Proceedings of the 2014 Federated Conference on
Computer Science and Information Systems pp. 623–629

DOI: 10.15439/2014F91
ACSIS, Vol. 2

MuPAD codes which implement limit-computable functions that cannot be bounded by any computable function

Apoloniusz Tyszka
University of Agriculture
Faculty of Production and Power Engineering
Balicka 116B, 30-149 Kraków, Poland
Email: rttyszka@cyf-kr.edu.pl

***Abstract*—Let $E_n = \{x_k = 1,\ x_i + x_j = x_k,\ x_i \cdot x_j = x_k :\ i, j, k \in \{1, \dots, n\}\}$. For a positive integer n, let $f(n)$ denote the smallest non-negative integer b such that for each system $S \subseteq E_n$ with a solution in non-negative integers $x_1, \dots, x_n$ there exists a solution of S in non-negative integers not greater than b. We prove that if a function $\Gamma : \mathbb{N} \setminus \{0\} \to \mathbb{N}$ is computable, then f dominates Γ i.e. there exists a positive integer m such that $\Gamma(n) < f(n)$ for any $n \geq m$. For positive integers n, m, let $g(n, m)$ denote the smallest non-negative integer b such that for each system $S \subseteq E_n$ with a solution in $\{0, \dots, m-1\}^n$ there exists a solution of S in $\{0, \dots, b\}^n$. Then,**

$$g(n, m) \leq m - 1, \tag{1}$$

$$0 = g(n, 1) < 1 = g(n, 2) \leq g(n, 3) \leq g(n, 4) \leq \dots \tag{2}$$

and

$$g(n, f(n)) < f(n) = g(n, f(n) + 1) =$$

$$g(n, f(n) + 2) = g(n, f(n) + 3) = \dots \tag{3}$$

We present an infinite loop in *MuPAD* which takes as input a positive integer n and returns $g(n, m)$ on the m-th iteration.

***Index Terms*—Hilbert's Tenth Problem, infinite loop, limit-computable function, *MuPAD*, trial-and-error computable function.**

LIMIT-computable functions, also known as trial-and-error computable functions, have been thoroughly studied, see [6, pp. 233–235] for the main results. Our first goal is to present an infinite loop in *MuPAD* which finds the values of a limit-computable function $f : \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0\}$ by an infinite computation, where f dominates all computable functions. There are many limit-computable functions $f : \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0\}$ which cannot be bounded by any computable function. For example, this follows from [2, p. 38, item 4], see also [5, p. 268] where Janiczak's result is mentioned. Unfortunately, for all known such functions f, it is difficult to write a suitable computer program. The sophisticated choice of a function f will allow us to do so.

Let

$$E_n = \{x_k = 1,\ x_i + x_j = x_k,\ x_i \cdot x_j = x_k :\ i, j, k \in \{1, \dots, n\}\}.$$

For a positive integer n, let $f(n)$ denote the smallest non-negative integer b such that for each system $S \subseteq E_n$ with a solution in non-negative integers $x_1, \dots, x_n$ there exists a solution of S in non-negative integers not greater than b. This definition is correct because there are only finitely many subsets of E_n. For positive integers n, m, let $g(n, m)$ denote the smallest non-negative integer b such that for each system $S \subseteq E_n$ with a solution in $\{0, \dots, m-1\}^n$ there exists a solution of S in $\{0, \dots, b\}^n$. Then, conditions (1)-(3) stated in the abstract hold.

Obviously, $f(1) = 1$. The system

$$\begin{cases} x_1 = 1 \\ x_1 + x_1 = x_2 \\ x_2 \cdot x_2 = x_3 \\ x_3 \cdot x_3 = x_4 \\ \dots \\ x_{n-1} \cdot x_{n-1} = x_n \end{cases}$$

has a unique integer solution, namely $\left(1, 2, 4, 16, \dots, 2^{2^{n-3}}, 2^{2^{n-2}}\right)$. Therefore, $f(n) \geq 2^{2^{n-2}}$ for any $n \geq 2$.

The Davis-Putnam-Robinson-Matiyasevich theorem states that every recursively enumerable set $\mathcal{M} \subseteq \mathbb{N}^n$ has a Diophantine representation, that is

$$(a_1, \dots, a_n) \in \mathcal{M} \Longleftrightarrow$$

$$\exists x_1, \dots, x_m \in \mathbb{N} \quad W(a_1, \dots, a_n, x_1, \dots, x_m) = 0 \tag{R}$$

for some polynomial W with integer coefficients, see [3]. The polynomial W can be computed, if we know the Turing machine M such that, for all $(a_1, \dots, a_n) \in \mathbb{N}^n$, M halts on $(a_1, \dots, a_n)$ if and only if $(a_1, \dots, a_n) \in \mathcal{M}$, see [3]. The representation (R) is said to be single-fold, if for any $a_1, \dots, a_n \in \mathbb{N}$ the equation $W(a_1, \dots, a_n, x_1, \dots, x_m) = 0$ has at most one solution $(x_1, \dots, x_m) \in \mathbb{N}^m$. Yu. Matiyasevich conjectures that each recursively enumerable set $\mathcal{M} \subseteq \mathbb{N}^n$ has a single-fold Diophantine representation, see [4].

Let $\mathcal{R}ng$ denote the class of all rings $\boldsymbol{K}$ that extend $\mathbb{Z}$.

Lemma ([8, p. 720]). *Let $D(x_1, \dots, x_p) \in \mathbb{Z}[x_1, \dots, x_p]$. Assume that $\deg(D, x_i) \geq 1$ for each $i \in \{1, \dots, p\}$. We can compute a positive integer $n > p$ and a system $T \subseteq E_n$ which satisfies the following two conditions:*

`Condition 1.` *If* $\boldsymbol{K} \in \mathcal{R}ng \cup \{\mathbb{N},\ \mathbb{N} \setminus \{0\}\}$, *then*

$$\forall \tilde{x}_1, \dots, \tilde{x}_p \in \boldsymbol{K} \ \Big(D(\tilde{x}_1, \dots, \tilde{x}_p) = 0 \Longleftrightarrow$$

$$\exists \tilde{x}_{p+1}, \dots, \tilde{x}_n \in \boldsymbol{K} \ (\tilde{x}_1, \dots, \tilde{x}_p, \tilde{x}_{p+1}, \dots, \tilde{x}_n) \text{ solves } T\Big)$$

`Condition 2.` *If* $\boldsymbol{K} \in \mathcal{R}ng \cup \{\mathbb{N},\ \mathbb{N} \setminus \{0\}\}$, *then for each* $\tilde{x}_1, \dots, \tilde{x}_p \in \boldsymbol{K}$ *with* $D(\tilde{x}_1, \dots, \tilde{x}_p) = 0$, *there exists a unique tuple* $(\tilde{x}_{p+1}, \dots, \tilde{x}_n) \in \boldsymbol{K}^{n-p}$ *such that the tuple* $(\tilde{x}_1, \dots, \tilde{x}_p, \tilde{x}_{p+1}, \dots, \tilde{x}_n)$ *solves* T.

Conditions 1 and 2 imply that for each $\boldsymbol{K} \in \mathcal{R}ng \cup \{\mathbb{N},\ \mathbb{N} \setminus \{0\}\}$, *the equation* $D(x_1, \dots, x_p) = 0$ *and the system* T *have the same number of solutions in* $\boldsymbol{K}$.

Theorem 1. *If a function* $\Gamma : \mathbb{N} \setminus \{0\} \to \mathbb{N}$ *is computable, then there exists a positive integer* m *such that* $\Gamma(n) < f(n)$ *for any* $n \geq m$.

Proof. The Davis-Putnam-Robinson-Matiyasevich theorem and the Lemma for $\boldsymbol{K} = \mathbb{N}$ imply that there exists an integer $s \geq 3$ such that for any non-negative integers x_1, x_2,

$$(x_1, x_2) \in \Gamma \Longleftrightarrow \exists x_3, \dots, x_s \in \mathbb{N} \ \ \Phi(x_1, x_2, x_3, \dots, x_s), \quad \texttt{(E)}$$

where the formula $\Phi(x_1, x_2, x_3, \dots, x_s)$ is a conjunction of formulae of the forms $x_k = 1$, $x_i + x_j = x_k$, $x_i \cdot x_j = x_k$ ($i, j, k \in \{1, \dots, s\}$). Let $[\cdot]$ denote the integer part function. For each integer $n \geq 6 + 2s$,

$$n - \left[\frac{n}{2}\right] - 3 - s \geq 6 + 2s - \left[\frac{6+2s}{2}\right] - 3 - s \geq 6 + 2s - \frac{6+2s}{2} - 3 - s = 0$$

For an integer $n \geq 6 + 2s$, let S_n denote the following system

$$\left\{\begin{array}{rcl}
\multicolumn{3}{l}{\text{all equations occurring in } \Phi(x_1, x_2, x_3, \dots, x_s)} \\
\multicolumn{3}{l}{n - \left[\frac{n}{2}\right] - 3 - s \text{ equations of the form } z_i = 1} \\
t_1 & = & 1 \\
t_1 + t_1 & = & t_2 \\
t_2 + t_1 & = & t_3 \\
& \dots & \\
t_{[\frac{n}{2}]-1} + t_1 & = & t_{[\frac{n}{2}]} \\
t_{[\frac{n}{2}]} + t_{[\frac{n}{2}]} & = & w \\
w + y & = & x_1 \\
y + y & = & y \text{ (if } n \text{ is even)} \\
y & = & 1 \text{ (if } n \text{ is odd)} \\
x_2 + t_1 & = & u
\end{array}\right.$$

with n variables. By the equivalence `(E)`, S_n is satisfiable over $\mathbb{N}$. If a n-tuple $(x_1, x_2, x_3, \dots, x_s, \dots, w, y, u)$ of non-negative integers solves S_n, then by the equivalence `(E)`,

$$x_2 = \Gamma(x_1) = \Gamma(w + y) = \Gamma\left(2 \cdot \left[\frac{n}{2}\right] + y\right) = \Gamma(n)$$

Therefore, $u = x_2 + t_1 = \Gamma(n) + 1 > \Gamma(n)$. This shows that $\Gamma(n) < f(n)$ for any $n \geq 6 + 2s$. □

Theorem 2. *There exists a computable function* $\varphi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *which satisfies the following conditions:*

`1)` *For each non-negative integers* n *and* l,

$$\varphi(n, l) \leq l$$

`2)` *For each non-negative integer* n,

$$0 = \varphi(n, 0) < 1 = \varphi(n, 1) \leq \varphi(n, 2) \leq \varphi(n, 3) \leq \dots$$

`3)` *For each non-negative integer* n, *the sequence* $\{\varphi(n, l)\}_{l \in \mathbb{N}}$ *is bounded from above.*

`4)` *The function*

$$\mathbb{N} \ni n \xrightarrow{\theta} \theta(n) = \lim_{l \to \infty} \varphi(n, l) \in \mathbb{N} \setminus \{0\}$$

dominates all computable functions.

`5)` *For each non-negative integer* n,

$$\varphi(n, \theta(n) - 1) < \theta(n) = \varphi(n, \theta(n)) =$$

$$\varphi(n, \theta(n) + 1) = \varphi(n, \theta(n) + 2) = \dots$$

Proof. Let us say that a tuple $y = (y_1, \dots, y_n) \in \mathbb{N}^n$ is a *duplicate* of a tuple $x = (x_1, \dots, x_n) \in \mathbb{N}^n$, if

$$(\forall k \in \{1, \dots, n\} \ (x_k = 1 \Longrightarrow y_k = 1)) \ \wedge$$
$$(\forall i, j, k \in \{1, \dots, n\} \ (x_i + x_j = x_k \Longrightarrow y_i + y_j = y_k)) \ \wedge$$
$$(\forall i, j, k \in \{1, \dots, n\} \ (x_i \cdot x_j = x_k \Longrightarrow y_i \cdot y_j = y_k))$$

For non-negative integers n and l, we define $\varphi(n, l)$ as the smallest non-negative integer b such that for each $x \in \{0, \dots, l\}^{n+1}$ there exists a duplicate of x in $\{0, \dots, b\}^{n+1}$. Theorem 1 implies the claim of item `4)` whereas the following *MuPAD* code performs a Turing computation of $\varphi(n, l)$.

```
input("input the value of n",n):
input("input the value of l",l):
n:=n+1:
X:=[i $ i=0..l]:
Y:=combinat::cartesianProduct(X $i=1..n):
W:=combinat::cartesianProduct(X $i=1..n):
for s from 1 to nops(Y) do
for t from 1 to nops(Y) do
m:=0:
for i from 1 to n do
if Y[s][i]=1 and Y[t][i]<>1
then m:=1 end_if:
for j from i to n do
for k from 1 to n do
if Y[s][i]+Y[s][j]=Y[s][k] and
Y[t][i]+Y[t][j]<>Y[t][k]
then m:=1 end_if:
if Y[s][i]*Y[s][j]=Y[s][k] and
Y[t][i]*Y[t][j]<>Y[t][k]
then m:=1 end_if:
end_for:
end_for:
end_for:
if m=0 and
max(Y[t][i] $i=1..n)<max(Y[s][i] $i=1..n)
then W:=listlib::setDifference(W,[Y[s]])
end_if:
```

```
end_for:
end_for:
print(max(max(W[z][u] $u=1..n) $z=1..nops(W))):
```

Code 1
A Turing computation of $\varphi(n, l)$

□

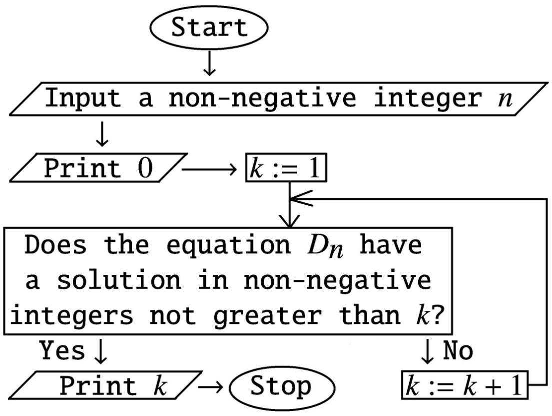
Code 1 is also stored in [10]. The following algorithm performs an infinite computation of $f(n)$, because it returns $g(n, m)$ on the m-th iteration, where m stands for any positive integer.

```
input("input the value of n",n):
i:=0:
while TRUE do
print(φ(n-1,i)):
i:=i+1:
end_while:
```

Algorithm 1
An infinite computation of $f(n)$

A slightly changed *MuPAD* code that implements Algorithm 1 is stored in [10, Code 4].

Let us fix a computable enumeration $D_0, D_1, D_2, \ldots$ of all Diophantine equations. The following flowchart illustrates an infinite computation of a limit-computable function that cannot be bounded by any computable function.



Algorithm 2
A loop whose execution does not always terminate, and that defines a partially computable function that cannot be bounded by any computable function from $\mathbb{N}$ to $\mathbb{N}$

For each non-negative integer n, the function has a non-zero value at n if and only if the equation D_n has a solution in non-negative integers. Unfortunately, the function does not have any easy implementation.

The following *MuPAD* code is stored in [10].

```
input("input the value of n",n):
print(0):
A:=op(ifactor(210*(n+1))):
B:=[A[2*i+1] $i=1..(nops(A)-1)/2]:
S:={}:
for i from 1 to floor(nops(B)/4) do
if B[4*i]=1 then
S:=S union {B[4*i-3]} end_if:
if B[4*i]=2 then S:=S union
{[B[4*i-3],B[4*i-2],B[4*i-1],"+"]}
end_if:
if B[4*i]>2 then S:=S union
{[B[4*i-3],B[4*i-2],B[4*i-1],"*"]}
end_if:
end_for:
m:=2:
repeat
C:=op(ifactor(m)):
W:=[C[2*i+1]-1 $i=1..(nops(C)-1)/2]:
T:={}:
for i from 1 to nops(W) do
for j from 1 to nops(W) do
for k from 1 to nops(W) do
if W[i]=1 then T:=T union {i} end_if:
if W[i]+W[j]=W[k] then
T:=T union {[i,j,k,"+"]} end_if:
if W[i]*W[j]=W[k] then
T:=T union {[i,j,k,"*"]} end_if:
end_for:
end_for:
end_for:
m:=m+1:
until S minus T={} end_repeat:
print(max(W[i] $i=1..nops(W))):
```

Code 2
A loop whose execution does not always terminate, and that defines a partially computable function that cannot be bounded by any computable function from $\mathbb{N}$ to $\mathbb{N}$

Theorem 3. *The above code implements a limit-computable function $\xi : \mathbb{N} \to \mathbb{N}$ that cannot be bounded by any computable function. The code takes as input a non-negative integer n, returns 0, and computes a system S of polynomial equations. If the loop terminates for S, then the next instruction returns $\xi(n)$. If the loop does not terminate, then $\xi(n) = 0$. The loop defines a partially computable function that cannot be bounded by any computable function from $\mathbb{N}$ to $\mathbb{N}$.*

Proof. Let $n \in \mathbb{N}$, and let ${p_1}^{t(1)} \cdot \ldots \cdot {p_s}^{t(s)}$ be a prime factorization of $210 \cdot (n+1)$, where $t(1), \ldots, t(s)$ denote positive integers. Obviously, $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and $p_4 = 7$.

For each positive integer i that satisfies $4i \le s$ and $t(4i) = 1$, the code constructs the equation $x_{t(4i-3)} = 1$.

For each positive integer i that satisfies $4i \le s$ and $t(4i) = 2$, the code constructs the equation $x_{t(4i-3)} + x_{t(4i-2)} = x_{t(4i-1)}$.

For each positive integer i that satisfies $4i \le s$ and $t(4i) > 2$, the code constructs the equation $x_{t(4i-3)} \cdot x_{t(4i-2)} = x_{t(4i-1)}$.

The last three facts imply that the code assigns to n a finite and non-empty system S which consists of equations of the forms: $x_k = 1$, $x_i + x_j = x_k$, and $x_i \cdot x_j = x_k$. Conversely, each such system S is assigned to some non-negative integer n.

Starting with the instruction $m := 2$, the code tries to find a solution of S in non-negative integers by performing a brute-force search. If a solution exists, then the search terminates and the code returns a non-negative integer $\xi(n)$ such that the system S has a solution in non-negative integers not greater than $\xi(n)$. In the opposite case, the execution of the code never terminates.

A negative solution to Hilbert's Tenth Problem ([3]) and the Lemma for $\boldsymbol{K} = \mathbb{N}$ imply that the code implements a limit-computable function $\xi : \mathbb{N} \to \mathbb{N}$ that cannot be bounded by any computable function. □

The execution of the last code does not terminate for $n = 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 - 1 = 323322$, when the code tries to find a solution of the system $\{x_1 + x_1 = x_1,\ x_1 = 1\}$. Execution terminates for any $n < 323322$, when the code returns 0 and next 1 or 0. The last claim holds only theoretically. In fact, for $n = 2^{18} - 1 = 262143$, the algorithm of the code returns 1 solving the equation $x_{19} = 1$ on the $\left(2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 \cdot 61 \cdot 67^2 - 1\right)$-th iteration.

Let $\mathcal{P}$ denote a predicate calculus with equality and one binary relation symbol, and let Λ be a computable function that maps $\mathbb{N}$ onto the set of sentences of $\mathcal{P}$. The following pseudocode in *MuPAD* implements a limit-computable function $\sigma : \mathbb{N} \to \mathbb{N}$ that cannot be bounded by any computable function.

```
input("input the value of n",n):
print(0):
k:=1:
while Λ(n) holds in all models of size k do
k:=k+1:
end_while:
print(k):
```

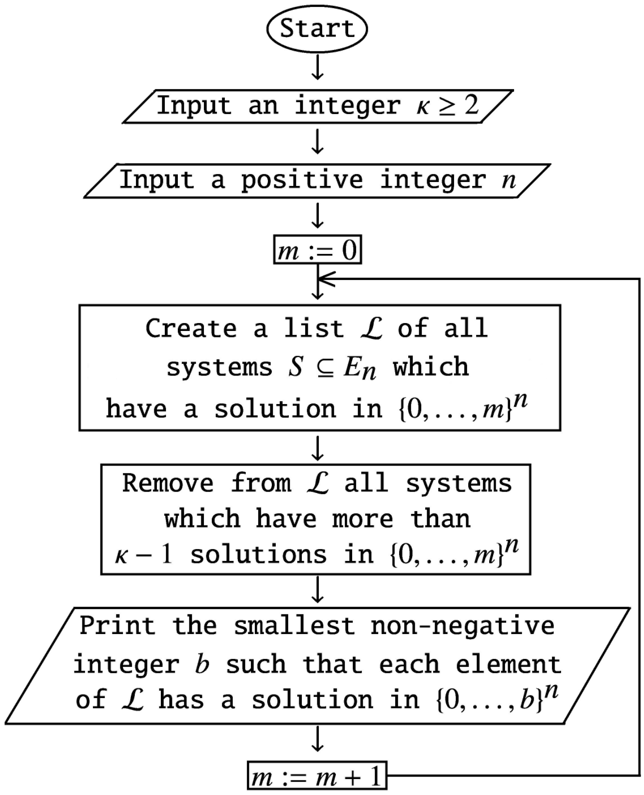
Algorithm 3
A loop whose execution does not always terminate, and that defines a partially computable function that cannot be bounded by any computable function from $\mathbb{N}$ to $\mathbb{N}$

The proof follows from the fact that the set of sentences of $\mathcal{P}$ that are true in all finite and non-empty models is not recursively enumerable, see [1, p. 129], where it is concluded from Trakhtenbrot's theorem. The author has no idea how to transform the pseudocode into a correct computer program.

The commercial version of *MuPAD* is no longer available as a stand-alone product, but only as the *Symbolic Math Toolbox* of *MATLAB*. Fortunately, the presented codes can be executed by *MuPAD Light*, which was and is free, see [11]. Similar codes in *MuPAD Light* are presented and discussed at http://arxiv.org/abs/1310.5363.

Limit-computable functions are related to the question of the decidability of Diophantine equations with a finite number of solutions in non-negative integers. Let $\kappa \in \{2, 3, 4, \ldots, \omega, \omega_1\}$. For a positive integer n, let $f_\kappa(n)$ denote the smallest non-negative integer b such that for each system $S \subseteq E_n$ which has a solution in non-negative integers $x_1, \ldots, x_n$ and which has less than κ solutions in non-negative integers $x_1, \ldots, x_n$, there exists a solution of S in non-negative integers not greater than b. Since $f_{\omega_1} = f$, f_{ω_1} is limit-computable by Algorithm 1.

Obviously, $f_2(n)$ is the smallest non-negative integer b such that for each system $S \subseteq E_n$ with a unique solution in non-negative integers $x_1, \ldots, x_n$ this solution belongs to $[0, b]^n$. If $\kappa < \omega$, then the function f_κ is limit-computable as the flowchart below describes an infinite computation of $f_\kappa(n)$.



Algorithm 4
An infinite computation of $f_\kappa(n)$

The following *MuPAD* code is stored in [10, Code 3] and performs an infinite computation of $f_2(n)$.

```
input("input the value of n",n):
X:=[0]:
while TRUE do
Y:=combinat::cartesianProduct(X $i=1..n):
W:=combinat::cartesianProduct(X $i=1..n):
for s from 1 to nops(Y) do
for t from 1 to nops(Y) do
m:=0:
for i from 1 to n do
if Y[s][i]=1 and Y[t][i]<>1 then m:=1 end_if:
for j from i to n do
for k from 1 to n do
```

```
if Y[s][i]+Y[s][j]=Y[s][k] and
Y[t][i]+Y[t][j]<>Y[t][k] then m:=1 end_if:
if Y[s][i]*Y[s][j]=Y[s][k] and
Y[t][i]*Y[t][j]<>Y[t][k] then m:=1 end_if:
end_for:
end_for:
end_for:
if m=0 and s<>t then
W:=listlib::setDifference(W,[Y[s]]) end_if:
end_for:
end_for:
print(max(max(W[z][u] $u=1..n) $z=1..nops(W))):
X:=append(X,nops(X)):
end_while:
```

Code 3
An infinite computation of $f_2(n)$

Theorem 5 implies that f_2 dominates any function $h : \mathbb{N} \setminus \{0\} \to \mathbb{N}$ with a single-fold Diophantine representation. Therefore, Matiyasevich's conjecture on single-fold Diophantine representations implies that f_2 dominates all computable functions from $\mathbb{N} \setminus \{0\}$ to $\mathbb{N}$.

Obviously, $f_K(1) = 1$ and $f_K(n) \geq 2^{2^{n-2}}$ for any $n \geq 2$. Theorem 1 implies that the equality

$$f_K = \{(1,1)\} \cup \left\{\left(n, 2^{2^{n-2}}\right) : n \in \{2,3,4,\dots\}\right\}$$

is false for $\kappa = \omega_1$. The above equality is also false for any $\kappa \in \{2,3,4,\dots,\omega\}$. The conjecture in [8] is false. The conjecture in [9] is false. The last three results were recently communicated to the author.

The representation (R) is said (here and further) to be κ-fold, if for any $a_1,\dots,a_n \in \mathbb{N}$ the equation $W(a_1,\dots,a_n,x_1,\dots,x_m) = 0$ has less than κ solutions $(x_1,\dots,x_m) \in \mathbb{N}^m$

Theorem 4. *([7, Theorem 2]) Let us consider the following three statements:*

(*a*) *There exists an algorithm $\mathcal{A}$ whose execution always terminates and which takes as input a Diophantine equation D and returns the answer* YES *or* NO *which indicates whether or not the equation D has a solution in non-negative integers, if the solution set $Sol(D)$ satisfies* $\mathrm{card}(Sol(D)) < \kappa$.

(*b*) *The function f_K is majorized by a computable function.*

(*c*) *If a set $\mathcal{M} \subseteq \mathbb{N}^n$ has a κ-fold Diophantine representation, then $\mathcal{M}$ is computable.*

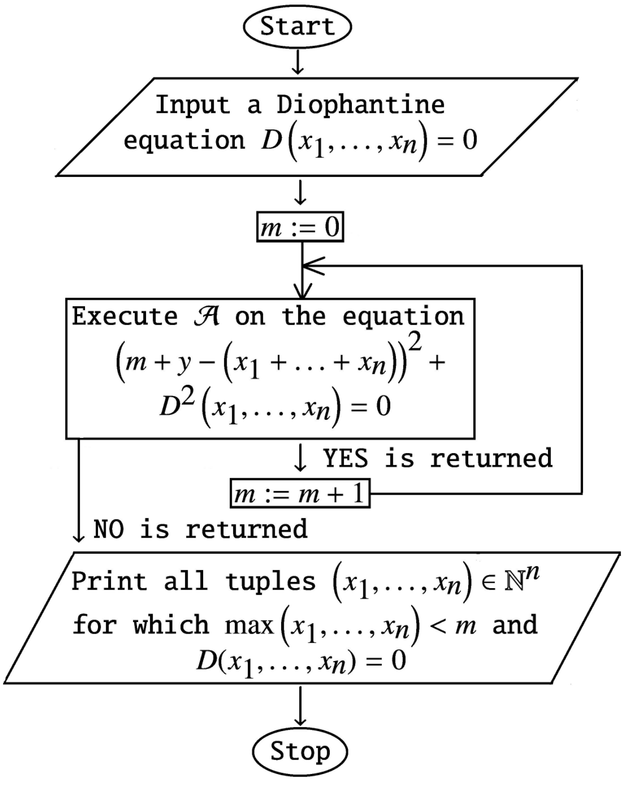
We claim that (*a*) *is equivalent to* (*b*) *and* (*a*) *implies* (*c*).

Proof. The implication $(a) \Rightarrow (c)$ is obvious. We prove the implication $(a) \Rightarrow (b)$. There is an algorithm Dioph which takes as input a positive integer m and a non-empty system $S \subseteq E_m$, and returns a Diophantine equation $\mathrm{Dioph}(m,S)$ which has the same solutions in non-negative integers $x_1,\dots,x_m$. Item (a) implies that for each Diophantine equation D, if the algorithm $\mathcal{A}$ returns YES for D, then D has a solution in non-negative integers. Hence, if the algorithm $\mathcal{A}$ returns YES for $\mathrm{Dioph}(m,S)$, then we can compute the smallest non-negative integer $i(m,S)$ such that $\mathrm{Dioph}(m,S)$ has a solution in non-negative integers not greater than $i(m,S)$. If the algorithm $\mathcal{A}$ returns NO for $\mathrm{Dioph}(m,S)$, then we set $i(m,S) = 0$. The function

$$\mathbb{N} \setminus \{0\} \ni m \to \max\Big\{i(m,S) : \emptyset \neq S \subseteq E_m\Big\} \in \mathbb{N}$$

is computable and majorizes the function f_K. We prove the implication $(b) \Rightarrow (a)$. Let a function h majorizes f_K. By the Lemma for $\boldsymbol{K} = \mathbb{N}$, a Diophantine equation D is equivalent to a system $S \subseteq E_n$. The algorithm $\mathcal{A}$ checks whether or not S has a solution in non-negative integers $x_1,\dots,x_n$ not greater than $h(n)$. □

The implication $(a) \Rightarrow (c)$ remains true with a weak formulation of item (a), where the execution of $\mathcal{A}$ may not terminate or $\mathcal{A}$ may return nothing or something irrelevant, if D has at least κ solutions in non-negative integers. The weakened item (a) implies that the following flowchart



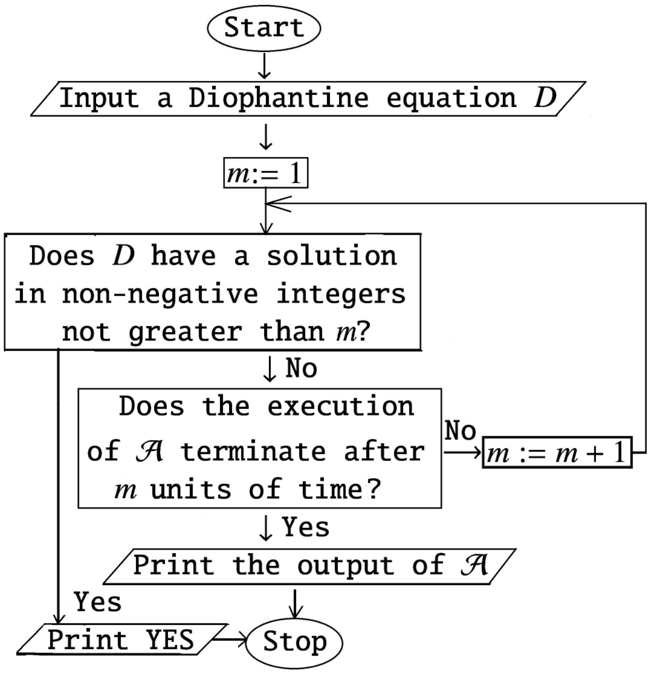
Algorithm 5
An algorithm that conditionally finds all solutions to a Diophantine equation which has less than κ solutions in non-negative integers

describes an algorithm whose execution terminates, if the set

$$Sol(D) := \{(x_1,\dots,x_n) \in \mathbb{N}^n : \; D(x_1,\dots,x_n) = 0\}$$

has less than κ elements. If this condition holds, then the weakened item (a) guarantees that the execution of the flowchart prints all elements of $Sol(D)$. However, the weakened item (a) is equivalent to the original one. Indeed, if the algorithm $\mathcal{A}$

satisfies the weakened item (a), then the flowchart below illustrates a new algorithm $\mathcal{A}$ that satisfies the original item (a).



Algorithm 6
The weakened item (a) implies the original one

The equality $f_{\omega_1} = f$ and Theorem 1 imply that item (b) is false for $\kappa = \omega_1$. By this and Theorem 4, we alternatively obtain a negative solution to Hilbert's Tenth Problem.

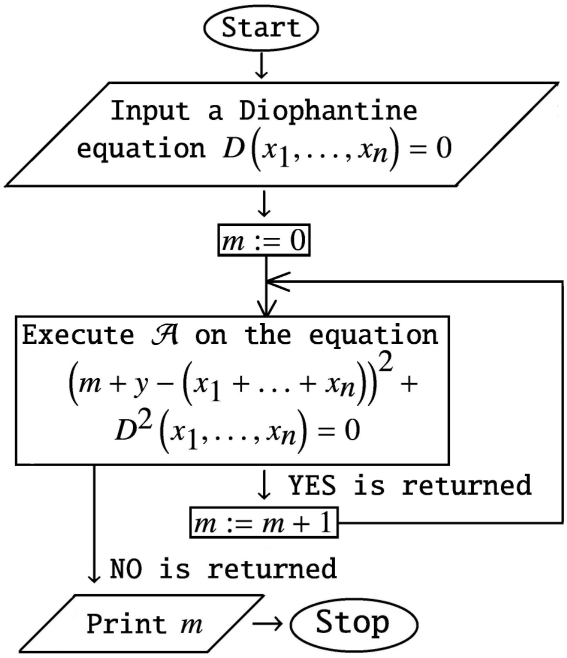
Theorem 5. *([7, Theorem 1]) If a function* $h : \mathbb{N} \setminus \{0\} \to \mathbb{N}$ *has a* κ*-fold Diophantine representation, then there exists a positive integer* m *such that* $h(n) < f_\kappa(n)$ *for any* $n \geq m$.

By the Davis-Putnam-Robinson-Matiyasevich theorem, Theorem 1 is a special case of Theorem 5 when $\kappa = \omega_1$. Let us pose the following two questions:

Question 1. *Is there an algorithm* $\mathcal{B}$ *which takes as input a Diophantine equation D, returns an integer, and this integer is greater than the heights of non-negative integer solutions, if the solution set has less than* κ *elements? We allow a possibility that the execution of* $\mathcal{B}$ *does not terminate or* $\mathcal{B}$ *returns nothing or something irrelevant, if D has at least* κ *solutions in non-negative integers.*

Question 2. *Is there an algorithm C which takes as input a Diophantine equation D, returns an integer, and this integer is greater than the number of non-negative integer solutions, if the solution set is finite? We allow a possibility that the execution of C does not terminate or C returns nothing or something irrelevant, if D has infinitely many solutions in non-negative integers.*

Obviously, a positive answer to Question 1 implies the weakened item (a). Conversely, the weakened item (a) implies that the flowchart below describes an appropriate algorithm $\mathcal{B}$.



Algorithm 7
The weakened item (a) implies a positive answer to Question 1

Theorem 6. *A positive answer to Question 1 for* $\kappa = \omega$ *is equivalent to a positive answer to Question 2.*

Proof. Trivially, a positive answer to Question 1 for $\kappa = \omega$ implies a positive answer to Question 2. Conversely, if a Diophantine equation $D(x_1, \ldots, x_n) = 0$ has only finitely many solutions in non-negative integers, then the number of non-negative integer solutions to the equation

$$D^2(x_1, \ldots, x_n) + (x_1 + \ldots + x_n - y - z)^2 = 0$$

is finite and greater than $\max(a_1, \ldots, a_n)$, where $(a_1, \ldots, a_n) \in \mathbb{N}^n$ is any solution to $D(x_1, \ldots, x_n) = 0$. □

References

[1] H.-D. Ebbinghaus and J. Flum, *Finite model theory,* Springer-Verlag, Berlin, 2006.

[2] A. Janiczak, *Some remarks on partially recursive functions,* Colloquium Math. 3 (1954), 37–38.

[3] Yu. Matiyasevich, *Hilbert's tenth problem,* MIT Press, Cambridge, MA, 1993.

[4] Yu. Matiyasevich, *Towards finite-fold Diophantine representations,* Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI) 377 (2010), 78–90, ftp://ftp.pdmi.ras.ru/pub/publicat/znsl/v377/p078.pdf, http://dx.doi.org/10.1007/s10958-010-0179-4.

[5] R. Murawski, *The contribution of Polish logicians to recursion theory,* in: K. Kijania-Placek and J. Woleński (eds.), *The Lvov-Warsaw School and Contemporary Philosophy,* 265–282, Kluwer Acad. Publ., Dordrecht, 1998.

[6] R. I. Soare, *Interactive computing and relativized computability,* in: *Computability: Turing, Gödel, Church, and beyond* (eds. B. J. Copeland, C. J. Posy, and O. Shagrir), MIT Press, Cambridge, MA, 2013, 203–260.

[7] A. Tyszka, *A condition equivalent to the decidability of Diophantine equations with a finite number of solutions in non-negative integers,* http://arxiv.org/abs/1404.5975.

[8] A. Tyszka, *Conjecturally computable functions which unconditionally do not have any finite-fold Diophantine representation,* Inform. Process. Lett. 113 (2013), no. 19–21, 719–722, http://dx.doi.org/10.1016/j.ipl.2013.07.004.

[9] A. Tyszka, *Does there exist an algorithm which to each Diophantine equation assigns an integer which is greater than the modulus of integer solutions, if these solutions form a finite set?* Fund. Inform. 125(1): 95–99, 2013, http://dx.doi.org/10.3233/FI-2013-854.

[10] A. Tyszka, *Four MuPAD codes,* http://www.cyf-kr.edu.pl/~rttyszka/codes.txt.

[11] A. Tyszka, *Links to an installation file for MuPAD Light,* http://www.ts.mah.se/utbild/ma7005/mupad_light_scilab_253.exe, http://caronte.dma.unive.it/info/materiale/mupad_light_scilab_253.exe, http://www.cyf-kr.edu.pl/~rttyszka/mupad_light_scilab_253.exe, http://www.cyf-kr.edu.pl/~rttyszka/mupad_light_253.exe, http://www.projetos.unijui.edu.br/matematica/amem/mupad/mupad_light_253.exe.