arXiv:2302.06248v4 [cs.FL] 15 Jul 2024

Decision Problems on Copying and Shuffling

Vesa Halava[*][†]
Department of Mathematics and Statistics
University of Turku, Finland
vesa.halava@utu.fi

Tero Harju
Department of Mathematics and Statistics
University of Turku, Finland
harju@utu.fi

Dirk Nowotka
Department of Computer Science
Kiel University, Germany
dn@zs.uni-kiel.de

Esa Sahla
Department of Mathematics and Statistics
University of Turku, Finland
etsahla@gmail.com

Abstract. We study decision problems of the form: given a regular or linear context-free language L, is there a word of a given fixed form in L, where given fixed forms are based on word operations copy, marked copy, shuffle and their combinations.

Keywords: Regular language, linear context-free language, shuffle, marked copy, reverse copy, membership problem

1. Introduction

We consider classic problems on decidability issues of formal languages. We shall fill in gaps that have remained for elementary operations copy and shuffle and their variants on words and languages. The presented results, as well as the known results on the topic, are presented in the table affixed in the second page leaving two open cases for further studies.

We investigate the decidability status of several special membership problems for regular and linear context-free (linear CF) languages where it is asked whether or not the language contains a word of a certain form. Let L be a given language. The operations and the question are the following:

---

1. *copy*, i.e., does there exist a square $ww \in L$ for some word w?

2. *reversed copy*, i.e., does there exist a word $ww^r \in L$ for some word w, where w^r denotes the reversal of the word w?

3. *marked copy*, i.e., does there exist a word $w\overline{w} \in L$ for some word w, where $\overline{w}$ denotes a marked copy of the word w? (For a definition of a marked copy, see page 271)

4. *self-shuffle*, i.e., does there exist a word $u \in w \sqcup\!\sqcup w$ with $u \in L$ for some word w, where $\sqcup\!\sqcup$ denotes the shuffle operations of two words?

5. *shuffle with reverse*, i.e., does there exist a word $u \in w \sqcup\!\sqcup w^r$ with $u \in L$ for some word w?

6. *marked shuffle*, i.e., does there exist a word $u \in w \sqcup\!\sqcup \bar{w}$ with $u \in L$ for some word w?

The decidability statuses of these questions are listed in the following table, where D (resp. U) means that the problem is *decidable* (resp. *undecidable*) and the question mark denotes problems that remain unsettled. After the symbols D and U we give a reference for the proof in the text. Here Reg stands for the regular languages and Lin for the linear context-free languages.

	Reg	Lin
$ww \in L$	D (Cor. 3.2)	U (Thm. 4.1)
$ww^r \in L$	D (Cor. 3.7)	U (Thm. 4.14)
$w\overline{w} \in L$	D (Thm. 3.5)	U (Thm. 4.3)
$w \sqcup\!\sqcup w \cap L \neq \emptyset$	?	U (Thm. 4.9)
$w \sqcup\!\sqcup w^r \cap L \neq \emptyset$	?	U (Thm. 4.13)
$w \sqcup\!\sqcup \overline{w} \cap L \neq \emptyset$	U (Thm. 3.10)	U (Thm. 4.16)

We also study decidability of some special inclusion problems related to the above problems. For example, we investigate the problem of whether a given regular, linear context-free or context-free language is closed under taking squares, and also, the problem whether the set all squares generated by another given language is a subset of a given language.

There are naturally many related language operations to be investigated; see Rampersad and Shallit [1], where among other results it was shown that it is undecidable whether a context-free grammar generates a square. We deal with this problem in Theorem 4.1 for linear CF-languages.

2. Preliminaries

Let Σ be a finite alphabet. A *word* over Σ is a finite sequence of symbols of Σ. The *empty word* is denoted by ε. The *length* of a word $w = a_1 \cdots a_k$, where $a_i \in \Sigma$ for all $i = 1, \ldots, k$, is k and it is denoted by $|w|$. The set of all words over Σ is denoted by Σ^* and the set of all non-empty words by Σ^+.

For two words $u, v \in \Sigma^*$, their *concatenation* is $u \cdot v = uv$. A *factorization* of a word $w \in \Sigma^*$ is a finite sequence $u_1, \ldots, u_k$, where $u_i \in \Sigma^*$ for all i, such that $w = u_1 \cdots u_k$. A word u is a *prefix* of a word w if $w = uv$ for some word v.

The powers of words are defined inductively: $w^0 = \varepsilon$ and for all $n \in \mathbb{N}$, $w^{n+1} = w^n \cdot w$. We say that a word w is *primitive*, if for all $u \in \Sigma^*$, $w = u^n$ implies that $n = 1$.

Let w^r denote the *reversal* (or the *mirror image*) of w, that is, $w^r = a_n \cdots a_1$ for $w = a_1 \cdots a_n$, where $a_i \in \Sigma$ for all i.

With an alphabet Σ we accompany a *marked copy alphabet* $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$, where $\Sigma \cap \bar{\Sigma} = \emptyset$. For a word $w = a_1 a_2 \cdots a_n$, let $\overline{w} = \bar{a}_1 \bar{a}_2 \cdots \bar{a}_n$ be the *marked copy* of w.

A subset of Σ^* is called *language*. Denote by L^c the complement of L, that is, $L^c = \Sigma^* \setminus L$.

We assume that the reader is familiar with the basic notions of language theory; see e.g., Salomaa [2] for definitions of *regular languages*, *finite automata*, *context-free (CF) languages*, *pushdown automata*, *context-sensitive languages* and *pumping lemmas* for regular and context-free languages.

We briefly recall a few basic facts. First of all, recall that a language L is a *linear CF-language* if it is accepted by a pushdown automaton (PDA) that makes at most one reversal (from increasing to decreasing mode) on its stack. Equivalently, each linear CF-language is generated by a linear context-free grammar, where the productions have at most one non-terminal on the right hand side.

Example 2.1. The language of all palindromes of even length,

$$E = \{ww^r \mid w \in \Sigma^*\},$$

is a linear CF-language. Indeed, E can be accepted by a non-deterministic linear PDA which first reads symbols onto the stack until it (non-deterministically) decides to check by popping symbols whether the rest of the input word agrees with the stack content.

We use the Pumping lemma for regular languages to show that certain languages are not regular.

Lemma 2.2. For a regular language L, there exists a natural number $p \geq 1$ such that, if $w \in L$ is of length $|w| \geq p$, then it has a factorization $w = xyz$ with $|y| \geq 1$ and $|xy| \leq p$, such that $xy^n z \in L$ for all $n \in \mathbb{N}$.

The Pumping lemma for CF-languages has two simulaneous pumps.

Lemma 2.3. For a CF-language L, there exists a natural number $p \geq 1$ such that, if $w \in L$ has length $|w| \geq p$, then it has a factorization $w = uvwxy$ with $|vx| \geq 1$ and $|vwx| \leq p$, such that $uv^n wx^n y \in L$ for all $n \in \mathbb{N}$.

Next we define two special languages. Firstly, let $P \subseteq \Sigma^*$ be a language. The *copy language* of P is defined as the set of all second powers of words of P:

$$C_P = \{ww \mid w \in P\}.$$

It is well-known that for a regular language P, the copy language C_P is context-sensitive, but not context-free (see, for example, [3]). We state the following open problem concerning the copy languages. A *one-counter automaton* is a pushdown automaton, with a single stack letter, which is able to check the emptiness of the stack.

Problem 1. Is the complement of C_{Σ^*} a one-counter language?

Considering the marked copy, the problem becomes easier.

Lemma 2.4. For a regular language P, the complement of the marked copy language $\{w\overline{w} \mid w \in P\}$ is a one-counter language.

Proof:
Sketch of the proof: Assume P is accepted by a finite automaton A, that is, $P = L(A)$. Let $\bar{A}$ be the *copied version* of A, i.e., where the letters in the transitions are changed to marked letters.

We construct a one reversal nondeterministic PDA B for A and its marked copy automaton $\bar{A}$ as follows:

1. B simulates A and reads symbols from Σ and adds one to the counter to count the length of the prefix read so far.

2. At one point B remembers the symbol $a \in \Sigma$ under its reading head and stops writing to the stack. B continues by reading the rest of the non-marked part w.

3. If $w \notin P$, the input is accepted.

4. Otherwise B simulates $\bar{A}$ for the marked part of the input and decreases the counter on each step.

5. When the stack is empty and the input is not read fully, B checks if the current symbol of the input is equal to $\overline{a}$. If not, then the input is accepted.   □

Secondly, we define the *shuffle* (language) of two words $u, v \in \Sigma^*$ as follows:

$$u \sqcup v = \{u_1 v_1 \cdots u_n v_n \mid u_i, v_i \in \Sigma^* \text{ for all } i = 1, \ldots, n$$
$$\text{and } u = u_1 u_2 \cdots u_n, \; v = v_1 v_2 \cdots v_n\}.$$

In the above factorizations of u and v we allow that some of the factors u_i and v_i are empty.

Let Σ and Δ be two alphabets. A mapping $g \colon \Sigma^* \to \Delta^*$ is a *morphism* if, for all $u, v \in \Sigma^*$, $g(uv) = g(u)g(v)$.

For the undecidability proofs, we use reductions from the *Post's Correspondence Problem* (PCP, for short). The PCP was introduced and proved to be undecidable by E. Post in 1946; see [4]. We shall use the modern form of the problem and define the PCP using monoid morphisms: assume that g and h are two morphisms from Σ^* into Δ^*, where $\Sigma = \{a_1, \ldots, a_n\}$ is an alphabet of n letters. The pair (g, h) is called an *instance* of the PCP, a word w satisfying

$$g(w) = h(w). \tag{1}$$

is called a *solution* of the instance (g, h). The *size* of an instance is the size of the domain alphabet, i.e., the size is equal to $|\Sigma|$.

Theorem 2.5. It is undecidable for instances (g, h) whether or not it has a solution.

It is known that for the size $n \leq 2$, the PCP is decidable; see [5] and [6]. On the other hand, for sizes $n \geq 5$, the PCP is known to be undecidable; see [7]. The decidability statuses for $n = 3$ and 4 are open. Note that in basic undecidability proofs, the morphisms g and h are *non-erasing*, that is, $g(a) \neq \varepsilon \neq h(a)$ for all $a \in \Sigma$. This is also the case in [7].

3. Regular languages

In this section we study the problems defined in the first section for regular languages.

3.1. Powers and copies

Our first theorem is well-known and can be regarded as folklore.

Theorem 3.1. Let $n \geq 2$ be a fixed integer and $P \subseteq \Sigma^*$ a regular language. It is decidable for regular languages $R \subseteq \Sigma^*$ if there exists a power $w^n \in R$ for some word $w \in P$. Indeed, the existence of a power $w^n \in R$ is a PSPACE complete problem.

Proof:
Let A be a finite automaton accepting R, i.e., $R = L(A)$. Let the states of A be $q_0, q_1, \ldots, q_m$, where q_0 is the initial state. Define, for all i and j, the regular language R_{ij} by

$$R_{i,j} = \{w \mid q_i \xrightarrow{w} q_j\} \cap P,$$

where $q_i \xrightarrow{w} q_j$ denotes that there is a computations from the state q_i to the state q_j reading the word w in A. Then there is an nth power w^n with $w \in P$ accepted by A if and only if there is an accepting sequence,

$$q_0 \xrightarrow{w} q_{i_1} \xrightarrow{w} \ldots \xrightarrow{w} q_{i_n},$$

where q_{i_n} is a final state, i.e., if $R_{0,i_1} \cap R_{i_1,i_2} \cap \cdots \cap R_{i_{n-1},i_n} \neq \emptyset$. For each sequence $0, i_1, \ldots, i_n$, the intersection is a regular language. Moreover, there are only finitely many such sequences of length $n + 1$. Since the emptiness problem is decidable for regular languages, the claim follows.

For PSPACE completeness we need to do the reduction above to the other direction. Indeed, let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be finite automata accepting languages $L_1, \ldots, L_n$. Now construct a new automaton $\mathcal{A}$ by adding transitions reading a new symbol $\#$ from the final states of $\mathcal{A}_i$ to initial state of $\mathcal{A}_{i+1}$ for $i = 1, \ldots, n - 1$, and add new final state f to $\mathcal{A}$, such that from all final states of $\mathcal{A}_n$, there is a transition to f reading $\#$. It is immediate that $L_1 \cap \cdots \cap L_n \neq \emptyset$ if and only if there exists $w(= u\#)$ such that $w^n \in L(\mathcal{A})$. The PSPACE completeness now follows from the PSPACE completeness of emptyness of the intersection problem, see [8]. □

By setting $P = \Sigma^*$ and $n = 2$ in Theorem 3.1, we have, see also Anderson et al. [9].

Corollary 3.2. It is decidable for a given regular language $L \subseteq \Sigma^*$ whether or not there exists $w \in \Sigma^*$ such that $ww \in L$.

The proof of Theorem 3.1 also gives a well-known result on the *roots* of words: the nth root of a regular language R,

$$\sqrt[n]{R} = \{w \mid w^n \in R\},$$

as well as, the collection of all the roots,

$$\sqrt[*]{R} = \bigcup_{n \geq 2} \sqrt[n]{R} = \{w \mid w^n \in R \text{ for some } n \geq 2\},$$

are regular. Let us mention that the regularity does not hold in the limit case as we see from the following lemma.

Lemma 3.3. For a regular language R, the language

$$Pr(\sqrt[*]{R}) = \{w \mid w \text{ primitive and } w^n \in R \text{ for some } n \geq 2\}$$

is not necessarily regular.

Proof:
Let $Q = Pr(\sqrt[*]{\Sigma^*})$, that is, Q is the language of all primitive words over Σ. Assume that Q is regular over $\Sigma = \{a, b\}$, and consider the (regular) complement Q^c of Q consisting of all non-primitive words over Σ. Then $a^n b a^n b \in Q^c$ for all n, and thus for sufficiently large n, (indeed, larger that p in the Pumping Lemma 2.2) $a^{n+k} b a^n b \in Q^c$ for some $k \geq 1$ by the Pumping Lemma, but the word $a^{n+k} b a^n b$ is clearly primitive; a contradiction.

Indeed, Q is not even deterministic CF-language; see Lischke [10].   □

For the sake of completeness, we state the following theorem on the inclusion problem $C_P \subseteq L$ for the copy languages C_P.

Theorem 3.4. For regular languages R and P, it is decidable if $C_P \subseteq R$ holds. In particular, it is decidable if a regular language R is closed under taking squares, i.e., if $C_R \subseteq R$.

Proof:
The claim follows from Theorem 3.1. Indeed, $C_P \not\subseteq R$ if and only if the complement R^c of R contains a square ww with $w \in P$. Since the complement of regular language is also regular, the claim follows.
                                                                                        □

The technique in the proof of Theorem 3.1 can also be used for the marked copy problem.

Theorem 3.5. It is decidable for regular languages R, if $w\overline{w} \in R$ for some w.

Proof:
Let C be a finite automaton accepting R, i.e., $R = L(C)$. As the regular languages are closed under intersection, let A be a finite automaton accepting the language $R \cap \Sigma^* \overline{\Sigma}^*$. Furthermore, let B be a copy of A where all letters in the transitions are changed from marked to unmarked and vice versa. Therefore, $L(B) = \{\overline{u}v \mid u\overline{v} \in R\}$. We may assume that the state set of A as well as that of B is $\{q_0, \ldots, q_m\}$ and q_0 is the initial state.

Define, for all i and j, the regular language R_{ij} by

$$R_{i,j}^A = \{w \mid q_i \xrightarrow{w} q_j \text{ in } A\}$$
$$R_{i,j}^B = \{w \mid q_i \xrightarrow{w} q_j \text{ in } B\}.$$

Now, there is a word $w\overline{w} \in R$ if and only if for some state q_j and a final state q_n,

$$q_0 \xrightarrow{w} q_j \xrightarrow{\overline{w}} q_n,$$

in A (and C), i.e., if $R_{0,j}^A \cap R_{j,n}^B \cap \Sigma^* \neq \emptyset$ for some j and n. There are only finitely many intersections of regular languages to be checked, so the claim follows again from the decidability of the emptiness problem for regular languages. □

For the reverse copy problem requesting if $ww^r \in R$ for some w, we first state a more general case.

Theorem 3.6. Let $k \geq 1$ be fixed. It is decidable for regular languages R if $w_1 w_1^r \cdots w_k w_k^r \in R$ for some $w_1, \ldots, w_k$.

Proof:
Let, for fixed k,
$$E_k = \{w_1 w_1^r \cdots w_k w_k^r \mid w_i \in \Sigma^*, i = 1, 2, \ldots, k\}.$$
As a concatenation of k copies of the linear CF-language E from Example 2.1, E_k is a (nondeterministic) CF-language. Therefore, also the language

$$L_k = E_k \cap R$$

is a CF-language as context-free languages are closed under intersection with regular languages. Since the emptiness problem is decidable for context-free languages, the claim follows. □

Setting $k = 1$ in the previous theorem yields a result for the reverse copy problem.

Corollary 3.7. It is decidable for a regular language R, if $ww^r \in R$ for some word w.

The problem of Theorem 3.6 turns out to be undecidable for context-free languages with $k = 1$; see Corollary 4.15.

The proof of Theorem 3.6 can also be used for the following theorem.

Theorem 3.8. It is decidable if a regular language R contains a word of the form $w_1 w_1^r \cdots w_k w_k^r$ for some $w_1, \ldots, w_k$ and $k \geq 1$.

Proof:
Let
$$L = \bigcup_{k=1}^{\infty} E_k \,.$$

Clearly, L is a CF-language, and as in the proof of Theorem 3.6, the language $L \cap R$ is a CF-language and the claim follows from the decidability of the emptiness problem of CF-languages. □

3.2. Shuffles

We begin by defining the language

$$L_\Sigma = \bigcup_{w \in \Sigma^*} w \sqcup \overline{w}.$$

Lemma 3.9. The language L_Σ is not a CF-language if $|\Sigma| \geq 2$.

Proof:
Indeed, let $L = L_\Sigma \cap \Sigma^* \overline{\Sigma}^* = \{w\overline{w} \mid w \in \Sigma^*\}$. If L_Σ were a CF-language, then L would be a CF-language as $\Sigma^* \overline{\Sigma}^*$ is regular. However, the copy language C_{Σ^*} is a morphic image of L, and as the CF-languages are closed under morphic images, that would make C_{Σ^*} a CF-language; a contradiction. □

Engelfriet and Rozenberg [11, Theorem 15] showed in 1980 that each recursively enumerable language K can be represented in the form $K = h(L_\Sigma \cap M)$, where h is a letter-to-letter morphism, Σ a binary alphabet and M a regular language. It follows, as stated in [11], that it is undecidable for regular languages $R \subseteq \Sigma^*$ if R contains a word from $w \sqcup \overline{w}$ for some $w \in \Sigma^*$; see also [12] for a direct proof of this along different lines.

We give a simple proof of the result based on reduction to the Post Correspondence Problem. The proof below does not apply to small alphabets as the PCP is known to be undecidable only for alphabet sizes of at least five.

Theorem 3.10. It is undecidable for regular languages R if R contains an element of $w \sqcup \overline{w}$ for some w.

Proof:
Let (g, h) be an instance of the PCP for $g, h \colon \Sigma^* \to \Delta^*$, where g and h are both non-erasing. Define a (generalized) finite automaton A with the set of states

$$Q = \{v \mid v \text{ is a prefix of } h(a),\, a \in \Sigma\}.$$

We set the state $q_f = \varepsilon$ to be the initial and the unique final state. The transitions of A are of the form: for $a \in \Sigma$ and $z \in \Sigma^+$

$$u \xrightarrow{a\overline{z}} v \quad \text{if there exists } x \in \Sigma^* \text{ and } v \in Q \text{ such that } g(a) = xv \text{ and } h(z) = ux,$$
$$u \xrightarrow{a} vg(a), \quad \text{if } vg(a) \in Q.$$

Note that the words z in the above can be found algorithmically, as the images $g(a)$ are of finite length and h is non-erasing.

The states $w \in Q$ correspond to *overflows* of the instance (g, h) when g is leading: an overflow $w \in Q$ occurs in the situation, where $g(u) = h(v)w$ for some words u and v. Indeed, if the automaton A is in state w after reading the word v, then necessarily $v = a_1\overline{z_1}a_2\overline{z_2}\cdots a_n\overline{z_n}$, where $a_i \in \Sigma$ and $z_i \in \Sigma^*$ (note that z_i is empty for the transitions of the latter form) for all $i = 1, \ldots, n$, and

$$g(a_1)g(a_2)\cdots g(a_n) = h(z_1)h(z_2)\cdots h(z_n)w.$$

As $q_f = \varepsilon$, we have that A accepts the language

$$L(A) = \{a_1\overline{z_1}a_2\overline{z_2}\cdots a_n\overline{z_n} \mid n \geq 0,\ a_i \in \Sigma,\ z_i \in \Sigma^*,$$
$$g(a_1 a_2 \cdots a_n) = h(z_1 z_2 \cdots z_n)\}.$$

Therefore, $(w \sqcup \overline{w}) \cap L(A) \neq \emptyset$ for some w if and only if there exists a word $a_1\overline{z_1}a_2\overline{z_2}\cdots a_n\overline{z_n}$ in $L(A)$ such that $w = a_1 a_2 \cdots a_n = z_1 z_2 \cdots z_n$. This is equivalent to saying that the instance (g, h) of the PCP has a solution. $\qquad\square$

The self-shuffle and the shuffle with reverse are left as open problems.

Problem 2. Is it decidable for regular languages R if R contains an element of $w \sqcup w$ for some w?

Problem 3. Is it decidable for regular languages R if R contains an element of $w \sqcup w^r$ for some w?

The problems with shuffles of words tend to be algorithmically difficult. Indeed, the following was shown by Biegler and McQuillan [13].

Theorem 3.11. Consider an instance consisting of a DFA A and two words $u, v \in \Sigma^*$ with $|\Sigma| \geq 2$. It is NP-complete to determine if there exists a word $w \in L(A)$ such that $w \notin u \sqcup v$.

Problem 4. Does the above problem stay NP-complete if an instance consists of A and a single word u, and the problem is to determine if there exists a word w with $w \notin u \sqcup u$?

4.   Linear CF-languages

In this section we study the problems defined in the introduction for linear CF-languages and show that they are all undecidable.

4.1.   Powers and copies

Let (g, h) be an instance of the PCP, where $g, h \colon \Gamma^* \to \Delta^*$ with $\Gamma \cap \Delta = \emptyset$. Define the language

$$L_2(g, h) = \{zu^r xw^r : u, w \in \Gamma^+,\ z, x \in \Delta^*,\ z = g(w),\ x = h(u)\}.$$

It is an easy exercise of language theory to show that $L_2(g, h)$ is accepted by a deterministic linear pushdown automaton. Indeed, the reversal happens between u^r and x as the automaton can recognize the alternation in the disjoint alphabets. Checking if $z = g(w)$ and $x = h(u)$ can be easily performed deterministically while decreasing the stack after the reversal. Therefore, $L_2(g, h)$ is a linear CF-language.

Theorem 4.1. It is undecidable for deterministic linear CF-languages L if $ww \in L$ for some w.

Proof:
Suppose $g, h\colon \Gamma^* \to \Delta^*$ are morphisms where Γ and Δ are disjoint, and let $\Sigma = \Gamma \cup \Delta$. We rewrite the above language in a more convenient form:

$$L_2(g,h) = \{g(w)u^r h(u)w^r : u, w \in \Gamma^+\}. \tag{2}$$

Obviously, there is a square in $L_2(g,h)$ if and only if $g(w)u^r = h(u)w^r$ for some non-empty words u and w, that is, if and only if the instance (g,h) of the PCP has a nonempty solution: $g(w) = h(u)$ and $w = u$.

Since $L_2(g,h)$ is a deterministic linear CF-language, the claim follows. $\square$

Since the deterministic linear CF-languages are closed under taking complements, we have, corresponding to Theorem 3.4, the following corollary.

Corollary 4.2. It is undecidable for deterministic linear CF-languages L if all squares are in L, i.e., if $C_{\Sigma^*} \subseteq L$.

Proof:
We consider the negation of the proposition in Theorem 4.1:

$$\neg \exists w : ww \in L \iff \forall w : ww \notin L \iff \forall w : ww \in L^c.   \square$$

For the marked copy, we transform the language $L_2(g,h)$ in (2) into the language

$$\bar{L}_2(g,h) = \{g(w)u^r \overline{h(u)}\overline{w}^r : u, w \in \Gamma^*\}$$

which is clearly also a deterministic linear CF-language.

Theorem 4.3. It is undecidable for deterministic linear CF-languages L, if $w\overline{w} \in L$ for some word w.

Proof:
As in the proof of Theorem 4.1, because the alphabets Δ and Γ are disjoint (therefore, so are $\overline{\Delta}$ and $\overline{\Gamma}$) we obtain that there is a word of the form $w\overline{w}$ in $\bar{L}_2(g,h)$ if and only if $g(w)u^r = h(u)w^r$ which again is equivalent to saying that the instance (g,h) of the PCP has a solution. $\square$

Example 4.4. The language

$$F = \{w \in \Sigma^* \mid w = uxxv \text{ for some nonempty word } x\}$$

is not regular. Indeed, by the Pumping lemma, its complement, the language of all square-free words, is not even context-free. For this, assume contrary that F^c is a CF-language. As it is infinite, pumping in Lemma 2.3, implies that for every sufficiently long word $w \in F^c$ contains a factor of the form xx for some nonempty word x. Therefore, there is a word of the form $uxxv$ in F^c; a contradiction.

We extend Theorem 4.1 for arbitrary powers $n \geq 2$ as follows.

Theorem 4.5. Let $n \geq 2$ be a fixed integer. It is undecidable for deterministic linear CF-languages $L \subseteq \Sigma^*$ if there exists a power $w^n \in L$ for some nonempty $w \in \Sigma^*$.

Proof:
Our construction relies on the language $L_2(g, h)$ defined in (2). Let

$$L_n(g, h) = L_2(g, h) \cdot (\#\Delta^+\Gamma^+)^{n-2}.$$

It is a deterministic linear CF-language since the end portion $(\Delta^+\Gamma^+)^{n-2}$ is regular. Recall that $\Delta \cap \Gamma = \emptyset$.

Now,
$$g(w)u^r h(u)w^r \cdot w_1 u_1 \cdots w_{n-2} u_{n-2} \in L_n(g, h)$$

is an nth power if and only if $g(w) = h(u) = w_1 = \ldots = w_{n-2}$ and $u^r = w^r = u_1 = \ldots = u_{n-2}$, and thus if and only if the instance (g, h) has a solution. This proves the claim. □

Finally, if we replace $L_n(g, h)$ by the deterministic linear CF-language

$$L_\omega(g, h) = L_2(g, h) \cdot (\Delta^+\Gamma^+)^*,$$

we have,

Theorem 4.6. It is undecidable for deterministic linear CF-languages $L \subseteq \Sigma^*$ if there exists a power $w^n \in L$ for some $n \geq 2$ and nonempty $w \in \Sigma^*$.

Regarding the decidability result in Theorem 3.4 for regular languages, we state an open problem for linear CF-languages.

Problem 5. Is it decidable for linear CF-languages L if $C_L \subseteq L$, i.e., if L is closed under taking squares?

In many special cases the answer to the above problem is positive. Indeed, according to Greibach [14] if a language $L_1 c L_2$, with $L_i \subseteq (\Sigma \setminus \{c\})^*$ for $i = 1, 2$, is a linear CF-language then L_i is regular for $i = 1$ or $i = 2$.

The inclusion problem of squares in a language becomes undecidable "just above" the regular languages. Recall that counter languages are accepted with (nondeterministic) pushdown automata with a single pushdown letter for the stack.

Theorem 4.7. It is undecidable for counter languages $L \subseteq \Sigma^*$ if $C_{\Sigma^*} \subseteq L$ holds.

Proof:
We prove the claim by reduction from the PCP. Let (g, h) be an instance of the PCP for $g, h \colon \Sigma^* \to \Delta^*$. Let $\Gamma = \Sigma \cup \{\#\}$, where $\#$ is a new letter.

We now describe a (nondeterministic) counter language $L \subseteq \Gamma^*$ such that $w \in L$ if

(1) $w \neq u \# v \#$, for all $u, v \in \Sigma^*$, or

(2) $w = u\#v\#$ but $u \neq v$ or $g(u) \neq h(v)$.

Clearly, L contain all squares of Γ^* except those words $w\#w\#$ with $g(w) = h(w)$. Thus $C_{\Sigma^*} \subseteq L$ if and only if the instance (g, h) has no solutions, and the claim follows from the undecidability of the PCP.

Starting from its initial state the automaton M branches to one of the three separate lines of actions, and it accepts the input w if one of these lines leads to acceptance. Note that counter automata are nondetermistic, and their languages are closed under union.

The automaton M accepts if

1. $w \notin \Sigma^*\#\Sigma^*\#$. Since $\Sigma^*\#\Sigma^*\#$ is regular this can be decided with the states, without counter actions.

   From this on, we suppose that $w = u\#v\#$, where $u, v \in \Sigma^*$.

2. $u \neq v$. While reading the prefix u and increasing the counter, the automaton guesses a position $n \leq |u|$, say with the letter a. The counter stack has then c^n. The automaton reads on until it reaches the first $\#$, after which it pops the counter to gain the nth letter, say b, of v. It accepts if $a \neq b$, that is the nth letter of u is not equal to nth letter of v.

3. $g(u) \neq h(v)$. As in case 2 the automaton can guess a position of different letter while in the images $g(u)$ and $h(v)$. E.g., while reading $u = a_1 \cdots a_m$, the automaton guesses a position $n = |g(a_1 a_2 \cdots a_k)| + j$ by pushing $c^{|g(a_i)|}$ $i = 1, 2, \ldots, k$, to the counter when reading $a_1, \ldots, a_k$ and then pushing c^j, for $j < |g(a_{k+1})|$, and remembers the symbol a which is the ith symbol of $g(a_{k+1})$ and, therefore, the nth symbol of $g(u)$.

   Then, when reading $v = b_1 \cdots b_t$, M decreases the counter with $c^{|h(b_i)|}$ until $|h(b_1 \cdot b_\ell)| + s = n$ for some ℓ such that $s < |h(b_{\ell+1})|$ and checks that the symbol in position s of $h(b_{\ell+1})$, that is, the nth symbol of $h(v)$ are different.

It is straightforward to see that M accepts the language L.   □

4.2. Shuffles

We begin with the shuffle operation on CF-languages. By considering the generating grammars, it is evident that the family of context-free languages is closed under concatenation. However, it is not closed under shuffle of languages. To see this, consider the languages

$$L_1 = \{a^n b^n \mid n \geq 1\}, \; L_2 = \{c^m a^m \mid m \geq 1\} \; \text{ and } \; L = L_1 \shuffle L_2 \cap a^* c^* b^* a^*.$$

If $L_1 \shuffle L_2$ is a CF-language, then so is L as CF-languages are closed under intersection with regular languages and $a^* c^* b^* a^*$ is regular. But $L = \{a^n c^m b^n a^m \mid n, m \geq 1\}$ is a well-known non-CF-language. Note that the languages L_1 and L_2 are even deterministic linear CF-languages.

The following example showing that CF-languages over binary alphabets are not closed under shuffle is due to Chris Köcher [15].

Example 4.8. Let $L_1 = \{a^n b a^n \mid n \geq 1\}$ and $L_2 = \{b^n a b^n \mid n \geq 1\}$. Now

$$(L_1 \shuffle L_2) \cap a^* b^* a^* b^* = \{a^m b^{n+1} a^{m+1} b^n \mid m, n \geq 1\},$$

which is not a context-free language. As the language $a^* b^* a^* b^*$ is regular, and intersection of a context-free language and a regular language is always context-free, we get that $L_1 \shuffle L_2$ is not a context-free language.

Next, we study the shuffle problem for linear CF-languages.

Theorem 4.9. It is undecidable for deterministic linear CF-languages L if L contains an element of $w \shuffle w$ for some w.

Proof:
Recall the language $L_2(g, h)$ from (2), and consider its modification

$$L_\#(g, h) = \{\$g(w)u^r \# \$h(u)w^r \# : u, w \in \Gamma^*\}, \tag{3}$$

where the markers \$ and \# appears only in the given positions. Clearly, $L_\#(g, h)$ is deterministic linear CF-language.

The only word of the form $\$u\#\$v\#$ in the shuffle $\$w\# \shuffle \$w\#$ is $\$w\#\$w\#$, which in its turn belongs to $L_\#(g, h)$ if and only if the instance (g, h) has a solution, the claim follows. $\quad\square$

Corollary 4.10. It is undecidable for deterministic linear CF-languages $L \subseteq \Sigma^*$ if all shuffles $w \shuffle w$ for $w \in \Sigma^*$ are in L.

Proof:
We consider the negation of the proposition in Theorem 4.9:

$$\neg\exists w : w \shuffle w \cap L \neq \emptyset \iff \forall w : w \shuffle w \cap L = \emptyset \iff \forall w : w \shuffle w \subseteq L^c.$$

Since the family of deterministic linear CF-languages is closed under complement, the claim follows.
$$\square$$

By Rizzi and Vialette [16] and Buss and Soltys [17], it is an NP-complete problem if a word v is a self-shuffle, i.e., if there exists a word w such that $v = w \shuffle w$.

Consider the shuffle $w \shuffle w^r$, where w^r is the reverse of w. For a language P, let

$$M_P = \bigcup_{w \in P} w \shuffle w^r.$$

See Henshall et al. [18] for the following result.

Theorem 4.11. The language M_{Σ^*} is not context-free.

We give a bit simpler result related to the previous theorem as an example.

Example 4.12. We show that the language M_P need not be regular for regular P. Indeed, let $P = a^+b^+$ over $\Sigma = \{a, b\}$. Then $L = M_P \cap a^*b^*a^* = \{a^n b^{2m} a^n \mid n, m \geq 1\}$ is non-regular (by the Pumping Lemma). Therefore, M_P is not regular.

Similarly, let $L = \{a^n b^n \mid n \geq 1\}$. Now,

$$M_L \cap a^*b^*a^* = \bigcup_{n \geq 1} (a^n b^n \amalg b^n a^n) \cap a^*b^*a^* = \{a^n b^{2n} a^n \mid n \geq 1\}$$

is not context-free, and, therefore, M_L is not context-free, since $a^*b^*a^*$ is regular.

Our next theorem concerns linear CF-languages that need not be deterministic.

Theorem 4.13. It is undecidable for linear CF-languages if L contains an element of $w \amalg w^r$ for some w.

Proof:
The proof is by reduction from the PCP. Let (g, h) be an instance of the PCP with $g, h \colon \Sigma^* \to \Delta^*$ and let $\#$ be a new symbol $\# \notin \Delta$. Consider the linear CF-language $L_1 = L(G)$ generated by the grammar G with two non-terminals S and T together with the production rules

$$S \to \#g(a)Th(a)^r\# \ \text{ for all } a \in \Sigma,$$
$$T \to g(a)Th(a)^r \ \text{ for all } a \in \Sigma,$$
$$T \to \#\#.$$

Hence

$$L_1 = \{\#g(v)\#\#h(v)^r\# \mid v \in \Sigma^+\}. \tag{4}$$

Now, L_1 contains an element of $w \amalg w^r$ for a word w if and only if $w = \#g(v)\# = \#h(v)\#$ for some nonempty word v for which then $g(v) = h(v)$. The claim follows from the undecidability of the PCP. $\qquad \square$

The linear CF-language L_1 in (4) also gives immediately the following result.

Theorem 4.14. It is undecidable for linear CF-languages L, if $ww^r \in L$ for some word w.

The proof of Theorem 4.13 also gives the following corollary.

Corollary 4.15. It is undecidable for linear CF-languages L if L contains a palindrome, i.e, a word w such that $w = w^r$.

The following result is a trivial consequence of Theorem 3.10, which stated the result already for regular languages.

Theorem 4.16. It is undecidable for linear context-free languages if L contains an element of $w \amalg \overline{w}$ for some w.

In contrast to the decidability result in Theorem 3.6 we can extend the proof of Theorem 4.13 for the next claim.

Theorem 4.17. Let $k \geq 1$ be fixed. It is undecidable for CF-languages L if L contains a word of the form $w_1 w_1^r \cdots w_k w_k^r$ for some $w_1, \ldots, w_k$.

Proof:
We modify the linear CF-language from (4). Consider the 'next' linear CF-language L_k for the instance (g, h): $L_k = L_1 \cdot (cc)^{k-1}$, where c is a new letter. □

For regular languages R it is clearly decidable if $\{w, w^r\}^* \subseteq R$ for given w.

Problem 6. Is it decidable for regular languages R if $\{w, w^r\}^* \subseteq R$ for some w? How about CF-languages?

On the other hand, it is decidable for context-free languages L if $L \subseteq \{w, w^r\}^*$ for some w. Indeed, by the Pumping property, the length of possible words w has an effective upper bound.

5.   Conclusions

Our aim was to present a survey on the decidability statuses of special membership problems for copies, marked copies, reversed copies, self-shuffles and shuffles with marked or reversed copies. Two cases, the self-shuffle and shuffle with the reversed copy, remain unsolved. We also studied special inclusion problems regarding powers and especially squares of words. Several related open problem were stated.

References

[1] Rampersad N, Shallit J. Detecting patterns in finite regular and context-free languages. *Inf. Process. Lett.*, 2010. **110**(3):108–112. doi:10.1016/j.ipl.2009.11.002.

[2] Salomaa A. Formal Languages. Academic Press, New York, 1973.

[3] Kutrib M, Malcher A, Wendlandt M. Queue Automata: Foundations and Developments, pp. 385–431. Springer International Publishing. ISBN 978-3-319-73216-9, 2018. doi:10.1007/978-3-319-73216-9_19. URL https://doi.org/10.1007/978-3-319-73216-9_19.

[4] Post EL. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 1946. **52**:264–268. doi:10.1090/S0002-9904-1946-08555-9. URL http://dx.doi.org/10.1090/S0002-9904-1946-08555-9.

[5] Ehrenfeucht A, Karhumäki J, Rozenberg G. The (generalized) Post correspondence problem with lists consisting of two words is decidable. *Theoret. Comput. Sci.*, 1982. **21**(2):119–144. doi:10.1016/0304-3975(89)90080-7. URL http://dx.doi.org/10.1016/0304-3975(89)90080-7.

[6] Halava V, Harju T, Hirvensalo M. Binary (generalized) Post correspondence problem. *Theoret. Comput. Sci.*, 2002. **276**(1-2):183–204. doi:10.1016/S0304-3975(01)00157-8. URL http://dx.doi.org/10.1016/S0304-3975(01)00157-8.

[7] Neary T. Undecidability in binary tag systems and the Post correspondence problem for five pairs of words. In: 32nd International Symposium on Theoretical Aspects of Computer Science, volume 30 of *LIPIcs. Leibniz Int. Proc. Inform.*, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015 pp. 649–661. doi:10.5167/uzh-121761.

[8] Rabin MO, Scott D. Finite Automata and Their Decision Problems. *IBM J. Res. Dev.*, 1959. **3**(2):114–125. doi:10.1147/rd.32.0114. URL `https://doi.org/10.1147/rd.32.0114`.

[9] Anderson T, Loftus J, Rampersad N, Santean N, Shallit J. Detecting palindromes, patterns and borders in regular languages. *Inf. Comput.*, 2009. **207**(11):1096–1118. doi:10.1016/j.ic.2008.06.007.

[10] Lischke G. Primitive words and roots of words. *Acta Univ. Sapientiae, Inform.*, 2011. **3**(1):5–34.

[11] Engelfriet J, Rozenberg G. Fixed point languages, equality languages, and representation of recursively enumerable languages. *J. Assoc. Comput. Mach.*, 1980. **27**:499–518.

[12] Halava V, Harju T, Sahla E. On shuffling a word with its letter-to-letter substitution. *Fundamenta Informaticae*, 2020. **175**:201–206.

[13] Biegler F, McQuillan I. On comparing deterministic finite automata and the shuffle of words. In: Implementation and application of automata. 19th international conference, CIAA 2014, Giessen, Germany, July 30 – August 2, 2014. Proceedings, pp. 98–109. Berlin: Springer. ISBN 978-3-319-08845-7/pbk, 2014.

[14] Greibach S. The unsolvability of the recognition of linear context-free languages. *J. Assoc. Comput. Mach.*, 1966. **13**:582–587.

[15] Köcher C. personal communication.

[16] Rizzi R, Vialette S. On recognizing words that are squares for the shuffle product. In: Computer science – theory and applications. 8th international computer science symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25–29, 2013. Proceedings, pp. 235–245. Berlin: Springer. ISBN 978-3-642-38535-3/pbk, 2013.

[17] Buss S, Soltys M. Unshuffling a square is NP-hard. *J. Comput. Syst. Sci.*, 2014. **80**(4):766–776.

[18] Henshall D, Rampersad N, Shallit J. Shuffling and Unshuffling. *Bulletin of the EATCS*, 2012. (107):131–142. `1106.5767v4`.