Theory Comput Syst (2010) 46: 27–58
DOI 10.1007/s00224-008-9139-5

Conjunctive Grammars over a Unary Alphabet: Undecidability and Unbounded Growth

Artur Jeż · Alexander Okhotin

Published online: 21 August 2008

Abstract It has recently been proved (Jeż, DLT 2007) that conjunctive grammars (that is, context-free grammars augmented by conjunction) generate some non-regular languages over a one-letter alphabet. The present paper improves this result by constructing conjunctive grammars for a larger class of unary languages. The results imply undecidability of a number of decision problems of unary conjunctive grammars, as well as non-existence of a recursive function bounding the growth rate of the generated languages. An essential step of the argument is a simulation of a cellular automaton recognizing positional notation of numbers using language equations.

Keywords Conjunctive grammars · Unary languages · Language equations · Trellis automata · Cellular automata

1 Introduction

Formal languages over an alphabet consisting of a single letter, known as *unary languages*, can be regarded as sets of natural numbers, and the questions of representation of such sets by the standard devices of formal language theory form a special topic of study. Regular unary languages are just ultimately periodic sets, though

A. Jeż supported by MNiSW grant number N206 024 31/3826, 2006–2008.
A. Okhotin supported by the Academy of Finland under grant 118540.

A. Jeż
Institute of Computer Science, University of Wrocław, Wrocław, Poland
e-mail: aje@ii.uni.wroc.pl

A. Okhotin (✉)
Department of Mathematics, University of Turku, Turku, Finland
e-mail: alexander.okhotin@utu.fi

A. Okhotin
Academy of Finland, Helsinki, Finland

there remain nontrivial questions, such as their descriptional complexity studied by Chrobak [2]. Context-free unary languages are well-known to be regular, though, as shown by Domaratzki et al. [6], context-free grammars give very succinct descriptions of ultimately periodic sets. Simple types of cellular automata, such as *trellis automata* studied by Culik et al. [3–5], Ibarra and Kim [9] and others, are also limited to regular languages when considered over $\{a\}$.

All the above families of languages are characterized by systems of language equations of the general form

$$\begin{cases} X_1 = \varphi_1(X_1, \ldots, X_n), \\ \quad \vdots \\ X_n = \varphi_n(X_1, \ldots, X_n) \end{cases} \tag{*}$$

with different operations allowed in their right-hand sides. As established by Ginsburg and Rice [7], context-free languages are obtained by using concatenation and union in (*). If concatenation is restricted to one-sided linear (that is, every concatenation appearing in some right-hand side is of the form $w \cdot \varphi$ for some constant string w) then the solutions represent exactly the regular languages. Finally, trellis automata, as shown by Okhotin [14], can be simulated by equations with union, intersection and two-sided linear concatenation.

The first example of a non-regular solution of a language equation over a unary alphabet was given by Leiss [11], who constructed a single equation $X = \varphi(X)$ using concatenation and complementation, with a unique solution $\{a^n \mid$ the octal notation of n starts with 1, 2 or $3\}$. The general case of such language equations was recently studied by Okhotin and Yakimova [16].

While equations with complementation as the only Boolean operation are of a purely theoretical interest, some classes of language equations (*) constitute natural extensions of the context-free grammars. One of these classes are *conjunctive grammars* introduced by Okhotin [12], which are represented by language equations with union, intersection and concatenation. These grammars have good practical properties (in particular, efficient parsing algorithms) and are surveyed in a recent article [15].

The expressive power of conjunctive grammars over a unary alphabet was early identified by Wotschke [18] as one of the main open problems in the area. Initially, in the absence of any examples of non-regularity, it was conjectured that only regular languages can be generated [12, 15]. Several unsuccessful attempts of proving this conjecture had been made by different researchers, until recently it has been disproved by Jeż [10], who constructed a grammar for the language $\{a^{4^n} \mid n \in \mathbb{N}\}$. Note that this is the language of all strings a^ℓ with the base-4 positional notation of ℓ given by a digit 1 followed by zeroes. This example was developed into a general theorem stating that for every regular language L over the alphabet $\{0, 1, \ldots, k-1\}$ of digits in base-k positional system, there is a conjunctive grammar generating the language $\{a^n \mid k$-ary notation of n is in $L\}$ [10]. All these languages have at most exponential growth.

This result leaves us with some natural questions to ponder. How far does the expressive power of unary conjunctive languages extend? Are these languages re-

stricted to exponential growth? Can their standard decision problems, such as emptiness, equivalence, etc., be effectively decided? These questions are answered in the present paper by showing that the emptiness problem and other related decision problems are undecidable, while the growth is not bounded by any fixed recursive function, which by far exceeds earlier expectations on the power of conjunctive grammars over $\{a\}$.

All these results are derived from the following general theorem established in Sect. 4. Suppose a set of positional notations of numbers is recognized by a *trellis automaton* [3–5, 9], which is the simplest type of cellular automata known to be equivalent to linear conjunctive grammars [14]. Then the operation of this automaton on positional notations can be simulated by a conjunctive grammar generating unary notations of the same numbers. Since trellis automata are powerful enough to recognize the language of computation histories of a Turing machine, the undecidability results given in Sect. 5 and the construction of a fast growing language in Sect. 6 follow by the standard methods of formal language theory.

Let us begin with the definitions of models used in this paper.

2 Conjunctive Grammars and Trellis Automata

Disjunction is the only Boolean connective expressible in the formalism of context-free grammars: it is represented by multiple rules for a single nonterminal. *Conjunctive grammars* are a direct generalization of the context-free grammars, in which the body of every rule may contain an explicit conjunction:

Definition 1 (Okhotin [12]) A conjunctive grammar is a quadruple $G = (\Sigma, N, P, S)$, in which Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively; P is a finite set of grammar rules, each of the form

$$A \to \alpha_1 \& \ldots \& \alpha_n \quad (\text{with } A \in N,\ n \geqslant 1 \text{ and } \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*) \tag{1}$$

and $S \in N$ is a nonterminal designated as the start symbol.

Informally, a rule (1) states that if a string is generated by each α_i, then it is generated by A. This semantics can be formalized using term rewriting, which generalizes Chomsky's string rewriting.

Definition 2 [12] Given a grammar G, consider terms over concatenation and conjunction with symbols from $\Sigma \cup N$ as atomic terms. The relation $\Rightarrow$ of immediate derivability on the set of terms is defined as follows:

- Using a rule $A \to \alpha_1 \& \ldots \& \alpha_n$, a subterm $A \in N$ of any term $\varphi(A)$ can be rewritten as $\varphi(A) \Rightarrow \varphi(\alpha_1 \& \ldots \& \alpha_n)$.
- A conjunction of several identical strings can be rewritten by one such string: $\varphi(w \& \ldots \& w) \Rightarrow \varphi(w)$, for every $w \in \Sigma^*$.

The language generated by a term φ is $L_G(\varphi) = \{w \mid w \in \Sigma^*, \varphi \Rightarrow^* w\}$. The language generated by the grammar is $L(G) = L_G(S) = \{w \mid w \in \Sigma^*, S \Rightarrow^* w\}$.

An equivalent definition can be given using language equations. This definition generalizes the well-known characterization of the context-free grammars by equations, due to Ginsburg and Rice [7].

Definition 3 [13] For every conjunctive grammar $G = (\Sigma, N, P, S)$, the associated system of language equations [13] is a system of equations in variables N, in which each variable assumes the value of a language over Σ, and which contains the following equation for every variable A:

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \bigcap_{i=1}^{m} \alpha_i \quad \text{(for all } A \in N\text{)}. \tag{2}$$

Each occurrence of a symbol $a \in \Sigma$ in such a system defines a constant language $\{a\}$, while each empty string denotes a constant language $\{\varepsilon\}$. A solution of a system is a vector of languages $(\ldots, L_C, \ldots)_{C \in N}$, such that the substitution of L_C for C, for all $C \in N$, turns each equation (2) into an equality.

It is known that every such system has at least one solution, and among them the *least solution* with respect to componentwise inclusion, and this solution consists of exactly the languages generated by the nonterminals of the original conjunctive grammar: $(\ldots, L_G(C), \ldots)_{C \in N}$ [13]. Thus the representation by language equations constitutes an equivalent semantics of conjunctive grammars, and it is this semantics, and not the rather technical derivation, that accounts for the intuitive clarity of conjunctive and context-free grammars.

Let us give a conjunctive grammar for a standard example of a non-context-free language:

Example 1 (Okhotin [12]) The following conjunctive grammar generates the language $\{wcw \mid w \in \{a, b\}^*\}$:

$$\begin{aligned}
S &\to C \& D,\\
C &\to aCa \mid aCb \mid bCa \mid bCb \mid c,\\
D &\to aA \& aD \mid bB \& bD \mid cE,\\
A &\to aAa \mid aAb \mid bAa \mid bAb \mid cEa,\\
B &\to aBa \mid aBb \mid bBa \mid bBb \mid cEb,\\
E &\to aE \mid bE \mid \varepsilon.
\end{aligned}$$

The nonterminal D generates the language $\{uczu \mid u, z \in \{a, b\}^*\}$. This is done as follows: the rules for D match a single symbol in the left part to the corresponding symbol in the right part using A or B, and the recursive reference to aD or bD makes the remaining symbols be compared in the same way. The intersection with the language $\{ucv \mid u, v \in \{a, b\}^*, |u| = |v|\}$ generated by C completes the grammar.

Grammars for some other standard examples of non-context-free languages, such as $\{a^n b^n c^n \mid n \geqslant 0\}$, can be obtained by a straightforward use of conjunction [12]. Let

us now give an important example of a conjunctive grammar over a unary alphabet for a non-regular language, which introduces an entirely different idea:

Example 2 (Jeż [10]) The following conjunctive grammar with the start symbol A_1 generates the language $\{a^{4^n} \mid n \geqslant 0\}$:

$$\begin{aligned} A_1 &\to A_2A_2\&A_1A_3 \mid a, \\ A_2 &\to A_{12}A_2\&A_1A_1 \mid aa, \\ A_3 &\to A_{12}A_{12}\&A_1A_2 \mid aaa, \\ A_{12} &\to A_3A_3\&A_1A_2. \end{aligned}$$

Each nonterminal A_i generates the language of all strings a^ℓ, with the base-4 notation of ℓ given by the digit(s) i followed by zeroes.

An explanation of this grammar in terms of positional notation of numbers will be given later in Example 3. This paper actually presents a far-going generalization of this construction to a large class of unary languages.

Let us now define an important subclass of conjunctive grammars analogous to linear context-free grammars.

Definition 4 A conjunctive grammar is called *linear conjunctive*, if every rule it contains is either of the form $A \to u_1B_1v_1\& \ldots \&u_nB_nv_n$ with $n \geqslant 1$, $u_i, v_i \in \Sigma^*$ and $B_i \in N$, or of the form $A \to w$ with $w \in \Sigma^*$.

Note that the grammar in Example 1 is linear, while the grammar in Example 2 is not.

The family of languages defined by linear conjunctive grammars [12] has actually been known for almost thirty years before these grammars were introduced: this is the family recognized by one of the simplest types of cellular automata. These are *trellis automata*, also known as one-way real-time cellular automata, which were studied by Culik, Gruska and Salomaa [3–5], Ibarra and Kim [9], and others. Let us explain this concept generally following Culik et al. [3, 4].

A trellis automaton (TA) processes an input string of length $n \geqslant 1$ using a uniform triangular array of $\frac{n(n+1)}{2}$ processor nodes, as presented in the figure below. 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 formalized in the below definition.

Definition 5 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 that sets the initial states,

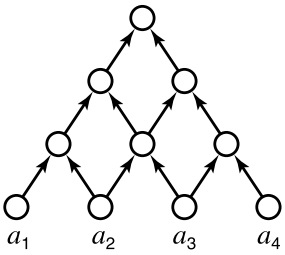
– $\delta : Q \times Q \to Q$ is the transition function, and
– $F \subseteq Q$ is the set of final states.

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 $L(M) = \bigcup_{q \in F} L_M(q)$.



The computational equivalence of linear conjunctive grammars and trellis automata is stated as follows:

Theorem 1 (Okhotin [14]) *A language $L \subseteq \Sigma^+$ is defined by a linear conjunctive grammar if and only if L is recognized by a trellis automaton. These representations can be effectively transformed into each other.*

The family of linear conjunctive languages is known to be closed under all Boolean operations. On the other hand, it is not closed under concatenation and Kleene star [14]. This family is not closed under homomorphisms in general, but it is closed under block codes [5]. It is also not closed under quotient with regular languages; however, it is closed under quotient with singletons, that is, whenever a language $L \in \Sigma^*$ is recognized by a trellis automaton M, the languages $u^{-1}L = \{w \mid uw \in L\}$ and $Lv^{-1} = \{w \mid wv \in L\}$, for every $u, v \in \Sigma^*$, are also recognized by some trellis automata. These trellis automata can be effectively computed from M, u and v.

Trellis automata over a unary alphabet recognize only regular languages [14]. Together with the above closure property, this implies the following simple result, which will be used later on:

Lemma 1 *Let L be a linear conjunctive language over an alphabet Σ, let $u, v \in \Sigma^*$ and $a \in \Sigma$. Then the language $L \cap ua^*v$ is regular.*

Proof Let $K = L \cap ua^*v$. Then the language

$$\widetilde{K} = u^{-1} \cdot K \cdot v^{-1} = \{w \mid uwv \in K\}$$

is linear conjunctive by the closure of this family under quotient with singletons. Since $\widetilde{K}$ is a unary linear conjunctive language, it is regular. Then $K = u\widetilde{K}v$ is regular as well. □

3 Representing Numbers in Positional Notation

Strings over a unary alphabet $\{a\}$ can be regarded as natural numbers, and, accordingly, languages over $\{a\}$ represent sets of numbers. Regular unary languages correspond to ultimately periodic sets. Concatenation of unary strings constitutes sum of numbers, and concatenation of languages is represented by the following pairwise addition of sets of numbers:

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

Language equations over a unary alphabet can be regarded as equations over sets of numbers using addition of sets instead of concatenation. In particular, conjunctive grammars over $\{a\}$ can be rephrased as resolved systems of equations over sets of numbers using union, intersection and addition, as well as singleton constants. The constructions and arguments in this section will be done in terms of such systems of equations.

This paper deals with sets of numbers of a particular form, which are naturally defined and described using positional notation. Fix a base $k \geqslant 2$ and define the alphabet $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$ of k-ary digits. Every string $w = d_{n-1} \ldots d_1 d_0$ with $d_i \in \Sigma_k$ represents the following number:

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

In particular, the empty string ε represents the number 0. Accordingly, every language $L \subseteq \Sigma_k^*$ defines a certain set of numbers:

$$(L)_k = \big\{(w)_k \big| w \in L\big\}.$$

Note that every number has infinitely many notations with different number of leading zeroes. In some cases it will be required that no notations have leading zeroes, that is, $L \subseteq \Sigma_k^* \setminus 0\,\Sigma_k^*$. There is a bijection between such strings (languages) and nonnegative integers (sets thereof).

For example, the set $\{4^n \mid n \geqslant 0\}$ can be written down as $(10^*)_4$, where $10^* \subseteq \{0, 1, 2, 3\}^*$ is a regular language of base-4 representations of numbers in this set. This set is represented by the following system of equations over sets of numbers, which transcribes the conjunctive grammar from Example 2 using a different notation:

Example 3 The least solution of the system

$$\begin{cases} X_1 = (X_2+X_2 \cap X_1+X_3) \cup \{1\}, \\ X_2 = (X_{12}+X_2 \cap X_1+X_1) \cup \{2\}, \\ X_3 = (X_{12}+X_{12} \cap X_1+X_2) \cup \{3\}, \\ X_{12} = X_3+X_3 \cap X_1+X_2 \end{cases}$$

is $X_1 = \{4^n \mid n \geqslant 0\}$, $X_2 = \{2 \cdot 4^n \mid n \geqslant 0\}$, $X_3 = \{3 \cdot 4^n \mid n \geqslant 0\}$, $X_{12} = \{6 \cdot 4^n \mid n \geqslant 0\}$, or $((10^*)_4, (20^*)_4, (30^*)_4, (120^*)_4)$ in base-4 notation. This is the unique solution of this system in sets of positive integers.

Here, as well as everywhere in the following, addition is assumed to have a higher precedence than the Boolean operations.

To see that this vector is indeed a solution, let us substitute the values $X_i = (i0^*)_4$ into the equation for X_1. Then the first sum can be transformed as follows:

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

and similarly the second sum equals

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

Then their intersection is

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

and after taking union with $\{1\}$, exactly $(10^*)_4 = X_1$ is obtained.

Each of the three remaining equations is checked similarly. It remains to show that the given solution is the unique solution of this system in sets of positive numbers. This follows from the general form of the right-hand sides of the system, in which all occurrences of variables are in sums of two variables, and no constant set contains 0. Since systems of this form will appear in the following, it is useful to state a general solution uniqueness lemma.

Definition 6 Consider expressions with union, intersection and addition. Such an expression is called *positive* if it is of the following inductively defined form:

1. a constant set not containing 0 is a positive expression;
2. a sum $\varphi + \psi$ of any two expressions, in which no constant set contains 0, is a positive expression;
3. a union or intersection of two positive expressions is a positive expression.

Lemma 2 *Let $X_i = \varphi_i(X_1, \ldots, X_n)$ be a system of equations over natural numbers, in which every φ_i is a positive expression. Then it has a unique solution in sets of positive integers.*

This class of systems is a variant of the notion of a *proper system* of language equations defined by Autebert et al. [1]. Lemma 2 is proved by a straightforward use of the same methods.

This concludes the explanation of Example 3, which has a least solution with a regular base-4 notation of all components. It is already known that every set with a regular set of positional notations can be specified in a similar way:

Theorem 2 (Jeż [10]) *For every $k \geqslant 2$ and for every nondeterministic finite automaton (NFA) $M = (\Sigma_k, Q, q_0, \delta, F)$ with $\delta : Q \times \Sigma_k \to 2^Q$, there exists a system of equations over sets of natural numbers in variables X, $Y_{i,j,q}$ and $Z_{i,j}$, with $1 \leqslant i < k$, $0 \leqslant j < k$ and $q \in Q$, which has the least solution $X = (L(M)^R)_k$, $Y_{i,j,q} = (ij(L_M(q))^R)_k$, $Z_{i,j} = (ij0^*)_k$.*

This system, which basically simulates the NFA reading base-k notation of a number from the right to the left, has been given in the cited paper, and is included here in the new notation for completeness. The construction is first done assuming $k \geqslant 9$, and the case of $2 \leqslant k \leqslant 8$ is dealt with later

$$X = \bigcup_{\substack{i,j,q:\\ \delta(q,ij)\cap F\neq\emptyset}} Y_{i,j,q} \cup (L(M) \cap \Sigma_k)_k,$$

$$Y_{i,j,q} = \bigcup_{\substack{\ell,q':\\ q\in\delta(q',\ell)}} \left(\bigcap_{n=0}^{3} Y_{j-n,\ell,q'} + Z_{i,n} \right) \cup \{(ij)_k \mid \text{if } q = q_0\} \quad (j \geqslant 4, 0 \leqslant i \leqslant k-1),$$

$$Y_{i,j,q} = \bigcup_{\substack{\ell,q':\\ q\in\delta(q',\ell)}} \left(\bigcap_{n=1}^{4} Y_{k-n,\ell,q'} + Z_{i-1,j+n} \right) \cup \{(ij)_k \mid \text{if } q = q_0\} \quad (j \leqslant 3, i \neq 1),$$

$$Y_{1,j,q} = \bigcup_{\ell,q':q\in\delta(q',\ell)} \left(\bigcap_{n=1}^{4} Y_{j+n,\ell,q'} + Z_{k-n,0} \right) \cup \{(ij)_k \mid \text{if } q = q_0\} \quad (j \leqslant 3),$$

$$Z_{1,j} = \bigcap_{n=1}^{2} Z_{k-n,0} + Z_{j+n,0} \cup \{1 \mid \text{if } j = 0\} \quad (j \leqslant 2),$$

$$Z_{i,j} = \bigcap_{n=1}^{2} Z_{i-1,k-n} + Z_{j+n,0} \cup \{i \mid \text{if } j = 0\} \quad (j \leqslant 2, i \geqslant 2),$$

$$Z_{i,j} = Z_{i,0} + Z_{j,0} \cap \bigcap_{n=1}^{2} Z_{i,j-n} + Z_{n,0} \quad (j \geqslant 3).$$

To prove the theorem, it is sufficient to substitute the given solution into the system and verify that each equation holds true. Then, since the equations for $Y_{i,j,q}$ and $Z_{i,j}$ are of the form required by Lemma 2, the given solution for these variables is unique

in sets of positive integers, and hence the least in sets of non-negative integers. Since the equation for X refers only to other components of this solution, the whole vector is the least solution of the system.

The given construction is applicable only for $k \geqslant 9$. It can be extended to base-k notation with $k \in \{2, \ldots, 8\}$ using the following lemma:

Lemma 3 *Let $S \subseteq \mathbb{N}$ be a set of numbers, let k and k^m (with $k \geqslant 2$ and $m \geqslant 2$) be two bases of positional notation. Then the language $L \subseteq \Sigma_k^* \setminus 0\Sigma_k^*$ of base-k notations of numbers in S is regular (linear conjunctive) if and only if the language $L' \subseteq \Sigma_{k^m}^* \setminus 0\Sigma_{k^m}^*$ of their base-k^m notations is regular (linear conjunctive, respectively).*

Proof Define a block code $h : \Sigma_{k^m}^* \to \Sigma_k^*$ as follows: for any k-ary digits $d_0, \ldots, d_{m-1} \in \Sigma_k$, let

$$h\left(\sum_{i=0}^{m-1} d_i \cdot k^i\right) = d_{m-1} \ldots d_1 d_0,$$

where $\sum_{i=0}^{m-1} d_i \cdot k^i$ is a single digit in base-k^m notation. Then, clearly, $(w)_{k^m} = (h(w))_k$ for every $w \in \Sigma_{k^m}$, and $(L)_{k^m} = (h(L))_k$ for every $L \subseteq \Sigma_{k^m}$.

If $S = (L')_{k^m}$, consider the language $\widehat{L} = h(L') \subseteq \Sigma_k^*$. Since regular languages are closed under homomorphisms, $\widehat{L}$ is regular provided that L' is regular. In the case of linear conjunctive L', the language $\widehat{L}$ is linear conjunctive due to a theorem by Culik et al. [5], because h is a block code. It is known that $S = (\widehat{L})_k$; however, some strings in $\widehat{L}$ may have leading zeroes, if the image of the leading digit in the source string in L' has leading zeroes. The number of such leading zeroes is at most $m - 1$. Let

$$L = \left(\{\varepsilon, 0, 0^2, \ldots, 0^{m-1}\}^{-1} \cdot \widehat{L}\right) \setminus 0\Sigma_k^*.$$

Then every string in $\widehat{L}$ has a representative in L with all leading zeroes removed. Therefore, $S = (L)_k$, and this language remains regular (linear conjunctive) by the closure properties of both families.

Conversely, let $S = (L)_k$ and construct a modified version of L by appending a limited number of leading zeroes:

$$\widetilde{L} = \left(\{\varepsilon, 0, 0^2, \ldots, 0^{m-1}\} \cdot L\right) \cap \left((\Sigma_k)^m\right)^*.$$

Obviously, it is possible to append some number of zeroes to every string in L, so that the number of digits in the result is divisible by m. Hence every string in L has a representative in $\widetilde{L}$ with the same numerical value, and $S = (\widetilde{L})_k$. As long as L is regular (linear conjunctive), the language $\widetilde{L}$ is regular (linear conjunctive) as well.

Consider the language $L' = h^{-1}(\widetilde{L}) \subseteq \Sigma_{k^m}^*$, which is regular (linear conjunctive) by the closure of these families under inverse homomorphisms [5, 9]. Every element of $\widetilde{L}$ is represented by an element of L' with the same numerical value, and therefore $S = (L')_{k^m}$. □

The basic new result of this paper is a construction of systems of equations for a large class of sets of numbers with *non-regular notation*. A simple example of

such a set is $\{(1^\ell 0^\ell)_2 \mid \ell \geqslant 1\}$, where the underlying set of binary notations is linear context-free but not regular. For convenience, the given construction will use sets with a regular notation as constant sets of numbers; according to Theorem 2, every such "constant set" can be expressed using additional equations with singleton constants only.

Different operations on formal languages of positional notations will be used to define these sets and to reason about their properties. Concatenation and Kleene star of positional notations have already been used in the explanation of Example 3. Other operations will be the Boolean operations, as well as the following operations of symbolic addition or subtraction of one. For every $w \in \Sigma_k^* \setminus (k-1)^*$, the string $w' = w \boxplus 1$ is defined as the unique string with $|w| = |w'|$ and $(w)_k + 1 = (w')_k$. Similarly, for every $w \in \Sigma_k^* \setminus 0^*$, define $w' = w \boxminus 1$ as the unique string with $|w| = |w'|$ and $(w)_k - 1 = (w')_k$.

For example, in decimal notation, $0099 \boxplus 1 = 0100$ and $0100 \boxminus 1 = 0099$. This notation shall never be used for strings on which it is undefined, such as $999 \boxplus 1$ and $000 \boxminus 1$. Symbolic addition and subtraction of one is extended to languages as

$$L \boxplus 1 = \{w \boxplus 1 \mid w \in L \setminus (k-1)^*\},$$
$$L \boxminus 1 = \{w \boxminus 1 \mid w \in L \setminus 0^*\}.$$

These operations obviously preserve regularity, and accordingly they can be used inside regular expressions for sets of positional notations. The sets thus defined will remain regular, and hence subject to Theorem 2.

Equations constructed below will often use expressions of a particular form, which are *distributive over infinite union*, in the sense that the value of an expression for a set S can be evaluated by substituting all singletons $\{n\}$ with $n \in S$ into the expression, and then taking the union of all results. The following sufficient condition of distributivity will cover all such cases in the constructions below.

Lemma 4 *Let $\varphi(X)$ be an expression defined as a composition of the following operations*: (i) *the variable X*; (ii) *constant sets*; (iii) *union*; (iv) *intersection with a constant set*; (v) *addition of a constant set*. *Then φ is distributive over infinite union, that is, $\varphi(X) = \bigcup_{n \in X} \varphi(\{n\})$.*

Proof Induction on the structure of φ.

Basis. If $\varphi(X) = X$ or $\varphi(X) = C \subseteq \mathbb{N}$, the statement trivially holds.

Induction step I. Let $\varphi(X) = \psi(X) \cup \xi(X)$. By the induction hypothesis, $\psi(X) = \bigcup_{n \in X} \psi(\{n\})$ and $\xi(X) = \bigcup_{n \in X} \xi(\{n\})$. Therefore,

$$\varphi(X) = \psi(X) \cup \xi(X) = \bigcup_{n \in X} \psi(\{n\}) \cup \bigcup_{n \in X} \xi(\{n\})$$
$$= \bigcup_{n \in X} \big(\psi(\{n\}) \cup \xi(\{n\})\big) = \bigcup_{n \in X} \varphi(\{n\}).$$

Induction step II. If $\varphi(X) = \psi(X) \cap C$ for some $C \subseteq \mathbb{N}$, then, by the induction hypothesis, $\psi(X) = \bigcup_{n \in X} \psi(\{n\})$. Since union and intersection are distributive,

$$\varphi(X) = \psi(X) \cap C = \left(\bigcup_{n \in X} \psi(\{n\}) \right) \cap C = \bigcup_{n \in X} \big(\psi(\{n\}) \cap C \big) = \bigcup_{n \in X} \varphi(\{n\}).$$

Induction step III. The case of $\varphi(X) = \psi(X) + C$ is handled similarly to the previous case, using the distributivity of union and addition. □

4 A Representation of Trellis Automata

Theorem 2 dealt with positional notations of numbers recognized by finite automata, and it asserted that every such set of numbers can be represented by a system of equations. This result will now be extended from regular to linear conjunctive languages of positional notations as follows.

Theorem 3 *For every $k \geqslant 2$ and for every trellis automaton M over the alphabet $\Sigma_k = \{0, \ldots, k-1\}$, such that $L(M) \cap 0\Sigma_k^* = \emptyset$, there exists and can be effectively constructed a resolved system of equations over sets of numbers using the operations $\cup, \cap$ and $+$ and singleton constants, such that its least solution contains a component $(L(M))_k = \{n \mid k\text{-ary notation of } n \text{ is in } L(M)\}$.*

The proof of the theorem is given in the rest of this section. The basis of the argument is the following simulation of the computation of a trellis automaton by a system of equations over sets of numbers.

Lemma 5 *For every $k \geqslant 4$ and for every trellis automaton M over Σ_k, there exists and can be effectively constructed a system using constant sets with regular base-k notation, such that one of the components of the least solution of this system is*

$$(1((L(M) \setminus 0^*) \boxminus 1)10^*)_k = \{(1w10^\ell)_k \mid \ell \geqslant 0, w \notin (k-1)^*, w \boxplus 1 \in L(M)\}.$$

In this way every string recognized by M is represented by a set of numbers $\{(1w1)_k, (1w10)_k, (1w100)_k, \ldots\}$, in which the digits forming w are placed between two 1s, and there may be any number of zeroes after the last 1. The reason to have multiple representatives is that trellis automata require appending symbols at each side of the string, and there is no apparent uniform way to obtain a number $(wi)_k$, with $w \in \Sigma_k^*$ and $i \in \Sigma_k$, from a number $(w)_k$ using addition. On the other hand, a number $(1wi10^\ell)_k$ may be obtained from $(1w10^{\ell+1})_k$ by adding a number of a simple form $((i-1)10^\ell)_k$. Such an addition consumes one zero in the lower digits of the number, and hence the below construction requires an unbounded supply of these zeroes.

Proof 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 $q \in Q$, and with an additional variable Y. It will be proved that the least solution of that system is $X_q = L_q$,

$Y = L$, where

$$\begin{aligned} L_q &= (1((L_M(q) \setminus 0^*) \boxminus 1)10^*)_k \\ &= \{(1w10^\ell)_k \mid \ell \geqslant 0, w \notin (k-1)^*, w \boxplus 1 \in L_M(q)\}, \\ L &= (1((L(M) \setminus 0^*) \boxminus 1)10^*)_k \\ &= \{(1w10^\ell)_k \mid \ell \geqslant 0, w \notin (k-1)^*, w \boxplus 1 \in L(M)\}. \end{aligned}$$

The equations are rather complex and will be constructed and explained in stages.

Define expressions λ_i and ρ_j, for $i, j \in \Sigma_k$, which depend on the variables X_q, and which will be used as building blocks for constructing equations for X_q. For convenience, the inner subexpressions of λ_i are denoted by $\kappa_{i'}$ and the inner subexpressions of ρ_j are $\pi_{j'}$

$$\kappa_{i'}(X) = \big(X \cap (1i'\Sigma_k^*10^*)_k\big) + (10^*)_k \cap (2i'\Sigma_k^*)_k, \quad \text{for all } i' \in \Sigma_k,$$

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

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

$$\pi_{j'}(X) = \big(X \cap (1\Sigma_k^*j'10^*)_k\big) + (10^*)_k \cap (1\Sigma_k^*j'20^*)_k, \quad \text{for all } j' \in \Sigma_k,$$

$$\rho_j(X) = \bigcup_{j' \in \Sigma_k} \big(\pi_{j'}(X) + ((k+j-2)10^*)_k \cap (1\Sigma_k^*j10^*)_k\big), \quad \text{for } j = 0, 1,$$

$$\rho_j(X) = \bigcup_{j' \in \Sigma_k} \big(\pi_{j'}(X) + (1(j-2)10^*)_k \cap (1\Sigma_k^*j10^*)_k\big), \quad \text{for } 2 \leqslant j \leqslant k-2,$$

$$\rho_{k-1}(X) = \bigcup_{j' \in \Sigma_k} \big(\pi_{j'}(X) + ((k-3)10^*)_k \cap (1\Sigma_k^*(k-1)10^*)_k\big).$$

As elsewhere, addition has a higher precedence than intersection.

Also define the following constant sets $R_q \subseteq \mathbb{N}$ for every $q \in Q$:

$$R_q = \{(1(w \boxminus 1)10^*)_k \mid w \in 0^*(\Sigma_k \setminus 0) \cup (\Sigma_k \setminus 0)0^*, w \in L_M(q)\}.$$

By Lemma 1, the language of positional notations of these numbers is regular, so those are allowed constants.

Using the functions λ_i and ρ_j, as well as the constant sets R_q, the system of equations over $\mathbb{N}$ can be succinctly represented in the following form:

$$\begin{cases} X_q = R_q \cup \displaystyle\bigcup_{\substack{q', q'' : \delta(q', q'') = q \\ i, j \in \Sigma_k}} \lambda_i(X_{q''}) \cap \rho_j(X_{q'}) \quad (\text{for all } q \in Q), \\ Y = \bigcup_{q \in F} X_q. \end{cases}$$

As already mentioned, every string $w \in \Sigma^+ \setminus 0^*$ is represented by numbers of the form $(1(w \boxminus 1)10^\ell)_k$ with $\ell \geqslant 0$. 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 $(1(iwj \boxminus 1)10^\ell)_k$ should belong to X_q if and only if there are states q', q'' with $\delta(q', q'') = q$, $(1(iw \boxminus 1)10^{\ell+1})_k \in X_{q'}$ and $(1(wj \boxminus 1)10^\ell)_k \in X_{q''}$. The above system of equations represents exactly this dependence.

The purpose of the expression λ_i is to take a number of the form $(1(wj \boxminus 1)10^\ell)_k$ and append the digit i to the left of the encoded string, obtaining a number $(1(iwj \boxminus 1)10^\ell)_k$. Similarly, ρ_j starts with a number $(1(iw \boxminus 1)10^{\ell+1})_k$ and appends j to the right of the string, also obtaining $(1(iwj \boxminus 1)10^\ell)_k$; note that one of the zeroes in the tail has to be consumed. 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.

Generally, 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 $j > i$. However, since the equations deal with *sets* of numbers, in which different numbers have to be modified at different positions ℓ, this requires adding a set $((j-i)0^*)_k$. Then these digits are added to each number at all possible positions, of which only one position is intended, while the rest produce some corrupted numbers. In order to force the addition of digits at the appropriate positions, these corrupted numbers must be eliminated. This is done by adding sets in two phases, checking the form of the intermediate and final results. In each phase, a digit is added and the sum is intersected with another constant set of numbers. The intersection filters out malformed sums, thus ensuring that the addition took place at the appropriate position.

Once these elementary operations on encoded strings are available, the high-level construction given in the equations for X_q works rather straightforwardly. The sets R_q contain the starting part of L_q representing elements of $L_M(q)$ of a very simple form: either a nonzero digit followed by zeroes, or a sequence of zeroes ending with one nonzero digit. The union over states q', q'' and digits i, j in the equation for X_q represents one step of computation of M exactly according to the definition of a trellis automaton.

The overall goal is to prove the following statement:

Main Claim The unique solution of the system in sets of non-negative integers is $X_q = L_q$ (for all $q \in Q$), $Y = L$.

Consider the system for the variables X_q, for all q. By Lemma 2, the form of this system ensures the uniqueness of its solution in non-negative integers. The equation for Y is just a union of some of these variables and it cannot yield any extra solutions. It remains to substitute the vector $(\ldots, L_q, \ldots, L)$ into the system and verify that all equations hold true.

The first to be calculated is the value of each λ_i on each L_q, beginning with its subexpression $\kappa_{i'}$.

Claim 1 For each $q \in Q$ and $i' \in \Sigma_k$,

$$\kappa_{i'}(L_q) = \{(2i'w10^m)_k \mid m \geqslant 0, i'w \notin (k-1)^*, i'w \boxplus 1 \in L_M(q)\}.$$

Proof The inner subexpression $X \cap (1i'\Sigma_k^*10^*)_k$ clearly has value

$$L_q \cap (1i'\Sigma_k^*10^*)_k = \{(1i'w10^m)_k \mid m \geqslant 0, i'w \notin (k-1)^*, i'w \boxplus 1 \in L_M(q)\}. \quad (3)$$

The evaluation of entire expression $\kappa_{i'}$ is first done for singletons $X = \{n\}$, and then Lemma 4 is applied to obtain its value on $X = L_q$. In view of the above intersection (3), it is sufficient to consider numbers of the form

$$n = (1i'w10^m)_k,$$

with $w' \in \Sigma_k^*$, $j' \in \Sigma_k$ and $m \geqslant 0$. Then any number $n' = (10^\ell)_k = k^\ell$ with $\ell \geqslant 0$ can be added, and it is required that the base-k notation of the sum $n + n'$ has $2i'$ as its first two digits.

- If the number of digits in n and n' is the same, that is, if $|i'w10^m| = \ell$, then the leading 1s in n and n' are in the same position, and the sum is $n + n' = (2i'w10^m)_k \in (2i'\Sigma_k^*)_k$.
- If n' has more digits than n, then $n+n'$ begins with 1, and hence $n+n' \notin (2i'\Sigma_k^*)_k$.
- If n' has fewer digits than n, then the form of the sum depends on whether there is a carry into the first position. If there is no carry, then the sum of these numbers does not begin with 2. Otherwise, if it begins with 20 due to a carry, then i', the second digit of n, is $k-1$, but at the same time the second digit of $n+n'$ is 0 and hence not i'.

These wrong combinations are filtered out by an intersection with $(2i'\Sigma_k^*)_k$. Altogether the substitution of $X = \{n\}$ yields

$$\big(\{(1i'w10^m)_k\} + (10^*)_k\big) \cap (2i'\Sigma_k^*)_k = \{(2i'w10^m)_k\}.$$

Since this expression is a superposition of intersection with a constant set and addition of a constant set, by Lemma 4, it is distributive: that is, for any $T \subseteq (1i'\Sigma_k^*10^*)_k$,

$$\begin{aligned} T + (10^*)_k \cap (2i'\Sigma_k^*)_k &= \bigcup_{n \in T} \big(\{n\} + (10^*)_k \cap (2i'\Sigma_k^*)_k\big) \\ &= \{2i'w10^m \mid 1i'w10^m \in T\}. \end{aligned}$$

It remains to substitute the value (3) of the inner subexpression for T, obtaining

$$\begin{aligned} \kappa_{i'}(L_q) &= \big(L_q \cap (1i'\Sigma_k^*10^*)_k\big) + (10^*)_k \cap (2i'\Sigma_k^*)_k \\ &= \{(1i'w10^m)_k \mid m \geqslant 0, i'w \notin (k-1)^*, i'w \boxplus 1 \in L_M(q)\} + (10^*)_k \\ &\quad \cap (2i'\Sigma_k^*)_k \\ &= \{(2i'w10^m)_k \mid m \geqslant 0, i'w \notin (k-1)^*, i'w \boxplus 1 \in L_M(q)\}, \end{aligned} \quad (4)$$

which completes the proof of Claim 1. □

Now, knowing the value of $\kappa_{i'}(L_q)$, one can evaluate λ_i on L_q.

Claim 2 For each $i \in \Sigma_k$ and $q \in Q$,

$$\lambda_i(L_q) = \{(1(iw \boxminus 1)10^m)_k \mid w \in L_M(q) \setminus 0^*, m \geqslant 0\}.$$

Proof The proof is again by evaluating each subexpression, which is first done for singletons $X = \{n\}$. Addition followed by intersection used in λ_i follows the same principle as in $\kappa_{i'}$. The set being added has a different form depending on i, and, according to the definition of λ_i, there are two cases.

Consider first the case of $i \in \{0, 1\}$. By Claim 1, every number in $\kappa_{i'}(L_q)$ is of the form

$$n = (2i'w'10^m)_k,$$

where $w = i'w'$. Then a number $n' = ((k+i-2)0^\ell)_k$ is added to n, and it is required that the result begins with digits $1i$. Since n has 2 as the leading digit, this digit should be modified to obtain a result of such a form.

- If the number of digits in n and n' is the same, that is, $|i'w'10^m| = \ell$, then the digits 2 and $(k+i-2)$ are in the same position, and so the result $n+n' = (1ii'w'10^m)_k$ is as intended.
- If n' has more digits than n, then the sum $n+n'$ has the leading digit $k+i-2 \in \{k-2, k-1\}$, which is not 1, because $k \geqslant 4$. Hence, $n+n' \notin (1i\Sigma_k^*)_k$.
- If n' has fewer digits than n, then the leading digit of their sum is 2 or 3, and again the sum is not in $(1i\Sigma_k^*)_k$.

After an intersection, again, only one number remains:

$$\big(\{(2i'w'10^m)_k\} + ((k+i-2)0^*)_k\big) \cap (1i\Sigma_k^*)_k = \{(1ii'w'10^m)_k\}.$$

As in the previous case, this subexpression is distributive by Lemma 4, that is, its value on any set $T \subseteq (2i'\Sigma_k^*10^*)_k$ is obtained from its value on singletons as follows:

$$\begin{aligned} T + ((k+i-2)0^*)_k \cap (1i\Sigma_k^*)_k &= \bigcup_{n\in T} \big(\{n\} + ((k+i-2)0^*)_k \cap (1i\Sigma_k^*)_k\big) \\ &= \{(1ii'w'10^m)_k \mid 2i'w'10^m \in T\}. \end{aligned}$$

The value (4) of the nested subexpression $\kappa_{i'}(L_q)$ is known from Claim 1. Substituting this value for T, one obtains

$$\begin{aligned} &\kappa_{i'}(L_q) + ((k+i-2)0^*)_k \cap (1i\Sigma_k^*)_k \\ &\quad = \{(2i'w'10^m)_k \mid m \geqslant 0, i'w' \notin (k-1)^*, i'w' \boxplus 1 \in L_M(q)\} + ((k+i-2)0^*)_k \\ &\qquad \cap (1i\Sigma_k^*)_k \\ &\quad = \{(1ii'w'10^m)_k \mid m \geqslant 0, i'w' \notin (k-1)^*, i'w' \boxplus 1 \in L_M(q)\}. \end{aligned} \tag{5}$$

Now consider the case of $i \geqslant 2$, where a number $n' \in (1(i-2)0^*)_k$ is added to n.

– If n' has exactly one digit more than n, that is, the digits 2 and $i-2$ are in the same position, then the sum equals $n+n' = (1ii'w'10^m)_k$, as intended.
– If n' has exactly as many digits as n, then there are three subcases:
  – if $i'+i-2<k$, then the result is $n+n' = (3(i'+i-2)w'10^m)_k \notin (1i\Sigma_k^*)_k$;
  – if $i'+i-2\geqslant k$ and $k\geqslant 5$, the sum is $n+n' = (4(i'+i-2-k)w'10^m)_k \notin (1i\Sigma_k^*)_k$;
  – and if $i'+i-2\geqslant k$ and $k=4$, then $n+n' = (10(i'+i-2-k)w'10^m)_k$, which is again not in $(1i\Sigma_k^*)_k$ since $i\neq 0$.
– If the number of digits in n' is greater than the number of digits in n plus one, then the result is $n+n' = (1(i-2)0^t2i'w'10^m)_k$ for some $t\geqslant 0$, and hence $n+n'\notin (1i\Sigma_k^*)_k$.
– Finally, if there are fewer digits in n' than in n, then the leading digit in $n+n'$ is 2 or 3.

Thus, all unintended results are filtered by intersection with $(1i\Sigma_k^*)_k$, and the result is the same as in the previous case:

$$\big(\{(2i'w'10^m)_k\}+(1(i-2)0^*)_k\big)\cap(1i\Sigma_k^*)_k = \{(1ii'w'10^m)_k\}.$$

Then, using Lemma 4 and the value (4) of the nested subexpression $\kappa_{i'}(L_q)$, the following is obtained for each $i\geqslant 2$:

$$\begin{aligned}
&\kappa_{i'}(L_q)+(1(i-2)0^*)_k \cap (1i\Sigma_k^*)_k\\
&\quad=\{(2i'w'10^m)_k \mid m\geqslant 0, i'w'\notin(k-1)^*, i'w'\boxplus 1\in L_M(q)\}+(1(i-2)0^*)_k\\
&\qquad\cap(1i\Sigma_k^*)_k\\
&\quad=\{(1ii'w'10^m)_k \mid m\geqslant 0, i'w'\notin(k-1)^*, i'w'\boxplus 1\in L_M(q)\},
\end{aligned}\tag{6}$$

and the result is the same as in the case of $i\in\{0,1\}$.

As the whole expression λ_i is defined as the union of subexpressions evaluated above, the set $\lambda_i(L_q)$ equals the union of the values (5,6) for all $i'\in\Sigma_k$. Taking a union over i', one obtains:

$$\begin{aligned}
\lambda_i(L_q)&=\bigcup_{i'}\{(1ii'w10^m)_k \mid m\geqslant 0, i'w\notin(k-1)^*, i'w\boxplus 1\in L_M(q)\}\\
&=\{(1iw10^m)_k \mid m\geqslant 0, w\notin(k-1)^*, w\boxplus 1\in L_M(q)\}\\
&=\{(1iw10^m)_k \mid m\geqslant 0, w\in(L_M(q)\setminus 0^*)\boxminus 1\}\\
&=(1(i(L_M(q)\setminus 0^*)\boxminus 1)10^*)_k,
\end{aligned}\tag{7}$$

and the claim is proved. □

The expressions ρ_j operate in a way similar to λ_i. While λ_i modifies the first, most significant digits of numbers $(1w10^\ell)_k$, ρ_j should modify the digits around the last non-zero digit. This is also done by addition in two stages, but instead of intersecting the result with sets of the form $(x\Sigma_k^*10^*)_k$, where $x\in\Sigma_k^+$ are the intended digits, this time one has to use intersection with $(1\Sigma^*x0^*)_k$. The corresponding correctness

statement for ρ_j is established using generally the same argument as the previous Claim 2. First its inner subexpression $\pi_{j'}$ is evaluated on L_q.

Claim 3 For all $q \in Q$ and $j' \in \Sigma_k$,

$$\pi_{j'}(L_q) = \{(1w'j'20^m)_k \mid m \geqslant 0, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}.$$

Proof As in the proof of Claim 1, every subexpression of $\pi_{j'}(X)$ will be evaluated on $X = L_q$. The inner subexpression of $\pi_{j'}(L_q)$ is

$$L_q \cap (1\Sigma_k^* j'10^*)_k = \{(1w'j'10^m)_k \mid m \geqslant 0, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}. \tag{8}$$

Consider the next subexpression in $\pi_{j'}$, which is $\big(X \cap (1\Sigma_k^* j'10^*)_k\big) + (10^*)_k \cap (1\Sigma_k^* j'20^*)_k$. The first task is to evaluate it for a singleton $X = \{n\}$, where the number is of the form

$$n = (1w'j'10^m)_k.$$

In this subexpression, any number $n' = (10^\ell)_k$ with $\ell \geqslant 0$ can be added to n, and the results are restricted to be in $(1\Sigma_k^* j'20^*)_k$, that is, $n + n'$ must have $j'2$ as its last non-zero digits.

- If $m = \ell$, that is, the last digit 1 in n is in the same position as the leading digit 1 in n', then $n + n' = (1iw'j'20^m)_k$.
- If these digits are not aligned, then the last non-zero digit in $n + n'$ is 1, and hence $n + n' \notin (1\Sigma_k^* j'20^*)_k$.

Thus it has been shown that

$$\{(1w'j'10^m)_k\} + (10^*)_k \cap (1\Sigma_k^* j'20^*)_k = \{(1w'j'20^m)_k\}.$$

As in the previous proofs, the subexpression is distributive by Lemma 4, that is, for any $T \subseteq (1\Sigma_k^* j'10^*)_k$,

$$\begin{aligned} T + (10^*)_k \cap (1\Sigma_k^* j'20^*)_k &= \bigcup_{n \in T} \big(\{n\} + (10^*)_k \cap (1\Sigma_k^* j'20^*)_k\big) \\ &= \{1w'j'20^m \mid 1w'j'10^m \in T\}. \end{aligned}$$

Now let T be the value (8) of the inner subexpression, which gives

$$\begin{aligned} \pi_{j'}(L_q) &= \big(L_q \cap (1\Sigma_k^* j'10^*)_k\big) + (10^*)_k \cap (1\Sigma_k^* j'20^*)_k \\ &= \{(1w'j'20^m)_k \mid m \geqslant 0, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}, \end{aligned} \tag{9}$$

and thus proves Claim 3. □

Claim 4 For each $j \in \Sigma_k$ and $q \in Q$,

$$\rho_j(L_q) = \{(1(w(j+1 \bmod k) \boxminus 1)10^m)_k \mid w \in L_M(q) \setminus 0^*, m \geqslant 0\}.$$

Proof The evaluation of subexpressions of $\rho_j(X)$ on $X = L_q$ splits into three cases according to the digit j. In each case the innermost subexpression is $\pi_{j'}(X)$, and its value $\pi_{j'}(L_q)$ contains only numbers of the form

$$n = (1w'j'20^m)_k,$$

with $w = w'j' \notin (k-1)^*$.

The first case is $j \in \{0, 1\}$, where ρ_j has a subexpression $\pi_{j'}(X) + ((k+j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k$. Here any number $n' = ((k+j-2)10^\ell)_k$ with $\ell \geqslant 0$ is added to n, and it is required that the sum $n+n'$ has $j1$ as its last non-zero digits. There are several subcases depending on the number of ending zeroes in n and in n', which is m and ℓ, respectively:

- If $\ell = m-1$, that is, the last non-zero digit 2 in n is aligned with the leading digit $(k+j-2)$ of n', and they sum up to j, with a carry to the next digit, j'. Accordingly, the sum of two numbers is $n+n' = (1(w'j' \boxplus 1)j10^{m-1})_k$, as it is intended to be. Note that since $w'j' \notin (k-1)^*$, the string $w'j' \boxplus 1$ is well-defined.
- If $\ell < m-1$, then the last two non-zero digits of $n+n'$ are $(k+j-2)1$. As $k \geqslant 4$ and $k+j-2 \neq j$, the last non-zero digits of $n+n'$ are different from $j1$, and therefore $n+n' \notin (1\Sigma_k^* j10^*)_k$.
- If $\ell = m$, then the last non-zero digit of $n+n'$ is 3 and not 1, and the sum is again not in $(1\Sigma_k^* j10^*)_k$.
- In case of $\ell > m$, the last digit is 2 inherited from n, and again the sum is not of the required form.

It follows that all unintended sums are filtered out by intersection, and the result is

$$(1w'j'20^m)_k + ((k+j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k = \{(1(w'j' \boxplus 1)j10^{m-1})_k\},$$

which, by Lemma 4, extends to any set $T \subseteq (1\Sigma_k^* j'20^*)_k \setminus (1(k-1)^*20^*)_k$ as follows:

$$T + ((k+j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k = \{(1(w'j' \boxplus 1)j10^{m-1})_k \mid 1w'j'20^m \in T\}.$$

Now let T be the value of $\pi_{j'}(L_q)$ known from Claim 3. Then, after the addition and intersection in the subexpression for $j \in \{0, 1\}$, the intermediate result is

$$\begin{aligned}&\pi_{j'}(L_q) + ((k+j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k \\ &\quad = \{(1(w'j' \boxplus 1)j10^{m-1})_k \mid m \geqslant 1, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}. \quad (10)\end{aligned}$$

Next, consider the case of $j \in \{2, \ldots, k-2\}$, when ρ_j has inner subexpression $\pi_{j'}(X) + (1(j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k$. Here a number $n' = (1(j-2)10^\ell)_k$ is added to n, and the subsequent intersection requires that $n+n'$ has $j1$ as its last non-zero digits.

- Again, for $\ell = m-1$ the sum is $(1(w'j' \boxplus 1)j10^{m-1})_k$, which is of the required form.
- If $\ell < m-1$, then the last two non-zero digits of $n+n'$ are $(j-2)1$. Since $j-2 \neq j$, the sum is not in $(1\Sigma_k^* j10^*)_k$.

– If $\ell = m$, then the last non-zero digit of $n + n'$ is 3.
– If $\ell > m$, then the last non-zero digit is 2 (coming from n).

Therefore, as in the previous case, the only result that passes through the intersection is $(1(w'j' \boxplus 1)j10^{m-1})_k$, and this subexpression evaluates to

$$(1w'j'20^m)_k + (1(j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k = \{(1(w'j' \boxplus 1)j10^{m-1})_k\}.$$

Hence the value of the inner subexpression for $2 \leqslant j \leqslant k-2$ is

$$\pi_{j'}(L_q) + ((j-2)10^*)_k \cap (1\Sigma_k^* j10^*)_k = \{(1(w'j' \boxplus 1)j10^{m-1})_k \mid m \geqslant 1, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}, \quad (11)$$

which is the same as for $j \in \{0, 1\}$.

To conclude this case, as well as the previous one, it has been proved that for all $j \neq k-1$, regardless of the value of j', ρ_j transforms $(1w'j'10^m)_k$ into $(1(w'j' \boxplus 1)j0^{m-1})_k$, or equivalently, $(1w'j'j10^m)_k$ is obtained from $(1(w'j' \boxminus 1)10^{m+1})_k$.

Finally, consider the case of $j = k-1$. Here any number $n' = ((k-3)10^\ell)_k$ with $\ell \geqslant 0$ can be added to $n = (1w'j'20^m)_k$, and their sum $n + n'$ is required to have $(k-1)1$ as its last non-zero digits.

– If $\ell = m-1$, then the sum is $(1w'j'(k-1)10^{m-1})_k$, which passes the intersection with $(1\Sigma_k^*(k-1)10^*)_k$.
– If $\ell < m-1$, then the last two digits of $n+n'$ are $(k-3)1$, and since $k-3 \neq k-1$, this number is not of the required form.
– If $\ell = m$, then the last digit of $n + n'$ is 3.
– If $\ell > m$, then the last digit is 2 from n.

As all wrong values have been filtered out, the value of the expression is again a singleton:

$$(1w'j'20^m)_k + ((k-3)10^*)_k \cap (1\Sigma_k^*(k-1)10^*)_k = \{(1w'j'(k-1)10^{m-1})_k\}.$$

Thus the subexpression corresponding to $j = k-1$ has the following value:

$$\pi_{j'}(L_q) + ((k-3)10^*)_k \cap (1\Sigma_k^*(k-1)10^*)_k = \{(1w'j'(k-1)10^{m-1})_k \mid m \geqslant 1, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\}. \quad (12)$$

Now the value of ρ_{k-1} on L_q is obtained as the union of (12) over j':

$$\begin{aligned}
&\rho_{k-1}(L_q) \\
&\quad = \bigcup_{j'} \{(1w'j'(k-1)10^{m-1})_k \mid m \geqslant 1, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\} \\
&\quad = \{(1w(k-1)10^{m-1})_k \mid m \geqslant 1, w \notin (k-1)^*, w \boxplus 1 \in L_M(q)\} \\
&\quad = \{(1(w \boxminus 1)(k-1)10^m)_k \mid m \geqslant 0, w \notin 0^*, w \in L_M(q)\}.
\end{aligned}$$

Note that $(w \boxminus 1)(k-1) = w0 \boxminus 1$, and so the latter set can be rewritten as

$$\rho_{k-1}(L_q) = \{(1(w0 \boxminus 1)10^m)_k \mid m \geqslant 0, w \in L_M(q) \setminus 0^*\},$$

which is of the form stated in the claim.

Similarly, for each $j \neq k-1$, the union of (10,11) over j' gives the value of ρ_j:

$$\begin{aligned}\rho_j(L_q) &= \bigcup_{j'} \{(1(w'j' \boxplus 1)j10^{m-1})_k \mid m \geqslant 1, w'j' \notin (k-1)^*, w'j' \boxplus 1 \in L_M(q)\} \\ &= \{(1(w \boxplus 1)j10^{m-1})_k \mid m \geqslant 1, w \notin (k-1)^*, w \boxplus 1 \in L_M(q)\} \\ &= \{(1wj10^m)_k \mid m \geqslant 0, w \notin 0^*, w \in L_M(q)\}.\end{aligned}$$

Here, since $j \neq k-1$, the string wj is equal to $w(j+1) \boxminus 1$, and hence the result equals

$$\{(1(w(j+1) \boxminus 1)10^m)_k \mid m \geqslant 0, w \in L_M(q) \setminus 0^*\},$$

which completes the proof of Claim 4. □

Now the intended solution can be substituted into the system.

Proof of the Main Claim For every $i, j \in \Sigma_k$ and $q', q'' \in Q$, consider the expression $\lambda_i(X_{q''}) \cap \rho_j(X_{q'})$. The values of $\lambda_i(L_{q''})$ and $\rho_j(L_{q'})$ are known from Claim 2 and Claim 4, and the form of the expressions can be unified as follows (all sums $j+1$ are modulo k):

$$\begin{aligned}\lambda_i(L_{q''}) &= \{(1(iw(j+1) \boxminus 1)10^m)_k \mid w(j+1) \in L_M(q'') \setminus 0^*, m \geqslant 0\}, \\ \rho_j(L_{q'}) &= \{(1(iw(j+1) \boxminus 1)10^m)_k \mid iw \in L_M(q') \setminus 0^*, m \geqslant 0\}.\end{aligned}$$

The intersection of these sets therefore is

$$\begin{aligned}\lambda_i(L_{q''}) \cap \rho_j(L_{q'}) = \big\{(1(iw(j+1) \boxminus 1)10^m)_k \big| \\ iw \in L_M(q'), w(j+1) \in L_M(q''), \\ iw \notin 0^*, w(j+1) \notin 0^*, m \geqslant 0\big\}.\end{aligned}$$

Fix $q \in Q$. According to the definition of a trellis automaton, $iw(j+1) \in L_M(q)$ if and only if $iw \in L_M(q')$ and $w(j+1) \in L_M(q'')$ for some q' and q'' with $\delta(q', q'') = q$. Then the union of $\lambda_i(L_{q''}) \cap \rho_j(L_{q'})$ over all such states q' and q'' equals

$$\begin{aligned}&\bigcup_{q',q'':\delta(q',q'')=q} \lambda_i(L_{q''}) \cap \rho_j(L_{q'}) \\ &\quad = \{(1(iw(j+1) \boxminus 1)10^m)_k \mid iw(j+1) \in L_M(q), iw \notin 0^*, \\ &\qquad w(j+1) \notin 0^*, m \geqslant 0\}.\end{aligned}$$

Taking another union over all digits i and j, the following expression is obtained:

$$\bigcup_{i,j\in\Sigma_k}\bigcup_{q',q'':\delta(q',q'')=q}\lambda_i(L_{q''})\cap\rho_j(L_{q'})$$
$$=\{(1(w\boxminus 1)10^m)_k \mid w\in L_M(q), w\notin \Sigma_k 0^*\cup 0^*\Sigma_k, m\geqslant 0\}.$$

The cases of $w\in\Sigma_k 0^*\cup 0^*\Sigma_k$ are not reflected in the above expression. However, they are included in R_q, and therefore

$$R_q\cup\bigcup_{i,j\in\Sigma_k}\bigcup_{q',q'':\delta(q',q'')=q}\lambda_i(L_{q''})\cap\rho_j(L_{q'})$$
$$=\{(1(w\boxminus 1)10^m)_k \mid w\in L_M(q), w\notin 0^*, m\geqslant 0\}$$
$$=\{(1w10^m)_k \mid w\boxplus 1\in L_M(q), w\notin (k-1)^*, m\geqslant 0\}=L_q,$$

that is, the equation for X_q turns into on equality.

This concludes the proof of the Main Claim and hence of the entire Lemma 5. □

The system constructed in Lemma 5 represents the set $(1(L(M)\boxminus 1)10^*)_k$ for every trellis automaton M. Every number in this set represents an encoding of a string $w\in L(M)$ modified by decrementing w as well as by introducing a pair of sentinel digits 1 and a tail of zeroes. The next step towards representing the set $(L(M))_k$ for every M with $L(M)\cap 0\Sigma_k^*=\emptyset$ is the following lemma, in which a string $w\in\Sigma_k$ is encoded as a number $(1w)_k$, that is, using only one sentinel digit, no zeroes and no decrementation.

Lemma 6 *For every $k\geqslant 4$ and for every trellis automaton M over Σ_k there exists and can be effectively constructed a system using constants with a regular base-k notation, such that one of the components of its least solution is $(1\cdot L(M))_k$.*

Proof For every $j\in\Sigma_k$ and $q\in Q$, consider the language $L_{j,q}=L_M(q)\cdot j^{-1}\setminus 0^*$. By the closure properties of trellis automata, this language is generated by a trellis automaton $M_{j,q}$. Then, by Lemma 5, there exists a system of language equations, such that one of its variables, $Y_{j,q}$, has value

$$Y_{j,q}=\{(1w10^\ell)_k \mid \ell\geqslant 0, w\notin (k-1)^*, w\boxplus 1\in L_{j,q}\}$$

in the least solution.

For each $q\in Q$, let us combine the equations for $Y_{j,q}$, for all j, into a single system, adding a new equation

$$Z_q=(1L_M(q)\cap 10^*\Sigma_k)_k\cup\bigcup_{j=0}^{k-1}(Y_{j,q}\cap(1\Sigma_k^*1)_k)+(1j\boxminus 1)_k.$$

The constant set $(1L_M(q)\cap 10^*\Sigma_k)_k$ has a regular base-k notation by Lemma 1. The value of Z_q in the least solution can be calculated by substituting the values of $Y_{j,q}$ into the equation for Z_q.

First, the inner intersection with $(1\Sigma_k^*1)_k$ filters out the elements of $Y_{j,q}$ with one or more zeroes in the end:

$$\begin{aligned}&\{(1w10^\ell)_k \mid \ell \geqslant 0, w \notin (k-1)^*, w \boxplus 1 \in L_{j,q}\} \cap (1\Sigma_k^*1)_k\\ &\quad = \{(1w1)_k \mid w \notin (k-1)^*, w \boxplus 1 \in L_{j,q}\}.\end{aligned} \tag{13}$$

The subsequent addition of a number $(1j \boxminus 1)_k = k + j - 1$ to each number $(1w1)_k$ changes the lowest digit from 1 to j and produces a carry to the second digit, thus changing w to $w \boxplus 1$. Hence this subexpression has the following value:

$$\begin{aligned}&\{(1w1)_k \mid w \notin (k-1)^*, w \boxplus 1 \in L_{j,q}\} + (1j \boxminus 1)_k\\ &\quad = \{(1w1)_k + (1j \boxminus 1)_k \mid w \notin (k-1)^*, w \boxplus 1 \in L_{j,q}\}\\ &\quad = \{(1(w \boxplus 1)j)_k \mid w \notin (k-1)^*, w \boxplus 1 \in L_{j,q}\} = (1(L_{j,q} \setminus 0^*)j)_k\\ &\quad = (1(L_M(q) \cdot j^{-1} \setminus 0^*)j)_k = (1(L_M(q) \cdot j^{-1})j)_k \setminus (10^*\Sigma_k)_k\\ &\quad = (1(L_M(q) \cap \Sigma_k^* j))_k \setminus (10^*\Sigma_k)_k.\end{aligned} \tag{14}$$

It remains to substitute these values of subexpressions into the full expression for Z, obtaining

$$\begin{aligned}&(1L_M(q) \cap 10^*\Sigma_k)_k \cup \bigcup_{j=0}^{k-1} \Big((1(L_M(q) \cap \Sigma_k^* j))_k \setminus (10^*\Sigma_k)_k \Big)\\ &\quad = (1L_M(q) \cap 10^*\Sigma_k)_k \cup \Big((1L_M(q))_k \setminus (10^*\Sigma_k)_k \Big) = (1L_M(q))_k,\end{aligned} \tag{15}$$

which is accordingly the value of Z_q in the least solution.

Now the equation

$$Z = \bigcup_{q \in F} Z_q,$$

clearly has the least solution $(1L(M))_k$. □

The last major step of the argument is eliminating the leading digit 1 in the representation given by Lemma 6.

Lemma 7 *For every $k \geqslant 4$ and for every trellis automaton M over Σ_k with $L(M) \cap 0\Sigma_k^* = \emptyset$ there exists and can be effectively constructed a system over sets of natural numbers using constants with a regular base-k notation, such that one of the components of its least solution is $(L(M))_k$.*

Proof For every $i \in \Sigma_k \setminus \{0\}$ and for every $q \in Q$, the language $i^{-1}L_M(q)$ is recognized by a certain trellis automaton. Then, by Lemma 6, there is a system of equations over sets of numbers, such that one of its variables, $Z_{i,q}$, represents the set $(1(i^{-1}L_M(q)))_k$.

These systems are combined into one, and a new variable T_q is added, along with the equation

$$T_q = (L_M(q) \cap (\Sigma_k \setminus \{0\}))_k \cup Z_{1,q} \cup \bigcup_{i \in \Sigma_k \setminus \{0,1\}} \tau_i(Z_{i,q}), \quad \text{where}$$

$$\tau_i(X) = \bigcup_{i' \in \Sigma_k} \Big((X \cap (1i'\Sigma_k^*)_k) + ((i-1)0^*)_k \cap (ii'\Sigma_k^*)_k \Big) \quad (\text{for } i \neq 0, 1).$$

The purpose of the subexpression τ_i is to convert a number $(1w)_k$ with $w \in \Sigma_k^+$ to a number $(iw)_k$. Then the right-hand side of the equation for T_q should evaluate to $T_q = (L_M(q) \setminus 0\Sigma_k^*)_k$ under the substitution $Z_{i,q} = (1(i^{-1}L_M(q)))_k$.

The proof starts with evaluating τ_i on each $Z_{i,q}$:

Claim 5 For every $i \in \Sigma_k \setminus \{0, 1\}$ and $q \in Q$,

$$\tau_i(Z_{i,q}) = (L_M(q) \cap i\Sigma_k^+)_k.$$

Proof For every $i' \in \Sigma_k$, consider the component of the union for i' in τ_i. Clearly, the innermost subexpression has the following value:

$$\begin{aligned} Z_{i,q} \cap (1i'\Sigma_k^*)_k &= \{(1w)_k \mid iw \in L_M(q)\} \cap (1i'\Sigma_k^*)_k \\ &= \{(1i'w)_k \mid ii'w \in L_M(q)\}. \end{aligned} \tag{16}$$

The next subexpression involves an addition of $((i-1)0^*)_k$ followed by intersection with $(ii'\Sigma_k)_k$. As in Claims 1–4 in the proof of Lemma 5, this subexpression is first evaluated on a singleton $\{n\}$. The nested intersection with $(1i'\Sigma_k^*)_k$ leaves only numbers of the form

$$n = (1i'w)_k,$$

with $w \in \Sigma_k^*$. Afterwards, any number $n' = ((i-1)0^\ell)_k$ with $\ell \geqslant 0$ can be added to n, and it is required that the sum $n + n'$ starts with two digits ii'.

- If the number of digits in n and n' is the same, that is, if $|i'w| = \ell$, then the sum is $n + n' = (ii'w)_k$, as intended.
- If n' has more digits than n, then the sum $n + n'$ has the leading digit $i - 1$, and hence it is not in $(ii'\Sigma_k^*)_k$.
- Suppose n' has exactly one digit less than n, that is, $|w| = \ell$. Then the second digit i' in n is aligned with the leading digit $i - 1$ of n', and the second digit of the sum $n + n'$ equals $i' + (i - 1)$ modulo k. Since $i' < i' + i - 1 < i' + k$, it follows that this second digit cannot be i', and therefore $n + n'$ is not in $(ii'\Sigma_k^*)_k$.
- If the number of digits in n' is less by more than one than the number of digits in n, there are two subcases:
  - If the addition of n' to n results in a carry to the second digit, then the second digit of the result is $i' + 1 \pmod k$, hence it is different from i'.
  - If there is no carry, then the leading digit of the sum is $1 \neq i$. In both cases $n + n' \notin (ii'\Sigma_k^*)_k$.

This concludes the case study: all wrong combinations are excluded by an intersection, and therefore

$$\{(1i'w)_k\}+((i-1)0^*)_k \cap (ii'\Sigma_k^*)_k = \{(ii'w)_k\}.$$

Hence,

$$\begin{aligned}&\big(Z_{i,q} \cap (1i'\Sigma_k^*)_k\big)+((i-1)0^*)_k \cap (ii'\Sigma_k^*)_k\\ &\quad= \{(ii'w)_k \mid (1i'w)_k \in Z_{i,q}\}\\ &\quad= \{(ii'w)_k \mid ii'w \in L_M(q)\} = (L_M(q) \cap ii'\Sigma_k^*)_k. \end{aligned} \tag{17}$$

Taking the union over i', one obtains

$$\tau_i(Z_{i,q}) = (L_M(q) \cap i\Sigma_k^+)_k, \tag{18}$$

which completes the proof of the claim. □

Getting back to the proof of Lemma 7, now the right-hand side of the equation for each T_q can be evaluated on $Z_{i,q} = (1(i^{-1}L_M(q)))_k$. This is a union of k expressions, the first of them representing a finite set, the second being the variable $Z_{1,q}$, and the rest are the subexpressions $\tau_i(Z_{i,q})$ with $i \in \Sigma_k \setminus \{0,1\}$, evaluated in Claim 5. Altogether, the value of T_q in the least solution of the constructed system is the union of the following sets:

$$\begin{aligned}&(L_M(q) \cap (\Sigma_k \setminus \{0\}))_k \cup \underbrace{(1(1^{-1}L_M(q)))_k}_{(L_M(q)\cap 1\Sigma_k^+)_k} \cup \bigcup_{i=2}^{k-1} (L_M(q) \cap i\Sigma_k^+)_k\\ &\quad= (L_M(q) \setminus 0\Sigma_k^*)_k. \end{aligned} \tag{19}$$

Since $L(M) \cap 0\Sigma_k^* = \emptyset$ by assumption, the least solution of the equation

$$T = \bigcup_{q\in F} T_q$$

is $(L(M))_k$. □

Now the proof of the theorem can be easily inferred from Lemma 7.

Proof of Theorem 3 First assume $k \geqslant 4$. Then Lemma 7 gives a system of equations over sets of numbers with the desired least solution. This system uses constants with a regular notation. By Theorem 2, each of these constants can be expressed by a separate system of equations using singleton constants. The resulting system satisfies the statement of Theorem 3.

It remains to consider the cases of $k = 2, 3$. Define the language L' of base-k^2 notations of numbers whose base-k notation is in $L(M)$. Then, by Lemma 3, L' is generated by another trellis automaton M'. Applying the above argument to M', a

system of equations specifying the given set of numbers is obtained. This completes the proof for this remaining case. □

Finally, the system of equations constructed in Theorem 3 can be represented as a conjunctive grammar over a unary alphabet, which yields the following general result on the expressive power of these grammars:

Corollary 1 *Let $k \geqslant 2$. For every trellis automaton M over Σ_k, with $L(M) \cap 0\Sigma_k^* = \emptyset$, there exists and can be effectively constructed a conjunctive grammar over the alphabet $\{a\}$ that generates the language $\{a^n \mid k\text{-ary notation of } n \text{ is in } L(M)\}$.*

This result will now be used to establish some quite unexpected properties of unary conjunctive grammars.

5 Decision Problems for Unary Conjunctive Grammars

One of the main techniques of proving undecidability results in formal language theory is by representing one or another form of the language of computations of a Turing machine. Given a TM T over an input alphabet Ω, its computations are represented as strings over an auxiliary alphabet Γ. For every $w \in L(T)$, let $C_T(w) \in \Gamma^*$ denote some representation of the accepting computation of T on w. The language

$$\mathrm{VALC}(T) = \{w\natural C_T(w) \mid w \in \Omega^* \text{ and } C_T(w) \text{ is an accepting computation}\}$$

over the alphabet $\Omega \cup \Gamma \cup \{\natural\}$ is the language of valid accepting computations of T. It was shown by Hartmanis [8] that for a certain simple encoding $C_T : \Omega^* \to \Gamma^*$ the language $\mathrm{VALC}(T)$ is an intersection of two context-free languages, while the complement of $\mathrm{VALC}(T)$, denoted $\mathrm{INVALC}(T)$, is context-free. Being able to represent these languages is one of the crucial properties of trellis automata.

Proposition 1 ([14]) *For every Turing machine T there exists an encoding $C_T : \Omega^* \to \Gamma^*$ of its computations, such that $\mathrm{VALC}(T)$ is recognized by a trellis automaton.*

This leads to a number of undecidability results for TA, which are inherited by linear conjunctive grammars [14] and hence by conjunctive grammars of the general form [12]. However, it appears hard to replicate these results for the case of a unary alphabet: a straightforward approach fails due to the apparent lack of structure in strings, on which all known encodings of $\mathrm{VALC}(T)$ rely. Contrary to this intuition, Theorem 3 asserts that if the computation histories in $\mathrm{VALC}(T)$ are regarded as notations of numbers, then, as a linear conjunctive language, $\mathrm{VALC}(T)$ can be specified in unary encoding by a unary conjunctive grammar.

Let us make some further technical assumptions on the encoding of these languages. Assume that $\mathrm{VALC}(T)$ is defined over an alphabet of digits $\Sigma_k = \{0, \ldots,$

$k-1\}$, for a suitable k, and that $0 \notin \Omega$, so that no string in $\mathrm{VALC}(T)$ has a leading zero. Define the language $\mathrm{INVALC}(T)$ as $(\Sigma_k^* \setminus 0\Sigma_k^*) \setminus \mathrm{VALC}(T)$. These elaborations do not affect Proposition 1, so that Theorem 3 can be used to obtain the following result:

Lemma 8 *For every Turing machine T there exist and can be effectively constructed conjunctive grammars G and G' over the alphabet $\{a\}$, such that $L(G) = \{a^n \mid n \in (\mathrm{VALC}(T))_k\}$ and $L(G') = \{a^n \mid n \in (\mathrm{INVALC}(T))_k\}$, where k is the size of the alphabet used for encoding the computations.*

The undecidability of basic decision problems for unary conjunctive grammars, such as whether a given grammar generates $\emptyset$ or whether a given grammar generates a^*, can be easily inferred from this. Let us, however, establish a more general result:

Theorem 4 *For every fixed unary conjunctive language $L_0 \subseteq a^*$, the problem of whether a given conjunctive grammar over $\{a\}$ generates the language L_0 is co-RE-complete.*

Proof The containment of the problem in co-RE is evident, since the equivalence problem for two given recursive languages is in co-RE. It is the co-RE-hardness that has to be established.

Let $G_0 = (\Sigma, N_0, P_0, S_0)$ be a fixed conjunctive grammar generating L_0. Suppose there is an algorithm to check whether $L(G) = L_0$ for any given conjunctive grammar G over $\{a\}$. Let us use this algorithm to solve the emptiness problem for Turing machines, which is known to be co-RE complete. Depending on the form of L_0, let us consider two cases.

Case I: L_0 contains no subset of the form $a^\ell(a^p)^*$, where $\ell \geqslant 0$ and $p \geqslant 1$. Given a Turing machine T, construct a conjunctive grammar $G_T = (\{a\}, N_T, P_T, S_T)$ for $\{a^n \mid n \in (\mathrm{VALC}(T))_k\}$. On the basis of G_T and G_0, construct a new conjunctive grammar $G = (\{a\}, N_T \cup N_0 \cup \{S, A\}, P_T \cup P_0 \cup P, S)$, where P contains the following new rules:

$$
\begin{aligned}
S &\to S_0 \mid S_T A,\\
A &\to aA \mid \varepsilon,\\
S_T &\to \ldots \quad \text{(rules generating } \{a^n \mid n \in (\mathrm{VALC}(T))_k\}),\\
S_0 &\to \ldots \quad \text{(rules generating } L_0).
\end{aligned}
$$

Now, if $L(T) = \emptyset$, then $L(G_T) = \emptyset$, the rule $S \to S_T A$ in G generates nothing, and therefore $L(G) = L(G_0) = L_0$.

Suppose $L(T) \neq \emptyset$. Then there is a string $w \natural C_T(w) \in \mathrm{VALC}(T)$, and accordingly there exists $a^n \in L(G_T)$. Hence the rule $S \to S_T A$ in G can be used to generate all strings in $a^n a^*$, and therefore $L(G)$ contains the subset $a^\ell(a^p)^*$ for $\ell = n$ and $p = 1$. As L_0 contains no such subset by assumption, $L(G) \neq L_0$.

It has been proved that $L(G) = L_0$ if and only if $L(T) = \emptyset$, and therefore the supposed algorithm solves the Turing machine emptiness problem, which yields a contradiction. This proves the theorem for this form of L_0.

Case II: L_0 contains a subset $a^\ell(a^p)^*$, where $\ell \geqslant 0$ and $p \geqslant 1$. Assume that p is larger than the cardinality of the alphabet used for $\mathrm{INVALC}(T)$ (if p is too small, any of its multiples can be taken). Define $\mathrm{INVALC}(T)$ over a p-letter alphabet, consider the set of numbers $(\mathrm{INVALC}(T) \cdot 0)_p$, and construct a conjunctive grammar $G'_T = (\{a\}, N'_T, P'_T, S'_T)$ generating $\{a^n \mid n \in (\mathrm{INVALC}(T) \cdot 0)_p\}$. Using G_0 and G'_T, construct a new grammar $G = (\{a\}, N'_T \cup N_0 \cup \{S, B, C\}, P_T \cup P_0 \cup P, S)$, where the new rules in P are as follows:

$$
\begin{aligned}
S &\to S_0 \& B \mid a^\ell S'_T,\\
B &\to a^i \quad (\text{for all } 0 \leqslant i < \ell),\\
B &\to a^{\ell+i} C \quad (\text{for all } 1 \leqslant i < p),\\
C &\to a^p C \mid \varepsilon,\\
S'_T &\to \ldots \quad (\text{rules generating } (\mathrm{INVALC}(T))_p),\\
S_0 &\to \ldots \quad (\text{rules generating } L_0).
\end{aligned}
$$

Note that $L_G(B) = a^* \setminus a^\ell(a^p)^*$.

If $L(T) = \emptyset$, then $\mathrm{INVALC}(T) = \Sigma_p^* \setminus 0\Sigma_p^*$, and hence $\{a^n \mid n \in (\mathrm{INVALC}(T) \cdot 0)_p\} = (a^p)^*$. Then the rule $S \to a^\ell S'_T$ generates the language $a^\ell(a^p)^* \subseteq L_0$, while the rule $S \to S_0 \& B$ generates $L_0 \setminus a^\ell(a^p)^*$. Therefore, $L(G) = L_0$.

Otherwise, if $L(T) \neq \emptyset$, then there exists $w \notin \mathrm{INVALC}(T)$, which implies $a^{(w0)_p} \notin L(G'_T)$. Let $(w0)_p = ip$, for $i \geqslant 0$. Then the string $a^{ip+\ell} \in L_0$ is not generated by the rule $S \to a^\ell S'_T$, and it is also not generated by $S \to S_0 \& B$, because it is not in $L_G(B)$. Therefore, $a^{ip+\ell} \notin L(G)$ and $L(G) \neq L_0$.

In this case, again, $L(G) = L_0$ if and only if $L(T) = \emptyset$, which proves the undecidability of the problem. □

If L_0 is not generated by a conjunctive grammar, then the problem of testing whether a given conjunctive grammar generates L_0 becomes trivial. Hence, the following characterization is obtained:

Corollary 2 *For every fixed language $L_0 \subseteq a^*$, the problem of testing whether a given conjunctive grammar over $\{a\}$ generates L_0 is either co-RE-complete or trivial.*

The same method can be used to establish undecidability of some further decision problems.

Theorem 5 *For conjunctive grammars over a unary alphabet there exist no algorithm to decide whether a given grammar generates a finite language* (*a regular language*).

Proof The proof is by reduction from the Turing machine emptiness problem.

For any given Turing machine T, construct another TM T' that works as follows. Given an input string $w \in \Sigma^*$, the machine T' first ensures that $w = \varepsilon$, rejecting

otherwise. Then it begins simulating T in parallel on all possible strings from Σ^*, starting one more computation after every simulated step, so that after n simulated steps of the first computation it has a total of n computations simultaneously running. This continues until any of the simulated computations accepts, in which case T' accepts as well. If none of the simulated computations ever accepts, then accordingly T' does not halt. By this construction, $L(T') = \{\varepsilon\}$ if $L(T) \neq \emptyset$, and $L(T') = \emptyset$ otherwise.

Next, construct a conjunctive grammar G for the language

$$\{a^\ell \mid \ell \in (\mathrm{VALC}(T'))_k\} \cdot \{a^{k^n} \mid n \geqslant 0\},$$

which can be done according to Lemma 8 and to Theorem 2.

If $L(T) \neq \emptyset$, then $\mathrm{VALC}(T')$ is a singleton. Let $(\mathrm{VALC}(T'))_k = \ell_0$. Then $L(G) = \{a^{k^n+\ell_0} \mid n \geqslant 0\}$, which is a non-regular language.

Otherwise, let $L(T) = \emptyset$. Then the language $\mathrm{VALC}(T')$ of computation histories is empty as well, and $L(G) = \emptyset$, that is, $L(G)$ is finite.

This shows that an algorithm for testing finiteness or regularity of conjunctive grammars over a unary alphabet would solve the emptiness problem for Turing machines, which completes the proof of the theorem. □

Having seen the above results, it is natural to ask whether unary conjunctive languages have any nontrivial decidable properties. It is known that the membership of a string can be decided in cubic time [12], even in square time in the case of a one-letter alphabet, but nothing besides this problem and its Boolean combinations is known to be decidable. Finding such an example (or perhaps proving its nonexistence) is left as a problem for future study.

6 Growth of Unary Conjunctive Languages

Every infinite unary language $L = \{a^{i_1}, a^{i_2}, \ldots, a^{i_n}, \ldots\}$, where $0 \leqslant i_1 < i_2 < \ldots < i_n < \ldots$, can be regarded as an increasing integer sequence, and it is natural to consider the *growth rate* of such sequences, represented by an increasing function $g(n) = i_n$. Obviously, the growth of every regular language is bounded by a linear function. The example of a conjunctive grammar for the language $\{a^{4^n} \mid n \geqslant 0\}$ [10], see Example 2, shows that the growth of unary conjunctive languages can be exponential, which raises two questions. First, can this growth be superexponential, and is there any upper bound for the growth rate of unary conjunctive languages? Second, can this growth be superlinear but subexponential, such as polynomial?

The following theorem gives the strongest possible answer to the first question:

Theorem 6 *For every infinite recursively enumerable set of natural numbers S there exists a conjunctive grammar G over an alphabet $\{a\}$, such that the growth function of $L(G)$ is greater than that of S at any point. Given a Turing machine recognizing S, the grammar G can be effectively constructed.*

Proof Let f be the growth function of S, that is, $0 \leqslant f(1) < f(2) < \ldots < f(n) < \ldots$ and $S = \{f(1), f(2), \ldots, f(n), \ldots\}$. Let T be a Turing machine that recognizes S and the numbers are given to it in unary notation. Consider the language $\mathrm{VALC}(T)$, which consists of strings $w_n = 1^{f(n)} \natural C_T(f(n))$ with $n \geqslant 1$, and assume it is defined over the alphabet $\Sigma_k = \{0, 1, \ldots, k-1\}$ for some $k \geqslant 2$. By Lemma 8, there exists a conjunctive grammar G over an alphabet $\{a\}$ that generates $L = \{a^n \mid n \in (\mathrm{VALC}(T))_k\}$.

Let $g(n)$ be the growth function of L. It is sufficient to show that $g(n) \geqslant f(n)$ for each $n \geqslant 1$. To see this, consider the values $g(1), g(2), \ldots, g(n)$. Obviously, all of these values cannot be in $\{(w_1)_k, \ldots, (w_{n-1})_k\}$, and hence one of them must be $(w_{n'})_k$ for some $n' \geqslant n$. Let $g(\ell) = (w_{n'})_k$, with $\ell \in \{1, 2, \ldots, n\}$. Since g is an increasing function, $g(n) \geqslant g(\ell)$. At the same time, $(w_{n'})_k > f(n')$, because a computation history on n' is longer than n' itself. It has thus been shown that

$$g(n) \geqslant g(\ell) = (w_{n'})_k > f(n') \geqslant f(n),$$

which completes the proof. □

Corollary 3 *For every growing recursive function $f : \mathbb{N} \to \mathbb{N}$ there exists a conjunctive language $L \subseteq a^*$ with the growth function greater than f at any point.*

Note that this quick-growing language is bound to be computationally very easy, as the upper bound on parsing complexity for conjunctive grammars over a unary alphabet is $DTIME(n^2) \cap DSPACE(n)$ [12, 15].

The next example gives a unary conjunctive language of a polynomial growth.

Proposition 2 *There exists a conjunctive grammar G over an alphabet $\{a\}$, such that the growth function g of $L(G)$ satisfies $g(n) = \Theta(n^2)$.*

Proof Consider the set of numbers

$$S = \{(2^m + 3i) \cdot 2^m \mid m \geqslant 0, 2^m \leqslant 2^m + 3i < 2^{m+2}\}.$$

Let $g(n)$ denote the nth largest element of S; this is the growth function of the corresponding unary language $L = \{a^n \mid n \in S\}$. The set of binary notations of the numbers in S is

$$\begin{aligned}&\left\{1w0^m \middle| w \in \{0,1\}^m; (w)_2 \text{ is divisible by } 3\right\}\\ &\quad\cup \left\{10w0^m \middle| w \in \{0,1\}^m; (1w)_2 \text{ is divisible by } 3\right\}\\ &\quad\cup \left\{11w0^m \middle| w \in \{0,1\}^m; (10w)_2 \text{ is divisible by } 3\right\}.\end{aligned}$$

This is clearly a linear context-free language, hence L is conjunctive by Theorem 3.

Let us derive an explicit expression for g. Consider the elements of S in the range $\{2^{2m}, \ldots, 2^{2m+2} - 1\}$. These values are given by different i with $2^m \leqslant (2^m + 3i) < 2^{m+2}$, which implies $0 \leqslant 3i < 3 \cdot 2^m$. Clearly, S contains exactly 2^m elements from this interval. It follows that the set $S \cap \{0, 1, \ldots, 2^{2m} - 1\}$ contains $1 + 2 + 4 + \ldots + 2^{m-1} = 2^m - 1$ elements. Therefore, $g(2^m) = 2^{2m}$ and $g(2^m + j) = (2^m + 3j)2^m$.

In order to prove that S has a quadratic growth rate, it is sufficient to establish the following inequality:

$$n^2 \leqslant g(n) \leqslant 4n^2.$$

The lower bound is obtained as follows:

$$g(2^m + i) = (2^m + 3i)2^m = 2^{2m} + 3i \cdot 2^m \geqslant 2^{2m} + 2i \cdot 2^m + i^2 = (2^m + i)^2,$$

where the inequality is due to $i \cdot 2^m \geqslant i^2$. On the other hand,

$$g(2^m + i) \leqslant g(2^{m+1}) = 2^{2m+2} \leqslant 4(2^m + i)^2,$$

which proves the upper bound $g(n) \leqslant 4n^2$ and completes the proof. □

This construction can be generalized to obtain the following result:

Theorem 7 *For every rational number $\frac{p}{q} \geqslant 1$ there exists a conjunctive grammar G over the alphabet $\{a\}$, such that the growth function of $L(G)$ is $g(n) = \Theta(n^{\frac{p}{q}})$.*

Sketch of a proof The proof follows the same steps as in the case of $\frac{p}{q} = 2$ treated above. The set of numbers to be represented is

$$S = \left\{2^{pk} + \lfloor C \cdot i \rfloor \cdot 2^{(p-q)k} \middle| i, k \geqslant 0, 2^{pk} \leqslant 2^{pk} + C \cdot i \cdot 2^{(p-q)k} < 2^{p(k+1)}\right\},$$

where $C = \frac{2^p - 1}{2^q - 1}$. □

7 Conclusion

Two years ago no conjunctive grammars generating non-regular unary languages were known. Following the first examples of non-regularity [10], a large nontrivial class of unary languages has been proved to be conjunctive. In particular, this has led to unexpected undecidability results for unary conjunctive grammars. It has also been established that the growth of unary conjunctive languages can be as fast as theoretically possible.

However, there are still no means of proving that some particular unary languages in $DSPACE(n)$ *cannot* be represented by conjunctive grammars. Hence the family of conjunctive languages still could not be separated from $DSPACE(n)$. Inventing a method for producing any non-representability results for unary conjunctive grammars remains a task for future research, and solving this problem would make the next major step in the study of this noteworthy family of languages.

References

1. Autebert, J., Berstel, J., Boasson, L.: Context-free languages and pushdown automata. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 111–174. Springer, New York (1997)

2. Chrobak, M.: Finite automata and unary languages. Theor. Comput. Sci. **47**, 149–158 (1986)
3. Culik, K. II, Gruska, J., Salomaa, A.: Systolic trellis automata, I. Int. J. Comput. Math. **15**, 195–212 (1984)
4. Culik, K. II, Gruska, J., Salomaa, A.: Systolic trellis automata, II. Int. J. Comput. Math. **16**, 3–22 (1984)
5. Culik, K. II, Gruska, J., Salomaa, A.: Systolic trellis automata: stability, decidability and complexity. Inf. Control **71**, 218–230 (1984)
6. Domaratzki, M., Pighizzini, G., Shallit, J.: Simulating finite automata with context-free grammars. Inf. Process. Lett. **84**, 339–344 (2002)
7. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. J. ACM **9**, 350–371 (1962)
8. Hartmanis, J.: Context-free languages and Turing machine computations. In: Proceedings of Symposia in Applied Mathematics, vol. 19, pp. 42–51. Am. Math. Soc., Providence (1967)
9. Ibarra, O.H., Kim, S.M.: Characterizations and computational complexity of systolic trellis automata. Theor. Comput. Sci. **29**, 123–153 (1984)
10. Jeż, A.: Conjunctive grammars can generate non-regular unary languages. Int. J. Found. Comput. Sci. **19**(3), 597–615 (2008)
11. Leiss, E.L.: Unrestricted complementation in language equations over a one-letter alphabet. Theor. Comput. Sci. **132**, 71–93 (1994)
12. Okhotin, A.: Conjunctive grammars. J. Autom. Lang. Comb. **6**(4), 519–535 (2001)
13. Okhotin, A.: Conjunctive grammars and systems of language equations. Program. Comput. Softw. **28**, 243–249 (2002)
14. Okhotin, A.: On the equivalence of linear conjunctive grammars to trellis automata. Inform. Théor. Appl. **38**(1), 69–88 (2004)
15. Okhotin, A.: Nine open problems for conjunctive and Boolean grammars. Bull. EATCS **91**, 96–119 (2007)
16. Okhotin, A., Yakimova, O.: On language equations with complementation. In: Developments in Language Theory, DLT 2006, Santa Barbara, USA, June 26–29, 2006. LNCS, vol. 4036, pp. 420–432. Springer, Berlin (2006)
17. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: STOC 1973, pp. 1–9 (1973)
18. Wotschke, D.: Personal communication to A. Okhotin, August 2000