Conjunctive Grammars over a Unary Alphabet: Undecidability and Unbounded Growth

Artur Jeż[1,⋆] and Alexander Okhotin[2,3,⋆⋆]

[1] Institute of Computer Science, University of Wrocław, Poland
aje@ii.uni.wroc.pl
[2] Academy of Finland
[3] Department of Mathematics, University of Turku, Finland
alexander.okhotin@utu.fi

Abstract. It has recently been proved (Jeż, DLT 2007) that conjunctive grammars (that is, context-free grammars augmented by conjunction) generate some nonregular 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 nonexistence of an r.e. bound on the growth rate of generated languages. An essential step of the argument is a simulation of a cellular automaton recognizing positional notation of numbers using language equations.

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 there remain nontrivial questions, such as the economy of description studied by Chrobak [1]. Context-free unary languages are well-known to be regular, though, as shown by Domaratzki et al. [3], 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. [2], Ibarra and Kim [6] 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 [4], context-free languages are obtained by using concatenation and union in (*). If concatenation is restricted to one-sided linear then the

⋆ Supported by MNiSW grant number N206 024 31/3826, 2006–2008.
⋆⋆ Supported by the Academy of Finland under grant 118540.

V. Diekert, M. Volkov, and A. Voronkov (Eds.): CSR 2007, LNCS 4649, pp. 168–181, 2007.

solutions represent exactly the regular languages. Finally, trellis automata, as shown by Okhotin [12], can be simulated by equations with union, intersection and two-sided linear concatenation.

The first example of a nonregular unary solution of a language equation was given by Leiss [9], who constructed a single equation with concatenation and completementation, such that its unique solution is $\{a^n \,|\, \text{the octal notation of } n \text{ starts with 1, 2 or 3}\}$. The general case of such language equations was recently studied by Okhotin and Yakimova [14].

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 [10], represented by language equations with union, intersection and concatenation. These grammars have good practical properties (such as efficient parsing algorithms) and are surveyed in a recent article [13].

The expressive power of conjunctive grammars over a unary alphabet was one of the most important open problems in the area [10,13], and it was conjectured that only regular languages are generated. This conjecture has recently been disproved by Jeż [7], by constructing a grammar for the language $\{a^{4^n} \mid n \in \mathbb{N}\}$, as well as grammars for a large class of languages of an 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 restricted to exponential growth? Can their standard decision problems, such as emptiness, equivalence, etc., be effectively decided? In this paper we answer these questions by showing that the emptiness problem and other related decision problems are undecidable, while the growth is not recursively bounded, which by far exceeds earlier expectations on the power of this class of unary languages.

2 Conjunctive Grammars and Trellis Automata

Definition 1. *A conjunctive grammar [10] 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*

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

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

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

Definition 2. *Given a grammar G, the relation $\Longrightarrow$ of immediate derivability on the set of terms is defined as follows: (I) Using a rule $A \to \alpha_1 \& \ldots \& \alpha_n$, a subterm A of any term $\varphi(A)$ can be rewritten as $\varphi(A) \Longrightarrow \varphi(\alpha_1 \& \ldots \& \alpha_n)$. (II) A conjunction of several identical words can be rewritten by one such word:*

$\varphi(w\&\ldots\&w) \Longrightarrow \varphi(w)$, *for every* $w \in \Sigma^*$. *The language generated by a term* $\mathcal{A}$ *is* $L_G(\alpha) = \{w \mid w \in \Sigma^*,\ \mathcal{A} \Longrightarrow^* w\}$. *The language generated by the grammar is* $L(G) = L_G(S) = \{w \mid w \in \Sigma^*,\ S \Longrightarrow^* w\}$.

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 [4].

Definition 3. *For every conjunctive grammar* $G = (\Sigma, N, P, S)$, *the associated system of language equations [11] is a system of equations in variables* N, *in which each variable assumes a value of a language over* Σ, *and which contains the following equation for every variable* A*:*

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

Each instance 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 such 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 solutions, 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: $(\ldots, L_G(C), \ldots)_{C \in N}$ [11]. This representation by language equations constitutes an equivalent semantics of conjunctive grammars, and it is this semantics, and not the fairly artificial derivation, that accounts for the intuitive clarity of conjunctive and context-free grammars.

Let us give conjunctive grammars for some standard examples of non-context-free languages:

Example 1. [10] The following conjunctive grammar generates the language $\{wcw \mid w \in \{a,b\}^*\}$:

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

The nonterminal D generates the language $\{uczu \mid u, z \in \{a,b\}^*\}$. This is done as follows: the rules for D match a 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. An intersection with the language $\{ucv \mid u, v \in \{a,b\}^*, |u| = |v|\}$ generated by C completes the grammar.

Example 2. [7] The following conjunctive grammar with the start symbol A_1 generates the language $\{a^{4^n} \mid n \geqslant 0\}$:

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

To understand this grammar, note that each A_i generates the language of all words a^ℓ, such that the base-4 notation of ℓ is given by the digit(s) i followed by zeroes.

This construction can be generalized to the following:

Theorem 1 (Jeż, 2007 [7]). *For every $k \geqslant 2$ and for every finite automaton A over $\{0, \ldots, k-1\}$, there exists a conjunctive grammar over $\{a\}$ generating all strings a^n, such that the k-ary notation of n is in $L(A)$.*

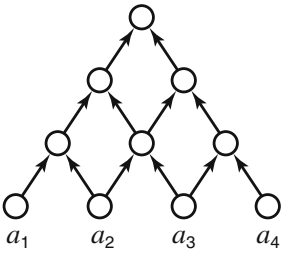
Let us now define an important subclass of conjunctive grammars, defined by analogy with linear context-free grammars. A conjunctive grammar is called *linear conjunctive* [10], if every rule it contains is either of the form $A \to u_1 B_1 v_1 \& \ldots \& u_n B_n v_n$, where $n \geqslant 1$, $u_i, v_i \in \Sigma^*$ and $B_i \in N$, or of the form $A \to w$, where $w \in \Sigma^*$. 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 has actually been known for almost thirty years before these grammars were introduced [12]: 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 [2], Ibarra and Kim [6], and others. Let us explain this concept following Culik et al. [2], who proposed it as a model of parallel computation in some electronic circuits.

A trellis automaton (TA), defined as a quintuple $(\Sigma, Q, I, \delta, F)$, processes an input string of length $n \geqslant 1$ using a uniform array of $n(n+1)/2$ processor nodes, as presented in the figure below. Each processor computes a value from a fixed finite set Q. The processors in the bottom row obtain their values directly from the input symbols using a function $I : \Sigma \to Q$. The rest of the processors 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 processor belongs to the set of accepting states $F \subseteq Q$.

Definition 4. *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 $L(M) = \bigcup_{q \in F} L_M(q)$.

Linear conjunctive grammars and trellis automata are computationaly equivalent:

Theorem 2 ([12]). *A language $L \subseteq \Sigma^+$ is recognised 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.*

Trellis automata over a unary alphabet recognize only regular languages, since in this case they behave as DFAs. Another property of TA which we often use is their closure under quotient with singletons [12], that is, whenever a language $L \in \Sigma^*$ is recognized by some trellis automaton M, then for every $a \in \Sigma$ the languages $a^{-1}L$ and La^{-1} are recognized by some TA M' and M''. These M', M'' can be effectively computed from M and a. From this the following result can be concluded:

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 $\widetilde{K} = \{u\}^{-1} \cdot K \cdot \{v\}^{-1}$ 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 Grammars in Positional Notation

Words over a unary alphabet $\{a\}$ can be regarded as natural numbers, and, accordingly, languages over $\{a\}$ represent sets of numbers. Our constructions are based upon representing these numbers in positional notation, that is, we fix a number $k \geqslant 2$, define the alphabet $\Sigma_k = \{0, 1, 2, \ldots, k-1\}$ of k-ary digits and consider words over this alphabet that represent numbers. Unary languages are accordingly represented by languages over Σ_k.

Let the empty string $\varepsilon \in \Sigma_k^*$ denote the number 0. No representation of a number shall begin with 0, that is, the set of valid representations of numbers is $\Sigma_k^* \setminus 0\Sigma_k^*$. Define the bijection $f_k : \Sigma_k^* \setminus 0\Sigma_k^* \leftrightarrow a^*$ as

$$f_k(w) = \{a^n \mid w \text{ read as } k\text{-ary notation represents } n\} .$$

We extend this function for languages in a usual way, preserving the bijectiveness.

Let us define a language-theoretic operator $\boxplus_k : \Sigma_k^* \times \Sigma_k^* \to \Sigma_k^*$, $\boxplus$ for simplicity, which represents addition of numbers in k-ary notation:

$$u \boxplus v = \{\text{the } k\text{-ary notation of } i+j \mid u \text{ is the } k\text{-ary notation of } i, \\ v \text{ is the } k\text{-ary notation of } j\} .$$

It is extended to languages in the standard way: $K \boxplus L = \{u \boxplus v \,|\, u \in K, v \in L\}$. Now this operation can be used in the context of language theory, e.g., if $k = 10$, then one can say that $9^+ \boxplus 2 = 10^*1$. Let us assume that concatenation has the highest precendence, followed by $\boxplus$ and then by intersection and union.

Define the corresponding subtraction operator on languages as follows:

$$K \boxminus L = \{\text{the } k\text{-ary notation of } i-j \mid i \geqslant j, \text{ the } k\text{-ary notation of } i \text{ is in } K, \\ \text{the } k\text{-ary notation of } j \text{ is in } L\} .$$

Consider systems of language equations of the form $X_i = \varphi(X_1, \ldots, X_n)$ (i=$1, \ldots, n$) over the alphabet Σ_k with $\cup, \cap, \boxplus$ as allowed operations and with

regular constants. Since all operations are monotone, any such system has a least solution of the same kind as in systems from Definition 3. The following tight correspondence between these two types of systems can be established:

Definition 5. *Let $k \geqslant 2$. Let $X_i = \varphi(X_1, \ldots, X_n)$ be a system of language equations over the alphabet $\{a\}$ using concatenation and Boolean operations. The corresponding k-ary system of language equations over Σ_k is $Y_i = \psi(Y_1, \ldots, Y_n)$, obtained by replacing each X_j with Y_j, each concatenation operator with $\boxplus$ and each constant language $L \subseteq a^*$ with the language $f_k^{-1}(L)$.*

Lemma 2. *Let $X_i = \varphi_i(X_1, \ldots, X_n)$ be a system of language equations over the alphabet $\{a\}$ and let $Y_i = \psi_i(Y_1, \ldots, Y_n)$ be the corresponding language equations over Σ_k. Then a vector of languages $(f_k(L_1), \ldots, f_k(L_n))$ is a solution of the former system if and only if the vector of languages $(L_1, \ldots, L_n)$ is a solution of the latter system. In particular, if $(f_k(L_1), \ldots, f_k(L_n))$ is the least solution of the former system, then $(L_1, \ldots, L_n)$ is the least solution of the latter system.*

Note that for all languages $K, L \subseteq \Sigma_k^*$, $f_k(K \boxplus L) = f_k(K) \cdot f_k(L)$. The rest of the proof is by a straightforward structural induction.

Example 3. Consider the conjunctive grammar from Example 2, consider the system of language equations over $\{a\}$ corresponding to it by Definition 3, and then consider the system over $\Sigma_4 = \{0, 1, 2, 3\}$ corresponding to that system according to Definition 5.

$$\begin{aligned} X_1 &= (X_2 \boxplus X_2 \cap X_1 \boxplus X_3) \cup \{1\} \\ X_2 &= (X_{12} \boxplus X_2 \cap X_1 \boxplus X_1) \cup \{2\} \\ X_3 &= (X_{12} \boxplus X_{12} \cap X_1 \boxplus X_2) \cup \{3\} \\ X_{12} &= X_3 \boxplus X_3 \cap X_1 \boxplus X_2 \end{aligned}$$

The least solution of the system is $(10^*, 20^*, 30^*, 120^*)$, cf. Example 2.

4 A Representation of Trellis Automata

Let us now show how the computation of a trellis automaton can be simulated by the class of language equations introduced in the previous section.

Lemma 3. *For every $k \geqslant 4$ and for every trellis automaton M over Σ_k, such that $L(M) \cap \mathtt{0}^* = \varnothing$, there exists and can be effectively constructed a resolved system of language equations over the alphabet Σ_k using operations $\cup$, $\cap$ and $\boxplus$ and regular constants, such that the least solution of this system contains a component $((\mathtt{1} \cdot L(M)) \boxminus \mathtt{1}) \cdot \mathtt{10}^*$.*

Proof (Outline). In this proof we abuse the notation of $\boxplus$ and $\boxminus$ by allowing their arguments and the result to have leading zeroes. We shall do this only for the second argument equal to $\mathtt{1}$. Under these conditions we define the result to have the same length as the first argument, e.g., $\mathtt{0100} \boxminus \mathtt{1}$ is deemed to be $\mathtt{0099}$.

We shall never use this notation in a context where these requirements cannot be fulfilled, that is, for $(k-1)^+ \boxplus \mathtt{1}$ and for $\mathtt{0}^* \boxminus \mathtt{1}$. This abused notation is used only in the text of the proof, while language equations strictly adhere to the definition.

Given a trellis automaton $M = (\Sigma_k, Q, I, \delta, F)$, we represent its input words $w \in \Sigma_k^*$ as words of the form $1(w \boxminus 1)10^m$, for all $m \geqslant 0$. This representation allows us to concatenate symbols to words both on the right and on the left by adding some appropriate numbers.

For a given M, we shall construct language equations with the set of variables X_q for $q \in Q$, and with an additional variable Y, such that their least solution is $X_q = L_q$, $Y = L$, where

$$L_q = 1\Big(\big(L_M(q) \setminus 0^*\big) \boxminus 1\Big)10^* = \{1w10^\ell \mid \ell \geqslant 0, w \notin 9^+, w \boxplus 1 \in L_M(q)\} \quad \text{and}$$
$$L = 1\Big(\big(L(M) \setminus 0^*\big) \boxminus 1\Big)10^* = \{1w10^\ell \mid \ell \geqslant 0, w \notin 9^+, w \boxplus 1 \in L(M)\}\,.$$

Let us define expressions λ_i and ρ_j, for $i, j \in \Sigma_k$, which depend upon the variables X_q, and which we use as building blocks for the equations for X_q.

$$\lambda_i(X) = 1i\Sigma_k^* \cap \bigcup_{i'} \Big((X \cap 1i'\Sigma^*) \boxplus 10^* \cap 2i'\Sigma_k^* \Big) \boxplus (k+i-2)0^* \text{ , for } i=0,1$$

$$\lambda_i(X) = 1i\Sigma_k^* \cap \bigcup_{i'} \Big((X \cap 1i'\Sigma^*) \boxplus 10^* \cap 2i'\Sigma_k^* \Big) \boxplus 1(i-2)0^* \text{ , for } i \geqslant 2$$

$$\rho_j(X) = \bigcup_{j'} \Big(\Big((X \cap 1\Sigma_k^* j'10^*) \boxplus 10^* \cap 1\Sigma_k^* j'20^* \Big) \boxplus (k+j-2)10^* \Big) \cap$$
$$\cap\, 1\Sigma_k^* j10^* \text{ for } j = 0,1$$

$$\rho_j(X) = \bigcup_{j'} \Big(\Big((X \cap 1\Sigma_k^* j'10^*) \boxplus 10^* \cap 1\Sigma_k^* j'20^* \Big) \boxplus 1(j-2)10^* \Big) \cap$$
$$\cap\, 1\Sigma_k^* j10^* \text{ for } 2 \leqslant j \leqslant k-2$$

$$\rho_{k-1}(X) = \bigcup_{j'} \Big(\Big((X \cap 1\Sigma_k^* j'10^*) \boxplus 10^* \cap 1\Sigma_k^* j'20^* \Big) \boxplus (k-3)10^* \Big) \cap$$
$$\cap\, 1\Sigma_k^*(k-1)10^*$$

Let us also define constants R_q, for all $q \in Q$, which are regular by Lemma 1.

$$R_q = 1\Big(\big((0^*(\Sigma_k \setminus 0) \cup (\Sigma_k \setminus 0)0^*) \cap L_M(q)\big) \boxminus 1\Big)10^*$$

Using this notation, the system of language equations is constructed as follows:

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

In these equations the sets R_q represent the starting part of X_q used to compose longer words. A word $w \in \Sigma^{\geqslant 2}$ belongs to $L_M(q)$ if and only if there are

states q', q'' such that $\delta(q', q'') = q$ and $\Sigma_k^{-1} w \in L_M(q'')$ and $w\Sigma_k^{-1} \in L_M(q')$. And so a word $1(w \boxminus 1)10^*$ should belong to X_q if and only if there are two witnesses belonging to $X_{q''}$ and $X_{q'}$ (with some additional constraints). This is specified in λ and ρ, respectively. These expressions represent adding digits at some specific positions, so that selected digits in the original word could be modified in the resulting word, while the rest of the digits remain the same. The main technical difficulty is to force the addition of digits at proper positions. This is achieved by adding the digits in two phases, and by checking the form of intermediate and final results using intersection with regular constants.

The correctness of the construction is stated as follows.

Main Claim. The least solution of the system is $(\ldots, L_q, \ldots, L)$.

The first to be established are the correctness statements for the expressions λ_i and ρ_j. For each λ_i, its value on any singleton of the form $\{1w10^m\}$ is characterized as follows:

Claim 1. For every word $1w10^m \in 1\Sigma_k^+ 10^* \setminus 1(k-1)^*10^*$,

$$\lambda_i(\{1w10^m\}) = \{1iw10^m\}.$$

Given a number with the notation $1w10^m$, the expression λ_i, adds a number notated $10^{|w|+1+m}$ to it to obtain $2w10^m$. The subsequent addition of $(k+i-2)0^{|w|+1+|m|}$ results in a number with the notation $1iw10^m$. The intersections used in λ_i ensure that no other additions are possible.

Since λ_j is an operation on languages defined as a superposition of $\cup$, $\cap$ and $\boxplus$, it is additive in the sense that $\lambda_i(K) = \bigcup_{w \in K} \lambda_i(\{w\})$ for every language K. Hence Claim 1 actually describes $\lambda_i(K)$ for every $K \subseteq 1\Sigma_k^+ 10^* \setminus 1(k-1)^*10^*$.

A similar statement is established for ρ_j:

Claim 2. For every word $u \in 1\Sigma_k^+ 10^* \setminus 1(k-1)^*10^*$,

$$\rho_j(\{u\}) = \begin{cases} \{1wj10^m\}, & \text{if } u = 1w10^{m+1} \text{ and } j = k-1\ , \\ \{1wj10^m\}, & \text{if } u = 1(w \boxminus 1)10^{m+1} \text{ and } j \neq k-1\ . \end{cases}$$

Now let us substitute the intended solution $(\ldots, L_q, \ldots, L)$ into λ_i and ρ_i. Then Claims 1 and Claim 2 easily imply the following equalities:

Claim 3. $\lambda_i(L_q) = 1(i(L_M(q) \setminus 0^*) \boxminus 1)10^*$.

Claim 4. $\rho_j(L_q) = 1((L_M(q) \setminus 0^*)(j+1 \mod k) \boxminus 1)10^*$.

The next claim is that the words in the intended solution $(\ldots, L_q, \ldots, L)$ belong to the corresponding components of every solution of the system.

Claim 5. For the least solution $(\ldots, S_q, \ldots)$ of the system and for every $q \in Q$, $L_q \subseteq S_q$.

For every $1w10^n \in L_q$ it has to be shown that $1w10^n$ is contained in S_q as well. The proof is done inductively on $|w|$, following the structure of the computation of M. The basis is given by the set R_q, which places in S_q all words $1w10^n$ for all w such that $w \boxplus 1$ has one nonzero digit, which is the last one or the first one. For the induction step, one has $w \boxplus 1 = iuj \in L_M(q)$ and one has to prove that $1w10^n$ is in S_q. The definition of a trellis automaton implies $iu \in L_M(q')$ and $uj \in L_M(q'')$ for some q', q'', such that $\delta(q', q'') = q$. Then the induction hypothesis gives that $1(iu \boxminus 1)10^{n+1} \in S_{q'}$ and $1(uj \boxminus 1)10^n \in S_{q''}$, and from these two words the equation for X_q produces $1w10^n \in S_q$, according to Claims 1 and 2.

Let us now show that the substitution of the intended solution $(\ldots, L_q, \ldots, L)$ into the system is contained in this intended solution.

Claim 6. For every $q \in Q$, $L_q \supseteq \varphi_q(\ldots, L_q, \ldots)$, where φ_q denotes the right-hand side of the equation for X_q.

Any word $1w10^n$ in $\varphi_q(\ldots, L_q, \ldots)$ is contained both in $\lambda_i(L_{q''})$ and in $\rho_j(L_{q'})$, for some q', q'' such that $\delta(q', q'') = q$. By Claim 4, $(w \boxplus 1)\Sigma_k^{-1} \in L_M(q')$, and by Claim 3, $\Sigma_k^{-1}(w \boxplus 1) \in L_M(q'')$. Then $w \boxplus 1 \in L_M(q)$, which yields the claim.

The proof of the main claim follows by the elementary properties of fixed points. Taking the componentwise inclusions of vectors

$$(\ldots, \varnothing, \ldots) \sqsubseteq (\ldots, L_q, \ldots) \sqsubseteq (\ldots, S_q, \ldots),$$

which holds by Claim 5, one can iteratively apply the right-hand sides of the system to each of the three components, obtaining the following limits:

$$(\ldots, S_q, \ldots) \sqsubseteq (\ldots, L_q, \ldots) \sqsubseteq (\ldots, S_q, \ldots).$$

This shows that the intended solution of the system is its least solution. □

Lemma 4. *For every $k \geqslant 4$ and for every trellis automaton M over Σ_k there exists and can be effectively constructed a resolved system of language equations over the alphabet Σ_k using the operations $\cup$, $\cap$ and $\boxplus$ as well as regular constants, such that its least solution contains a component $1 \cdot L(M)$.*

Proof. For every $j \in \Sigma_k$, consider the language $L(M) \cdot \{j\}^{-1}$. By the closure properties of trellis automata, this language is generated by a trellis automaton M_j. Then, by Lemma 3, there exists a system of language equations, such that one of its variables, Y_j, represents the language $(L(M) \cdot \{j\}^{-1}) \boxminus 1$.

Let us combine these equations for all j into a single system, and add a new equation

$$Z = \bigcup_{j=0}^{k-1} (Y_j \cap 1\Sigma^*1) \boxplus (1j \boxminus 1).$$

This equation uses the same technique as in Lemma 3. The value of Z is $L(M)$. □

Theorem 3. *For every $k \geqslant 4$ and for every trellis automaton M over Σ_k, such that no words in $L(M)$ start with 0, there exists and can be effectively constructed a resolved system of language equations over the alphabet Σ_k using the operations $\cup$, $\cap$ and $\boxplus$ as well as regular constants, such that its least solution contains a component $L(M)$.*

Proof. For every $i \in \Sigma_k \setminus \{0\}$, the language $\{i\}^{-1} \cdot L(M)$ is generated by a certain trellis automaton. By Lemma 4, there is a system of language equations, such that one of its variables, Z_i, represents the language $\{i\}^{-1} \cdot L(M)$.

Combine these systems and add a new variable T with the following equation:

$$T = \bigcup_{i,i'} \Big((Z_i \cap 1i'\Sigma_k^*) \boxplus (i-1)0^* \Big) \cap ii'\Sigma_k^*.$$

The right-hand side of this equation evaluates to $L(M)$ on the least solution, which is shown by the same method as in Lemmata 3 and 4. □

We now translate the above result to the languages generated by the unary conjunctive grammars. We need a small technical lemma to handle the cases of $k = 2, 3$.

Lemma 5. *Let $\ell = k^n$ for some natural $n > 0$. Then for every languages L, L' such that $f_k(L) = f_\ell(L')$, L is linear conjunctive if and only if L' is linear conjunctive. Given a trellis automaton for either of the languages, a trellis automaton for the other language can be effectively constructed.*

The proof is by a straightforward grouping of digits, and it is omitted.

Theorem 4. *For every $k \geqslant 2$ and for every trellis automaton M over $\Sigma_k = \{0, 1, \ldots, k-1\}$, such that no words in $L(M)$ start with 0, there exists and can be effectively constructed a conjunctive grammar generating $f_k(L(M))$.*

Proof. For big enough k, that is $k \geqslant 4$, by Theorem 3, there exists a resolved system of language equations over Σ_k with regular constants that defines $L(M)$. According to Lemma 2, there exists a corresponding system over $\{a\}$, which uses constants of the form $f_k(K)$ for regular languages $K \subseteq \Sigma_k^*$. But, according to Theorem 1, every such constant is generated by a conjunctive grammar. Combining these grammars with the system of equations, the requested conjunctive grammar is obtained.

For $k < 4$ we use Lemma 5 and the already proved results for $k = 4, 9$. □

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, one represents its computations as words 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 [5] 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 ([12]). *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 [12] and hence by conjunctive grammars of the general form [10]. 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 words, on which all known encodings of $\mathrm{VALC}(T)$ rely. Contrary to this intuition, Theorem 4 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 $\Sigma_k = \{0, \ldots, k-1\}$, for a suitable k, and that $0 \in \Omega$, so that no word 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 we can use Theorem 4 to obtain the following result:

Lemma 6. *For every Turing machine T there exist and can be effectively constructed conjunctive grammars G and G' over the alphabet $\{a\}$, such that $L(G) = f_k(\mathrm{VALC}(T))$ and $L(G') = f_k(\mathrm{INVALC}(T))$.*

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

Theorem 5. *For every fixed unary conjunctive language $L_0 \subseteq a^*$ there is no algorithm to decide whether a given conjunctive grammar over $\{a\}$ generates L_0.*

Proof. 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. Depending on the form of L_0, there are 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 $f_k(\mathrm{VALC}(T))$. 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 four new rules: $S \to S_0$, $S \to S_T A$, $A \to aA$ and $A \to \varepsilon$. Then it can be proved that $L(G) = L_0$ if and only if $L(G_T) = \varnothing$, and the latter is equivalent to $L(T) = \varnothing$.

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 $\mathrm{INVALC}(T)$. Define $\mathrm{INVALC}(T)$ over a p-letter alphabet and consider the language $f_p(\mathrm{INVALC}(T) \cdot \mathtt{0})$, which is generated by some conjunctive grammar $G'_T = (\{a\}, N'_T, P'_T, S'_T)$. 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: $S \to S_0 \& B$, $S \to a^\ell S'_T$, $B \to a^i$ (for all $0 \leqslant i < \ell$), $B \to a^{\ell+i}C$ (for all $1 \leqslant i < p$), $C \to a^p C$ and $C \to \varepsilon$. Note that $L_G(B) = a^* \setminus a^\ell(a^p)^*$. For this grammar one can establish that $L(G) = L_0$ if and only if $\mathrm{INVALC}(T) = \Sigma_p^* \setminus \mathtt{0}\Sigma_p^*$, which in turn holds if and only if $L(T) = \varnothing$.

In each case we have shown that the emptiness problem of Turing machines would be decidable, which forms a contradiction. □

If L_0 is not generated by a conjunctive grammar, then this problem becomes trivial. Hence, the following characterization is obtained:

Corollary 1. *For different constant languages $L_0 \subseteq a^*$, the problem of testing whether a given conjunctive grammar over $\{a\}$ generates L_0 is either Π_1-complete or trivial.*

Theorem 6. *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 (a sketch). Given a Turing machine T, construct another TM T' that recognizes $\{\varepsilon\}$ if $L(T) \neq \varnothing$, and $\varnothing$ otherwise. Construct a conjunctive grammar G for $f_k(\mathrm{VALC}(T')) \cdot \{a^{k^n} \mid n \geqslant 0\}$. Then $L(G)$ is either nonregular (if $L(T) \neq \varnothing$) or empty. □

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 word can be decided in cubic time [10], 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 a function

$g(n) = i_n$. Obviously, the growth of every regular language is linearly bounded. The example of a conjunctive grammar for the language $\{a^{4^n} \mid n \geqslant 0\}$ [7], 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, e.g., polynomial?

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

Theorem 7. *For every recursively enumerable set of natural numbers X there exists a conjunctive grammar G over an alphabet $\{a\}$, such that the growth function of $L(G)$ is greater than that of X at any point.*

Proof. Let T be a Turing machine, which recognizes the set $X = \{i_1, i_2, \ldots, i_j, \ldots\}$, where $0 \leqslant i_1 < i_2 < \ldots < i_j < \ldots$, and the numbers are given to it in a binary notation. Consider the language $\mathrm{VALC}(T)$, which contains words $w_n = f_2^{-1}(a^n)\#C_T(n)$. By Lemma 6, there exists a conjunctive grammar G over an alphabet $\{a\}$ that generates $L = f_k(\mathrm{VALC}(T))$ for some $k \geqslant 2$.

Let $g(n)$ be the growth function of L. It is sufficient to show that $g(j) \geqslant i_j$ for each $j \geqslant 1$. To see this, consider the values $g(1), g(2), \ldots, g(j)$. At least one of them describes a computation $w_{j'}$ for $j' \geqslant j$. Since g is an increasing function, we obtain $g(j) \geqslant f_k(w_{j'}) > j' \geqslant j$. □

Note that this quick-growing language is bound to be computationally very easy, as the upper bound of parsing complexity for conjunctive grammars is $DTIME(n^3) \cap DSPACE(n)$ [10,13].

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 of $L(G)$ satisfies is $g(n) = \Theta(n^2)$.*

Proof (a sketch). Consider the set of numbers $X = \{(2^m + 3i) \cdot 2^m \mid m \geqslant 0,\ 2^m \leqslant 2^m + 3i < 2^{m+2}\}$. Let $g(n)$ denote the n-th largest element of X; this is the growth function of the corresponding unary language $L = \{a^n \mid n \in X\}$. The set of binary notations of the numbers in X is

$$f_2^{-1}(L) = \left\{\, 1w0^m \;\middle|\; |w| = m-1 \text{ or } |w| = m, \text{ and } f_2(w) \text{ divides by } 3 \,\right\}.$$

This is clearly a linear context-free language, hence L is conjunctive by Theorem 4. Let us prove that $n^2 \leqslant g(n) \leqslant 4n^2$.

It is not hard to prove that for any number $n = 2^m + j$, where $0 \leqslant j < 2^m$, it holds that $g(2^m + j) = (2^m + 3j)2^m$; in particular, $g(2^m) = 2^{2m}$. Then, to see that $g(n) \geqslant n^2$, consider $g(2^m + j)$ for $0 \leqslant j < 2^m$. We have

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

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

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

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

This construction can be generalized to obtain the following result:

Theorem 8. *For every rational number $p/q \geqslant 1$ there exists a conjunctive grammar G over an alphabet $\{a\}$, such that the growth function of $L(G)$ is $g(n) = \Theta(n^{p/q})$.*

7 Conclusion

We can conclude that though we have established that the growth of unary conjunctive grammars can be as high as theoretically possible, we still have no means of proving that some particular unary languages cannot be represented by conjunctive grammars, and hence the class of conjunctive languages still could not be separated from $NSPACE(n)$. Inventing a method for producing such nonrepresentability results for unary conjunctive grammars is left as an open problem.

References

1. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47, 149–158 (1986)
2. Culik II, K., Gruska, J., Salomaa, A.: Systolic trellis automata, I and II. International Journal of Computer Mathematics 15, 195–212 (1984) and 16, 3–22 (1984)
3. Domaratzki, M., Pighizzini, G., Shallit, J.: Simulating finite automata with context-free grammars. Information Processing Letters 84, 339–344 (2002)
4. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. Journal of the ACM 9, 350–371 (1962)
5. Hartmanis, J.: Context-free languages and Turing machine computations. Proceedings of Symposia in Applied Mathematics 19, 42–51 (1967)
6. Ibarra, O.H., Kim, S.M.: Characterizations and computational complexity of systolic trellis automata. Theoretical Computer Science 29, 123–153 (1984)
7. Jeż, A.: Conjunctive grammars can generate non-regular unary languages. In: DLT 2007, Turku, Finland, July 3–6, 2007 (to appear)
8. Jeż, A., Okhotin, A.: Language equations with addition in positional notation, TUCS Technical Report No 824, Turku Centre for Computer Science, Turku, Finland (June 2007)
9. Leiss, E.L.: Unrestricted complementation in language equations over a one-letter alphabet. Theoretical Computer Science 132, 71–93 (1994)
10. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)
11. Okhotin, A.: Conjunctive grammars and systems of language equations. Programming and Computer Software 28, 243–249 (2002)
12. Okhotin, A.: On the equivalence of linear conjunctive grammars to trellis automata. Informatique Théorique et Applications 38(1), 69–88 (2004)
13. Okhotin, A.: Nine open problems for conjunctive and Boolean grammars. Bulletin of the EATCS 91, 96–119 (2007)
14. Okhotin, A., Yakimova, O.: On language equations with complementation. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 420–432. Springer, Heidelberg (2006)