Algorithmic properties of some fragments of concatenation theory

B Karlov

Department of Applied Mathematics and Cybernetics, Tver State University, 33, Zhelyabova str., Tver, 170100, Russia
E-mail: bnkarlov@gmail.com

Abstract. The paper considers two fragments of the word theory with concatenation. The first fragment has two relations denoting that one of the words is a prefix (respectively, a suffix) of another one. It is proved that this theory is algorithmically equivalent to elementary arithmetic and, therefore, undecidable. The second fragment has a countable set of power operations. It is proved that this theory admits effective quantifier elimination and is, therefore, decidable.

1. Introduction

The study of the concatenation theory began in [1] where a second order axiomatization of theory of syntax based on concatenation was given. After that the concatenation theory and its different variants were studied in many papers. A detailed review of history of this research can be found in [2]. In particular, in [3] it was proved that some finitely axiomatizable concatenation theory is undecidable for a two-symbol alphabet, and in [4] it was proved that it is essentially undecidable. In [2, 5] it was established that some variant of Robinson arithmetic can be interpreted in this theory. At the same time it is easy to see that in some particular cases the concatenation theory can be decidable. For example, if the alphabet \( \Sigma \) contains a single symbol \( a \), then the set of all words \( \Sigma^* \) is isomorphic to the set of natural numbers with addition since \( a^x \cdot a^y = a^{x+y} \). Therefore, the theory of this structure coincides with Presburger arithmetic and is, therefore, decidable. More difficult results on decidability and undecidability of different fragments of concatenation theory were obtained in [6, 7]. In [8–10] some results were established on decidability of theories of some structures when the universe in not the set of all words but some set of all languages over some alphabet. In particular, it was proved that the theory of regular languages with concatenation only is undecidable for all alphabets.

In this paper we study the algorithmic complexity of two fragments of the word theory with concatenation. Section 2 contains main definitions. In Section 3 we study the theory \( T_1 \) such that its language contains only the symbols of constants and two predicate symbols Pref and Suf. The relations Pref\( (x, y) \) and Suf\( (x, y) \) mean that the word \( x \) is a prefix or a suffix of the word \( y \), respectively. We prove that the theory of this structure is undecidable and equivalent to elementary arithmetic. In Section 4 we study the theory \( T_2 \) with the language containing power operations \( x^i = x \cdots x \) for all natural numbers \( i > 1 \). We prove that if the language is expanded with some additional symbols, then this theory admits effective quantifier elimination and is, therefore, decidable.
2. Preliminaries

An alphabet is a finite set of symbols. A word over an alphabet \( \Sigma \) is a finite sequence of symbols from \( \Sigma \). The length of the word \( w \) is the number of symbols in \( w \), it is denoted by \(|w|\). The empty word is a word of zero length, it is denoted by \( \varepsilon \). The set of all words over an alphabet \( \Sigma \) is denoted by \( \Sigma^* \).

### Concatenation

Two words \( u \) and \( v \) are concatenated to form a word which is obtained by appending \( v \) in the end of \( u \). Concatenation of \( u \) and \( v \) is denoted by \( u \cdot v \) or simply by \( uv \). The \( i \)-th power of the word \( v \) is the word \( v^i = vv \cdots v \) where \( v \) is repeated \( i \) times. In particular, \( v^0 = \varepsilon \), \( v^1 = v \).

The word \( u \) is a prefix of the word \( v \) if \( v = uw \) for some word \( w \). The word \( u \) is a suffix of the word \( v \) if \( v = wx \) for some words \( x \) and \( y \). In all three definitions the cases \( u = v \) and \( u = \varepsilon \) are possible.

A semi-Thue system is a pair \( T = (\Sigma, U) \) where \( \Sigma \) is an alphabet, \( U \) is a finite set of productions of the form \( \alpha \rightarrow \beta \) where \( \alpha, \beta \in \Sigma^* \). The relation \( \Rightarrow_T \) is defined on the set \( \Sigma^* \) as follows: \( x \Rightarrow_T y \) if and only if there exist some words \( u, v, \alpha, \beta \) such that \( x = u\alpha v \), \( y = u\beta v \), \( \alpha \rightarrow \beta \in U \), i.e. \( x \Rightarrow_T y \) means that \( y \) is derivable from \( x \) in one step in the semi-Thue system \( T \). Derivability in \( n \) steps is denoted by \( \Rightarrow^n_T \), and \( \Rightarrow^*_T \) denotes a reflexive and transitive closure of the relation \( \Rightarrow_T \).

A theory \( T \) is a set of first-order formulas closed under logical inference. The formulas \( \varphi \) and \( \psi \) are equivalent in the theory \( T \) if \( (\varphi \iff \psi) \in T \). The equivalence of the formulas is denoted by \( \varphi \equiv_T \psi \). An n-ary relation \( P(x_1, \ldots, x_n) \) is definable in the theory \( T \) if there exists a formula \( \varphi(x_1, \ldots, x_n) \) such that \( \varphi \) does not contain \( P \) and \( P(x_1, \ldots, x_n) \equiv_T \varphi(x_1, \ldots, x_n) \). An n-ary function \( f(x_1, \ldots, x_n) \) is definable in the theory \( T \) if there exists a formula \( \varphi(x_1, \ldots, x_n) \) such that \( \varphi \) does not contain \( f \) and \( f(x_1, \ldots, x_n) = y \equiv_T \varphi(x_1, \ldots, x_n, y) \). The theory of a structure \( \mathfrak{A} \) is the set of all formulas which are true in \( \mathfrak{A} \). The theory \( T \) admits quantifier elimination if for every formula \( \varphi \) there exists a quantifier-free formula \( \psi \) such that \( \varphi \equiv_T \psi \). The theory \( T \) admits effective quantifier elimination if there exists an algorithm which given an arbitrary formula \( \varphi \) constructs an equivalent quantifier-free formula \( \psi \).

### 3. Undecidability of the word theory with prefix and suffix relations

In this section we study the theory \( T_1 \) of the structure with the universe \( \Sigma^* \) for some alphabet \( \Sigma = \{ a_1, a_2, \ldots, a_r \} \), \( r \geq 2 \), and with the language \( \Omega_1 = (\text{Pref}(2), \text{Suf}(2); a_1^{(0)}, a_2^{(0)}, \ldots, a_r^{(0)}) \). The predicate and constant symbols are interpreted as follows:

- \( \text{Pref}(x, y) \) — \( x \) is a prefix of \( y \);
- \( \text{Suf}(x, y) \) — \( x \) is a suffix of \( y \);
- \( a_i \) — the symbol \( a_i \) from the alphabet \( \Sigma \).

At first we prove definability of some additional operations and relations in the theory \( T_1 \). The relation \( \text{Subw}(x, y) \) means that the word \( x \) is a subword of the word \( y \):

\[
\text{Subw}(x, y) \equiv_{T_1} (\exists u)(\text{Pref}(u, y) \land \text{Suf}(x, u)).
\]

If \( x \) is a subword of \( y \), then \( y = vxw \) for some \( v, w \). Then we may choose \( u = vx \), so the formula is true. If the formula is true, then \( y = uw, u = vx \) for some \( v, w \). Then \( y = vxw \), i.e. \( x \) is a subword of \( y \).

The relation \( \text{Subw}_1(x, y) \) means that \( y \) has exactly one occurrence of \( x \):

\[
\text{Subw}_1(x, y) \equiv_{T_1} \text{Subw}(x, y) \land \neg(\exists u)(\exists v)(u \neq v \land \text{Pref}(u, y) \land \text{Pref}(v, y) \land \text{Suf}(x, u) \land \text{Suf}(x, v)).
\]

If \( y \) has an occurrence of \( x \), then the formula \( \text{Subw}(x, y) \) is true. Let us suppose that the formula from the right-hand side of the equivalence is false, i.e. that there exist two different words \( u \) and \( v \) such that the formula \( \text{Pref}(u, y) \land \text{Pref}(v, y) \land \text{Suf}(x, u) \land \text{Suf}(x, v) \) is true.
Then \( y = uz_1, y = vz_2, u = w_1x, v = w_2x \) for some \( z_1, z_2, w_1, w_2 \); therefore, \( y = w_1xz_1, y = w_2xz_2 \). Since \( u \neq v \), then \( z_1 \neq z_2 \) and \( w_1 \neq w_2 \). This means that there are at least two different occurrences of \( x \) in \( y \); contradiction. Now let us suppose that the formula is true. Since \( \text{Subw}(x, y) \) is true, then \( x \) is a subword of \( y \). Let us suppose that there are at least two different occurrences of \( x \) in \( y \). Then \( y = w_1xz_1, y = w_2xz_2 \) for some words \( w_1, w_2, z_1, z_2 \) such that \( w_1 \neq w_2, z_1 \neq z_2 \). If we choose \( u = w_1x, v = w_2x \), then the formula \( \text{Pref}(u, y) \land \text{Pref}(v, y) \land \text{Suf}(x, u) \land \text{Suf}(x, v) \) is true, and the whole formula is false; again contradiction.

The main definable operation is concatenation of a word with an arbitrary fixed word. At first we define concatenation of a word and a symbol, \( y = xa \) and \( y = ax \) where \( a \in \{ a_1, \ldots, a_r \} \):

\[
\begin{align*}
y = xa & \equiv_{T_1} y \neq x \land \text{Pref}(x, y) \land \text{Suf}(a, y) \land (\forall z)(\text{Pref}(z, y) \rightarrow (\text{Pref}(z, x) \lor z = y)); \\
y = ax & \equiv_{T_1} y \neq x \land \text{Suf}(x, y) \land \text{Pref}(a, y) \land (\forall z)(\text{Suf}(z, y) \rightarrow (\text{Suf}(z, x) \lor z = y)).
\end{align*}
\]

We prove the correctness of the formula for the relation \( y = xa \). The proof for \( y = ax \) is analogous. If \( y = xa \), then obviously \( x \neq y \), \( x \) is a prefix of \( y \), and \( a \) is a suffix of \( y \). Let \( z \) be an arbitrary prefix of \( y \). If \( z = y \), then the conclusion of the implication is true. If \( z \neq y \), then \( z \) does not contain the ending symbol \( a \); therefore, \( z \) is a prefix of \( x \). Now let us suppose that the formula from the right-hand side of the equivalence is true. Then \( y \) begins with \( x \) and ends with \( a \). Therefore, \( y = xwa \) for some word \( w \) since \( y \neq x \). If \( w \neq \varepsilon \), then let \( z \) be the word \( xw \). The word \( z \) is a prefix of \( y \), it is not a prefix of \( x \), and \( z \neq y \). Therefore, the implication is false. This contradiction shows that \( w = \varepsilon \) and so \( y = xa \).

Now we define concatenation with a word \( w \), \( y = xw \) and \( y = wx \) where \( w \in \{ a_1, \ldots, a_r \}^* \).

We use induction on the length of \( w \):

\[
\begin{align*}
y = x\varepsilon & \equiv_{T_1} y = \varepsilon x \equiv_{T_1} y = x; \\
y = x(wa) & \equiv_{T_1} (\exists z)(z = xw \land y = za); \\
y = (aw)x & \equiv_{T_1} (\exists z)(z = wx \land y = az).
\end{align*}
\]

Let us emphasize that this construction does not allow to express concatenation of two variables.

Now we show how to describe computations of an arbitrary deterministic Turing machine using formulas of the language \( \Omega_1 \). We write the configurations of the Turing machine \( M = (Q, \Delta, P, q_0, q_f) \) as \textit{Post words}, i.e. as words of the form \#qav\# where \( q \in Q \) is current state, \( a \in \Delta \) is an observed symbol, \# \( \notin Q \cup \Delta \) is a special symbol (marker of the used part of the tape), \( u, v \in \Delta^* \) are words written on the tape to the left and to the right of the observed cell. Here \( u \) does not begin with the empty symbol \( \Lambda \), \( v \) does not end with \( \Lambda \). The relation \( \alpha \vdash_M \beta \) means that the machine \( M \) moves in one step from the configuration \( \alpha \) to the configuration \( \beta \). It is known (see [11]) that for every Turing machine \( M = (Q, \Delta, P, q_0, q_f) \) one can effectively construct a semi-Thue system \( T = (Q \cup \Delta \cup \{ \# \}, U) \) such that for every two Post words \( \alpha \) and \( \beta \) the following property holds: \( \alpha \vdash_M \beta \) if and only if \( \alpha \Rightarrow_T^* \beta \). Moreover, both used production and a place of its application are defined unambiguously for every step of the derivation \( \alpha \Rightarrow_T^* \beta \).

In the proof of the following lemma we assume that the semi-Thue system is fixed.

**Lemma 1.** Let the relation \( \text{Der}(x, y) \) mean that \( x \Rightarrow_T^* y \) for the semi-Thue system \( T \). Then \( \text{Der} \) is definable in the theory \( T_1 \).

**Proof.** We assume that the alphabet \( \Sigma \) contains all the symbols of the semi-Thue system and a special symbol \$\$, i.e. that \( Q \cup \Delta \cup \{ \#, \$ \} \subseteq \Sigma \). Later in the proof of the main theorem we will show how to encode the computations using only two symbols. The derivation \( \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_m \) is encoded as one word \$\alpha_1\$\$\alpha_2\$\ldots\$\alpha_m\$. The symbol \$ serves as
a separator of two consecutive Post words. Note that the border of two Post words can be identified by two consecutive symbols \#'. The symbol $ is used for convenience.

Let us define some relations which describe derivability in the semi-Thue system. The relation \( \text{Step}_\alpha \to \beta(x,y) \) means that \( x \Rightarrow_T y \) by using a production \( \alpha \to \beta \), \( x \) has exactly one occurrence of \( \alpha \), and \( y \) has exactly one occurrence of \( \beta \):

\[
\text{Step}_\alpha \to \beta(x,y) \equiv T_1 \quad \text{Subw}_1(\alpha,x) \wedge \text{Subw}_1(\beta,y) \wedge \\
(\exists u)(\exists v)(\text{Pref}(u\alpha,x) \wedge \text{Pref}(u\beta,y) \wedge \text{Suf}(\beta v,y)).
\]

The formulas \( \text{Subw}_1(\alpha,x) \) and \( \text{Subw}_1(\beta,y) \) express that the occurrences of \( \alpha \) and \( \beta \) are unique. If \( x \Rightarrow_T y \), then \( x = u_\alpha z \), \( y = u_\beta z \) for some \( u \) and \( z \). Then we may choose \( u = w, v = z \). Conversely, if the formula is true, then \( x \) begins with the word \( u_\alpha \) and ends with the word \( \alpha \). Since the occurrence of \( \alpha \) in \( x \) is unique, this means that \( x = u_\alpha \). Similarly, \( y = u_\beta \).

The relation \( \text{Consec}(x,y,z) \) means that \( x \) and \( y \) are two consecutive words over \( \Sigma \) in \( z \):

\[
\text{Consec}(x,y,z) \equiv T_1 \quad (\exists u)(\text{Subw}(u\$, z) \wedge \text{Subw}(\$, u) \wedge \text{Pref}(x\$, u) \wedge \text{Suf}(\$, u)).
\]

Note that this relation does not verify whether \( x \Rightarrow_T y \). It only checks that \( y \) immediately follows \( x \).

If the relation is true, then \( z = v\$x y\$w \) for some words \( v \) and \( w \). We may choose \( u = x\$y \). This word is a subword of \( z \), contains exactly one symbol $, begins with \( x\$ \), and ends with \( \$y \). Now let the formula be true. Then \( x\$ \) is a prefix of \( u \), and \( \$y \) is a suffix of \( u \). Moreover, since \( \text{Subw}(\$, u) \) holds, then this symbol $ occurs in the same place. This means that \( u = x\$y \), and also that neither \( x \) nor \( y \) contains the separator $$. Then \( z \) has a subword $$x\$y\$; therefore, \( x \) and \( y \) are two consecutive words over \( \Sigma \).

The relation \( \text{Der}_1(x) \) means that \( x \) encodes a derivation in the semi-Thue system:

\[
\text{Der}_1(x) \equiv T_1 \quad x \neq \$ \wedge \text{Pref}(x,\$) \wedge \text{Suf}(\$, x) \wedge (\forall u)(\forall v)(\text{Consec}(u, v, x) \Rightarrow \bigvee_{\alpha \to \beta \in U} \text{Step}_\alpha \to \beta(u, v)).
\]

Let \( x \) encode some derivation \( \alpha_1 \Rightarrow_T \alpha_2 \Rightarrow_T \cdots \Rightarrow_T \alpha_m \). Then \( x = \$\alpha_1\$\alpha_2\$ \cdots \$\alpha_m\$ \) where the words \( \alpha_i \) do not contain $, \( \alpha_i \Rightarrow_T \alpha_{i+1} \) for all \( 1 \leq i \leq m - 1 \). By definition \( x \) begins and end with $$. Moreover, \( x \neq $ \) since \( m \geq 1 \). Let \( u \) and \( v \) be two arbitrary words. If one of them is not a subword of \( x \) or contains $$, then the implication is true since \( \text{Consec}(u, v, x) \) is false. If they are not consecutive configurations, then the implication is also true. Finally, if \( u = \alpha_i, v = \alpha_{i+1} \), then \( \text{Step}_\alpha \to \beta(u, v) \) is true for some production \( \alpha \to \beta \). In all cases the formula is true. Conversely, let the formula be true. Then \( x \neq $$, \( x \) begins and ends with $$. This means that \( x \) can be written as \( x = \$\alpha_1\$\alpha_2\$ \cdots \$\alpha_m\$ \) where neither of the words \( \alpha_i \) contains $$. For every \( i \) the formula \( \text{Consec}(\alpha_i, \alpha_{i+1}, x) \) is true; therefore, the conclusion of the implication is also true, and \( \alpha_i \Rightarrow_T \alpha_{i+1} \) for some production.

The main relation \( \text{Der}(x,y) \) is defined as follows:

\[
\text{Der}(x,y) \equiv T_1 \quad \neg \text{Subw}(\$, x) \wedge \neg \text{Subw}(\$, y) \wedge (\exists z)(\text{Der}_1(z) \wedge \text{Pref}(\$, z) \wedge \text{Suf}(\$, z)).
\]

If \( x \Rightarrow_T^* y \), then there exists a derivation which is encoded by the word \( z = \$\alpha_1\$\alpha_2\$ \cdots \$\alpha_m\$ \) where \( \alpha_1 = x, \alpha_m = y \). This word begins with $$ and ends with $$$. Moreover, neither \( x \) nor \( y \) contains the symbol $$. This means that the formula is true. Now, let the formula be true. Then \( x \) and \( y \) do not contain $, and also there exists a word \( z \) encoding some derivation. If \( z \) contains exactly two symbols $$, then \( z = $$; therefore, \( x = y \) and \( x \Rightarrow_T^* y \). If \( z \) contains more than two symbols $$, then \( z = $$ where \( \alpha_1 = x, \alpha_m = y \). In this case also \( x \Rightarrow_T^* y \) due to the fact that the formula \( \text{Der}_1(z) \) is true.
Now we prove a technical lemma which will be used later to encode natural numbers.

**Lemma 2.** Let the relation \( \text{Num}(x) \) mean that \( x = (bab)^k \) for some natural number \( k > 0 \). Then \( \text{Num} \) is definable in the theory \( T_1 \).

**Proof.** The relation \( \text{Num}(x) \) is defined as follows:

\[
\text{Num}(x) \equiv T_1 \quad \text{Pref}(b, x) \land \text{Suf}(b, x) \land \neg \text{Pref}(bb, x) \land \neg \text{Suf}(bb, x) \land \\
\text{Subw}(a, x) \land \neg \text{Subw}(aa, x) \land \neg \text{Subw}(bbb, x) \land \neg \text{Subw}(aba, x).
\]

It is straightforward to check that the words of the form \( babab\ldots bab \) satisfy the formula. Now, let the formula be true. The word \( x \) cannot be empty because \( x \) begins with \( b \). We prove the following statement by induction on \( n \): if \( |x| > 3n \) for some natural number \( n \), then \( x \) begins with the prefix \( (bab)^{n+1} \).

**Base case.** Let \( n = 0 \). Due to \( \text{Pref}(b, x) \) the word \( x \) begins with \( b \). The case \( x = b \) is impossible since \( x \) contains \( a \) due to \( \text{Subw}(a, x) \). But the second symbol can be only \( a \) because otherwise \( x \) would start with \( bb \) which contradicts \( \neg \text{Pref}(bb, x) \). Thus, \( x \) begins with \( ba \). The case \( x = ba \) is also impossible since \( x \) can only end with \( b \) due to \( \text{Suf}(b, x) \). The third symbol can be only \( b \) because \( x \) does not contain \( aa \) as a subword due to \( \neg \text{Subw}(aa, x) \). Thus, \( x \) necessarily begins with \( bab \).

**Induction step.** Let \( |x| > 3(n + 1) \). By induction hypothesis \( x \) begins with the prefix \( y = (bab)^{n+1} \). The next symbol can be only \( b \) because otherwise \( x \) would contain a subword \( aba \) contradicting \( \neg \text{Subw}(aba, x) \). This symbol \( b \) cannot be the last one because \( \neg \text{Suf}(bb, x) \) is true. The next symbol cannot be \( b \) since the word \( bbb \) has no occurrences in \( x \) due to \( \neg \text{Subw}(bbb, x) \). Therefore, \( x \) begins with \( yba \). Since both formulas \( \text{Suf}(b, x) \) and \( \neg \text{Subw}(aa, x) \) are true, then the next symbol is \( b \). Thus, \( x \) begins with \( ybab = (bab)^{n+2} \).

Now, let \( x \) be some word for which the formula is true, and let \( 3n < |x| \leq 3(n + 1) \). Then \( x = (bab)^{n+1}y \) for some \( y \). Since \( x \leq 3(n + 1) \), then \( y = \varepsilon \) and \( x = (bab)^{n+1} \). \( \Box \)

Now we establish our main result on the algorithmic complexity of \( T_1 \).

**Theorem 1.** The theory \( T_1 \) is algorithmically equivalent to elementary arithmetic even if \( \Sigma = \{ a, b \} \).

**Proof.** We need to define addition and multiplication of natural numbers in the theory \( T_1 \). Let \( M_+ = (Q, \Delta, P, q_1, q_f) \) be a Turing machine which adds numbers written in the unary system. We may assume that the number \( p \) is encoded by the word \( |p|^{+1} \) in order to avoid using the empty word when \( p = 0 \). Then

\[
|q_1|^{p+1}A|q^1+1\# \vdash_{M_+} ^* |q_f|^{p+q+1}\#.
\]

Let \( Q = \{ q_1, q_2, \ldots, q_m \} \), \( \Delta = \{ b_1, b_2, \ldots, b_n \} \). Let \( T_+ \) be a semi-Thue system corresponding to the machine \( M_+ \). Since we have only two symbols \( a \) and \( b \), we use the standard encoding:

- \( b_i \) is encoded as \( ba^ib \);
- \( q_j \) is encoded as \( ba^{m+n+1}b \);
- \( \# \) is encoded as \( ba^{m+n+1}b \);
- \( \$ \) is encoded as \( ba^{m+n+2}b \).

We may assume that \( b_1 = |, b_2 = \Lambda, q_2 = q_f \). Since the symbol \( | \) is encoded by the word \( bab \), then the natural number \( k \) is encoded as \( (bab)^{k+1} \). In particular, the number \( 0 \) is represented as the word \( bab \). By Lemma 2 the relation \( \text{Num}(x) \) verifies whether the word \( x \) is a code of some
natural number. Then for every three natural numbers $p$, $q$, $r$ the following holds: $p + q = r$ if and only if
\[ ba^{m+n+1}bb^n b(bab)^{p+1}baab(bab)^{q+1}ba^{m+n+1}b = T_+ ba^{m+n+1}bb^{n+2}b(bab)^{r+1}ba^{m+n+1}b. \]

Now we can define the relation $Add(x, y, z)$ which means that the word $z$ represents the sum of two numbers encoded by the words $x$ and $y$:
\[
Add(x, y, z) \equiv T_1 \text{Num}(x) \land \text{Num}(y) \land \text{Num}(z) \land (\exists u)(\text{Pref}(ba^{m+n+1}bb^n b bxbabab, u) \land \text{Suf}(baabba^{m+n+1}b, u) \land \text{Subw}_1(baab, u) \land \text{Der}(u, ba^{m+n+1}bb^{n+2}bzbba^{m+n+1}b)).
\]

Here we replace every symbol from the alphabet $Q \cup \Delta \cup \{\#, \}$ with its code in the subformula Der.

If $Add(x, y, z)$ is true, then $x = (bab)^{p+1}$, $y = (bab)^{q+1}$, $z = (bab)^{p+q+1}$ for some $p$, $q$. We may choose $u = ba^{m+n+1}bb^n b(bab)^{p+1}baab(bab)^{q+1}ba^{m+n+1}b$. This word encodes the initial configuration of the machine $M_+$. Therefore, there exists a derivation $u \Rightarrow T_+ ba^{m+n+1}bb^n b(bab)^{p+1}baab(bab)^{q+1}ba^{m+n+1}b$, and the formula from the right side is true. Conversely, let the formula be true. Then $x = (bab)^{p+1}$, $y = (bab)^{q+1}$ for some $p$ and $q$. Since the word $u$ contains exactly one occurrence of $baab$, then $u = ba^{m+n+1}bb^n b bxbababba^{m+n+1}b$. There exists a derivation $ba^{m+n+1}bb^n b(bab)^{p+1}baab(bab)^{q+1}ba^{m+n+1}b \Rightarrow T_+ ba^{m+n+1}bb^{n+2}b(bab)^{p+q+1}ba^{m+n+1}b$ in the semi-Thue system $T_+$. This is the only derivation ending with the final state $q_2$ because $M_+$ is deterministic. Therefore, $z = (bab)^{p+q+1}$.

In an analogous way we can define the relation $Mult(x, y, z)$ which means that the word $z$ represents the product of two numbers encoded by the words $x$ and $y$. The only difference is that the formula Der is constructed using a Turing machine for multiplication.

Let $\varphi$ be an arbitrary formula of arithmetic. We may assume that all its atomic subformulas are of the form $x = y$, $x + y = z$, or $x \times y = z$. For the formula $\varphi$ we define its translation $T(\varphi)$:

- $T(x = y)$ is $\text{Num}(x) \land \text{Num}(y) \land x = y$;
- $T(x + y = z)$ is $Add(x, y, z)$;
- $T(x \times y = z)$ is $\text{Mult}(x, y, z)$;
- $T(\psi \circ \theta)$ is $T(\psi) \circ T(\theta)$ for $\circ \in \{\land, \lor, \rightarrow\}$;
- $T(\neg \psi)$ is $\neg T(\psi)$;
- $T((\exists x) \psi)$ is $(\exists x)(\text{Num}(x) \land T(\psi))$;
- $T((\forall x) \psi)$ is $(\forall x)(\text{Num}(x) \rightarrow T(\psi))$.

It is easy to see that the formula $\varphi$ is true in arithmetic if and only if $T(\varphi) \in T_1$.

In order to prove the converse, we fix some “natural” enumeration of all words over the alphabet $\Sigma$. For example, we may order all the words by their lengths, and the words of the same length are ordered lexicographically. Let $w_k$ be the word with the number $k$. Then the relations $\text{Pref}(w_x, w_y)$ and $\text{Suf}(w_x, w_y)$ are computable, and, consequently, they are representable in arithmetic (see [12]). Therefore, for every formula $\varphi$ of the language $\Omega_1$ we can construct an arithmetical formula $\psi$ such that $\psi$ is true in arithmetic if and only if $\varphi \in T_1$.

Now we strengthen Theorem 1. We prove that the theory $T_1$ remains undecidable even if no constant symbols are available.

**Theorem 2.** The theory $T_1$ is algorithmically equivalent to elementary arithmetic even if $\Sigma = \{a, b\}$ and the language contains no constant symbols.
Proof. The empty word $\varepsilon$ is definable in the theory $T_1$:

$$x = \varepsilon \equiv_{T_1} \neg(\exists y)(y \neq x \land \text{Pref}(y, x)).$$

The correctness of this definition follows from the fact that $\varepsilon$ is the only word which has exactly one prefix — itself.

The relation Symb$(x)$ is also definable meaning that $x$ is a symbol:

$$\text{Symb}(x) \equiv_{T_1} x \neq \varepsilon \land \neg(\exists y)\left(\text{Pref}(y, x) \land y \neq \varepsilon \land y \neq x\right).$$

Every symbol is nonempty and has exactly two prefixes — the empty word and itself. Conversely, if $|x| \geq 2$, then $x = abz$ for some symbols $a, b$ and some word $z$. Therefore, the formula is false for $y = a$.

Let $\varphi$ be a formula of the language $\Omega_1$ where two constant symbols $a$ and $b$ are used. We replace them with two variables denoted also $a$ and $b$, and we construct the following formula $\psi$:

$$(\exists a)(\exists b)(a \neq b \land \text{Symb}(a) \land \text{Symb}(b)) \rightarrow (\exists a)(\exists b)(a \neq b \land \text{Symb}(a) \land \text{Symb}(b) \land \varphi).$$

Then $\varphi \in T_1$ if and only if $\psi \in T_1$. □

Let us note that if the language contains only one of the predicate symbols Pref or Suf, then such structure is automatic. Therefore, its theory is decidable (see [13, 14]).

Corollary 1. Let $x^{-1}$ denote the inverse of the word $x$: $(a_1a_2 \ldots a_n)^{-1} = a_n \ldots a_2a_1$. Then the theory $T'_1$ of the structure with the universe $\Sigma^*$ ($|\Sigma| \geq 2$) and the language $\Omega'_1 = (\text{Pref}^{(2)}; x^{-1(1)})$ is equivalent to elementary arithmetic.

Proof. The relation Suf is definable in this theory:

$$\text{Suf}(x, y) \equiv_{T'_1} \text{Pref}(x^{-1}, y^{-1}).$$

4. Decidability of the word theory with power operations

In this section we study the word theory with power operations $x^i = \underbrace{x \cdot \ldots \cdot x}_i$. We expand the language with the constant symbol $\varepsilon$ denoting the empty word. Also we add the predicate symbols $R_i$, $i = 2, 3, \ldots$, meaning that $x$ is the $i$-th power of some word. Let $T_2$ be the theory of the structure with the universe $\Sigma^*$ for some alphabet $\Sigma = \{a_1, a_2, \ldots, a_r\}$, $r \geq 1$, and with the language $\Omega_2 = (R_i; x^{(1)}, \varepsilon^{(0)}), i = 2, 3, \ldots$. Note that the symbols $\varepsilon$ and $R_i$ are definable in the original language:

$$x = \varepsilon \equiv_{T_2} x^2 = x, \quad R_i(x) \equiv_{T_2} (\exists y)x = y^i.$$

In the proof of decidability of $T_2$ we will use the following well-known result (see [15, 16]).

Theorem 3 (Lyndon-Schützenberger theorem). Let $x^i = y^j$ for some words $x, y$ and some numbers $i, j$. Then $x = z^k, y = z^l$ for some word $z$ and some numbers $k, l$.

At first we establish a simple property of the predicates $R_i$.

Lemma 3. The following equivalence holds in the theory $T_2$ (here lcm denotes the least common multiple):

$$\bigwedge_{i=1}^m R_{b_i}(x) \equiv_{T_2} R_{\text{lcm}(b_1, \ldots, b_m)}(x).$$
Proof. At first we consider the case \( m = 2 \). If the formula \( R_{\text{lcm}(b_1,b_2)}(x) \) is true, then \( x = y^{\text{lcm}(b_1,b_2)} \) for some word \( y \). Therefore, \( x = (y^{\text{lcm}(b_1,b_2)}/b_1)^{b_1} = (y^{\text{lcm}(b_1,b_2)}/b_2)^{b_2} \), and, consequently, the formulas \( R_{b_1}(x) \) and \( R_{b_2}(x) \) are true.

Now let the formulas \( R_{b_1}(x) \) and \( R_{b_2}(x) \) be true, i.e. \( x = y^{b_1} = z^{b_2} \) for some words \( y \) and \( z \). It follows from Lyndon-Schützenberger theorem that there exists a word \( v \) such that \( y = v^i \), \( z = v^j \) for some \( i \) and \( j \). Therefore, \( x = v^{ib_1} = v^{jb_2} \). It follows from this equality that \( ib_1 \) is divisible by both \( b_1 \) and \( b_2 \). So \( ib_1 \) is divisible by \( \text{lcm}(b_1,b_2) \), and, consequently, the formula \( R_{\text{lcm}(b_1,b_2)}(x) \) is true.

For an arbitrary \( m \) the equivalence is proved by induction on \( m \) using the equality \( \text{lcm}(\text{lcm}(a,b),c) = \text{lcm}(a,b,c) \).

Now we prove that the theory \( T_2 \) is decidable.

**Theorem 4.** The theory \( T_2 \) admits effective quantifier elimination and is, therefore, decidable.

Proof. It is sufficient to show how to eliminate a quantifier from the formula of the form \((\exists x)\theta\) where \( \theta \) is a conjunction of atomic formulas and their negations (see [12]). Let

\[
\varphi = (\exists x) \left( \bigwedge_{i=1}^{k} x^{p_i} = s_i \wedge \bigwedge_{i=1}^{l} x^{q_i} \neq t_i \wedge \bigwedge_{i=1}^{m} R_{b_i}(x^{p'_i}) \wedge \bigwedge_{i=1}^{n} \neg R_{c_i}(x^{q'_i}) \right)
\]

where the terms \( s_i \) and \( t_j \) do not contain the variable \( x \). We may assume that the formula has no subformulas of the form \( x^p = x^q \) since they are equivalent either to \( \top \) when \( p = q \), or to \( x = \varepsilon \) when \( p \neq q \) (here the symbol \( \top \) denotes a formula that is always true). It is easy to see that for every \( k \geq 1 \)

\[
x = y \equiv_{T_2} x^k = y^k, \quad R_{b_i}(x) \equiv_{T_2} R_{kb_i}(x^k).
\]

Let \( N \) be the least common multiple of all the numbers \( p_i, q_i, p'_i, q'_i \). Then

\[
x^{p_i} = s_i \equiv_{T_2} x^N = s_i^{N/p_i}, \quad x^{q_i} \neq t_i \equiv_{T_2} x^N \neq t_i^{N/q_i},
\]

\[
R_{b_i}(x^{p'_i}) \equiv_{T_2} R_{b_iN/p'_i}(x^N), \quad \neg R_{c_i}(x^{q'_i}) \equiv_{T_2} \neg R_{c_iN/q'_i}(x^N),
\]

therefore,

\[
\varphi \equiv_{T_2} (\exists x) \left( \bigwedge_{i=1}^{k} x^{s_i} = s_i \wedge \bigwedge_{i=1}^{l} x^N \neq t_i \wedge \bigwedge_{i=1}^{m} R_{b'_i}(x^N) \wedge \bigwedge_{i=1}^{n} \neg R_{c'_i}(x^N) \right)
\]

for some \( s'_i, t'_i, b'_i, c'_i \). Now the exponent of \( x \) is the same for all occurrences of \( x \) in \( \varphi \).

Let us suppose at first that the formula \( \varphi \) contains equalities. Then

\[
\varphi \equiv_{T_2} (\exists x)(x^N = t \wedge \varphi'(x^N)) \equiv_{T_2} \varphi'(t) \wedge R_N(t).
\]

Now it is left to show how to eliminate a quantifier from the formula without equalities. Let

\[
\varphi \equiv_{T_2} (\exists x) \left( \bigwedge_{i=1}^{l} x^N \neq t'_i \wedge \bigwedge_{i=1}^{m} R_{b'_i}(x^N) \wedge \bigwedge_{i=1}^{n} \neg R_{c'_i}(x^N) \right).
\]

Let \( B \) be the least common multiple of the numbers \( b'_1, \ldots, b'_m \). Then by Lemma 3

\[
\varphi \equiv_{T_2} (\exists x) \left( \bigwedge_{i=1}^{l} x^N \neq t'_i \wedge R_B(x^N) \wedge \bigwedge_{i=1}^{n} \neg R_{c'_i}(x^N) \right).
\]
If \( l = 0 \) or \( n = 0 \), then the respective conjunction is missing. If \( m = 0 \), then the atomic formula \( R_B(x^N) \) is also missing.

At first we consider the case when the formula \( \varphi \) contains atomic subformulas of all types, i.e. when \( l \neq 0 \), \( m \neq 0 \), \( n \neq 0 \). We attempt to find a number \( M > 0 \) such that \( MN \) is divisible by \( B \) but is not divisible by either of \( c_i' \). The numbers \( MN \) and \( (M + Bc_1' \ldots c_n')N \) give the same remainders when divided by \( B, c_1', \ldots, c_n' \); therefore, \( M \) exists if and only if it exists on the interval \([1; Bc_1' \ldots c_n']\), and it can be found by checking a finite set of numbers.

Let us suppose at first that \( M \) exists. We choose a prime number \( p \) such that \( p > c_i' \) for all \( i \), \( pMN > |t_i| \) for all \( i \). Let \( x = a_{pM} \), then \( x^N = a_{pMN} \) (here \( a \) is an arbitrary symbol). Therefore, the formula \( R_B(x^N) \) is true since \( pMN \) is divisible by \( B \), and all the formulas \( R_{c_i'}(x^N) \) are false since \( pMN \) is not divisible by \( c_i' \). The formulas \( x^N \neq t_i' \) are also true since \( |x^N| > |t_i'| \). Thus, in this case the formula \( \varphi \) is true, i.e. \( \varphi \equiv T_2 \top \).

Now let us suppose that \( M \) does not exist. Then for every \( M \) either \( MN \) is not divisible by \( B \), or \( MN \) is divisible by some \( c_i' \). Let us suppose that \( x \) exists. Then it can be represented as \( x = y^K \) where the word \( y \) cannot be written as \( z^L \) for \( L \geq 2 \). Therefore, \( x^N = y^{KN} \), and here either \( KN \) is divisible by \( B \), or it is divisible by some \( c_i' \).

Suppose that \( KN \) is not divisible by \( B \), and consider the formula \( R_B(x^N) \). If it is true, then \( x^N = u^B \) for some word \( u \), i.e. \( y^{KN} = u^B \). By Lyndon-Schützenberger theorem there exists a word \( v \) such that \( y = v^p, u = v^q \) for some \( p \) and \( q \). But since \( y \) cannot be represented in the form \( z^L \) for \( L \geq 2 \), then \( p = 1 \), \( y = v \), \( u = y^q \), and \( y^{KN} = y^{Bq} \). It follows from the last equality that \( KN \) is divisible by \( B \). We obtained a contradiction; therefore, the formula \( R_B(x^N) \) is false and \( \varphi \equiv T_2 \bot \).

Now suppose that \( KN \) is divisible by some \( c_i' \). Then the formula \( R_{c_i'}(x^N) \) is true; therefore, again \( \varphi \equiv T_2 \bot \).

Thus, if \( M \) does not exists, then \( \varphi \equiv T_2 \bot \).

Now let us consider the remaining cases when one or more of the numbers \( l, m, n \) are zero. The proof for these cases is analogous. If the inequalities are missing, then the proof remains correct because we may just omit the condition \( pMN > |t_i| \). If the subformula \( R_B(x^N) \) is missing, then we may omit the condition that \( MN \) is divisible by \( B \). In this case the number \( M \) can be found by checking all numbers from the interval \([1; c_1' \ldots c_n']\). Finally, if \( \varphi \) has no subformulas of the form \( \neg R_{c_i'}(x^N) \), then we omit the conditions that \( MN \) is divisible by neither of \( c_i' \). In this case \( M \) always exists since we may choose \( M \) to be a multiple of \( B \); therefore, \( \varphi \equiv T_2 \top \).

**Corollary 2.** The structures of the language \( \Omega_2 \) are elementarily equivalent for all alphabets \( \Sigma \).

**Proof.** This result follows from the fact that quantifier elimination does not depend on the alphabet.

**Corollary 3.** There is a decision procedure for the theory \( T_2 \) with the time complexity \( O(2^{2^{mp}}) \) for some constant \( p > 1 \).

**Proof.** Let \( \Sigma = \{ a \} \), and let \( \mathfrak{A} \) be the structure of the language \( \Omega_2 \) with the universe \( \Sigma^a \). By Corollary 2 the theory \( T_2 \) is the same for every alphabet, so it is sufficient to prove that the theory of the structure \( \mathfrak{A} \) is decidable. Let \( \mathfrak{B} \) be the structure with the universe \( \Sigma^a \) and with the only operation of concatenation. The structure \( \mathfrak{B} \) is isomorphic to Presburger arithmetic since \( a^m \cdot a^n = a^{m+n} \). Let \( \varphi \) be an arbitrary formula of language \( \Omega_2 \). If we replace every atomic subformula \( x = y^i \) with the subformula \( x = \underbrace{y + \cdots + y}_i \), then we obtain a formula \( \psi \) which is true in Presburger arithmetic if and only if \( \varphi \in T_2 \). Moreover, \( \psi \) is at least as long as \( \varphi \). But Presburger arithmetic can be decided with triply exponential time complexity (see [17]).
Note that this corollary establishes only the upper bound. The formula of Presburger arithmetic constructed in the proof has very special form; therefore, the lower bound on time complexity of Presburger arithmetic is not directly applicable to the theory $T_2$.

5. Conclusion
In this paper we studied the algorithmic complexity of two fragments of concatenation theory. It was proved that the theory $T_1$ with two relations Pref and Suf is algorithmically equivalent to elementary arithmetic and, therefore, undecidable. Also it was proved that the theory $T_2$ with the operations $x^i$ admits effective quantifier elimination and, therefore, is decidable. Some interesting problems remain open.

- Investigate algorithmic properties of other “natural” fragments of concatenation theory.
- Study the theories of analogous structures where the universe is not the set of all words but some set of languages.
- Is concatenation definable in the theory $T_1$?
- What is time and space complexity of the theory $T_2$?
- Will the theory $T_2$ remain decidable if we add another relation $R(x,y)$ meaning that $x$ is some power of $y$?

Acknowledgments
This work is funded by Russian Foundation for Basic Research (RFBR), project number 20-01-00435.

References
[1] Tarski A 1935 Stud. Philos. 1 261–405
[2] Visser A 2009 Notre Dame J. Formal Logic 50 61–85
[3] Grzegorczyk A 2005 Stud. Logica 79 163–230
[4] Grzegorczyk A and Zdanowski K 2008 Andrzej Mostowski and Foundational Studies ed Ehrenfeucht A et al. (Amsterdam: IOS Press) pp 72–91
[5] Švejdar V 2009 Notre Dame J. Formal Logic 50 87–95
[6] Kristiansen L and Murwanashyaka J 2018 Sailing Routes in the World of Computation. 14th Conf. on Computability in Europe, CiE 2018, Kiel, Germany, July 30–August 3, 2018 (Lecture Notes in Computer Science vol 10936) ed Manea F et al. (Cham: Springer) pp 244–53
[7] Kristiansen L and Murwanashyaka J 2020 Arch. Math. Logic URL https://link.springer.com/article/10.1007/s00224-020-09995-4
[8] Dudakov S and Karlov B 2019 Computer Science – Theory and Applications. Proc. 14th Int. Computer Science Symp. in Russia, CSR 2019, Novosibirsk, Russia, July 1–5, 2019 (Lecture Notes in Computer Science vol 11532) ed van Bevern R and Kucherov G (Cham: Springer) pp 119–30
[9] Dudakov S 2020 Lobachevskii J. Math. 41 168–75
[10] Dudakov S and Karlov B 2020 Theory Comput. Syst. URL https://link.springer.com/article/10.1007/s00224-020-09995-4
[11] Davis M D, Sigal R and Weyuker E J 1994 Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science 2nd ed (San Diego: Academic Press Professional, Inc.)
[12] Boolos G S, Burgess J P and Jeffrey R C 2007 Computability and Logic 5th ed (New York: Cambridge University Press)
[13] Blumensath A and Grädel E 2000 Proc. Fifteenth Annual IEEE Symp. on Logic in Computer Science, Santa Barbara, USA, June 26–28, 2000 pp 51–62
[14] Chiswell I 2008 A Course in Formal Languages, Automata and Groups (London: Springer)
[15] Lyndon R C and Schützenberger M P 1962 Michigan Math. J. 9 289–98
[16] Shallit J 2008 A Second Course in Formal Languages and Automata Theory (New York: Cambridge University Press)
[17] Oppen D C 1978 J. Comput. Syst. Sci. 16 323–32