, since $\mathsf{X}_k[j'] > 0$, we have $\mathsf{Z}_k[j'] = 0$. This forces $\mathsf{I}_{i'}[j'+1] = 1$, and instruction $I_{i'}$ is precisely the next instruction to execute following the $(j'+1)^{th}$ step taken by RM, as required.

Finally, the case for statement 2 when I_i is $G(k', i')$ or $G(k, i')$ is similar. In either case, $I_i \notin S_k$ and $\mathsf{S}(k, i)$ is not defined so the conditions in Lemma 11 assure us that $\mathsf{C}(k)[j] = \mathsf{X}_k[j]$. Once again, the equation in Lemma 13 forces $\mathsf{X}_k[j] = \mathsf{X}_k[j+1]$. Indeed, the content of register R_k is unchanged following the j^{th} step and following the $(j+1)^{th}$ step, as required. □

With all these lemmas out of the way, we can finally get to the main result.

4.2 Proof of Main Theorem

Proof of Theorem 1. From the equation $\mathsf{q}^{\mathsf{T}-1} = \mathsf{I}_n$ we can see that the last instruction executed is the implicit halting instruction, which does not change the contents of any registers. Since $\mathsf{X}_k[\mathsf{T}-1] = \mathsf{O}_k$, O_k is equal to the content of register R_k following the $(\mathsf{T}-1)^{th}$ step taken by RM. i.e. the output of register R_k.

Since a system constructed as outlined in Section 3 uniquely determines I_i to encode whether or not I_i is executed at each step of the computation, X_k to encode the content of register R_k over time, Z_k being uniquely determined by X_k, I^* being uniquely determined from q and T, T being uniquely determined by the number of steps taken by RM, $\mathsf{S}(k, i)$ being uniquely determined by I^*,I_i, X_k, and Z_k, and finally $\mathsf{C}(k)$ being uniquely determined by I^*,X_k and I_i for $I_i \in S_k$, it must be the case that this system has a unique solution corresponding to the output of RM. Furthermore, if RM never halts then T would not be able to be determined since the number of steps taken by RM would be infinite.

Taking the system constructed as specified in Section 3 and using the sum of squared differences technique yields a single (massive) equation which has a solution in natural numbers if and only if RM halts. □

5 Bibliography

References

[1] Gregory J. Chaitin. *Algorithmic Information Theory*. Cambridge Univ. Press, 2003, pp. 23–45.

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

[3] David Hilbert. "Mathematical problems". Trans. by Mary Winston Newson. In: *Bulletin of the American Mathematical Society* 8 (1902), pp. 437–479.

[4] David Hilbert. "Mathematische probleme". In: *Nachrichten von der Koniglichen Gesellschaft der Wissenschaften zu Gottingen* (1900).

[5] James P Jones. "Universal diophantine equation". In: *The journal of symbolic logic* 47.3 (1982), pp. 549–571.

[6] James P Jones and Yuri V Matijasevič. "Register machine proof of the theorem on exponential diophantine representation of enumerable sets". In: *The Journal of symbolic logic* 49.3 (1984), pp. 818–829.

[7] Ivan Korec. "Small universal register machines". In: *Theoretical Computer Science* 168.2 (1996), pp. 267–301.

[8] Alexandre Laugier and Manjil P Saikia. "A new proof of Lucas' Theorem". In: *arXiv preprint arXiv:1301.4250* (2013).

[9] Yu V Matijasevič. "Some purely mathematical results inspired by mathematical logic". In: *Logic, Foundations of Mathematics, and Computability Theory*. Springer, 1977, pp. 121–127.

[10] Yuri Vladimirovich Matiyasevich. "The Diophantineness of enumerable sets". In: *Doklady Akademii Nauk*. Vol. 191. 2. Russian Academy of Sciences. 1970, pp. 279–282.

[11] Thoralf Skolem. *Diophantische gleichungen*. Berlin, 1938.

[12] Alan Mathison Turing. “On computable numbers, with an application to the Entscheidungsproblem”. In: *J. of Math* 58.345-363 (1936), p. 5.

Appendix

In this section I will briefly outline the method for converting Exponential Diophantine Equations into Ordinary Diophantine Equations. For a proof see the paper by M. Davis [2].

Davis showed that for positive integers a, b, and c, $a = b^c$ if and only if the following system of Ordinary Diophantine Equations has at least one solution in positive integers:

$$x^2 - (d^2 - 1)y^2 = 1$$

$$u^2 - (d^2 - 1)v^2 = 1$$

$$s^2 - (e^2 - 1)t^2 = 1$$

$$v = ry^2$$

$$e = 1 + 4py = d + qu$$

$$s = x + fu$$

$$t = c + 4(g - 1)y$$

$$y = c + h - 1$$

$$(x - y(d - b) - a)^2 = (i - 1)^2(2db - b^2 - 1)^2$$

$$a + j = 2db - b^2 - 1$$

$$w = b + k = c + l$$

$$d^2 - (w^2 - 1)(w - 1)^2 z^2 = 1$$

Note some variable labels have been altered from the source cited. This system uses Diophantine Equations which take positive integer values. A system of equations can be constructed which has solutions in natural numbers if and only if the original system has solutions in positive integers by simply adding one to each variable. For example the positive integer variable a would be replaced by $(a' + 1)$ where a' takes on natural number values.

The above system can be used to convert any Exponential Diophantine Equation into an ordinary one by replacing any instance of exponentiation of indeterminates with a new indeterminate and equating it with the indeterminate a in the above system. For example, the Exponential Diophantine Equation

$$X_0 + X_0^{X_1} = X_2$$

could be replaced by the equations

$$X_0 + Y_0 = X_2$$

$$Y_0 = a$$

$$X_0 = b$$

$$X_1 = c$$

adjoined with the system of equations mentioned above. This results in a system of strictly Ordinary Diophantine Equations which has one or more solutions if and only if the original system of Exponential Diophantine Equations has one or more solutions.