CONJUNCTIVE GRAMMARS AND EQUATIONS OVER SETS OF NATURAL NUMBERS

ARTUR JEŻ

Abstract

This paper presents a short description of the PhD thesis of the author [4]. For shortness reasons, almost all technical part and proofs are omitted. Since the goal is to accessibly present the main results and give some intuition of the proof techniques, some of the details and subtleties are removed on purpose.

The thesis is devoted to the study of systems of equations

$$\varphi_i(X_1, \dots, X_n) = \psi_i(X_1, \dots, X_n) \quad \text{for } i = 1, \dots m \ ,$$

in which the variables $X_1, \dots, X_n$ represent sets of natural numbers. The allowed operations are union, intersection and addition, which is defined as

$$X + Y = \{x + y \mid x \in X,\ y \in Y\} \ .$$

Such systems can be equally interpreted as systems of language equations over a single-letter alphabet and operations of union, intersection and concatenation.

The study begins with considering the subclass of systems of equations over sets of numbers, consisting of systems of the *resolved form*, i.e.,

$$X_i = \varphi_i(X_1, \dots, X_n) \quad \text{for } i = 1, \dots, n \ .$$

The counterparts of such systems among the language equations are the resolved systems of language equations over a single-letter alphabet. These can be also viewed as appropriate grammars: when the allowed operations are union and concatenation, they correspond to context free grammars; when also intersection is allowed, to the conjunctive grammars.

It is shown that the resolved systems of equations over sets of natural numbers can have non-ultimately periodic sets as the least solutions. Equivalently, conjunctive grammars over a single-letter alphabet can generate non-regular languages, as opposed to context-free grammars. This claim is firstly demonstrated by giving a simple system, which has the least solution with $\{4^n : n \in \mathbb{N}\}$ as its first component. This system exploits the properties of the base-4 positional notation of sets, in particular, the aforementioned set should be viewed as the set of numbers with base-4 notation $\mathtt{10}^\ell$, for some $\ell \geq 0$.

Then, this example is generalised. For every set S of numbers, such that base-k positional notations of numbers from S are recognised by a certain type of a real-time cellular automaton, an explicit construction of a resolved system with S as the first component of the least solution is given.

Next the systems with no restrictions on the form of the equations imposed are investigated. They are shown to be computationally universal: the class of unique solutions of such systems coincides with the class of recursive sets. Similar characterisations are shown for the class of least (greatest) solutions: this class coincides with the class of recursively enumerable sets (co-recursively enumerable sets, respectively; these sets are called r.e. and co-r.e. from now on). These results hold even when only union and addition (or only intersection and addition) are allowed in the system.

Systems with addition as the only allowed operation are also considered, it is shown that the obtained class of sets is computationally universal, in the sense that their unique (least, greatest) solutions can represent encodings of all recursive (r.e., co-r.e., respectively) sets.

The computational complexity of decision problems for both formalisms is investigated. The membership problem for the resolved systems of equations is EXPTIME-complete. Many other decision problems for both types of systems are undecidable, and their exact undecidability level is settled. Most of these results hold even when the systems are restricted to the use of one equation with one variable.

1. Introduction

Systems of equations over sets of natural numbers are among the simplest and most intuitive mathematical formalisms. Each equation is of the form

$$\varphi(X_1, \ldots, X_n) = \psi(X_1, \ldots, X_n) \ ,$$

where expressions φ and ψ may use Boolean operations and arithmetical operations, applied pairwise to elements of sets, for example:

$$A + B = \{a + b \mid a \in A,\ b \in B\} \ .$$

In the thesis, equations with only union and intersection as the allowed Boolean operations and with addition as the only arithmetical operation are studied.

Consider the following example of an equation with an unknown $X \subseteq \mathbb{N}$, with the operations of addition and union, and with two singleton constant languages.

$$X = (X + \{2\}) \cup \{0\} \ .$$

This equation defines even numbers inductively, by stating that 0 is an even number, and that an even number plus two is an even number as well. Formally, the equation has a unique solution, which is the set of all even numbers.

Another example of a univariate equation

$$X + \{1\} = (X + X) \cup \{2\}$$

no longer defines an explicit induction, as none of its sides is X. This equation has multiple solutions, among them the solutions $X = \{1\}$ and $X = \{1, 2, \ldots\}$. Furthermore, every solution of this equation must contain 1 and be contained in $\{1, 2, \ldots\}$, which makes these two solutions the *least* and the *greatest solutions* of this equation.

As already shown by the above examples, an equation over sets of numbers may have a unique solution or multiple solutions, or sometimes no solutions at all, as in the example $X = X + \{1\}$. There is a common outlook on an equation as a *formalism for defining its solutions*. An equation with a unique solution may be seen as a definition of this solution, and in case of multiple solutions, least or greatest solutions (provided that they exist, as in the second example above) may be similarly considered.

All the examples considered so far define ultimately periodic sets, and in fact, every ultimately periodic set can be easily defined by a unique solution of an equation with finite constants. However, when one tries to write any such system of equations, its solution somehow tends to be ultimately periodic, and there seems not to be an easy way to avoid this. Prior to the thesis [4], only one example of a non-periodic set defined by an equation over sets of numbers has been known [6], and no general method of constructing any further examples.

In the thesis [4] a surprising result is shown: the class of unique solutions of equations of the general form is exactly the class of recursive sets, that is, the

sets with an effectively decidable membership. For equations in the resolved form $X = \varphi(X)$, the computational power of their solutions is limited, yet still much exceeds the class of ultimately periodic sets. Such results belong to the domain of the theory of computations, and the methods used to establish them come from formal language theory.

1.1. **Language equations.** Language equations are equations with formal languages as unknowns, their most general form can be given as

$$\begin{cases} \varphi_1(X_1, \ldots, X_n) &= \psi_1(X_1, \ldots, X_n) \\ & \vdots \\ \varphi_m(X_1, \ldots, X_n) &= \psi_m(X_1, \ldots, X_n) \ , \end{cases}$$

where each φ_i, ψ_i consist of variables, constants from some given class, and operations on languages, such as concatenation, union, intersection, complementation, and possibly others. While language equations are an old formalism, in the recent years they have attracted new attention.

Language equations have several useful properties. Their semantics is intuitively clear and can be mathematically well formalised. One can trim or extend them in many ways to obtain different classes of languages or to adapt them to a particular application. Such changes do not affect the way the semantics is given. Moreover, grammars and automata can be translated to systems of language equations. While this does not introduce any theoretical or practical novelty on its own, it often eases the proofs and formalisation and provides an elegant common generalisation.

Despite all the motivation, language equations are far away from being fully, or even partially, understood. The quest for their better understanding is crucial. Thus even a partial progress in a simple subcase is important, as it might eventually help in dealing with more general cases.

1.1.1. *General language equations.* When no restriction on the form of language equations is imposed, they possess great computational power: Okhotin determined that the family of sets representable by unique (least, greatest) solutions of such equations is exactly the family of recursive languages (r.e., co-r.e., respectively). In the general case, a solution is a vector of languages $(L_1, \ldots, L_n)$, and it is greater than another solution $(K_1, \ldots, K_n)$, if $K_i \subseteq L_i$ for each i-th coordinates. The least (greatest) solution is the one that is less (greater, respectively) than any other solution. Though such extremal solutions do not always exist, when they do, they are computationally complete:

Theorem 1 (Okhotin [9, 7])**.** *Consider a system of language equations using union, intersection, concatenation and recursive constants that has a unique (least, greatest) solution $(L_1, \ldots, L_n)$. Then each component L_i is recursive (r.e., co-r.e., respectively).*

Conversely, for every recursive (r.e., co-r.e.) language $L \subseteq \Sigma^*$ (with $|\Sigma| \geq 2$) there exists a system using concatenation, union and singleton constants (concatenation, intersection and singleton constants) with the unique (least, greatest, respectively) solution $(L, \ldots)$.

It is worth to mention that the lower bound demonstrated in the theorem essentially uses the fact that $|\Sigma| \geqslant 2$. On the other hand, the upper bound applies to the case of a unary alphabet.

1.1.2. *Resolved language equations.* Among all language equations a particularly useful subclass consists of *resolved language equations*, i.e., the one of the form:

$$\begin{cases} X_1 &= \varphi_1(X_1, \ldots, X_n) \\ & \vdots \\ X_n &= \varphi_n(X_1, \ldots, X_n) \ . \end{cases}$$

Such systems possess a nice property: if the expressions φ_i are monotone, the least solution exists.

Another property was discovered by Ginsburg and Rice [2], who gave a semantics of context-free grammars by using resolved language equations with concatenation and union as the only operations. In fact, this semantics is a simple (syntactic) transformation between context-free grammars and resolved language equations using union and concatenation. Such a transformation supports the claim that there is a strong connection between resolved language equations and grammars.

This connection was further explored by Okhotin, who defined *conjunctive grammars*. They can be viewed as an extension of context-free grammars by a possibility of an unrestricted intersection in the body of every rule. The particular importance of conjunctive grammars among other extensions of context-free grammars lies not in the possibility of the intersection itself, but in the semantics. It can be given not only in terms of string rewriting, but also in terms of language equations. In fact this is the (unique) semantics generalising the one of Ginsburg and Rice to the language equations using union, intersection and concatenation; this is the major justification both for conjunctive grammars and for resolved systems of language equations.

It should be noted that the conjunctive grammars inherit almost all good parsing properties of context-free grammars, while having much larger expressive power, stretching beyond the finite intersection of context-free languages.

1.2. **Systems of equations over sets of numbers.** Consider language equations over a single letter alphabet. By a trivial isomorphism $a^n \longleftrightarrow n$, unary languages can be regarded as sets of natural numbers, and language equations over a single-letter alphabet become *equations over sets of numbers*. Concatenation of languages accordingly turns into *addition* (or *sum*) of sets

$$S + T = \{m + n \mid m \in S,\ n \in T\} \ .$$

Such equations constitute a basic mathematical object on its own, the study of which can be traced up to the seminal paper of Stockmeyer and Meyer [11], who studied *integer expressions*, i.e., regular expressions (though with complementation) over unary languages. These expression were generalised to *integer circuits*, that is expressions which may share the subexpressions; recursive calls are still not allowed though. As such expressions (and circuits) define only finite sets, they were studied mainly with complexity of their decision problems in mind, while their expressive power was not investigated, as there was virtually nothing to investigate.

Equations over sets of numbers are a more general formalism, and its expressive power seems to be related to the allowed Boolean operations. For example, it is relatively easy to see that for resolved equations using union and addition, only ultimately periodic sets can be obtained (in terms of formal languages: unary context-free languages are regular).

Constructing any system of equations, even an unresolved one, with a non-periodic unique solution has proved to be a non-trivial task; the first example of such an equation using the operations of concatenation and complementation was presented by Leiss [6].

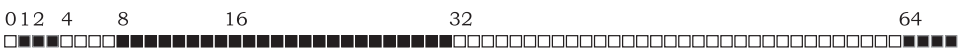


FIGURE 1. The Leiss construction depicted graphically. The black boxes represent the numbers from the unique solution, while the white ones represent the numbers outside the solution.

Example 1 (Leiss [6])**.** For every expression φ, denote $2\varphi = \varphi + \varphi$. Then the set $\{n | \text{base-8 notation of } n \text{ starts with } \mathtt{1}, \mathtt{2} \text{ or } \mathtt{3}\}$ is the unique solution of the equation

$$X = 2\Big(\overline{2\big(\overline{2\overline{X}}\big)}\Big) + \{1\} \ .$$

Naturally, it was asked, whether unary conjunctive grammars generate non-regular languages, or equivalently, whether there exists a non-periodic least solution of resolved system of equations over sets of numbers using union, intersection and addition.

This question remained open for some time and was considered as one of the most important in the field of conjunctive grammars, until a simple construction for a non-regular language $\{a^{4^n} \mid n \in \mathbb{N}\}$ was demonstrated by the author [5]; the generalisations of this construction is the cornerstone of the thesis.

1.3. **Preliminaries and notation.** In this section, the basic terminology and the general facts about language equations and equations over sets of natural numbers are stated.

1.3.1. *Systems of Equations and their Solutions.* A *system of equations of equations over sets of numbers* is a system of the form

$$\begin{cases} \varphi_1(X_1, \ldots, X_n) &= \psi_1(X_1, \ldots, X_n) \\ & \vdots \\ \varphi_m(X_1, \ldots, X_n) &= \psi_m(X_1, \ldots, X_n) \ , \end{cases}$$

where the unknowns X_i are subsets of $\mathbb{N}$, while φ_j and ψ_j are expressions using Boolean operations and addition, as well as some constants. A *solution* of such a system is a vector $(S_1, \ldots, S_n)$, such that substituting $X_i = S_i$ for $i = 1, \ldots, n$ in the system turns each equation into an equality.

When no other description is given, *systems of equations* are assumed (by default), to be over sets of natural numbers and to use operations $\cap$, $\cup$ and $+$, the allowed constants are only singletons. If some considerations apply to different systems, these systems will be explicitly characterised.

To simplify the notation, the following precedence of operations is assumed: addition has the highest precedence, followed by the intersection and with the union having the least precedence.

There is a natural partial order on the sets of solution of a given system. A solution $(S_1, \ldots, S_n)$ is greater than $(S'_1, \ldots, S'_n)$, denoted as $(S_1, \ldots, S_n) \supseteq (S_1, \ldots, S_n)$ if for each $i = 1, \ldots, n$ it holds that $S_i \supseteq S'_i$. Whenever we refer to a *least* or *greatest* solution, this is with respect to this order.

1.3.2. *Least solutions of resolved systems.* With every resolved system of equations of the form

$$(X_1, \ldots, X_n) = \varphi(X_1, \ldots, X_n) \ ,$$

an operator $\varphi : \left[2^{\mathbb{N}}\right]^n \mapsto \left[2^{\mathbb{N}}\right]^n$ can be associated. Then every solution is a fixpoint of such an operator and vice-versa, in particular, the least fixpoint is the least solution of the system.

By the Tarski's Fixpoint Theorem, each monotone operator has the least fixpoint. Still, this does not give any construction of this fixpoint. An older Kleene's Fixpoint Theorem can be used instead. While it has stronger assumptions on the operator, it can be routinely checked that they are satisfied in the case of systems of equations.

Proposition 1. *The least fixpoint of a monotone and $\cup$-continuous operator is given by*

$$\bigcup_{i=0}^{\infty} \varphi^i(\varnothing, \dots, \varnothing) \ .$$

1.3.3. *Representing numbers in positional notation.* While all the equations defined in the thesis deal with numbers as they are, the principal outlook on these numbers considers their base-k positional notation, for a suitable k. For example, the very first non-trivial example of a least solution of a resolved system of equations defines the set $\{4^n \mid n \in \mathbb{N}\}$, i.e., the set of all numbers that, written in base-4 notation, consist of $\mathtt{1}$ followed by some zeroes. For this reason, a special notation for sets of natural numbers is employed. Fix a base $k \geq 2$ and define the alphabet $\Sigma_k = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \dots, k-1\}$ of k-ary digits.

Every string $w = d_{n-1} \dots d_1 d_0$ with $d_i \in \Sigma_k$ represents the following number:

$$(w)_k = (d_{n-1} \dots d_1 d_0)_k = \sum_{i=0}^{n-1} d_i \cdot k^i \ .$$

Accordingly, every language $L \subseteq \Sigma_k^*$ defines a certain set of numbers:

$$(L)_k = \left\{ \ (w)_k \ \middle| \ w \in L \ \right\} \ .$$

1.3.4. *Simplifications.* The technical lemmata presented in this paper are sometimes simplified when compared to the full statements available in the thesis [4]. This can mean that some additional properties or assumptions are not listed, slightly more complicated construction are in fact needed, etc. This is done in order to streamline the presentation and give the idea more clearly. In particular, some of the lemmata may not be true as they are written here, though they resemble lemmata from the thesis. Still, all the statements of the theorems are as in the thesis and exactly these statements are proved there.

2. Toy example

D. Wotschke asked the question of the expressive power of conjunctive grammars over a single letter alphabet. This question was suppose to settle, whether conjunctive grammars possess more expressive power than context-free grammars even in the simplest case of a single-letter alphabet. It was conjectured that only regular languages can be generated by unary conjunctive grammars [8], similarly to the context-free grammars. In the terms of systems of equations, it was conjectured that the resolved systems of equations have only ultimately periodic least solutions. The contrary claim is shown: a construction of a resolved system of equations with non-periodic least solution is presented.

Theorem 2. *The least solution of the system*

$$\left\{ \begin{array}{rcl} X_1 & = & X_2{+}X_2 \ \cap \ X_1{+}X_3 \cup \{1\} \\ X_2 & = & X_{12}{+}X_2 \cap X_1{+}X_1 \cup \{2\} \\ X_3 & = & X_{12}{+}X_{12} \cap X_1{+}X_2 \cup \{3\} \\ X_{12} & = & X_3{+}X_3 \cap X_1{+}X_2 \ , \end{array} \right. \tag{1}$$

is $((\mathtt{10}^*)_4, (\mathtt{20}^*)_4, (\mathtt{30}^*)_4, (\mathtt{120}^*)_4)$.

This is the unique solution of this system in sets of positive integers.

Proof. To see that this vector is indeed a solution, let us substitute the given values into the equations. The proof follows by a simple principle: each sum of two sets consists of numbers with a few non-zero digits (in base-4 notation). Intersection of two such sum filters out numbers with unwanted base-4 notation.

The first sum can be transformed as follows:

$$X_2 + X_2 = (\mathtt{20}^*)_4 + (\mathtt{20}^*)_4 = (\mathtt{10}^+)_4 \cup (\mathtt{20}^*\mathtt{20}^*)_4 \ ,$$

and similarly the second sum equals

$$X_1 + X_3 = (\mathtt{10}^*)_4 + (\mathtt{30}^*)_4 = (\mathtt{10}^+)_4 \cup (\mathtt{10}^*\mathtt{30}^*)_4 \cup (\mathtt{30}^*\mathtt{10}^*)_4 \ .$$

Then their intersection is

$$(X_2 + X_2) \cap (X_1 + X_3) = (\mathtt{10}^+)_4 \ ,$$

and after taking union with $\{1\}$, exactly $(\mathtt{10}^*)_4 = X_1$ is obtained. The equations for variables X_{20}, X_{30} and X_{12} are verified in a similar way.

It remains to show that the given solution is the least solution in the sets of natural numbers. This follows from the general form of the right-hand sides of this system, in which all occurrences of variables are in sums of two variables, and no constant set contains 0; this is only a matter of a technique, which generalises the notion of *proper system* [1]. □

3. Expressive power of resolved systems

The example presented in Section 2 shows that systems of (resolved) equations over sets of natural numbers become substantially richer in their expressive power, when the intersection is also an allowed operation. Recall, that the construction effectively manipulates the leading digits of the numbers. It is natural to ask, how far this method can be extended, i.e., what sets can be constructed in this way. In this section it is shown that the method can be greatly generalised. The goal is obtained in stages: in each stage the previously defined sets are used as constants, in order to ease the construction of richer class of sets. This makes the construction much more modular and thus easier to both develop and follow.

3.1. Expressing $(ij\mathtt{0}^*)_k$.

The example from Section 2 is naturally extended to the sets of the form $(ij\mathtt{0}^*)_k$ for each $i \in \Sigma_k \setminus \{\mathtt{0}\}$, $j \in \Sigma_k$. The constructed system of equations are variations of the (1) and the proof follows the same principle as the proof of Theorem 2.

Theorem 3. *Let $k \geq 2$ be a natural number. There exists a resolved system of equations in variables $\{X_{i,j}\}$ for $i \in \Sigma_k \setminus \{0\}$, $j \in \Sigma_k$ such that its unique solution in the subset of positive numbers is $X_{i,j} = (ij\mathtt{0}^*)_k$ for each i, j.*

This solution is the least solution of this system in the natural numbers.

As an example, consider the equation which defines $(ij\mathtt{0}^*)_k$ for $i > 1$, $j > 2$ and $k \geq 9$:

$$X_{i,j} = \bigcap_{n=1}^{2} X_{i-1,k-n} + X_{j+n,0} \cup (ij)_k \ .$$

3.2. Regular notation.

In order to construct a larger class of sets the idea behind the construction has to be changed a bit. Until now, in the sets $X_{i,j} = (ij\mathtt{0}^*)_k$ the digits i, j were crucial, while the strings of 0's after them were just a gadget of the construction. For the new sets it works the other way around—they are of the form $X_{i,j,L} = (\{ijw \mid w \in L\})_k$ for some language L. So now i, j are just technical gadgets, which are used in the construction to manipulate the leading digits, while the important information is stored in the rest of the digits.

The construction uses several different variables, which represent different sets L and different two leading digits. One should imagine that for a fixed ℓ and varying i, j variables $X_{i,j,\ell}$ correspond to one language L_ℓ. Moreover, the technique, if properly generalised, allows expanding the notations of numbers by a fixed digit. So the intuition is that languages $\{L_\ell\}_\ell$ should express themselves using other languages from this group of languages and left-concatenation with a digit. This is almost the formal definition of the class of regular languages.

Consider an arbitrary regular language $L \subseteq \Sigma_k^+ \setminus 0\Sigma_k^*$. Let $M = \langle \Sigma_k, Q, \delta, F, q_0\rangle$ be a (deterministic) automaton recognizing L^r. A resolved system of equations with the least solution $X_{i,j,q} = \{(ijw)_k \mid \delta(q_0, w^r) = q\}$, for $i, j \in \Sigma_k$, $i > 0$ and $q \in Q$ will be presented. Then constructing a resolved equation with the least solution $(L^r)_k$ easily follows. The reversal of the word in the definition of $X_{i,j,q}$ is for technical reasons—while the automata read the word from the left to the right, the constructed system expands the numbers by digits to the left, i.e., by the digit read first by the automaton.

Lemma 1. *There exists a resolved system in variables $X_{i,j,q}$, $i, j, \ell \in \Sigma_k$, $i \neq 0$ and $q \in Q$ such that its unique solution in positive natural numbers is $X_{i,j,q} = \{(ijw)_k \mid \delta(q_0, w^r) = q\}$.*

This is also the least solution in the natural numbers.

As an example of such an equation, consider the one for $j < 4$ and $i > 1$ and $k \geq 9$.

$$X_{i,j,q} = \bigcup_{\substack{(\ell,q'):\\ \delta(q',\ell)=q}} \bigcap_{n=1}^{4} X_{k-n,\ell,q'} + ((i-1)(j+n)0^*)_k \cup (\{ij \mid q_0 = q\})_k \ .$$

The strings are generated recursively: the representation for $w = \varepsilon$ is given explicitly, otherwise $w = w'\ell$ and so it may be assumed that the representation for w' were already generated. Assume that w leads to a state q and w' to q'. Then

$$(ijw^r)_k = (ij\ell w'^r)_k = ((k-n)\ell w'^r)_k + ((i-1)(j+n)0^{|w|})_k \in X_{k-n,\ell,q'} + (i-1, j+n0^*)_k \ ,$$

for each $n \in \{1, \ldots, 4\}$However, the latter sums of sets may contain also some unwanted results. Using several such representations and intersecting them allows eliminating the unwanted sums, as in Theorem 2.

After establishing Lemma 1 the general representation theorem follows by taking a union of appropriate sets constructed in Lemma 1.

Theorem 4. *For every natural $k > 1$ and every regular language $L \subseteq \Sigma_k^*$ the set of numbers $(L)_k$ can be represented as a unique solution of a resolved system of equations.*

This solution is the least solution of this system in the natural numbers.

It should be pointed out, that after representing sets of numbers with a regular positional notation, the task of constructing equations with specified least solutions becomes substantially easier, as using constants with a regular positional notation is allowed. Generally speaking, this allows modifying a specified digit in numbers from the input set.

3.3. **Sets with trellis representation.** There is a natural attempt to extend the used method—since the notations of numbers are expanded to the left by a given digit, maybe the same can be done with the ending digit?

In order to enforce this approach, a different representation of a word is needed: in the previous construction, a single word w was represented by a number $(ijw)_k$, in particular one word was represented by one number. Now a word w is represented

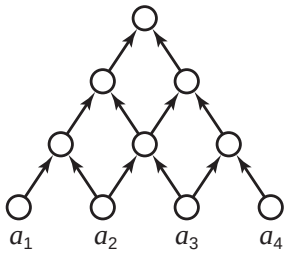
by the numbers $(\mathtt{1}w\mathtt{1000}\ldots\mathtt{0})_k$, i.e., by the numbers with their base-k notation consisting of $\mathtt{1}w\mathtt{1}$ followed by a series of $\mathtt{0}$'es. The two additional $\mathtt{1}$'s are 'sentinels', as they mark the beginning and the end of the word w—note that w may contain both leading and ending $\mathtt{0}$'es. Then going from w to, say, $w\mathtt{3}$ is just changing the $\mathtt{10}$ in the end into $\mathtt{31}$. Moreover, since it is not known in advance, how many ending $\mathtt{0}$'es will be consumed, w is represented as $(\mathtt{1}w\mathtt{10}^*)_k$, i.e., one word is represented by an infinite set of numbers.

Now one needs to understand what type of computational device can be simulated in this way. On a high level, it is an automaton with a finite state control and the state calculated on a word awb depends on the states calculated on aw and on wb. Such a class of automata is well-known—these are *trellis automata*. Quite surprisingly, the class of *trellis languages* already is connected with the conjunctive grammars: it coincides with the class of linear conjunctive languages, which are a strict subclass of conjunctive languages.

A *trellis automaton* processes an input string of length $n \geq 1$ using a uniform triangular array of $\frac{n(n+1)}{2}$ processor nodes. Each node computes a value from a fixed finite set Q. The nodes in the bottom row obtain their values directly from the input symbols using a function $I : \Sigma \to Q$. The rest of the nodes compute the function $\delta : Q \times Q \to Q$ of the values in their predecessors. The string is accepted if and only if the value computed by the topmost node belongs to the set of accepting states $F \subseteq Q$. This is formalised in the below definition.

Definition 1. A trellis automaton is a quintuple $M = (\Sigma, Q, I, \delta, F)$, in which:

- Σ is the input alphabet,
- Q is a finite non-empty set of states,
- $I : \Sigma \to Q$ is a function setting the initial states,
- $\delta : Q \times Q \to Q$ is the transition function,
- $F \subseteq Q$ is the set of final states.



Extend δ to a function $\delta : Q^+ \to Q$ by $\delta(q) = q$ and $\delta(q_1, \ldots, q_n) = \delta(\delta(q_1, \ldots, q_{n-1}), \delta(q_2, \ldots, q_n))$, while I is extended to a homomorphism $I : \Sigma^* \to Q^*$. Let $L_M(q) = \{w \mid \delta(I(w)) = q\}$ and define $L(M) = \bigcup_{q \in F} L_M(q)$.

A brief explanation how the informal intuition is changed into a specific encoding should be given. When devising the representation for word w some possible obstacles should be taken taken into account:

- while $L(M) \cap \mathtt{0}\Sigma_k^* = \varnothing$ can be assumed, and hence in the end $w \in L(M)$ does not start with $\mathtt{0}$, its substring can start with $\mathtt{0}$. Thus a way of marking the number of leading $\mathtt{0}$'es is needed;
- similar question arises for w's ending $\mathtt{0}$'es—they should be distinguishable from $\mathtt{0}$'es added in the end.

The natural approach is to add *sentinels* on both sides of the word, i.e., to represent w as $(\mathtt{1}w\mathtt{10}^*)_k$. This solves the two mentioned problems. Still it creates another: resolved equations are 'monotone' in the sense that if $(w)_k \in \varphi(\{(w')_k\})$ then $(w)_k \geq (w')_k$. The current encoding does not guarantee such property. This is answered in the thesis by representing w not by $(\mathtt{1}w\mathtt{10}^*)_k$ directly, but by $(\mathtt{1}w'\mathtt{10}^*)_k$, where w' represents a number smaller by 1 than w, i.e., $(w')_k + 1 = (w)_k$. This modified representation is properly explained and verified in the thesis, however, this technical detail makes it harder to understand the general idea, therefore we present a construction in a simplified way, which does not work, however is easier to comprehend.

For a given trellis automaton $M = (\Sigma_k, Q, I, \delta, F)$, define a system of equations over sets of numbers with the set of variables X_q for $q \in Q$, the desired solution of this variable is

$$S_q = (1L_M(q)10^*)_k = \{(1w10^\ell)_k \mid \ell \geq 0, w \in L_M(q)\} \ .$$

The main idea of the construction is to represent numbers corresponding to a string $iwj \in L_M(q)$, with $i, j \in \Sigma_k$ and $w \in \Sigma_k^*$, through numbers corresponding to the strings iw and wj. Consider that iwj belongs to $L_M(q)$ if and only if there are states q', q'' with $\delta(q', q'') = q$, $iw \in L_M(q')$ and $wj \in L_M(q'')$. In terms of the encodings, the number $(1iwj10^\ell)_k$ should belong to X_q if and only if there are states q', q'' with $\delta(q', q'') = q$, $(1iw10^{\ell+1})_k \in X_{q'}$ and $(1wj10^\ell)_k \in X_{q''}$.

Two expressions: λ_i and ρ_j for $i, j \in \Sigma_k$ depending on the variables X_q are devised. The purpose of λ_i is to take a number of the form $(1wj10^\ell)_k$ and append the digit i to the left of the encoded string obtaining a number $(1iwj10^\ell)_k$. Similarly, ρ_j starts with a number $(1iw10^{\ell+1})_k$ and appends j to the right of the string, also obtaining $(1iwj10^\ell)_k$. This is implemented by adding digits at some specific positions, so that selected digits in the original number (at the left and at the right of the encoding, respectively, whence the letters λ and ρ come from) could be modified in the resulting number, while the rest of the digits remain the same. This is to be done by adding a number of the form $((j-i)0^\ell)_k$, where ℓ is the position in which digit i is to be replaced by digit $j > i$. Below, we present an example of these expression for $i, j \geq 2$.

$$\lambda_i(X) = \bigcup_{i' \in \Sigma_k} \big((X \cap (1i'\Sigma_k^*10^*)_k) + (10^*)_k \cap (2i'\Sigma_k^*)_k\big) + (1(i-2)0^*)_k \cap (1i\Sigma_k^*)_k \ ,$$

$$\rho_j(X) = \bigcup_{j' \in \Sigma_k} \big((X \cap (1\Sigma_k^*j'10^*)_k) + (10^*)_k \cap (1\Sigma_k^*j'20^*)_k\big) + (1(j-2)10^*)_k \cap (1\Sigma_k^*j10^*)_k \ .$$

For example, the desired action of λ_i on a number $(1w10^\ell)_k$ is as follows: first it represents w as $w = i'w'$. Then it adds $(10^{|w|+1+\ell})_k$ to $(1i'w'10^\ell)_k$, obtaining $(2i'w'10^\ell)_k$. As a next operation, $(1(i-2)0^{|w|+1+\ell})_k$ is added, and the result is $(1ii'w'10^\ell)_k = (1iw10^\ell)_k$, as desired. The additional intersections in the equations assure that in fact everything goes as intended.

Using the functions λ_i and ρ_j, the resolved system of equations can be succinctly represented:

$$X_q = R_q \ \cup \bigcup_{\substack{q', q'' \colon \delta(q', q'') = q \\ i, j \in \Sigma_k}} \lambda_i(X_{q''}) \cap \rho_j(X_{q'}) \ ,$$

where R_q is the set of numbers which cannot be represented using the recursion. Intuition behind this equations is as follows: the sets R_q contain the starting part of S_q representing elements of $L_M(q)$ of a very simple form; it can be shown that R_q is regular. All other numbers are constructed by simulating the transition of M, as described earlier.

To obtain the main set out of this construction it is needed to extract the number $(w)_k$ out of the set $(1w10^*)_k$. This is done by fairly natural digit manipulation.

Theorem 5. *For every $k \geq 2$ and for every trellis automaton M over $\Sigma_k = \{0, \ldots, k-1\}$, such that $L(M) \cap 0\Sigma_k^* = \varnothing$, there exists and can be effectively constructed a resolved system of equations such that its least solution contains a component*

$$(L(M))_k = \{n \mid k\text{-ary notation of } n \text{ is in } L(M)\} \ .$$

This is the unique solution in sets of positive natural numbers.

4. Unresolved equations over sets of natural numbers

So far, only the resolved system of equations were considered. Now the attention is turned to the general (i.e., unresolved) case of systems of equations. As noted, systems of equations over sets of numbers can be viewed as systems of language equations over a unary alphabet. Thus the upper bound of Theorem 1 can be applied to systems of equations as they are, i.e., each unique (least, greatest) solution of such system is recursive (r.e., co-r.e., respectively). The main result presented in this section is a matching lower bound.

Theorem 6. *The family of sets of natural numbers representable by unique (least, greatest) solutions of systems of equations of the form* $\varphi_i(X_1, \ldots, X_n) = \psi_i(X_1, \ldots, X_n)$ *with union and addition, is exactly the family of recursive (r.e., co-r.e., respectively) sets. The same result holds for systems with intersection and addition.*

Then it is investigated, what happens if none Boolean operation is allowed, i.e., that addition is the only used operation. It is shown that the solutions are indeed trivial, when only finite constants are allowed; however, when ultimately-periodic can also be used, the least (greatest) solutions are r.e.-complete (co-r.e.-complete).

4.1. **Sketch of proof of Theorem 1.** A short sketch of the proof of lower bound of Theorem 1 is given to present its general idea, the proof of Theorem 6 is modelled on it. The main technical device used in the construction is the language of transcription of computation of a Turing machine. In short, for every TM T over an input alphabet Γ one can construct an alphabet $\Sigma \supseteq \Gamma$ and an encoding of computations $C_T : \Gamma^* \to \Sigma^*$, so that for every $w \in L(T)$ the string $C_T(w)$ lists the configurations of T on each step of its accepting computation on w. The language

$$\mathrm{VALC}(T) = \{C_T(w) \natural w \mid C_T(w) \text{ is an accepting computation}\} \ ,$$

where $\natural \notin \Sigma$, is an intersection of two linear context-free languages. Since unresolved equations can directly simulate context-free grammars and are equipped with intersection, for every Turing machine it is relatively easy to construct a system in variables $(X_1, \ldots, X_n)$ with a unique solution $(L_1, \ldots, L_n)$, so that $L_1 = \mathrm{VALC}(T)$.

It remains to 'extract' $L(T)$ out of $\mathrm{VALC}(T)$ using a language equation. Let Y be a new variable and consider the inequality

$$\mathrm{VALC}(T) \subseteq \Sigma^* \natural Y \ , \tag{2}$$

which can be rewritten as an equation $X_1 \cup \Sigma^* \natural Y = \Sigma^* \natural Y$ or an equation $X_1 \cap \Sigma^* \natural Y = X_1$. This inequality states that for every $w \in L(T)$, the string $C_T(w) \natural w$ should be in $\Sigma^* \natural Y$, that is, w should be in Y. This makes $L(T)$ the least solution of this inequality and proves the second part of Theorem 1 with respect to r.e. sets and least solutions. The construction for a co-r.e. set and a greatest solution is established by a dual argument, and these two constructions can be then combined to represent every recursive set.

4.2. **Systems of equations with one Boolean operation.** The construction of the lower bound from Theorem 1 is 'arithmetised': each word is reinterpreted as a number and language operations are simulated by operations on sets of numbers. This yields a basis of the proof of Theorem 6. For starters, treat the used alphabet Σ as a set of digits: simply assume that $\Sigma = \Sigma_6$. There is nothing specific about base-6, simply this is the smallest number for which the construction is easy to present. Moreover, the used encoding $C_T(w)$ needs to be of more specific form. This does not affect the fact that $\mathrm{VALC}(T)$ is an intersection of linear CFL's, as CFG's are flexible enough to express various encodings of $\mathrm{VALC}(T)$.

Trellis languages are known to properly contain linear CFL's and are closed under all Boolean operations, in particular, VALC(T) is a trellis language. Therefore $(\mathrm{VALC}(T))_6$ is a unique solution of a resolved system of equations.

The next step is to simulate (2). The crucial property needed here is that there is a set Q such that if $x \in Q$ and $u \in \Sigma_6^+$ satisfy $xu \in QY$, where Y is a variable, then $u \in Y$. This essentially means that the partition of a string xu into strings in Q and S is unique. Thus the equation $\mathrm{VALC}(T) \subseteq QY$ extracts $Y = L(T)$ from VALC(T), as each string in VALC(T) is partitioned in to $C_T(w) \in Q$ and $w \in Y$.

In our setting, this is simulated by the following lemma

Lemma 2. *Let $S \subseteq (\Sigma_6^+)_6$. There exists a set Q such that for all $(x)_6 \in Q$ and $(u)_6 \in (\Sigma_6^+)_6$, if*

$$(x)_6 + (u)_6 \in Q + Y \ ,$$

then $(u)_6 \in Y$.

While a little different in formulation, this is a needed reformulation of (2). Again, this lemma is stripped out of some technical details, which are taken care of in the thesis [4].

Using Lemma 2, the equation $\mathrm{VALC}(T) \subseteq Q + Y$ has the least solution $Y = (L(T))_6$. This gives a skeleton of the proof of Theorem 6.

The main idea of the encoding of set Q in the lemma above is that it the VALC(T) encodes the transcription of TM in binary using `300` for 1 and `30` for 0, the whole transcription ends with `1`. Then Q contains $(\{\mathtt{30}, \mathtt{300}\}^*\mathtt{10}^*)_6$. Then every number in $Q + Y$ can be uniquely represented as a sum $(x)_6 + (u)_6$.

However, the given sketch of the proof uses both $\cup$ and $\cap$ in the construction: these operations were in fact used at the very beginning, in the resolved systems that generated $(\mathrm{VALC}(T))_6$. To obtain the full proof of Theorem 6 this construction needs to be redesigned, so that they use only one Boolean operation.

To simplify the process, a general translation lemma is given. It is used to automate the redesigning process: as an input, it gets a resolved system (using union, intersection and addition), as an output it gives an unresolved system using addition and either intersection or union; the least solution of the input system is the unique solution of the output system. The lemma is applied to the systems constructed in Section 3. Note, that some modifications are needed during the process, due to strong assumptions of the lemma.

Lemma 3. *Let $X_i = \varphi_i(X_1, \ldots, X_n)$ be a resolved system of equations with union, intersection and addition and with finite constants. Let $(S_1, \ldots, S_n)$ be its least solution. Assume that for every variable X_{i_0} there exists a subset of variables $\{X_i\}_{i \in I}$ containing X_{i_0}, such that*

- *the sets $\{S_i\}_{i \in I}$ are pairwise disjoint and their union is a unique solution of an unresolved system using addition and union, and*
- *the equations for all $\{X_i\}_{i \in I}$ are either all of the form $X_i = \bigcup_j \alpha_{ij}$, or all of the form $X_i = \bigcap_j \alpha_{ij}$, where $\alpha_{ij} = A_1 + \ldots + A_\ell$, with $\ell \geq 1$ and with each A_t being a constant or a variable.*

Then there exists an unresolved system with union and addition and finite constants, which has the unique solution $(S_1, \ldots, S_n)$.

A similar construction that returns a system of equations using only intersection and addition can be given.

The promised equations look as follows: each equation $X_i = \bigcap_j \alpha_{ij}$, where α_{ij} is a sum of constants and variables, is replaced by the following collection of inequalities:

$$X_i \subseteq \alpha_{ij}, \quad \text{for all } j \ .$$

In addition, for each group of variables $\{X_i\}_{i\in I}$, whose union of the group is C_I, the following equation is added:

$$\bigcup_{i\in I} X_i = C_I \ . \tag{3}$$

It is easy to verify, that the least solution of the resolved system is a solution of the system constructed in Lemma 3. The technical part is showing that in fact this is the unique solution. The idea is that each other solution has some 'errors' with respect to the desired solutions and these errors propagates among variables: if the original equation was $X = Y \cup Z$ or $X = Y + Z$ and X has some error, at least one of Y, Z has an error. The problematic case is equation $X = Y \cap Z$ replaced by $X \subseteq Y$ and $X \subseteq Z$: note, that we may freely remove some number from X and keep the values of Y, Z as they were. However, this is what (3) is used for: the error propagate from X to some other variable using this new equation. It can be shown that we cannot propagate the errors ad infinitum, which makes the least solution of the original system the unique solution of the new system.

The last task is to apply Lemma 3 to resolved systems constructed in the proofs of Theorems 4 and 5. For the lemma to be applicable, these equations need to be decomposed into smaller parts and slightly changed. There are a few details to check, however, this is relatively straightforward and so not commented here. Then the variables can be grouped into subsets, as required by the lemma.

This ends the sketch of proof of Theorem 6.

4.3. **Systems with addition only.** Further restrictions on the computational model are now discussed, that is, systems of equations with addition only are considered. When ultimately periodic constants are allowed, such restricted systems are computationally universal.

The idea is to take an arbitrary system using addition and union and *encode* it in another system using addition only. The solutions of the two systems are not identical, but there is a bijection between solutions based upon an encoding of sets of numbers.

All constants in the construction are ultimately periodic; some of them are finite and some are infinite. The last question is whether infinite constants are necessary to specify any non-periodic sets, and an affirmative answer is given.

The general idea of the construction is as follows. Define an abstract encoding, that is an injective function

$$\sigma\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}} \ .$$

There are two goals to be acquired by this encoding. Firstly, it should be *verifiable by an equation*, meaning, that there is an equation such that X satisfies it if and only if $X = \sigma(\widehat{X})$ for some $\widehat{X}$. Secondly, this encoding should allow *simulating operations*, i.e., for union (addition, respectively) there is an equation such that $X = \sigma(\widehat{X})$, $Y = \sigma(\widehat{Y})$ and $Z = \sigma(\widehat{Z})$ satisfy this equation if and only if $\widehat{X} = \widehat{Y} \cup \widehat{Z}$ ($\widehat{X} = \widehat{Y} + \widehat{Z}$, respectively).

An arbitrary set of numbers $\widehat{S} \subseteq \mathbb{N}$ will be represented by another set $S \subseteq \mathbb{N}$, which contains a number $16n + 13$ if and only if n is in $\widehat{S}$. The membership of numbers i with $i \neq 13 \pmod{16}$ in S does not depend on $\widehat{S}$.

Definition 2. For each $i \in \{0, 1, \ldots, 15\}$,

$$\textsc{track}_i(S) = \{n \mid 16n + i \in S\}, \quad \tau_i(S') = \{16n + i \mid n \in S'\} \ .$$

The subset $S \cap \{16n + i \mid n \geq 0\}$ is called the i^{th} *track of* S. A set S is said to have an *empty (full) track* i if $\textsc{track}_i(S) = \varnothing$ ($\textsc{track}_i(S) = \mathbb{N}$, respectively).

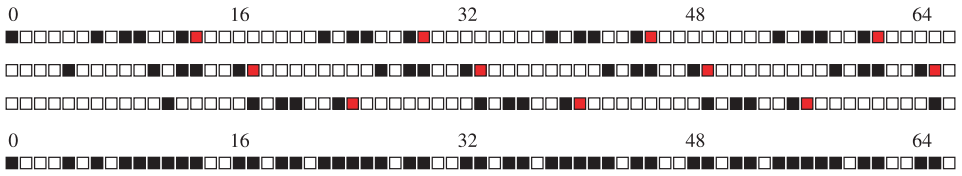


FIGURE 2. The addition of $\sigma(\widehat{X}) + \{0, 4, 11\}$. The following rows represent $\sigma(\widehat{X})$, $\sigma(\widehat{X}) + \{4\}$, $\sigma(\widehat{X}) + \{11\}$ and finally $\sigma(\widehat{X}) + \{0, 4, 11\}$. It can be seen that the last sum has only black and white cells.

In these terms a set $\widehat{S}$ shall be encoded in the 13^{th} track of a set S. The rest of the tracks of S contain technical information needed for the below constructions to work: track 0 contains a singleton $\{0\}$, tracks 6, 8, 9 and 12 are full and the rest of the tracks are empty.

Definition 3. For every set $\widehat{S} \subseteq \mathbb{N}$, its *encoding* is the set

$$S = \sigma(\widehat{S}) = \{0\} \cup \tau_6(\mathbb{N}) \cup \tau_8(\mathbb{N}) \cup \tau_9(\mathbb{N}) \cup \tau_{12}(\mathbb{N}) \cup \tau_{13}(\widehat{S}) \ .$$

The first property of the encoding is that there exists an equation with the set of all valid encodings as its set of solutions, see Fig. 2 for an illustration.

Lemma 4. *A set $X \subseteq \mathbb{N}$ satisfies an equation*

$$X + \{0, 4, 11\} = \bigcup_{\substack{i \in \{0,4,6,8,9,\\ 10,12,13\}}} \tau_i(\mathbb{N}) \cup \bigcup_{i \in \{1,3,7\}} \tau_i(\mathbb{N} + 1) \cup \{11\}$$

if and only if $X = \sigma(\widehat{X})$ for some $\widehat{X} \subseteq \mathbb{N}$.

The idea of the proof is that if a track t in $X + \{0, 4, 11\}$ is empty, then it can be inferred that for each $t' \in \{0, 4, 11\}$ and t'' such that $t' + t'' = t \mod 16$ the track t'' of X is empty as well. This shows that many tracks of X are empty. Similar considerations apply if the track t contains a singleton. On the other hand, the equations are constructed in a way that if track t of $X + \{0, 4, 11\}$ is full then there exists exactly one pair $t' \in \{0, 4, 11\}$ and t'' such that $t' + t'' = t \mod 16$ and track t'' of X is non-empty. Thus the t''-th track of X is in fact full.

The second property of the encoding σ is that a sum of encodings of two sets and a fixed constant set simulates the union of these two sets, while the addition of a different fixed constant set simulates the sum of the two original sets.

Lemma 5. *For all sets $X, Y, Z \subseteq \mathbb{N}$,*

$$\sigma(Y) + \sigma(Z) + \{0, 1\} = \sigma(X) + \sigma(\{0\}) + \{0, 1\} \text{ if and only if } Y + Z = X \ ,$$
$$\sigma(Y) + \sigma(Z) + \{0, 2\} = \sigma(X) + \sigma(X) + \{0, 2\} \text{ if and only if } Y \cup Z = X \ .$$

The addition of $\sigma(Y) + \sigma(Z)$ should be viewed as follows: first, choose a track t from $\sigma(Y)$ and t' from $\sigma(Z)$. Add the sets encoded on these tracks and encode it on the track $t + t' \mod 16$ of the result. Then take the union over all choices of t, t'. In particular, adding a 0-th track of $\sigma(Y)$, which contains a singleton, to a 13-th track of $\sigma(Z)$, which encodes Z, gives Z encoded on 13-th track; by symmetry, addition of a track 0 from $\sigma(Z)$, which encodes singleton, and a 13-th track of $\sigma(Y)$, which encodes Y, gives Y on track 13. It can be checked that no other information is encoded on this track and thus the 13-th track of $\sigma(Y) + \sigma(Z)$ encodes $Y \cup Z$.

On the other hand, when 13-th tracks of $\sigma(Y)$ and $\sigma(Z)$, which encode Y and Z, respectively, are added, their sum $X + Y$ is encoded on $13 + 13 \mod 16 = 10$-th track.

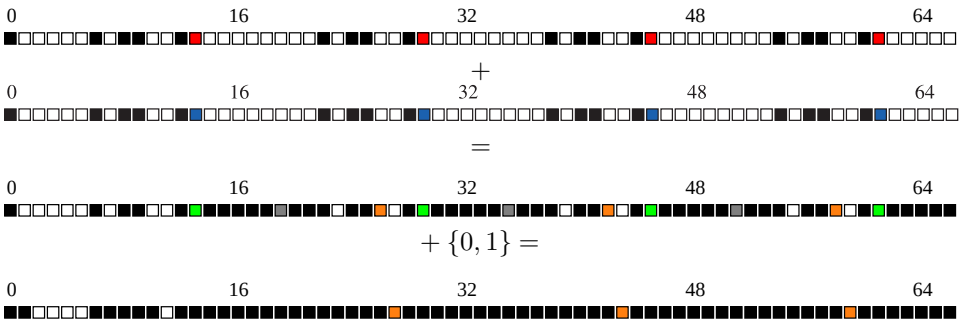


FIGURE 3. The graphical illustration of addition $\sigma(Y)+\sigma(Z)+\{0,1\}$. The first row represents $\sigma(Y)$, with red cells encoding the elements of Y. The second row represents $\sigma(Z)$, with blue cells encoding the elements of Z. In the sum $\sigma(Y)+\sigma(Z)$ the green cells are the encoded elements of $Y \cup Z$, the orange cells are the encoded elements of $\sigma(Y)+\sigma(Z)$ while the grey cells are some auxiliary 'junk' numbers. Then the addition of $\{0,1\}$ leaves only encoded numbers from $\sigma(Y)+\sigma(Z)$ and the numbers not depending on Y nor Z.

Thus the sum $\sigma(Y)+\sigma(Z)$ encodes both $Y \cup Z$ and $Y + Z$ on its tracks. It is left to modify $\sigma(Y)+\sigma(Z)$ so that it contains exactly one of these sets. It turns out that adding appropriate constant allows to 'spam' one of these tracks with a full track, while preserving the other.

Using the encoding defined above, it is now possible to represent a system with union and addition by a system with addition only.

Theorem 7. *For every recursive (r.e., co-r.e.) set $S \subseteq \mathbb{N}$ there exists a system of equations*

$$\begin{cases} \varphi_1(X_1,\ldots,X_n) & = & \psi_1(X_1,\ldots,X_n) \\ & \vdots & \\ \varphi_m(X_1,\ldots,X_n) & = & \psi_m(X_1,\ldots,X_n) \end{cases},$$

with φ_j, ψ_j using the operation of addition and ultimately periodic constants, which has a unique (least, greatest, respectively) solution with $X_1 = T$, where $S = \{n | 16n + 13 \in T\}$.

To construct the system it is enough to decompose the input system into simple equations and translate each of them using Lemma 5. Then a copy of the equation from Lemma 4 is added for each variable.

The constructions above essentially use infinite ultimately periodic constants. It is shown that the use of such constants is necessary, and systems using only addition and *finite* constants cannot specify any non-trivial infinite sets: every solution $(\ldots, S, \ldots)$ of such a system can be *pruned* in the sense that each of its infinite components can be replaced by an empty set and the resulting vector remains a solution. In a similar way, infinite components of a solution can be augmented to co-finite sets.

Lemma 6. *If a system of equations in variables $(\ldots, X_j, \ldots, Y_i, \ldots)$ using addition and only finite constants has a solution $(\ldots, F_j, \ldots, S_i, \ldots)$, where each F_j is finite and each S_i infinite, then $(\ldots, F_j, \ldots, \varnothing, \ldots)$ and $(\ldots, F_j, \ldots, S_i + \mathbb{N}, \ldots)$ are solutions of this system as well.*

Using this lemma, the following theorem is obtained.

Theorem 8. *If a system of equations using addition and finite constants has a least (greatest, unique) solution $(\dots, S_i, \dots)$, then each S_i is finite (finite or co-finite, finite, respectively).*

5. Univariate Equations

It is quite natural to ask, how many variables are needed to obtain any non-trivial solution using systems of equations, both the resolved and unresolved ones. Quite surprisingly, Okhotin and Rondogiannis [10] gave a construction of a univariate resolved equation with a non-periodic solution. In this section their example is generalised to a representability theorem: for every resolved system of equations the sets assigned to its variables can be encoded together in a single set that is a solution of a univariate resolved equation over set of numbers. The techniques used are not specific to resolved equations and are therefore applied to unresolved equations, and similar results for them are obtained.

5.1. **Resolved univariate equation.** The goal is to simulate an arbitrary resolved system by a univariate one. The intended encoding of a single set S_i is $S'_i = \{pn - d_i \mid n \in S_i\}$, for some properly chosen numbers p and $d_1, \dots, d_m$; all these sets are encoded in one by taking a union over i: $S = \bigcup_{i=1}^{m} S'_i$. This encoding use the idea of tracks, similarly to the previous section: for a set S its subset of the form $S \cap \{pn - t \mid n \in \mathbb{N}\}$ will be called its t^{th} *track*, where t is the *offset* of the track. Thus, the set S encodes S_i on the i-th track.

Let us shortly compare this encoding with the one used in Section 4.3. First of all, the offset is of opposite sign. Secondly, no tracks are filled with gadgets. Moreover, several tracks of the single variable are used to encode many variables of the original resolved system, while in the previous section only one track per variable was meaningful.

First, we describe the transformation of the system with a least solution $(S_1, \dots, S_m)$ into a system with a solution $(S'_1, \dots, S'_m)$. Consider the equation $X_\ell = \varphi_\ell(X_1, \dots, X_n)$. To obtain the new equation φ'_ℓ out of φ_ℓ, it is enough to replace each constant $\{n\}$ in φ_ℓ by $\{np - d_\ell\}$ and each addition $X_i + X_j$ by $X_i + X_j + (d_i + d_j - d_\ell)$. Indeed, when S'_i and S'_j are substituted into the new equation, the value of the addition is

$$\begin{aligned}(4)\quad &\{np - d_i \mid n \in S_i\} + \{np - d_i \mid n \in S_j\} + (d_i + d_j - d_\ell) = \\ &\qquad = \{np + n'p - d_\ell \mid n \in S_i,\ n' \in S_j\} = \{np - d_\ell \mid n \in S_i + S_j\}\ .\end{aligned}$$

By easy calculations, similar values are obtained for the Boolean operations, and in the end it is obtained that $\varphi'_\ell(S'_1, \dots, S'_n)$ has value

$$\{np - d_\ell \mid n \in \varphi(S_1, \dots, S_n)\} = \{np - d_\ell \mid n \in S_\ell\} = S'_\ell\ .$$

To obtain a single equation using φ'_ℓ's, it is 'enough' to take $X = \bigcup_{\ell=1}^{m} \varphi'_\ell(X, \dots, X)$. However, in this way the contents of the tracks may mix: it needs to be assured that (4) holds, when S'_i and S'_j are both replaced by S. In general, this is not true. However, it is possible to extract the contents of a track: we modify the original system of equations, so that whenever an addition $X_i + X_j$ appears in it, it is in fact in a subequation $X_i + X_j \cap X_{i'} + X_{j'}$. When the offsets d_i, d_j, $d_{i'}$ and $d_{j'}$ are chosen properly, it holds that

$$\begin{aligned}&S + S + (d_i + d_j - d_\ell) \cap S + S + (d_{i'} + d_{j'} - d_\ell) = \\ &\qquad = S'_i + S'_j + (d_i + d_j - d_\ell) \cap S'_{i'} + S'_{j'} + (d_{i'} + d_{j'} - d_\ell)\ ,\end{aligned}$$

i.e., the addition works as in (4). Having established this basic property, it can be shown by a trivial induction that the input system is properly simulated by the constructed resolved equation.

Theorem 9. *For every resolved system of equations in variables $X_1, \ldots, X_m$ there exist numbers $0 < d_1 < \ldots < d_{m'} < p$ depending only on m and an equation $X = \varphi(X)$ with the least solution $S = \{np - d_i \mid 1 \leq i \leq m',\ n \in S_i\}$, where $(S_1, \ldots, S_m)$ is the least solution of the original system and S_{m+1}, ..., $S_{m'}$ are functionally dependent on $(S_1, \ldots, S_m)$.*

5.2. **General univariate equation.** It was shown that a resolved system of equations can be encoded in one resolved equation. Now a similar result is shown for general (that is, unresolved) systems of equations. The first goal is to replicate Theorem 6 using a unique equation with a unique variable. This is achieved by taking an arbitrary system of equations with solutions of the form $X_1 = S_1, \ldots, X_m = S_m$, and constructing an equation whose solutions is the set $S = \bigcup_{i=1}^{m} \{pn - d_i \mid n \in S_i\}$, for some properly chosen numbers p and $d_1, \ldots, d_m$, as in Section 5.1.

The encoding of multiple sets on tracks of one set requires constructing a new equation out of the old system. This equation extracts the tracks contents out of a single variable and performs the operations on them, avoiding mixing the contents of several tracks. Also it checks that all tracks of the variable that should be empty according to the encoding are indeed empty, which ensures a bijection between the solutions of the original system and the solutions of the constructed equation.

Theorem 10. *For every system of equations $E_j(Y_1, \ldots, Y_m) = F_j(Y_1, \ldots, Y_m)$ with $j \in \{1, \ldots, \ell\}$, where each expression E_j and F_j is of the form $Y \cup Y'$ or $Y \cap Y'$ or $\{1\}$, there exist numbers $0 \leq d_1 < \ldots < d_m < p$ and an equation $\varphi(X) = \psi(X)$ using singleton constants, such that a set $S \subseteq \mathbb{N}$ is its solution if and only if*

$$S = \{kp, kp + \tfrac{p}{2} + 1 \mid k \geq 0\} \cup \bigcup_{i=1}^{m} \{kp - d_i \mid k \in S_i\}$$

for some solution $(S_1, \ldots, S_m)$ of the original system.

Accordingly, solutions of the resulting equation encode solutions of the original equation as follows. The numbers d_i are offsets of tracks for X: each set S_i is represented in a track $S \cap \{kp - d_i \mid k \geq 1\}$. For each variable Y_i, a unique offset d_i is assigned. The rest of the tracks should be empty.

The set $\varphi(S) = \psi(S)$ is as well split into tracks of its own, which are not directly related to the tracks of S. The tracks of $\varphi(S) = \psi(S)$ correspond to equations of the original system. A track $\{kp - e_j \mid k \geq 1\}$ with a unique offset e_j is assigned to an equation number j.

First, a simplified version of the theorem is explained, in which also a usage of an infinite ultimate periodic constant $\{pn \mid n \in \mathbb{N}\}$ is allowed, note, that the constants $\{pn - d \mid n \in \mathbb{N}\}$ can be expressed using it. These constants are used to extract the track contents of a variable X. To encode one of the original equations of the system, say j-th, first all of the used sets are extracted and, by an addition of a singleton, moved to the e_j-th track, which corresponds to this equation. Then the equation is rewritten, taking into the account that the sets are written on tracks. For example, the equation number j of the form $X_i \cup X_{i'} = \{1, 2\}$ is transformed into

$$\Big(\big(X \cap \{pn - d_i \mid n \in \mathbb{N}\}\big) + \{d_i - e_j\}\Big) \cup \Big(\big(X \cap \{pn - d_{i'} \mid n \in \mathbb{N}\} +\big) + \{d_{i'} - e_j\}\Big) = \{p - e_j, 2p - e_j\} \ .$$

In order to show the full statement of the theorem it is needed to eliminate the usage of the constant $\{np \mid n \in \mathbb{N}\}$. To this end, a copy of such constant is also encoded in the solution S. Properly devised equation allows making sure that

	equiv. to a ult. periodic set	equivalence	finiteness	co-finiteness
general	Π_1-complete	Π_1-complete	$\Sigma_1 \leq \cdot \leq \Sigma_2$	$\Sigma_1 \leq \cdot \leq \Sigma_2$
univariate	decidable	Π_1-complete	Σ_1-complete	Σ_1-complete

TABLE 1. Decision problems for resolved systems of equations over sets of numbers.

	satisfiability	unique satisfiability	finite satisfiability
general	Π_1-complete	Π_2-complete	Σ_3-complete
univariate	Π_1-complete	Π_2-complete	Σ_3-complete

TABLE 2. Decision problems for general systems of equations over sets of numbers.

indeed one track is full. It is only left to merge the equation simulating the system and the equation generating the constant.

6. Decision Problems

Each formalism defining sets of numbers or languages is judged from two main perspectives. On one hand, its expressive power is measured, i.e., how complicated and useful sets can be defined. This was addressed already in the previous sections. On the other hand, one wants to know the difficulty of the decision problems for such a formalism, or, informally speaking, how hard is to check the properties of a given instance of a formalism. It is natural to expect some trade-off between those two.

In this section we discuss the decision problems for both unresolved and resolved systems of equations. All usual problems, such as emptiness, equivalence, membership etc., are considered. It is natural to expect that in all cases majority of the decision problems are hard, as these systems define a rich class of sets. It is the case —most of the results presented here will establish the exact level of undecidability, or at least narrow down the position of the problem in the arithmetical hierarchy.

The complexity of decision problems usually decrease when some restrictions on the systems are imposed. This restrictions can be, for example, limiting the number of the equations or variables. Contrary to this intuition, it is shown that almost all considered problems are equally hard for a single equation with a single variable.

6.1. **Properties of the Solutions.** Since the resolved systems of equations can simulate trellis automata and unresolved equations can simulate Turing Machines, it is no wonder that hardness of their decision problems can be naturally inferred. Tables 1 and 2 summarise the findings. The results are proved in a relatively simple way, therefore no sketches are provided. The equivalence, finiteness and co-finiteness problems for unresolved equations have the same complexity as the corresponding problems for resolved equations.

6.2. **Membership problem.** Each formalism defining subsets of natural numbers is made with one goal in mind—to describe sets. As such, the most important question that can be asked about it, is the complexity of its membership problem. Since the unresolved equations can specify all r.e. sets as least solutions, their membership problem is hard. The situation is different for resolved equations: in this section its complexity is studied: it is shown that this problem is EXPTIME-complete.

Stockmeyer and Meyer [11] established that the membership problem for expressions with union and addition is NP-complete. The problem remains in NP when

systems of equations with union and addition is considered, as shown by Huynh [3]. Note, that in both of these cases the obtained sets are regular and the NP-hardness happens only for the general membership problem, i.e., if the system is part of the input.

It is expected that the membership problem for resolved systems of equations with union, intersection and addition should be harder, as it allows a larger set of operations. This intuition is also supported by the fact that resolved systems with these operations define much larger class of sets than resolved systems with addition and union only, i.e., the class of ultimately periodic sets.

In this section a construction of a fixed resolved system is given, such that testing the membership of numbers in its least solution is an EXPTIME-hard problem, with the numbers given in binary notation. The result is obtained by an arithmetisation of an alternating linear-space Turing Machine (ATM). It is relatively easy to show that the membership problem for resolved equations is in EXPTIME, even for the general membership problem, which makes the constructed set computationally hardest in its class.

Theorem 11. *The family of sets of numbers representable by resolved systems of equations with union, intersection and addition, as well as singleton constants, is a subset of EXPTIME and contains an EXPTIME-complete language.*

An EXPTIME-complete language can be generated even by a single equation using one variable.

The proof is by constructing a system of equations that encodes a computation of any linearly bounded ATM. It is known that such machines recognise some EXPTIME-complete sets.

The ATM operates on a *circular tape* and move to the right at every step. Such a machine can simulate an arbitrary linear-bounded ATM by marking the position of its head on the tape, and by making one transition of the simulated ATM.

The construction of a system of equations over sets of numbers simulating a computation is based upon representing instantaneous descriptions of the ATM as numbers considered in some positional notation. The entire argument is based upon mapping the symbols used by the machine to digits, and then using addition to manipulate individual digits in the positional notation of numbers. As usually, this positional notation is only a tool for understanding the constructions, while the actual equations, deal with numbers as they are.

Besides the configuration, we include in the description of an ATM a counter of rotations of the circular tape. This counter specifies the number of circles through the tape the machine is still allowed to make before it must halt. The key property of this representation is that *every transition of the machine reduces the numerical value of its representation*. This is relatively easy to assure for the transitions of the head to the right, but hard for the jump to the beginning of the tape. In this case the counter of rotations is essentially used, i.e., it is decreased and hence the whole numerical value of the representation decreases.

It is left to define the descriptions of the ATM that lead to an accepting computation. For sure, if the head is in the accepting state, the ATM accepts, set of such positions can be described by a constant set (with regular notation). Otherwise, a position is accepting if all of its successor positions (or at least one of its successor positions) is accepting. Such a recursive definition can be performed in two steps. Firstly, an equation which constructs the set of all previous configurations to the given one is presented. This is implemented by a simple digit manipulation on the encodings. Then the logic of the ATM, which includes a universal and existential transition, is simulated by intersection (union, respectively) of sets of such previous configurations.

References

[1] J. Autebert, J. Berstel, L. Boasson, "Context-free languages and pushdown automata", in: Rozenberg, Salomaa (Eds.), *Handbook of Formal Languages*, Vol. 1, Springer-Verlag, 1997, 111–174.

[2] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.

[3] D. T. Huynh, "Commutative grammars: the complexity of uniform word problems", *Information and Control*, 57:1 (1983), 21–39.

[4] A. Jeż, "Conjunctive Grammars and Equations over Sets of Natural Numbers", *University of Wrocław*, PhD thesis, 2010.

[5] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.

[6] E. L. Leiss, "Unrestricted complementation in language equations over a one-letter alphabet", *Theoretical Computer Science*, 132 (1994), 71–93.

[7] A. Okhotin, "Unresolved systems of language equations: expressive power and decision problems", *Theoretical Computer Science*, 349:3 (2005), 283–308.

[8] A. Okhotin, "Nine open problems for conjunctive and Boolean grammars", *Bulletin of the EATCS*, 91 (2007), 96–119.

[9] A. Okhotin, "Decision problems for language equations", *Journal of Computer and System Sciences*, 76:3–4 (2010), 251–266.

[10] A. Okhotin, P. Rondogiannis, "On the expressive power of univariate equations over sets of natural numbers", *IFIP Intl. Conf. on Theoretical Computer Science* (TCS 2008, Milan, Italy, 8–10 September, 2008), IFIP vol. 273, 215–227.

[11] L. J. Stockmeyer, A. R. Meyer, "Word problems requiring exponential time", *STOC 1973*, 1–9.