5 Complexity of Arithmetic

At the International Mathematical Congress in Paris (1900), the well-known mathematician David Hilbert put forth a list of 23 research problems for the new century. The 10th of these problems was as follows:

Is there an algorithm that decides for every Diophantine equation whether it has a solution? In other words: is there an algorithm that decides for every polynomial $p(x_1, \ldots, x_n)$ with integer coefficients whether $p(x_1, \ldots, x_n) = 0$ has an integer solution?

Hilbert assumed (as many of his colleagues) that it is only a question of time until such an algorithm is found. People believed at that time that, in principle, it should be possible to decide every arithmetical expression. An arithmetical expression can be any combination of arithmetic and quantified logical expressions. For instance,

$$(\exists y \forall x (x = y + 1)) \Rightarrow (\forall w \forall z (w = z))$$

is an arithmetical expression. In 1931, Kurt Gödel made an abrupt end to this belief by publishing his famous Incompleteness Theorem, which implies that in any axiomatic mathematical system there are propositions that cannot be proved or disproved within the axioms of the system. Or in other words, provability is a weaker notion than truth. We will investigate in this section with the help of the example of arithmetical expressions, why this is the case, and what causes arithmetical expressions to be hard to evaluate. For this we first need some definitions.

An *arithmetical term* can be any one of

(a) a number $n \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$

(b) a variable $x \in \mathbb{N}_0$,

(c) $-T$, where T is an arithmetical term,

(d) $(T_1 + T_2)$, where T_1 and T_2 are terms, or

(e) $(T_1 \cdot T_2)$, where T_1 and T_2 are terms.

Note the similarity with the definition of a Boolean expression. An *arithmetical expression* is any one of

(a) $(T_1 = T_2)$, where T_1 and T_2 are terms,

(b) $\neg\phi_1$, $(\phi_1 \vee \phi_2)$, or $(\phi_1 \wedge \phi_2)$, where ϕ_1 and ϕ_2 are arithmetical expressions, or

(c) $\exists x \phi$ or $\forall x \phi$, where ϕ is an arithmetical expression.

For convenience, we will also use the connectives $\Rightarrow$ and $\Leftrightarrow$.

An arithmetical expression that contains no free variables is called an *arithmetical sentence*. An arithmetical expression that contains no multplication is an *expression of the Presburger arithmetic*. Given a mapping F that assigns natural numbers to the variables of an arithmetical expression ϕ, the truth value of ϕ is inductively defined as follows:

- $\phi = (T_1 = T_2)$ for some terms T_1 and T_2: $F \models \phi$ if $T_1 = T_2$ is true for F

- $\phi = \neg\phi_1$, $\phi = (\phi_1 \vee \phi_2)$, or $\phi = (\phi_1 \wedge \phi_2)$: the same as for Boolean expressions
- $\phi = \exists x \phi_1$ for some arithmetical expression ϕ_1: $F \models \phi$ if $\bigvee_{n \in \mathbb{N}_0} F_{x=n} \models \phi_1$
- $\phi = \forall x \phi_1$ for some arithmetical expression ϕ_1: $F \models \phi$ if $\bigwedge_{n \in \mathbb{N}_0} F_{x=n} \models \phi_1$

Obviously, a necessary prerequisite for a class of problems to be undecidable is that there are problems that have an infinitely large domain. Since $\mathbb{N}$ contains an infinite number of elements, arithmetic expressions fulfill this property. However, is this also a sufficient condition for arithmetical expressions to be undecidable? In order to investigate this question, we will determine the complexity of the following two decision problems:

$$\begin{aligned} ARITH &= \{\langle\phi\rangle \mid \phi \text{ is a valid arithmetical sentence}\} \\ PARITH &= \{\langle\phi\rangle \mid \phi \text{ is a valid sentence of the Presburger arithmetic}\} \end{aligned}$$

First, we need some auxiliary results.

Lemma 5.1 *Let x and y be free variables. There are arithmetical expressions for*

1. *"$x < y$" (without multiplication)*
2. *"$x \leq y$" (without multiplication)*
3. *"x is divisible by p", where p is a constant (with multiplication)*
4. *"x is a power of p", where p is a prime number (with multiplication)*

that all have a constant length.

Proof.

1. $\exists c\,(x + c + 1 = y)$
2. $\exists c\,(x + c = y)$
3. $\exists c\,(c \cdot p = x)$
4. $\forall r \forall s\,((r \cdot s = x) \Rightarrow (r = 1 \vee \text{“}r \text{ is divisable by } p\text{”}))$ (recall that p is a prime!)

□

We will need to describe properties of words over some finite alphabet Σ as arithmetical expressions. For this we will encode these words as numbers. Let $\Sigma = \{c_1, \ldots, c_q\}$ and $p = q+1$. W.l.o.g. we assume that p is a prime (otherwise increase the alphabet by dummy characters). We encode a word $w = w_1 \ldots w_s \in \Sigma^*$ as follows:

$$\begin{aligned} a(w_1, \ldots, w_s) &= \sum_{i=1}^{s} a(w_i) \cdot p^{s-i}, \text{ where} \\ a(c_i) &= i \quad \text{for all } i \in \{1, \ldots, q\}. \end{aligned}$$

Lemma 5.2 *There is an expression of constant length for "x encodes a word over Σ" (that uses multiplication). If the expression is true, the encoded word is called $w(x)$.*

Proof. Obviously, every $x \in \mathbb{N}_0$ has a unique representation of the form $x = \sum_{i=1}^{s} \alpha_i \cdot p^{s-i}$, where $\alpha_i \in \{0, \dots, p-1\}$ (this is actually called the *p-ary representation* of x). Thus, it remains to test whether $\alpha_i \in \{1, \dots, p-1\}$ for all i. This is done as follows:

$$\begin{aligned} \forall c, n, d \quad & [((x = c \cdot n + d) \wedge (d < n) \wedge (c \neq 0) \wedge \text{“}n \text{ is a power of } p\text{”}) \Rightarrow \\ & (\neg \text{“}c \text{ is divisible by } p\text{”})] \end{aligned}$$

The expression is false if and only if there are c, n, d such that $((x = c \cdot n + d) \wedge (d < n) \wedge (c \neq 0) \wedge$ "n is a power of p") is true and ($\neg$ "c is divisible by p") is false. That is, $((x = c \cdot n + d) \wedge (d < n) \wedge (c \neq 0) \wedge$ "n is a power of p" $\wedge$ "c is divisible by p") is true. Suppose that $n = p^j$. Then we must have $\alpha_j = 0$. Since $j < s - 1$ because of $c \neq 0$, x cannot represent a word, which completes the proof. □

Lemma 5.3 *There is an expression of constant length that decides for x, y, and z whether x, y, and z are encodings of words and whether $w(x) = w(y)w(z)$.*

Proof. The following expression ensures the requested property:

$$\begin{aligned} & \text{“}x, y, \text{ and } z \text{ are encodings of words”} \wedge [(x = y \wedge z = 0) \vee (x = z \wedge y = 0) \vee \\ & [\neg(y = 0) \wedge \neg(z = 0) \wedge \exists a, b, c\, ((x = a \cdot b + c) \wedge \text{“}b \text{ is a power of } p\text{”} \wedge (c < b) \wedge \\ & (a = y) \wedge (c = z))]] \end{aligned}$$

□

Lemma 5.4 *There are expressions for*

a) "x encodes some $w \in \{c\}^*$", where $c \in \Sigma$ is fixed

b) "x encodes w", $w \in \Sigma^*$ fixed

c) "x encodes $w_1 w_2 w_3$ with $w_1, w_3 \in \{c\}^*$", where $c \in \Sigma$ and $w_2 \in \Sigma^*$ are fixed

d) "x encodes some $w \in (\Sigma - \{c\})^*$", where $c \in \Sigma$ is fixed

a) and d) have constant length, and b) and c) have a length of O(binary length of $a(w)$).

Proof. Assignment. □

The main result of this section is the following theorem.

Theorem 5.5 *ARITH is undecidable.*

Proof. Recall that the problem

$$L_a = \{\langle M\rangle x \mid M \text{ is a single-tape TM and } M \text{ started with } x \text{ accepts}\}$$

is not decidable. We will show that $L_a \leq ARITH$ by constructing for every Turing machine M and input x an arithmetical sentence $\phi(M, x)$ that is true if and only if M started with x accepts.

Let now $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a single-tape Turing machine and x be an input. W.l.o.g. it holds that

- $Q = \{q_0, \ldots, q_f\}$
- $F = \{q_f\}$
- if M halts, then M converts before that all tape symbols into blanks

Let $A = Q \cup \Gamma \cup \{/\}$ be the alphabet used for the encoding of a computation, where $/$ is not an element of Q or Γ. In order to construct an arithmetic sentence that is true if and only if M started with x halts, we have to encode again configurations and transitions between configurations. In contrast to the theorems by Cook and Meyer and Stockmeyer, configurations can have here an arbitrary length. However, we can also use here arithmetical instead of just Boolean expressions.

Lemma 5.6 *There are arithmetical expressions for*

a) "ℓ encodes a configuration" (we represent here configurations as $w_1 q w_2$ instead of (q, w_1, w_2), and the configuration encoded by ℓ is called $C(\ell)$)

b) "ℓ encodes a sequence of configurations $C_1/C_2/\ldots/C_T$" (for an arbitrary T)

c) "$C(\ell) \to C(\ell')$"

The expressions have a length that is independent of $|x|$.

Proof. a) and b) are assigments.
Proof of c): under the assumption that $C(d) = w_1 p\, w_2$ with $|w_1|, |w_2| > 0$ we obtain

$$\begin{aligned}
\text{“}C(d) \to C(e)\text{”} \quad \equiv \quad & \exists u \exists v \bigvee_{\delta(p,x)=(q,y,N)} (\text{“}C(d) = w(u)\, px\, w(v)\text{”} \wedge \text{“}C(e) = w(u)\, qy\, w(v)\text{”}) \vee \\
& \bigvee_{\delta(p,x)=(q,y,R)} (\text{“}C(d) = w(u)\, px\, w(v)\text{”} \wedge \text{“}C(e) = w(u)\, yq\, w(v)\text{”}) \vee \\
& \bigvee_{\delta(p,x)=(q,y,L)} \bigvee_{c \in \Gamma} (\text{“}C(d) = w(u)\, cpx\, w(v)\text{”} \wedge \text{“}C(e) = w(u)\, qcy\, w(v)\text{”})
\end{aligned}$$

This can be easily extended to the case that $|w_1|$ or $|w_2|$ is 0. □

Now we can construct an arithmetic sentence that is true if and only if M started with x halts.

"M started with x accepts"

$$
\begin{array}{lll}
\Leftrightarrow & \exists \ell(\text{“}\ell \text{ encodes a sequence of configurations } C_1/\ldots/C_T\text{”} & \\
& \wedge \forall i \in \{1,\ldots,T\} \text{ “}C_i \text{ is a configuration”} & \\
& \wedge \forall i \in \{1,\ldots T-1\} \text{ “}C_i \to C_{i+1}\text{”} & \\
& \wedge \text{ “}C_1 = q_0 x\text{”} & \\
& \wedge \text{ “}C_T = q_f\text{”} & \\
\Leftrightarrow & \exists \ell(\text{“}\ell \text{ encodes a sequence of configurations”} & \text{L. 5.6 b)} \\
& \wedge \forall c \forall d \forall e \forall f [(\text{ (“}w(\ell) = w(c)/w(d)/w(e)/w(f)\text{”} \vee & \text{L. 5.3} \\
& \text{“}w(\ell) = w(d)/w(e)/w(f)\text{”} \vee \text{“}w(\ell) = w(c)/w(d)/w(e)\text{”} & \\
& \vee \text{ “}w(\ell) = w(d)/w(e)\text{”}) & \\
& \wedge \text{ “}w(d) \text{ and } w(e) \text{ are configurations”}) \Rightarrow \text{“}C(d) \to C(e)\text{”}] & \text{L. 5.6 a),c)} \\
& \wedge \exists a \exists b (\text{“}w(\ell) = w(a)/w(b)\text{”} \wedge \text{“}w(a) = q_0 x\text{”}) & \text{L. 5.3, 5.4} \\
& \wedge \exists g \exists h (\text{ “}w(\ell) = w(g)/w(h)\text{”} \wedge \text{“}w(h) = q_f\text{” }) \,) & \text{L. 5.4, 5.4}
\end{array}
$$

Since this is an arithmetical sentence, the proof is completed. □

Matijasevič showed in 1970 undecidability also in the special case of the 10. Hilbert problem.

A *Diophantine equation* is an expression of the form "$T = 0$", where T is an arithmetic term. A Diophantine equation is *solvable* if the arithmetical sentence

$$\exists x_1 \exists x_2 \ldots \exists x_n (T = 0)$$

is valid. Let

$$SDE = \{\langle \phi \rangle \mid \phi \text{ is a solvable Diophantine equation}\} \,.$$

Theorem 5.7 (Matijasevič) *SDE is undecidable.*

One can show that if we restrict a Diophantine equation to have no multiplication, it can be decided efficiently whether or not the Diophantine equation has a nonnegative integer solution.

In general, it can be shown that within the Presburger arithmetic it is decidable whether arithmetic sentences are valid. In particular, one can show the following result.

Theorem 5.8

a) $PARITH \in \mathrm{DTIME}(2^{2^{2^{2pn\log n}}})$ for some constant $p > 0$.

b) $PARITH$ is $\mathrm{NTIME}(2^{2^{cn}})$-complete for some $c > 0$.

We only prove the PSPACE-hardness here.

Theorem 5.9 *$PARITH$ is PSPACE-hard.*

Proof. Let L be any language in PSPACE. Then there is a deterministic single-tape Turing machine M that decides L in polynomial space. Now, recall the construction of a quantified Boolean expression in the proof of Theorem 4.9 that is valid if and only if M started on input x accepts. W.l.o.g. we assume that it is in conjunctive normal form. Simply replace in it now

- all Boolean variables x_i by variables $x'_i \in \mathbb{N}_0$ with the additional requirement that $(x'_i = 0) \vee (x'_i = 1)$,
- all occurances of $\neg x'_i$ by $(1 - x'_i)$, and
- all expressions $\bigvee_i \ell_i$ with $\ell_i \in \{x'_i, (1 - x'_i)\}$ by $\neg(\sum_i \ell_i = 0)$.

Then we obtain an equivalent arithmetical expression that does not use multiplication. Hence, $PARITH$ is PSPACE-hard. $\square$

5.1 References

- Y.V. Matijasevič. Enumerable sets are Diophantine. *Soviet. Math. Dokl.* 11, pp. 354–357, 1970.
- B.M. Moret. *The Theory of Computation.* Addison Wesley, Reading, 1998.
- C.H. Papadimitriou. *Computational Complexity.* Addison Wesley, Reading, 1994.
- J.E. Savage. *Models of Computation – Exploring the Power of Computing.* Addison Wesley, Reading, 1998.
- M. Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.