Information and Computation 209 (2011) 344–352

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Decidability of operation problems for T0L languages and subclasses

Henning Bordihn [a,*], Markus Holzer [b,1], Martin Kutrib [b]

[a] Institut für Informatik, Universität Potsdam, August-Bebel-Straße 89, 14482 Potsdam, Germany
[b] Institut für Informatik, Universität Giessen, Arndtstraße 2, 35392 Giessen, Germany

ARTICLE INFO

Article history:
Available online 18 November 2010

Keywords:
L systems
Operation problem
Decidability
Unary languages

ABSTRACT

We investigate the decidability of the operation problem for T0L languages and subclasses. Fix an operation on formal languages. Given languages from the family considered (0L languages, T0L languages, or their propagating variants), is the application of this operation to the given languages still a language that belongs to the same language family? Observe, that all the Lindenmayer language families in question are anti-AFLs, that is, they are not closed under homomorphisms, inverse homomorphisms, intersection with regular languages, union, concatenation, and Kleene closure. Besides these classical operations we also consider intersection and substitution, since the language families under consideration are not closed under these operations, too. We show that for all of the above mentioned language operations, except for the Kleene closure, the corresponding operation problems of 0L and T0L languages and their propagating variants are not even semidecidable. The situation changes for unary 0L languages. In this case we prove that the operation problems with respect to Kleene star, complementation, and intersection with regular sets are decidable.

1. Introduction

Elementary undecidable questions for formal language families appeared first in [1], where it was shown that the family of languages defined by context-free grammars is too wide to admit a decidable theory for language equivalence. The same holds true for the family of 0L (E0L) languages generated by (extended) context independent Lindenmayer systems (L systems for short). This grammar formalism has been introduced in order to describe the development of lower organism [7,8]. One of the main reasons why L systems are interesting is that they form a parallel counterpart to sequential rewriting mechanisms such as context-free grammars. Moreover, they can be considered as finite substitutions over a free monoid, which are iteratively applied to a designated element of the monoid, the so-called axiom of the system. The basic properties of the derivation in L systems in contrast to sequential rewriting mechanisms can be summarized as follows:

- In every derivation step, all symbols in the sentential form have to be rewritten (in parallel).
- There is no distinction between terminal and nonterminal symbols. Therefore, we call those systems *pure* L systems which is in line with the theory of pure grammars [9].
- L systems have a word as axiom instead of a symbol as in the case of Chomsky grammar, or instead of a set of words as in the case of pure grammars.

It is known that for 0L and T0L languages, for example, inclusion, finiteness, and regularity are undecidable [3,11]. The family of T0L languages is generated by context independent tabled L systems. Roughly speaking, these are 0L systems with several production sets, which are also called tables, see, e.g., [11]. On the other hand, some related questions such as membership are decidable.

* Corresponding author.
E-mail addresses: henning@cs.uni-potsdam.de (H. Bordihn), holzer@informatik.uni-giessen.de (M. Holzer), kutrib@informatik.uni-giessen.de (M. Kutrib).

[1] Most of the work was done while the author was with Institut für Informatik, Technische Universität München, Boltzmannstraße 3, 85748 Garching bei München, Germany.

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2010.11.008

Another important class of decision problems can be stated as follows. Fix a family of languages and operations thereon. Given languages from this family, is it decidable or semidecidable whether or not the application of the operation to the given languages leads out of this family? In the forthcoming this problem is referred to as the *operation problem*. From an implementation point of view, the operation problem is related to the question whether, for example, a parser or acceptor for a given language can be decomposed into several simpler parsers. Advantages of simpler parsers, whose combination according to the operation is equivalent to the given device, are obvious. For example, the total size of the simpler devices could be smaller than the given parser, the verification is easier, etc. So, there is a natural interest in efficient decomposition algorithms. From this point of view, the complexity of the converse question, whether the composition of languages yields a given language, is interesting. The operation problem can be seen as a weaker class of such problems. Of course, the operation problem makes only sense for language operations under which the family under consideration is *not* closed, since the aforementioned problem becomes trivially decidable otherwise.

The operation problem for families of languages generated by L systems has been investigated only for the *union* of 0L and propagating (that is, non-erasing) 0L languages [4]—also the union problem for unary 0L languages was considered there. It was shown that in general the union problem for 0L languages is undecidable, while for the restricted variants of PD0L and unary 0L languages the problem becomes decidable. Other operations as well as tabled L systems have not been considered yet. The aim of the present paper is to investigate the operation problem for the families of 0L and T0L languages and their propagating variants to a large extent. The notation on 0L and T0L systems and their propagating variants is introduced in Section 2. Besides AFL operations, except the Kleene closure, we also consider intersection and substitution by languages from the families in question. For all of the above mentioned operations we prove non-semidecidability. The results on the Boolean operations can be found in Section 3, while the remaining operations are treated in Section 4. Similarly to the approach in [4], we show how to reduce Post's Correspondence Problem (PCP) to the problem in question [10]. Compared to some other proofs in the literature the constructions presented here and the argumentation are more involved. This is due to the fact that we have to deal with pure language families, which do not allow to hide any sentential form during the derivation process, as in context-free grammars or extended context-independent L systems. In Section 5 we show that the situation changes for *unary* 0L languages. In this case we show that the operation problems with respect to Kleene star, complementation, and intersection with regular sets are decidable. The results complement the decidability of the union problem for unary 0L languages given in [4]. Finally, we summarize our results and state some open problems in the last section.

2. Preliminaries and definitions

The reader is assumed to be familiar with the basic notions of formal language theory as contained, for example, in [11,12]. In the present paper we will use the following notational conventions. An alphabet is any non-empty finite set, its elements are called letters or symbols. For an alphabet V let V^+ and V^* denote the free semi-group and free monoid, respectively, generated by V. The unit element of V^* is the empty word denoted by λ. The reversal of a word $w \in V^*$ is denoted by w^R, and for the length of w we write $|w|$. For the number of occurrences of a symbol a in w we use the notation $|w|_a$. Generally, for a singleton set $\{a\}$ we simply write a. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*. The powerset of a set S is denoted by 2^S.

Let U and V be two alphabets and σ be a mapping from V into 2^{U^*}, that is, $\sigma(a) \subseteq U^*$, for all $a \in V$. The extension of σ to domain V^* defined by $\sigma(\lambda) = \{\lambda\}$ and $\sigma(w_1 \cdot w_2) = \sigma(w_1) \cdot \sigma(w_2)$, for $w_1, w_2 \in V^*$, is called a *substitution*. If $\sigma(a)$ is a finite set for all $a \in V$, then σ is a finite substitution. If $U = V$, then σ is called a substitution over V.

Now we give the formal definition of the L systems which will be considered in this paper.

Definition 1

1. A T0L system is a tuple $G = (V, P_1, P_2, \ldots, P_r, \omega)$, where r is a positive integer, V is an alphabet, $\omega \in V^+$ is the axiom, and P_i, for $1 \leq i \leq r$, is a finite subset of $V \times V^*$ such that for every $a \in V$, there is a word $v \in V^*$ with $(a, v) \in P_i$. The sets P_i are called the tables of G.
2. A T0L system is *propagating* (a PT0L system) if all tables of G are finite subsets of $V \times V^+$.
3. A 0L system is a T0L system with only one table, that is, $r = 1$.
4. A 0L system is *propagating* (a P0L system) if the only table of G is a finite subset of $V \times V^+$.

The elements of the tables are called rules and define how a symbol of the current sentential form may be rewritten. Such as in the case of phrase structure grammars we usually write $a \to v$ for (a, v) in P. Since in a single step of a T0L system all symbols are rewritten in parallel according to one of its tables, any table of a T0L system can be viewed as a finite substitution over V. More precisely, with every table $P \subseteq V \times V^*$ we associate the finite substitution σ_P defined by $\sigma_P(a) = \{ v \mid (a, v) \in P \}$. Now, we can define the language generated by a T0L system.

Definition 2. Let $r \geq 1$ and $G = (V, P_1, P_2, \ldots, P_r, \omega)$ be a T0L system. A word $x \in V^+$ directly derives a word $y \in V^*$ if there is i with $1 \leq i \leq r$, such that $y \in \sigma_{P_i}(x)$. We write $x \Rightarrow y$ in this case. The language $L(G)$ *generated* by G is defined to be the set $L(G) = \{ w \in V^* \mid \omega \Rightarrow^* w \}$, where $\Rightarrow^*$ refers to the reflexive, transitive closure of the derivation relation $\Rightarrow$.

For $X \in \{\text{P0L}, \text{0L}, \text{PT0L}, \text{T0L}\}$, a language is said to be an X language if there is an X system generating it. Let V be an alphabet and σ a substitution over V. If, for all letters $a \in V$, the set $\sigma(a)$ is an X language, then σ is called an X substitution. By definition, every P0L language is also a 0L, PT0L as well as a T0L language, and both every 0L and every PT0L language is also a T0L language.

In the remainder of this section we introduce the necessary notation from computability theory. A problem is called *decidable*, if there is a Turing machine that will halt on all inputs and, given an encoding of any instance of the question, will compute the correct answer "yes" or "no" for the instance. Otherwise the problem is *undecidable*. The problem is *semidecidable* or *recursively enumerable*, if the Turing machine halts on all instances for which the answer is "yes". Otherwise the problem is *non-semidecidable*. For example, the well-known halting problem is undecidable. But it is easy to see that it is semidecidable.

We will prove the non-semidecidability of operation problems for T0L systems and subclasses thereof by reduction of Post's Correspondence Problem (PCP) to the problem in question. Formally, the definition of PCP reads as follows, see, e.g., [12]: Let n be a positive integer, V be an alphabet containing at least two letters, and

$$I = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$$

be a finite set of pairs from $V^* \times V^*$. As n and V are implicitly specified when I is given, the set I determines an instance of the PCP. The PCP I has a solution if and only if there is a sequence of integers $i_1, i_2, \ldots, i_k$ with $k \geq 1$, $1 \leq i_j \leq n$, for $1 \leq j \leq k$, such that

$$u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}.$$

As an example, assume that $V = \{0, 1\}$ and furthermore, let the PCP instance be $I = \{(1, 111), (10111, 10), (10, 0)\}$. A solution to this instance of the PCP is the sequence 2, 1, 1, 3 obtaining $10111 \cdot 1 \cdot 1 \cdot 10 = 10 \cdot 111 \cdot 111 \cdot 0$. It is well-known that the PCP is undecidable [10]. Simply by enumerating all possibilities it is easy to see that it is still semidecidable. On the other hand, the problem to determine whether a PCP has *no* solution cannot be semidecidable. Otherwise the PCP would be decidable. So, to be more precise, we will prove our undecidability results by reduction of the question whether a PCP has no solution to the operation problem.

Our undecidability results (for binary operations) read as follows, where $\circ$ refers to the operation in question:

Let X and Y be in $\{\text{P0L}, \text{0L}, \text{PT0L}, \text{T0L}\}$. Given two X systems G_1 and G_2, it is non-semidecidable whether $L(G_1) \circ L(G_2)$ is a Y language.

More precisely, we show that the semidecidability of the operation problem under consideration would imply the semidecidability of the question whether a PCP has no solution. Because of the aforementioned inclusion structure of L language families it suffices to prove that, given an instance of the PCP, we can construct two P0L systems G_1 and G_2, such that the language $L(G_1) \circ L(G_2)$ is P0L if the PCP has *no* solution for the given instance, and if the PCP does have a solution, the resulting language is not even a T0L language. The results and proof structures for unary operations are given analogously. Our decidability results for unary 0L languages are obtained by deriving corresponding algorithms.

3. Boolean operations

In this section we consider the operation problem of the family of T0L languages and subclasses with respect to Boolean operations. Since there are regular languages which are not generated by any T0L system, in addition, we consider the intersection with regular languages. First we consider the union operation problem with respect to the four classes of L systems defined in the previous section.

Theorem 3. *Let X and Y be in $\{P0L, 0L, PT0L, T0L\}$. Given two X systems G_1 and G_2, it is non-semidecidable whether the union $L(G_1) \cup L(G_2)$ is a Y language.*

Proof. We prove the non-semidecidability by reducing Post's Correspondence Problem. Let $I = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ be a finite set of pairs from $\{a, b\}^* \times \{a, b\}^*$, an instance of the PCP, and let $\Sigma = \{a, b, c\}$. For technical reasons, we assume (x, x) be not contained in I, that is, we exclude instances for which the PCP is trivially decidable. We consider the P0L system

$$G_1 = (\{S, A, B, C, D, \#\} \cup \Sigma, P_1, S),$$

where P_1 is the union of the following sets of rules:

$$R_1 = \{S \to \#\#, S \to a^3 b^3 c^3, S \to A\},$$
$$R_2 = \{\, A \to xAx \mid x \in \Sigma \,\},$$
$$R_3 = \{\, A \to xBy \mid x, y \in \Sigma, x \neq y \,\} \cup \{\, B \to xBy \mid x, y \in \Sigma \,\},$$

$$R_4 = \{ A \to xC \mid x \in \Sigma \} \cup \{ B \to xC \mid x \in \Sigma \} \cup \{ C \to xC \mid x \in \Sigma \},$$
$$R_5 = \{ A \to Dx \mid x \in \Sigma \} \cup \{ B \to Dx \mid x \in \Sigma \} \cup \{ D \to Dx \mid x \in \Sigma \},$$
$$R_6 = \{B \to \#, C \to \#, D \to \#, \# \to \# \},$$

and

$$R_7 = \{ x \to x \mid x \in \Sigma \}.$$

The derivation process starts necessarily with an application of a rule from R_1. As only upper case letters can be replaced non-identically, the derivation can be continued only by rewriting A. As long as only rules from R_2 are used, strings in the set

$$K_1 = \{ wAw^R \mid w \in \Sigma^* \}$$

are obtained, and any string in K_1 can be generated. After the symbol A is rewritten by some rule in R_3, R_4, or R_5, the sets

$$K_2 = \{ wBw' \mid w, w' \in \Sigma^*, |w| = |w'|, w' \neq w^R \},$$
$$K_3 = \{ wCw' \mid w, w' \in \Sigma^*, |w| > |w'| \},$$

and

$$K_4 = \{ wDw' \mid w, w' \in \Sigma^*, |w| < |w'| \}$$

are generated by further applications of R_3, R_4, or R_5, respectively. Finally, with the help of R_6 we obtain

$$K_5 = \{ w\#w' \mid w, w' \in \Sigma^*, w' \neq w^R \}.$$

Since no further sentential forms can be derived, we have

$$L_1 = L(G_1) = \bigcup_{i=1}^{5} K_i \cup \{\#\#, a^3b^3c^3, S\}.$$

Next, we construct a P0L system G_2 dependent on the given instance of the PCP:

$$G_2 = (\{S, \#, a, b\}, P_2, S),$$

with

$$P_2 = \{ S \to u_i\#v_i^R \mid 1 \leq i \leq n \} \cup \{\# \to u_i\#v_i^R \mid 1 \leq i \leq n \} \cup \{a \to a, b \to b\},$$

and set $L_2 = L(G_2)$.

Now we consider the language $L_1 \cup L_2$ in more detail, and distinguish two cases, namely whether the PCP used in the construction of the L system G_2 has a solution or not.

Case 1. The PCP has no solution for the instance I. Then $L_2 \subset L_1$, hence $L_1 \cup L_2 = L_1$ is a P0L language.

Case 2. If the PCP has a solution for the instance I, then we claim that $L_1 \cup L_2$ cannot be generated by any T0L system. Assume the contrary and let $G = (V, P, \omega)$ be some T0L system with $L(G) = L_1 \cup L_2$.

First, we observe that ## is the only word in $L_1 \cup L_2$ which is of the form zz, for some word z. Therefore, in any table $\# \to \#$ is the only rule in P for the symbol #. The rule $\# \to \lambda$ would imply that λ belongs to $L_1 \cup L_2$, a contradiction.

As $a^3b^3c^3$ is contained in $L_1 \cup L_2$, in any table for any $x \in \Sigma$, the rules $x \to v$ are so that $|v|_X = 0$, for all $X \in \{S, A, B, C, D, \#\}$. Otherwise a word with at least three occurrences of X could be derived, which contradicts the structure of the words in $L_1 \cup L_2$. Next, $a^3b^3c^3$ is the only word in $L_1 \cup L_2$ over the alphabet Σ. In conclusion, $x \to x$ is the only possible rule, for any $x \in \Sigma$.

Now, let $i_1, i_2, \ldots, i_k$ be a solution of the PCP for the instance I, and let $y = u_{i_1}u_{i_2}\ldots u_{i_k}$. Then $L_0 = \{ y^m\#(y^R)^m \mid m \geq 1 \} \subseteq L_2$. Therefore, for all long enough $v \in L_0$, there are words z, z', and a derivation $\omega \Rightarrow^* zTz' \Rightarrow v$ (recall that $\# \to \#$ is the only rule in P for the symbol #) according to G such that $zTz' \in L_1$ with $T \in \{A, B, C, D\}$. Clearly, $T \neq B$ since otherwise $zBz' \in L_1$ implies $|z| = |z'|$ and $z' \neq z^R$. So, $zBz' \Rightarrow v$ is impossible.

Let $T \in \{A, C, D\}$. As $czTz'c$ is an element of L_1 either, also the derivation

$$\omega \Rightarrow^* czTz'c \Rightarrow cy^m\#(y^R)^mc$$

is possible according to G, but $cy^m\#(y^R)^m c$ belongs neither to L_1 nor to L_2, which is a contradiction to our assumption $L(G) = L_1 \cup L_2$. Hence, $L_1 \cup L_2$ is not generated by any T0L system.

The systems G_1 and G_2 are P0L systems and, therefore, of any type from the set {P0L, 0L, PT0L, T0L}. We have shown that the union $L(G_1) \cup L(G_2)$ is of any type from {P0L, 0L, PT0L, T0L}, if and only if the PCP has no solution. We conclude that it is non-semidecidable whether the union $L(G_1) \cup L(G_2)$ is a language of any type from {P0L, 0L, PT0L, T0L}. This proves the theorem. □

In the previous proof we have used the fact, that the union of two P0L languages may not even be a T0L language. At first glance, this seems to be not spectacular. Nevertheless, it shows that these Lindenmayer language families obey some very strong non-closure properties. The finite languages $\{a\}$ and $\{a^3\}$ are one of the simplest examples demonstrating the non-closure of P0L languages under union. Obviously, these languages are generated by the systems $G_1 = (\{a\}, P, a)$ and $G_2 = (\{a\}, P, a^3)$, where the finite substitution P contains the single (propagating) rule $a \to a$ only. Next, we argue that the union of these two languages, i.e., the finite language $\{a, a^3\}$, is *not* a T0L language. We argue as follows: Assume to the contrary that there is a T0L system $G = (V, P_1, P_2, \ldots, P_r, \omega)$, for some $r \geq 1$, generating the finite language $\{a, a^3\}$. Without loss of generality we may assume that $V = \{a\}$. Then we distinguish two cases: (1) either $\omega = a \Rightarrow a^3$ or (2) $\omega = a^3 \Rightarrow a$. In the former case we must have a table P_i, for some $1 \leq i \leq r$, satisfying $a^3 \in \sigma_{P_i}(a)$. This gives a contradiction, because then also a^9 must be a member of the generated language, since $\omega = a \Rightarrow a^3 \Rightarrow a^9$. In the latter case we also obtain a contradiction. Let $\omega = a^3$ and $a^3 \Rightarrow a$. Then there is a table P_j with $1 \leq j \leq r$ satisfying $\{\lambda, a\} \subseteq \sigma_{P_j}(a)$. Thus λ is an element of the generated language, since $\omega = a^3 \Rightarrow a \Rightarrow \lambda$ is a valid derivation. Hence, $\{a, a^3\}$ is *not* a T0L language. For the other operations considered in this paper, similar examples can be found:

Intersection: Consider the P0L system $G_1 = (\{a\}, P_1, a)$ with $P_1 = \{a \to a^3\}$ generating the language $L(G_1) = \{ a^{3^n} \mid n \geq 0 \}$, and the P0L system $G_2 = (\{a, b, c\}, P_2, b)$ with $P_2 = \{a \to a, b \to c, b \to a, c \to b, c \to a^3\}$ generating the finite language $L(G_2) = \{a, a^3, b, c\}$. Then $L(G_1) \cap L(G_2) = \{a, a^3\}$ which was previously shown not even to be a T0L language.

Intersection with regular sets: Let G_1 be the P0L system described above. Then $L(G_1) \cap R$ equals the non-T0L language $\{a, a^3\}$, if we set $R = \{a, a^3\}$.

(Non-erasing) homomorphism and substitution: Let $G = (\{a, b\}, P, b)$ with $P = \{a \to b, b \to a\}$ generating the language $L(G) = \{a, b\}$. But then the homomorphism $h : \{a, b\}^* \to \{a\}^*$ defined by $h(a) = a$ and $h(b) = a^3$ applied to $L(G)$ gives again our non-T0L language $\{a, a^3\}$. Note that h is also a P0L substitution. Thus, giving also an example for non-closure under the weakest form of substitution, namely P0L substitution.

Inverse homomorphism: Set $G = (\{a, b, c\}, P, a)$ with $P = \{a \to b, a \to c, b \to a, b \to c^3, c \to c\}$. Then it is easy to see that $L(G) = \{a, b, c, c^3\}$. Define the homomorphism $h : \{a\}^* \to \{a, b, c, d\}^*$ *via* $h(a) = c$. Then $h^{-1}(L(G)) = \{a, a^3\}$, which is a not a T0L language.

Concatenation: Define $G_1 = (\{a\}, P_1, aa)$ with $P_1 = \{a \to a\}$ generating the language $L(G_1) = \{aa\}$ and $G_2 = (\{a\}, P_2, aa)$ with $P_2 = \{a \to aa\}$ generating the language $L(G_2) = \{ a^{2^n} \mid n \geq 1 \}$. Then we obtain $L(G_1) \cdot L(G_2) = \{ a^{2^n+2} \mid n \geq 1 \}$. But the concatenation $L(G_1) \cdot L(G_2)$ is not even a T0L language, since no table can contain the rule $a \to \lambda$ and, thus, the axiom must be a^4. Further, if there is a rule $a \to a^k$, for some $k \geq 2$, then the word a^{4k} is generated whose length is a multiple of four. But the length of all words in $L(G_1) \cdot L(G_2)$ longer than four are not divisible by four.

We continue our investigation with the intersection operation and base our construction on that of Theorem 3, thus reducing the intersection problem to the undecidability problem of the union problem.

Theorem 4. *Let X and Y be in {P0L, 0L, PT0L, T0L}. Given two X systems H_1 and H_2, it is non-semidecidable whether the intersection $L(H_1) \cap L(H_2)$ is a Y language.*

Proof. Let $I = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ be a finite set from $\{a, b\}^* \times \{a, b\}^*$, an instance of the PCP, and let G_1 and G_2 be the P0L systems as in the proof of Theorem 3. Let P_1 be the sole table of G_1. We construct the P0L system

$$H_1 = (\{S, A, B, C, D, E, \#, a, b, c\}, P_1', S),$$

where $P_1' = P_1 \cup R_8$ with

$$R_8 = \{ S \to u_i E v_i^R \mid 1 \leq i \leq n \} \cup \{ E \to u_i E v_i^R \mid 1 \leq i \leq n \} \cup \{E \to \#\}.$$

Then

$$\begin{aligned} L(H_1) = L(G_1) \cup \{ u_{i_1} u_{i_2} \ldots u_{i_j} \# v_{i_j}^R \ldots v_{i_2}^R v_{i_1}^R \mid 1 \leq j, 1 \leq i_1, i_2, \ldots, i_j \leq n \} \\ \cup \{ u_{i_1} u_{i_2} \ldots u_{i_j} E v_{i_j}^R \ldots v_{i_2}^R v_{i_1}^R \mid 1 \leq j, 1 \leq i_1, i_2, \ldots, i_j \leq n \}. \end{aligned}$$

Since

$$L(G_2) = \{S\} \cup \{\, u_{i_1} u_{i_2} \dots u_{i_j} \# v_{i_j}^R \dots v_{i_2}^R v_{i_1}^R \mid 1 \le j, 1 \le i_1, i_2, \dots, i_j \le n \,\},$$

where G_2 is the P0L system from Theorem 3, we can alternatively write language $L(H_1)$ as follows, taking into account that $S \in L(G_1)$:

$$L(H_1) = L(G_1) \cup L(G_2) \cup \{\, u_{i_1} u_{i_2} \dots u_{i_j} E v_{i_j}^R \dots v_{i_2}^R v_{i_1}^R \mid 1 \le j, 1 \le i_1, i_2, \dots, i_j \le n \,\}.$$

Further, let H_2 be a P0L system generating the set $\{S, A, B, C, D, \#, a, b, c\}^+$. Since $L(H_1) \cap L(H_2) = L(G_1) \cup L(G_2)$, and it is non-semidecidable whether $L(G_1) \cup L(G_2)$ is a language of any type from {P0L, 0L, PT0L, T0L}, the proof is complete. □

It is known that there are regular languages which are not generated by any T0L system [11]. However since semi-groups like $\{S, A, B, C, D, \#, a, b, c\}^+$ belong to all language families in question, we immediately obtain the following corollary.

Corollary 5. *Let X and Y be in {P0L, 0L, PT0L, T0L}. Given an X system G and a regular language R, it is non-semidecidable whether the intersection $L(G) \cap R$ is a Y language.*

Whether the only remaining Boolean operation, the complementation, also yields a non-semidecidable operation problem for the four L language families is left open.

4. Non-erasing homomorphism, substitution, and concatenation

Here we consider the operation problem for non-erasing homomorphism, substitution, and finally for concatenation. We start with non-erasing homomorphisms. Observe, that the stated theorem even holds in case of letter-to-letter homomorphisms.

Theorem 6. *Let X and Y be in {P0L, 0L, PT0L, T0L}. Given an X system G and a (non-erasing) homomorphism h, it is non-semidecidable whether $h(L(G))$ is a Y language.*

Proof. Consider the P0L system H_1 over $V = \{S, A, B, C, D, E, \#, a, b, c\}$ of the proof of Theorem 4, and let $h : V^* \to V^*$ be the homomorphism defined by $h(x) = x$, for $x \in V \setminus \{E\}$ and $h(E) = \#$. Then $h(L(H_1)) = L(G_1) \cup L(G_2)$, where G_1 and G_2 are the P0L systems of the proof of Theorem 3. Since it is non-semidecidable whether $L(G_1) \cup L(G_2)$ is a language of any type from {P0L, 0L, PT0L, T0L}, the assertion follows. □

For inverse homomorphisms we obtain the following result:

Theorem 7. *Let X and Y be in {P0L, 0L, PT0L, T0L}. Given an X system G, a (non-erasing) homomorphism h, it is non-semidecidable whether $h^{-1}(L(G))$ is a Y language.*

Proof. We argue as in the previous proof, again by considering the P0L system H_1 over $V = \{S, A, B, C, D, E, \#, a, b, c\}$ of the proof of Theorem 4. Then we define the homomorphism $h : (V \setminus \{E\})^* \to V^*$ by $h(x) = x$, for $x \in V \setminus \{E\}$ and note that $h^{-1}(L(H_1)) = L(G_1) \cup L(G_2)$, where G_1 and G_2 are the P0L systems of the proof of Theorem 3. Thus, the non-semidecidability of the inversion homomorphism operation problem for the language families in question is immediate. □

For substitutions, whose languages are certain L languages, we find a similar result.

Theorem 8. *Let X, Y and Z be in {P0L, 0L, PT0L, T0L}. Given an X system G and a Z substitution σ, it is non-semidecidable whether $\sigma(L(G))$ is a Y language.*

Proof. Consider the P0L language $\{a, b\}$ and the P0L substitution σ defined by $\sigma(a) = L(G_1)$ and $\sigma(b) = L(G_2)$, where G_1 and G_2 are the P0L systems of the proof of Theorem 3. Then the undecidability of the union problem reduces to the substitution problem, and the assertion follows. □

In the remainder of this section we consider the concatenation.

Theorem 9. *Let X and Y be in {P0L, 0L, PT0L, T0L}. Given two X systems H_0 and H_1, it is non-semidecidable whether the concatenation $L(H_0) \cdot L(H_1)$ is a Y language.*

Proof. Given a PCP instance $I = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ over the alphabet $\{a, b\}$. Let $H_0 = (\{E, a, b, \#, \$\}, P_0, \#\#E\$\$)$, where

$$P_0 = \{ E \to u_i E v_i^R \mid 1 \leq i \leq n \} \cup \{a \to a, b \to b, \# \to \#, \$ \to \$\},$$

such that

$$L(H_0) = \{\#\#\} \cdot \{ u_{i_1} u_{i_2} \ldots u_{i_j} E v_{i_j}^R \ldots v_{i_2}^R v_{i_1}^R \mid 1 \leq j, 1 \leq i_1, i_2, \ldots, i_j \leq n \} \cdot \{\$\$\}.$$

Furthermore, let H_1 be taken from the proof of Theorem 4. This completes the description of the P0L systems.

Next we consider the language $L(H_0) \cdot L(H_1)$ in more detail. We distinguish two cases, namely whether the PCP used in the construction of the L systems has a solution or not:

Case 1. The PCP has no solution for the instance I. Then the rule $E \to \#$ can be omitted from the set of rules of H_1 without changing the generated language $L(H_1)$.

We construct a P0L system H from H_0 and H_1 as follows:

$$H = (\{S, A, B, C, D, E, a, b, c, \#, \$\}, P, \#\#E\$\$S),$$

where $P = P_1' \cup P_0 \cup \{ X \to X \mid X \in \{S, A, B, C, D, E\} \}$, and P does not contain the rule $E \to \#$. It is easy to see that $L(H) = L(H_0) \cdot L(H_1)$.

Case 2. The PCP has a solution for the given instance, say $j_1 j_2 \ldots j_k$. Let

$$w = (u_{j_1} u_{j_2} \ldots u_{j_k})^m$$

with m sufficiently large. Then any word in $L(H_0) \cdot w\#w$ is in $L(H_0) \cdot L(H_1)$. Assume there is a T0L system H' with $L(H') = L(H_0) \cdot L(H_1)$. As every word in $L(H')$ begins with ##, in any table the only rule replacing # is $\# \to \#$.

Next, by arguments similar to the ones given in the proof of Theorem 3 one shows that in any table, $x \to x$ are the only rules for the symbols $x \in \{a, b, c\}$.

Since w is long, we conclude that in order to derive the suffix $w\#w$ the symbol # has to be derived from symbol E. But then the corresponding rule can be applied to the left occurrence of E in any word of the language, yielding a word of the form $\#\#u\#v\$\$y$ which does not belong to $L(H_0) \cdot L(H_1)$. Hence, the words in $L(H_0) \cdot w\#w$ cannot be generated by H', a contradiction. In conclusion, $L(H_0) \cdot L(H_1)$ is not a T0L language.

This proves the assertion. □

5. Unary 0L languages

We continue our investigation on the operation problem for L systems with some results on unary 0L languages, i.e., languages over a singleton letter alphabet. Already in [4] it was shown that the union problem for unary 0L languages is decidable. Their proof heavily relies on a characterization result for unary 0L languages from [5]: If L is a 0L language over a unary alphabet $\{a\}$, then either L is regular, i.e., there is a finite set F and integers $d \geq 1$ and $1 \leq i_1 < i_2 < \cdots < i_k$, for some $k \geq 0$, such that

$$L = F \cup \{a^{i_1}, a^{i_2}, \ldots, a^{i_k}\}\{a^d\}^*,$$

or there are integers $i \geq 1$ and $k \geq 2$ such that

$$L = \{ a^{i \cdot k^n} \mid n \geq 0 \}.$$

Moreover, given a 0L system G, there is an algorithm to determine the parameters of the languages and L has the latter form if and only if is generated by the D0L system $G = (\{a\}, \{a \to a^k\}, a^i)$. So, regularity is decidable for unary 0L systems. Since on the other hand not every regular language is a 0L language, the problem arises whether a given regular language is also a 0L language. The decidability of this problem for unary 0L systems has been shown in [13].

Next we consider three operation problems, where we also use the structural characterization of unary 0L languages in more detail. The two theorems to come show that the Kleene star operation and the complementation problems for unary 0L languages are decidable.

Theorem 10. *Given a 0L language L over a unary alphabet, it is decidable, whether or not the Kleene star of L is a 0L language.*

Proof. It is folklore that the Kleene star of any unary language is regular (cf. [6]). Due to the effective characterization of unary 0L languages [5] mentioned above, clearly, given a unary 0L system G a regular expression for the language $L(G)^*$ can effectively be constructed. Since by Salomaa [13] it is decidable whether a given regular language is also a 0L language, we can decide whether or not $L(G)^*$ is a 0L language. □

Theorem 11. *Given a 0L language L over a unary alphabet, it is decidable, whether or not the complement of L is a 0L language.*

Proof. We have to consider two cases:

Case 1: If L is regular, then its complement is regular too, and thus by Salomaa [13] we can decide whether or not the complement of L is a 0L language.

Case 2: If language L is of the form $L = \{ a^{i \cdot k^n} \mid n \geq 0 \}$, for some $i \geq 1$ and $k \geq 2$, its complement can neither be of the exponential nor of regular form. Thus, in this case the complement of L is never a 0L language. □

Finally, we consider the intersection with regular languages. Not surprisingly we also obtain a decidability result.

Theorem 12. *Given a 0L language L over a unary alphabet and a regular language R, it is decidable, whether or not the intersection of L and R is a 0L language.*

Proof. We have to consider the following two cases:

Case 1: Language L is regular. Then $L \cap R$ is also regular. By Salomaa [13] we can decide whether or not $L \cap R$ is a 0L language.

Case 2: Language L is of the form $L = \{ a^{i \cdot k^n} \mid n \geq 0 \}$, for some $i \geq 1$ and $k \geq 2$. Since L is a 0L language it is an E0L language. The family of E0L languages is effectively closed under intersection with regular sets. So, the intersection $L \cap R$ is an E0L language, too. Since for E0L languages membership, emptiness, and finiteness can be decided as mentioned in [11], we obtain the following algorithm to determine whether $L' = L \cap R$ is a 0L language or not:

First we decide the finiteness of L'. If L' is finite, we search for a natural number k such that $L' \cap \{ a^\ell \mid \ell \geq k \}$ is empty. This can be tested effectively because L' is an E0L language and is therefore closed under intersection with regular sets, and emptiness for E0L languages is decidable. Then we have found a number such that there is no longer word in L'. In order to give an explicit representation of the finite language L', we successively check whether a^i, for $0 \leq i \leq k$, belongs L'. Then by Salomaa [13] we can decide whether or not L' is a 0L language.

In case L' is infinite we proceed by searching the smallest n_1 such that $a^{i \cdot k^{n_1}}$ is in L'. Moreover, we also determine the next smallest value n_2 such that $a^{i \cdot k^{n_2}}$ belongs to L'. Note that $n_1 < n_2$. Thus, the language generated by the 0L system

$$G = (\{a\}, \{a \to a^{k^{n_2 - n_1}}\}, a^{i \cdot k^{n_1}})$$

has to be checked for equivalence with L'. This is done in stages: (i) Check whether the intersection of $L(G)$ and the complement of R is empty. Thus, we know $L(G) \subseteq R$. (ii) Then verify that the finite set $\{ a^{i \cdot k^n} \mid 0 \leq n < n_1 \}$ has an empty intersection with R. This proves that we have chosen the correct axiom for G. (iii) Finally, verify that every language generated by the 0L system $G_n = (\{a\}, \{a \to a^{k^{n_2 - n_1}}\}, a^{i \cdot k^n})$ with $n_1 < n < n_2$ has an empty intersection with R. This shows that the single rule was chosen in the right way. Observe, that all questions can be effectively answered by our previous investigations. If all of these questions are answered in the affirmative, then the language $L' = L \cap R$ is a 0L language. Otherwise, it is not a 0L language.

Thus, in both cases we can decide whether or not $L \cap R$ is a 0L language or not. □

Whether other operations such as, e.g., intersection, homomorphism, inverse homomorphism, or concatenation, also yield decidable operation problems for unary 0L languages is left open.

6. Conclusions

We have investigated the operation problem for T0L languages and subclasses, that is, we fix an operation on formal languages, and consider the question: Given languages from the language family under consideration, is it decidable or at least semidecidable whether or not the application of the operation to the languages leads out of this family? For the AFL operations, except Kleene closure, we have shown that the considered problem is non-semidecidable, even already for P0L systems. Since we were dealing with pure language families some of our constructions and arguments were slightly more involved than usual. Contrary to these non-semidecidability results we have shown that for unary 0L languages some operation problems, namely the Kleene star operation problem, the complementation problem, and the intersection with regular sets, are decidable. This nicely fits to the result shown in [4], where the union problem for unary 0L languages, was classified to be decidable, too.

Nevertheless, some interesting problems remain open:

1. Where are the borderlines between decidability, undecidability, and non-semidecidability of the operation problems for T0L languages and subclasses exactly? Are there some nontrivial operations for which the problem is decidable? Can we characterize these cases?
2. What about the operation problems for deterministic T0L languages and subclasses? A T0L system $G = (V, P_1, P_2, \ldots, P_r, \omega)$ is *deterministic* if all tables $P_i \subseteq V \times V^*$ with $1 \leq i \leq r$ are such that $\sigma_{P_i}(a) = \{ v \mid (a, v) \in P_i \}$ is a singleton set for every letter $a \in V$. In particular, for the most simple L systems, namely propagating D0L systems, which are nothing other than iterated non-erasing homomorphisms, decidability of the union problem has been shown [4]. But the decidability status of the operation problem with respect to other operations is unknown.
3. Determine the decidability status of the operation problem with respect to complementation and Kleene closure for the L systems in question.

We have seen that all undecidability results show that the problems studied are not even semidecidable, but what is their exact status of unsolvability? To this end, one has to consider the *arithmetic hierarchy*, which is defined as follows:

$$\Sigma_1 = \{ L \mid L \text{ is recursively enumerable} \},$$
$$\Sigma_{n+1} = \{ L \mid L \text{ is recursively enumerable in some } A \in \Sigma_n \},$$

and Π_n is the class of all complements of languages in Σ_n, that is, define $\Pi_n = \{ L \mid \overline{L} \in \Sigma_n \}$, for $n \geq 1$. Here, a language L is said to be recursively enumerable in some B if there is a Turing machine with oracle B that semi-decides L. Alternatively, a more revealing characterization of the arithmetic hierarchy can be given in terms of alternation of quantifiers. More precisely, a language L is in Σ_n, for $n \geq 1$, if and only if there exists a *decidable* $(n+1)$-ary predicate R such that

$$L = \{ w \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q y_n \, R(w, y_1, y_2, \ldots, y_n) \},$$

where $Q = \exists$ if n is odd and $Q = \forall$ if n is even. Thus, the non-semidecidability implies that the problems in question are at least on level Π_1 of the arithmetic hierarchy, but what is their precise level? As the reader may easily verify, the upper bound on the operation problems under consideration is Σ_2, although we have to leave open whether the problems are complete for this class, but we conjecture it to be so. This conjecture is based on the fact that recently it was shown that for certain operations, under which the families of linear and deterministic context-free languages are not closed, the corresponding operation problems are Σ_2-complete [2].

References

[1] Y. Bar-Hillel, M. Perles, E. Shamir, On formal properties of simple phrase structure grammars, Z. Phonetik Sprachwiss. Kommun.forsch. 14 (1961) 143–172.
[2] H. Bordihn, M. Holzer, M. Kutrib, Unsolvability levels of operation problems for subclasses of context-free languages, Int. J. Found. Comput. Sci. 16 (2005) 423–440.
[3] J. Dassow, G. Păun, Regulated Rewriting in Formal Language Theory, Springer, 1989.
[4] J. Dassow, G. Păun, A. Salomaa, On the union of 0l languages, Inform. Process. Lett. 47 (1993) 59–63.
[5] G.T. Herman, P. Lee, J. van Leeuwen, G. Rozenberg, Characterization of unary developmental languages, Discrete Math. 6 (1973) 235–247.
[6] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[7] A. Lindenmayer, Mathematical models for cellular interactions in development I. Filaments with one-sided inputs, J. Theor. Biol. 18 (1968) 280–299.
[8] A. Lindenmayer, Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs, J. Theor. Biol. 18 (1968) 300–315.
[9] H.A. Maurer, A. Salomaa, D. Wood, Pure grammars, Inform. Control 44 (1980) 47–72.
[10] E.L. Post, A variant of a recursively unsolvable problem, Bull. AMS 52 (1946) 264–268.
[11] G. Rozenberg, A. Salomaa, The Mathematical Theory of L Systems, Academic Press, 1980.
[12] A. Salomaa, Formal Languages, Academic Press, 1973.
[13] A. Salomaa, Solutions of a decision problem concerning unary Lindenmayer systems, Discrete Math. 9 (1974) 71–77.