Annals of Pure and Applied Logic 157 (2009) 225–233

Contents lists available at ScienceDirect

Annals of Pure and Applied Logic

journal homepage: www.elsevier.com/locate/apal

Existential arithmetization of Diophantine equations☆

Yuri Matiyasevich

Steklov Isnstitute of Mathematics at St. Petersburg, Russia

ARTICLE INFO

Article history:
Available online 17 October 2008

MSC:
primary 03D25
secondary 11D99
11U05

Keywords:
Arithmetization
Diophantine equation

ABSTRACT

A new method of coding Diophantine equations is introduced. This method allows (i) checking that a coded sequence of natural numbers is a solution of a coded equation without decoding; (ii) defining by a purely existential formula, the code of an equation equivalent to a system of indefinitely many copies of an equation represented by its code.

The new method leads to a much simpler construction of a universal Diophantine equation and to the existential arithmetization of Turing machines, register machines, and partial recursive functions.

1. Introduction

In his seminal paper [3] of 1931 Kurt Gödel introduced his powerful technique of arithmetization. Among other things this technique allows us to define effectively enumerable sets by arithmetical formulas. It can easily be checked that all universal quantifiers used by Gödel can be made bounded, and that any arithmetical formula with bounded universal quantifiers defines an effectively enumerable set (as the set of values of its free variables for which the formula is true). Thus arithmetical formulae with bounded universal quantifiers can be used to serve as definitions of effectively enumerable sets and hence they can provide a foundation for computability theory.

Martin Davis [1] conjectured that in fact universal quantifiers aren't required at all for arithmetical representations of effectively enumerable sets. A weaker form of his conjecture was established in the outstanding paper [2] of Martin Davis, Hilary Putnam, and Julia Robinson. Namely, they showed that every effectively enumerable set $\mathfrak{M}$ of n-tuples of natural numbers (i.e., non-negative integers) has an *exponential Diophantine representation*

$$\begin{aligned}&\langle a_1, \ldots, a_n\rangle \in \mathfrak{M} \iff\\ &\exists x_1 \ldots x_m E_{\mathrm{L}}(a_1, \ldots, a_n, x_1, x_2, \ldots, x_m) =\\ &\qquad E_{\mathrm{R}}(a_1, \ldots, a_n, x_1, x_2, \ldots, x_m)\end{aligned} \tag{1}$$

where E_{L} and E_{R} are *exponential polynomials*, i.e., expressions constructed by combining variables and particular positive integers using the usual operations of addition, multiplication and exponentiation (lower case italic letters in formulae in this paper always range over natural numbers).

A method due to Gödel [3] for representing arbitrarily long sequences of natural numbers by finite sets of natural numbers using the Chinese Remainder Theorem played an essential technical role in the construction of (1) as presented in [2]. Namely, if $P(y)$ is a polynomial with non-negative integer coefficients and the natural numbers $y_1, \ldots, y_k$ are coded by a pair of natural numbers $\langle a, b\rangle$ such that

$$y_1 \equiv a \pmod{1b+1}, \qquad \ldots, \qquad y_k \equiv a \pmod{kb+1} \tag{2}$$

☆ The author is grateful to a juror of Kurt Gödel Society Fellowships for suggestions and corrections.

E-mail address: yumat@pdmi.ras.ru.
URL: http://logic.pdmi.ras.ru/~yumat.

0168-0072/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.apal.2008.09.009

and

$$y_1 < b, \qquad \ldots, \qquad y_k < b \tag{3}$$

then the pair $\langle P(a), b\rangle$ codes in a similar manner the sequence $P(y_1), \ldots, P(y_k)$ provided that

$$P(y_1) < b, \qquad \ldots, \qquad P(y_k) < b. \tag{4}$$

The representation (1) was improved in [6] (see also [7]) to a similar *Diophantine representation* where E_{L} and E_{R} are genuine polynomials, thus completing the proof of Davis's conjecture. The transition from exponential to genuine Diophantine representations was achieved by constructing a Diophantine representation

$$a^b = c \iff \exists z_1 \ldots z_m \{A(a, b, c, z_1, \ldots, z_m) = 0\} \tag{5}$$

of exponentiation (a function is called *Diophantine* if its graph has a Diophantine representation) on the basis of Julia Robinson's work [11].

The entire proof of Davis's conjecture was constructive; in particular, it enabled one to construct, for each n, a *universal Diophantine equation*

$$U_n(p_1, \ldots, p_n, k, y_1, \ldots, y_M) = 0 \tag{6}$$

with the following property: *for every Diophantine equation*

$$D(p_1, \ldots, p_n, x_1, \ldots, x_m) = 0, \tag{7}$$

one can effectively find a particular number k_D, such that, for given values of the parameters $p_1, \ldots, p_n$, Eq. (7) has a solution in $x_1, \ldots, x_m$ if and only if the equation

$$U_n(p_1, \ldots, p_n, k_D, y_1, \ldots, y_M) = 0 \tag{8}$$

has a solution in $y_1, \ldots, y_M$.

The assertion of the existence of universal Diophantine equations is a purely number-theoretical result, so the following question naturally arises: *Do we really need the full machinery of the elimination of bounded universal quantifiers and the general notion of effectively enumerable set in order to construct a universal Diophantine equation?* In [7] I gave a purely number-theoretical construction of universal Diophantine equations. It was based on a new method, different from the one used by K. Gödel, for representing arbitrarily long sequences of natural numbers by finite sets of natural numbers.

Unfortunately, this method was not suitable for eliminating bounded universal quantifiers, that is for proving the existence of purely existential representations of arbitrary effectively enumerable sets. In this paper yet another method for coding Diophantine equations is introduced. This method permits the following:

- checking that a given coded sequence is indeed a solution of a given coded equation can be directly expressed by a purely existential formula, which in turn enables a new, much simpler purely number-theoretical construction of universal Diophantine equations;
- given a code of a Diophantine equation, defining by a purely existential formula the code of an equation equivalent to a system of indefinitely many copies of the given equation (with indefinitely many unknowns); this allows us to eliminate bounded universal quantifiers and existentially arithmetize Turing machines, register machines, and partial recursive functions.

The new method of coding is based mainly on basic linear algebra which makes it suitable for presentation in courses on computability theory (taking the number-theoretical result of the existence of representation (5) for granted).

2. Auxiliary functions and relations

Our ultimate goal is the construction of Diophantine representations

$$\langle a_1, \ldots, a_n\rangle \in \mathfrak{M} \iff \exists x_1 \ldots x_m \{P(a_1, \ldots, a_n, x_1, x_2, \ldots, x_m) = 0\} \tag{9}$$

for particular effectively enumerable sets. Actually, it is sufficient to construct a representation by more complex arithmetical formulae provided that we have tools to transform these formulae into equivalent formulae of the form (9). For example, a system of Diophantine equations can be easily combined into a single Diophantine equation, so we can freely use conjunction &.

The equivalence (5) justifies the use of exponentiation. More generally, as soon as we establish that a particular function is Diophantine, we can use it for constructing new equations. In a similar way in addition to equality we can use any other predicate provided that we have proved that the set of tuples satisfying this predicate has a Diophantine representation. In particular, we can use inequalities because

$$a \le b \iff \exists x\{a + x = b\}, \quad a < b \iff \exists x\{a + x + 1 = b\}. \tag{10}$$

In this section we present Diophantine representations for several functions and relations which will be used in the sequel.

2.1. Positional representation

Let $\mathrm{Digit}(a, b, n)$ denote the n-th digit in the base-b positional representation of the number a, that is,

$$a = \sum_{n=0}^{\infty} \mathrm{Digit}(a, b, n) b^n \tag{11}$$

and for all n

$$\mathrm{Digit}(a, b, n) < b. \tag{12}$$

This function has the representation

$$d = \mathrm{Digit}(a, b, n) \iff \exists xy\{a = xb^{n+1} + db^n + y \,\&\, d < b \,\&\, y < b^n\} \tag{13}$$

so from now on we can use Digit along with addition, multiplication, and exponentiation.

2.2. Coding of tuples of natural numbers

Choosing the number c sufficiently large, we can code a tuple $\langle d_{l-1}, \ldots, d_0\rangle$ of arbitrary length l by the triple $\langle a, c, l\rangle$ with

$$a = \sum_{n=0}^{l-1} d_n 2^{cn}; \tag{14}$$

in other words, the numbers $d_{l-1}, \ldots, d_0$ are nothing else but the base-2^c digits of the number a. The number c will be called the *key of the code*, and the number a will be called the *key-c cipher of the tuple* $\langle d_{l-1}, \ldots, d_0\rangle$.

Not every triple is a code, but the Diophantine condition

$$\mathrm{Code}(a, c, l) \iff a < 2^{cl} \tag{15}$$

distinguishes those triples that are codes.

2.2.1. Codes of some special tuples

We shall take advantage of the fact that for ciphers of some special tuples we can easily provide explicit expressions using the formula for the sum of a geometric progression and its generalizations. Namely, for sufficiently large c

$$\mathrm{Const}(d, c, l) = \sum_{k=0}^{l-1} d2^{ck} = \frac{2^{cl} - 1}{2^c - 1} d \tag{16}$$

is the key-c cipher of the l-tuple all entries of which are equal to d,

$$\mathrm{Lin}(c, l) = \sum_{k=0}^{l-1} k2^{ck} = \frac{2^{cl}(2^c(l-1) - l) + 2^c}{(2^c - 1)^2} \tag{17}$$

is the key-c cipher of the l-tuple $\langle l-1, l-2, \ldots, 1, 0\rangle$, and

$$\mathrm{Sq}(c, l) = \sum_{k=0}^{l-1} k^2 2^{ck} = \frac{2^{cl}(2^{2c}(l-1)^2 - 2^c(2l^2 - 2l - 1) + l^2) - 2^{2c} - 2^c}{(2^c - 1)^3} \tag{18}$$

is the key-c cipher of the l-tuple $\langle (l-1)^2, (l-2)^2, \ldots, 1^2, 0^2\rangle$; the summations in (16)–(18) can also be easily verified by induction.

2.3. Binomial coefficients

We consider that the binomial coefficient $\binom{m}{n}$ is defined for all m and n by putting $\binom{m}{n} = 0$ for $n > m$.

2.3.1. Diophantine representation

By the Binomial Theorem, the key-$(m+1)$ cipher of the tuple

$$\left\langle \binom{m}{m}, \ldots, \binom{m}{0} \right\rangle \tag{19}$$

is equal to $(2^{m+1} + 1)^m$, so

$$\binom{m}{n} = \mathrm{Digit}((2^{m+1} + 1)^m, 2^{m+1}, n). \tag{20}$$

2.3.2. Divisibility properties

Being a positive integer, the binomial coefficient $\binom{a+b}{b}$ can be uniquely represented as the product of prime powers:

$$\binom{a+b}{b} = 2^{\alpha_2} 3^{\alpha_3} \dots p^{\alpha_p} \dots . \tag{21}$$

Kummer [4][1] gave the following striking method for calculating α_p: *Write the numbers a and b in base-p notation and add them; α_p is equal to the number of carries performed during this operation.*

2.4. Bitwise relations and operations

Let a, b, and c be three numbers with binary expansions

$$a = \sum_{k=0}^{\infty} a_k 2^k, \qquad b = \sum_{k=0}^{\infty} b_k 2^k, \qquad c = \sum_{k=0}^{\infty} c_k 2^k \tag{22}$$

where a_k, b_k, and c_k are either 0 or 1.

2.4.1. Orthogonality

The numbers a and b are said to be *orthogonal*, written $a \perp b$, if $a_k b_k = 0$ for every k. According to Kummer's theorem

$$a \perp b \iff \exists x \left\{ \binom{a+b}{b} = 2x + 1 \right\}. \tag{23}$$

In fact, both the left-hand side and the right-hand side of this equivalence are true if and only if no carry occurs during the addition of a and b in binary notation.

2.4.2. Binary masking

A number c is said to *mask* the number b if $b_k \le c_k$ for every k. The masking relation will be denoted as $\preceq$. Again, according to Kummer's theorem

$$b \preceq c \iff \exists x \left\{ \binom{c}{b} = 2x + 1 \right\}. \tag{24}$$

To see why (24) is true, note that both sides are false whenever $b > c$. Otherwise, put $a = c - b$. According to (23), the right-hand side of (24) means that $a \perp b$ so no carry occurs during the calculation of $c = a + b$ and hence b is masked by c.

2.4.3. Digit-by-digit product

A number c is said to be the *digit-by-digit product* of the numbers a and b, denoted by $a \wedge b$, if $c_k = a_k b_k$ for every k. It is not difficult to see that

$$c = a \wedge b \iff c \preceq a \,\&\, c \preceq b \,\&\, a - c \perp b - c. \tag{25}$$

The implication $\Longrightarrow$ is evident. In the opposite direction, the right-hand side of (25) implies that $c_k \le a_k b_k$ for every k. Suppose that there is k such that $c_k < a_k b_k$, that is, $a_k = b_k = 1$ but $c_k = 0$, and let k_0 be the smallest such k. Then the numbers $a - c$ and $b - c$ are not orthogonal because their k_0-th digits are both "1".

2.5. Bounding entries in a tuple

The binary notation of a number a can be obtained from its base-2^c notation by replacing each 2^c-digit by its binary notation padded by initial zeros as needed to make its length be c. The masking relation $d \preceq 2^e - 1$ is equivalent to the inequality $d \le 2^e - 1$. Thus the condition

$$\mathrm{SmallCode}(a, c, l, e) \iff e \le c \,\&\, a \preceq \mathrm{Const}(2^e - 1, c, l) \tag{26}$$

asserts that the triple $\langle a, c, l\rangle$ codes some l-tuples all entries of which are not greater than $2^e - 1$.

2.6. Comparison of coded tuples

Every tuple has infinitely many codes corresponding to different keys. In this section we provide a Diophantine representation for the relation $\mathrm{Equal}(a', a'', c', c'', l)$ defined to mean that the *triples $\langle a', c', l\rangle$ and $\langle a'', c'', l\rangle$ code the same l-tuple*.

[1] Kummer's result was several times rediscovered and re-proved in different manners; for a modern presentation see, for example, [13] (a proof of this Kummer theorem is also given in [7]).

If for some a, c, c', and c'' both SmallCode(a, c, l, c') and SmallCode(a, c, l, c'') hold, then clearly there are numbers a' and a'' such that the triples $\langle a, c, l\rangle$, $\langle a', c', l\rangle$, and $\langle a'', c'', l\rangle$ all code the same l-tuple. By (14)

$$a \equiv a' \pmod{2^c - 2^{c'}}, \quad a \equiv a'' \pmod{2^c - 2^{c''}}, \tag{27}$$

$$a' < 2^{lc'}, \quad a'' < 2^{lc''}. \tag{28}$$

If c is sufficiently large, namely, if $2^c - 2^{c'} > 2^{lc'}$ and $2^c - 2^{c''} > 2^{lc''}$, then conditions (27) and (28) uniquely determine a' and a''. Thus

$$\begin{aligned}\mathrm{Equal}(a', a'', c', c'', l) \iff \exists acq'q''\{&c = (l+2)(c' + c'' + 1)\ \&\\ &\mathrm{SmallCode}(a, c, l, c')\ \&\ \mathrm{SmallCode}(a, c, l, c'')\ \&\\ &\mathrm{Code}(a', c', l)\ \&\ \mathrm{Code}(a'', c'', l)\ \&\\ &a = a' + (2^c - 2^{c'})q'\ \&\ a = a'' + (2^c - 2^{c''})q''\}.\end{aligned} \tag{29}$$

3. Coding linear algebra

Vectors and matrices can be viewed as tuples of numbers, so we extend our coding technique to them.

3.1. Coding of vectors

In linear algebra one distinguishes row vectors from column vectors, but we will consider the code $\langle a, c, l\rangle$ of an l-tuple $\langle d_{l-1}, \ldots, d_0\rangle$ as code of both the row vector $[d_{l-1}, \ldots, d_0]$ and the column vector

$$\begin{bmatrix} d_0 \\ \vdots \\ d_{l-1} \end{bmatrix} = [d_0, \ldots, d_{l-1}]^{\mathrm{T}}. \tag{30}$$

3.2. Vector by vector product

The product of the row vector coded by the triple $\langle a', c', l\rangle$ and the column vector coded by the triple $\langle a'', c'', l\rangle$ is a 1×1 matrix the only entry of which will be denoted as Scalar(a', a'', c', c'', l). It is easy to see that if $c' = c'' = c$ and this number is sufficiently large, then $\mathrm{Scalar}(a', a'', c', c'', l) = \mathrm{Digit}(a'a'', c, l-1)$. We can reduce the general case to this special case as follows:

$$\begin{aligned}&\mathrm{Scalar}(a', a'', c', c'', l) = s \iff \exists b'b''c\{c = c' + c'' + l\ \&\\ &\quad \mathrm{Equal}(a', b', c', c, l)\ \&\ \mathrm{Equal}(a'', b'', c'', c, l)\ \&\ s = \mathrm{Digit}(b'b'', c, l-1).\end{aligned} \tag{31}$$

Yet another Diophantine representation can be obtained in terms of the digit-by-digit product:

$$\begin{aligned}\mathrm{Scalar}(a', a'', c', c'', l) = s \iff &\exists b'b''c\{c = c' + c'' + l\ \&\\ &\mathrm{Equal}(a', b', c', c, l)\ \&\ \mathrm{Equal}(a'', b'', c'', c, l)\ \&\\ &s2^{c(l-1)} = (b'b'') \wedge \left((2^c - 1)2^{c(l-1)}\right).\end{aligned} \tag{32}$$

3.3. Coding of matrices

The key-c cipher of an $m \times n$ matrix $M = ||d_{ij}||_{i=0\ j=0}^{m-1\ n-1}$ is defined to be equal to the key-cm cipher of the row vector consisting of the key-c ciphers of the columns of the matrix M, i.e., equal to the number

$$a = \sum_{j=0}^{n-1} \left(\sum_{i=0}^{m-1} d_{ij} 2^{ci} \right) 2^{cm(n-j-1)}. \tag{33}$$

The matrix M will be coded by the quadruple $\langle a, c, m, n\rangle$.

3.4. Vector by matrix product

Let Prod$(a', a'', a''', c', c'', c''', m, n)$ mean that product of the row vector V coded by the triple $\langle a', c', m\rangle$ and the matrix M coded by the quadruple $\langle a'', c'', m, n\rangle$ is equal to the row vector W coded by the triple $\langle a''', c''', n\rangle$. The entries of the vector W are just the products of the vector V by the column vectors comprising the matrix M, so we can proceed

as in (32): if Equal(a', b', c', c, m), Equal(a'', a, c'', c, mn), Equal(a, b'', cm, $2cm$, n), and Equal(a''', b''', c''', $2cm$, n) hold for $c = c' + c'' + m$ then

$$b'''2^{c(m-1)} = (b'b'') \wedge \big(\mathrm{Const}((2^c - 1), 2cm, n)2^{c(m-1)}\big). \tag{34}$$

3.5. Selective matrices

An important role in our construction will be played by matrices that have exactly one entry equal to 1 in each column with all other entries equal to 0; such matrices will be called *selective*. We can easily express the property Select(a, c, m, n) saying that *the quadruple* $\langle a, c, m, n\rangle$ *codes a selective matrix*, namely

$$\begin{aligned}&\mathrm{Select}(a, c, m, n) \iff\\ &\quad \mathrm{Prod}(\mathrm{Const}(1, 1, m), a, \mathrm{Const}(1, 1, n), c, 1, 1, m, n).\end{aligned} \tag{35}$$

3.6. Squaring

We need one non-standard operation. Let $V^{\square}$ denote the vector each entry of which is the square of the corresponding entry in V. The corresponding relation between the codes will be denoted by Squaring(a', a'', c', c'', l). Clearly, we have Squaring(Lin(m, m), Sq(m, $2m$), m, $2m$, m). In the general case we can apply a suitable selective matrix:

$$\begin{aligned}\mathrm{Squaring}(a', a'', c', c'', l) \iff\ & \exists ms\{\mathrm{Select}(s, 1, m, l)\&\\ & \mathrm{Prod}(\mathrm{Lin}(m, m), s, a', m, 1, c', m, l)\&\\ & \mathrm{Prod}(\mathrm{Sq}(m, 2m), s, a'', 2m, 1, c'', m, l)\}.\end{aligned} \tag{36}$$

4. Coding Diophantine equations

At the cost of introducing new unknowns, every Diophantine equation can be transformed into an equivalent system consisting of equations of three kinds:

$$x = 1, \tag{37}$$

$$x = y + z, \tag{38}$$

$$x = yz. \tag{39}$$

Clearly, it suffices to have only one equation of the form (37), and equations of the form (39) can be reduced to the special case

$$x = y^2 \tag{40}$$

by use of the identity $(a + b)^2 = a^2 + 2ab + b^2$. Using matrix notation, we can write such a system in the form

$$x_0 = 1, \tag{41}$$

$$XA = XB, \qquad XC = X^{\square}D \tag{42}$$

where $X = [x_{m-1}, x_{m-2}, \ldots, x_0]$ is the vector of unknowns and A, B, C, and D are particular matrices of sizes $m \times k$, $m \times k$, $m \times l$, and $m \times l$ respectively with entries that are either 0 or 1. Thus, we can code the equation by the 7-tuple $\langle a, b, c, d, k, l, m\rangle$ where $\langle a, 1, m, k\rangle$, $\langle b, 1, m, k\rangle$, $\langle c, 1, m, l\rangle$, and $\langle d, 1, m, l\rangle$ code matrices A, B, C, and D respectively.

4.1. Solvability

We can easily give a Diophantine representation for the relation Solution(a, b, c, d, k, l, m, x, z) saying that *the triple* $\langle x, z, m\rangle$ *codes a solution of the system of the form* (41)–(42) *coded by the* 7*-tuple* $\langle a, b, c, d, k, l, m\rangle$:

$$\begin{aligned}\mathrm{Solution}(a, b, c, d, k, l, m, x, z) \iff\ & \exists pqy\{\mathrm{Digit}(x, z, 0) = 1\ \&\\ & \mathrm{Prod}(x, a, p, z, 1, z, m, k)\ \&\ \mathrm{Prod}(x, b, p, z, 1, z, m, k)\ \&\\ & \mathrm{Prod}(x, c, q, z, 1, z, m, k)\ \&\mathrm{Prod}(y, d, q, 2z, 1, z, m, k)\ \&\\ & \mathrm{Squaring}(x, y, z, 2z, m).\end{aligned} \tag{43}$$

4.2. Universal Diophantine equations

Let $K(y_1, \ldots, y_7)$ be any polynomial which assumes each of its values only once. Using the techniques described above we can transform the system

$$K(y_1, \ldots, y_7) = k, \quad \text{Solution}(y_1, \ldots, y_9), \tag{44}$$

$$\text{Digit}(y_8, y_9, 1) = p_1, \ldots, \text{Digit}(y_8, y_9, n) = p_n \tag{45}$$

into a universal Diophantine equation (6).

4.3. Staircase systems

Let us consider the following system of equations analogous to (42) but with an even number of unknowns split into two groups (with and without $'$):

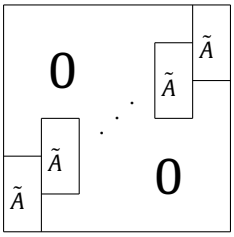
$$X\tilde{A} = X\tilde{B}, \qquad X\tilde{C} = X^{\square}\tilde{D} \tag{46}$$

where $X = [x'_{m-1}, \ldots, x'_0, x_{m-1}, \ldots, x_0]$ is the vector of unknowns and $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, and $\tilde{D}$ are particular matrices of sizes $2m \times k$, $2m \times k$, $2m \times l$, and $2m \times l$, respectively. Let us further consider the following system in the unknowns $x_0, x_1, \ldots, x_{(p+1)m-1}$:

$$\begin{aligned} &X_0\tilde{A} = X_0\tilde{B}, && X_0\tilde{C} = X_0^{\square}\tilde{D} \\ &\vdots && \vdots \\ &X_{p-1}\tilde{A} = X_{p-1}\tilde{B}, && X_{p-1}\tilde{C} = X_{p-1}^{\square}\tilde{D} \end{aligned} \tag{47}$$

where $X_k = [x_{(k+2)m-1}, x_{(k+2)m-2}, \ldots, x_{km}]$. This system can be written in the canonical form (42) with particular matrices A, B, C, and D of sizes $(p+1)m \times pk$, $(p+1)m \times pk$, $(p+1)m \times pl$, and $(p+1)m \times pl$ respectively.

It is easy to see that the matrix A (as well as the three other matrices) has the following block structure:



Thanks to this regular structure, we can easily determine the key-1 cipher a of the matrix A via the key-1 cipher $\tilde{a}$ of the matrix $\tilde{A}$. Namely, if $\text{Equal}(\tilde{a}, a', 2m, (p+1)m, k)$ holds, then the quadruple $\langle a', 1, (p+1)m, k\rangle$ codes the matrix formed by the last k columns of the matrix A, and $a = a'\text{Const}(1, (k(p+1)+1)m, p)$; the key-1 cyphers of the three other matrices can be obtained in a similar manner.

Now we can define the relation $\text{StairSolution}(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m, p, x, z)$ saying that *the triple* $\langle x, z, (p+1)m\rangle$ *codes a solution of the system consisting of Eqs.* (41) *and* (47) *where the matrices* $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, *and* $\tilde{D}$ *are coded, respectively, by triples* $\langle \tilde{a}, 1, m, k\rangle$, $\langle \tilde{b}, 1, m, k\rangle$, $\langle \tilde{c}, 1, m, l\rangle$, $\langle \tilde{d}, 1, m, l\rangle$:

$$\begin{aligned} &\text{StairSolution}(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m, p, x, z) \iff \\ &\exists a'b'c'd'e_1e_2\{\text{Equal}(\tilde{a}, a', 2m, (p+1)m, k)\&\text{Equal}(\tilde{b}, b', 2m, (p+1)m, k)\& \\ &\text{Equal}(\tilde{c}, c', 2m, (p+1)m, k)\&\text{Equal}(\tilde{d}, d', 2m, (p+1)m, k)\ \& \\ &e_1 = \text{Const}(1, (k(p+1)+1)m, p)\ \&\ e_2 = \text{Const}(1, (l(p+1)+1)m, p)\ \& \\ &\qquad\qquad \text{Solution}(a'e_1, b'e_1, c'e_2, d'e_2, k, l, m, x, z). \end{aligned} \tag{48}$$

5. Applications

The technique described above for constructing ciphers of staircase systems of equations allows us to arithmetize Turing machines, register machines, and partial recursive functions in a uniform manner, and to eliminate bounded universal quantifiers as well.

5.1. Elimination of bounded universal quantifier

Consider the formula

$$\forall x_2\{x_2 < p \Rightarrow \exists x_3 \ldots x_k P(x_1, x_2, \ldots, x_n) = 0\}. \tag{49}$$

Introducing new unknowns, rewrite the equation $P(x_1, x_2, \ldots, x_n) = 0$ as a system consisting of Eq. (41) and a number of equations of the form (38) and (40); add to these equations the three equations

$$x'_0 = x_0, \quad x'_1 = x_1, \quad x'_2 = x_2 + x_0 \tag{50}$$

and calculate the code $\langle \tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m\rangle$ of the resulting system. Formula (49) is equivalent to the existential formula

$$\exists xz\{\mathrm{StairSolution}(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m, p, x, z)\ \&\ \mathrm{Digit}(x, z, 1) = x_1\ \&\ \mathrm{Digit}(x, z, 2) = 0)\}. \tag{51}$$

5.2. Arithmetization of Turing machines

Let $\{\alpha_0, \ldots, \alpha_{b-1}\}$ be the tape alphabet of a Turing machine with α_0 denoting the blank cell. The content of a semi-infinite tape $\alpha_{d_0}\alpha_{d_1}, \ldots$ can be represented by the number $t = \sum_{i=0}^{\infty} d_i b^i$. A configuration of a Turing machine can be represented by the quadruple $\langle q, v, t_{\mathrm{L}}, t_{\mathrm{R}}\rangle$ where q is the current state, α_v is the content of the cell currently observed by the head, and t_{L} and t_{R} represent the content of the tape to the left and to the right of the head respectively in the sense described above.

If the next configuration $\langle q', v', t'_{\mathrm{L}}, t'_{\mathrm{R}}\rangle$ is obtained by writing the symbol α_w and moving the head to the right, then

$$t'_{\mathrm{L}} = t_{\mathrm{L}} b + w\ \&\ t'_{\mathrm{R}} b + v' = t_{\mathrm{R}}; \tag{52}$$

dual equalities can be written for the case of moving to the left. Transition from one configuration to another is described by a formula of the form

$$\bigvee_{i=1}^{m}\bigvee_{j=0}^{b-1}\{q = i\ \&\ v = j\ \&\ q' = Q_{i,j}\ \&\ T(i, j)\} \tag{53}$$

where m is the number of states, $Q_{i,j}$ is a number in the range $0 \ldots m$ with the zero value corresponding to the STOP state, and $T(i, j)$ is a copy of either (52) or its dual with a particular value of w.

Using the relation StairSolution we can easily describe an arbitrary number of steps by a Turing machine by a single Diophantine equation, so it only remains to add descriptions of the initial and final configurations, which can be easily done in a natural way.

5.3. Arithmetization of register machines

Register machines were introduced almost simultaneously by several authors: Lambek [5], Melzak [8], Minsky [9,10], and Shepherdson and Sturgis [12]. A register machine has a finite number of *registers* R1, ..., Rn each of which is capable of containing an arbitrarily large natural number. The machine, being in one of finitely many states, can perform instructions of three types:

I. Rl + +; Qi
II. Rl − −; Qi; Qj
III. STOP.

An instruction of type I means that the machine is to increase register Rl by 1 and pass to state Qi.

An instruction of type II means that the machine is to decrease register Rl by 1 and pass to state Si; however, if register Rl already contains 0, its value does not change and the machine goes to state Qj instead of Qi.

In spite of their very primitive instructions, register machines are in principle as powerful as all other standard abstract computing devices.

A configuration of a register machine is naturally represented by an $(n + 1)$-tuple $\langle q, r_1, \ldots, r_n\rangle$ consisting of the current state q and current content of the registers. The verbal description of the instructions of a register machine above can be straightforwardly translated into a Diophantine equation connecting the current configuration $\langle q, r_1, \ldots, r_n\rangle$ and the subsequent one $\langle q', r'_1, \ldots, r'_n\rangle$. The use of the relation StairSolution allows us to describe any number of steps of a register machine, and the description of the initial and terminating configurations is also straightforward.

5.4. Arithmetization of partial recursive functions

It is trivial to write down Diophantine representations for the initial functions: constants, projections, and the successor.

Let function F be defined by primitive recursion

$$F(i, 0) = G(i) \tag{54}$$

$$F(i, j + 1) = H(i, j, F(i, j)) \tag{55}$$

where functions G and H have Diophantine representations

$$G(i) = g \iff \exists v_1 \dots v_p\{P(i, g, v_1, \dots, v_p) = 0\}, \tag{56}$$

$$H(i, j, f) = h \iff \exists w_1 \dots w_q\{Q(i, j, f, h, w_1, \dots, w_q) = 0\}. \tag{57}$$

Introducing new unknowns, rewrite the equation $Q(x_1, x_2, x_3, x_3', x_4, \dots, x_{q+3}) = 0$ as a system consisting of Eq. (41) and a number of equations of the form (38) and (40); add to these equations the three equations

$$x_0' = x_0, \quad x_1' = x_1, \quad x_2' = x_2 + x_0 \tag{58}$$

and calculate the code $\langle \tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m\rangle$ of the resulting system. Then the function F can be defined by the following formula:

$$\begin{aligned} F(i, j) = f \iff \exists g v_1 \dots v_q xz\{&P(i, g, v_1, \dots, v_p) = 0\& \\ &\mathrm{Digit}(x, z, 1) = i\&\mathrm{Digit}(x, z, 2) = 0\&\mathrm{Digit}(x, z, 3) = g\& \\ &\mathrm{StairSolution}(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m, j, x, z)\&\mathrm{Digit}(x, z, jm + 3) = f\}. \end{aligned} \tag{59}$$

Now let function $F(i)$ be defined as the minimal value of j for which $H(i, j) = 0$ where the function H has the Diophantine representation

$$H(i, j) = h \iff \exists w_1 \dots w_q\{Q(i, j, h, w_1, \dots, w_q) = 0\}. \tag{60}$$

Introducing new unknowns, rewrite the equation $Q(x_1, x_2, x_3 + 1, x_4, \dots, x_{q+3}) = 0$ as a system consisting of Eq. (41) and a number of equations of the form (38) and (40); add to these equations the three equations

$$x_0' = x_0, \quad x_1' = x_1, \quad x_2' = x_2 + x_0 \tag{61}$$

and calculate the code $\langle \tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m\rangle$ of the resulting system. Now the function F can be defined by the following formula:

$$\begin{aligned} F(i) = f \iff \exists w_1 \dots w_q xz\{&Q(i, f, 0, w_1, \dots, w_q) = 0\& \\ &\mathrm{Digit}(x, z, 1) = i\&\mathrm{Digit}(x, z, 2) = 0\& \\ &\mathrm{StairSolution}(\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}, k, l, m, f, x, z)\}. \end{aligned} \tag{62}$$

References

[1] M. Davis, Arithmetical problems and recursively enumerable predicates, The Journal of Symbolic Logic 18 (1) (1953) 33–41.
[2] M. Davis, H. Putnam, J. Robinson, The decision problem for exponential Diophantine equations, Annals of Mathematics 74 (2) (1961) 425–436. Reprinted in The Collected Works of Julia Robinson, Collected Works, 6, pp.77–88, American Mathematical Society, Providence, RI, 1996, ISBN: 0-8218-0575-4.
[3] K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. I, Monatshefte für Mathematik und Physik 38 (1) (1931) 173–198. Reprinted with English translation in Solomon Feferman et al., editors, Kurt Gödel. Collected Works, volume I, pages 144–195, Oxford University Press, 1986. English translations can be found in many other collections, in particular in Martin Davis, editor, The Undecidable, pages 4–38, Raven Press, Hewlett, New York, 1965 (reprinted by Dover Publications, Inc., Mineola, NY, 2004), and in Jean van Heijenoort, editor, From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931, pages 596–616, Harvard University Press, Cambridge, MA, 1967.
[4] E.E. Kummer, Über die Ergänzungssätze zu den allgemeinen Reciprocitätsgesetzen, Journal für die Reine und Angewandte Mathematik 44 (1852) 93–146.
[5] J. Lambek, How to program an infinite abacus, Canadian Mathematical Bulletin 4 (1961) 295–302.
[6] Yu.V. Matiyasevich, Enumerable sets are Diophantine, Dokl. AN SSSR 191 (2) (1970) 278–282 (in Russian). English translation in Soviet Math. Doklady, 11 (2), 354–358 (1970), translation reprinted in Gerald E. Sacks, editor, Mathematical Logic in the 20th Century, pages 269–273, Singapore University Press, Singapore; World Scientific Publishing Co., Inc., River Edge, NJ, 2003.
[7] Yu.V. Matiyasevich, Desyataya Problema Gilberta, Fizmatlit, Moscow, 1993, English translation: Hilbert's Tenth Problem. MIT Press, Cambridge, MA, London, 1993. French translation: Le dixième Problème de Hilbert, Masson, Paris Milan Barselone, 1995. URL: http://logic.pdmi.ras.ru/~yumat/H10Pbook.
[8] Z.A. Melzak, An informal arithmetical approach to computability and computation, Canadian Mathematical Bulletin 4 (1961) 279–293.
[9] M.L. Minsky, Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines, Annals of Mathematics, Second Series 74 (3) (1961) 437–455.
[10] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, 1967.
[11] J. Robinson, Existential definability in arithmetic, Transactions of the American Mathematical Society 72 (1952) 437–449. Reprinted in The Collected Works of Julia Robinson, Collected Works, 6, pp.47–59, American Mathematical Society, Providence, RI, 1996, ISBN: 0-8218-0575-4.
[12] J.C. Shepherdson, H.E. Sturgis, Computability of recursive functions, Journal of the Association for Computing Machinery 10 (2) (1963) 217–255.
[13] D. Singmaster, Notes on binomial coefficients. I, II, III, The Journal of the London Mathematical Society (Second Series) 8 (3) (1974) 545–548, 549–554, 555–560.