Unique Decipherability in Formal Languages

Paul C. Bell[a,*], Daniel Reidenbach[b], Jeffrey O. Shallit[c,1]

[a] *Department of Computer Science, Liverpool John Moores University, Liverpool, Merseyside, L3 3AF, United Kingdom*
[b] *Department of Computer Science, Loughborough University, Loughborough, Leicestershire, LE11 3TU, United Kingdom*
[c] *School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada*

Abstract

We consider several language-theoretic aspects of various notions of *unique decipherability* (or unique factorization) in formal languages. Given a language L at some position within the Chomsky hierarchy, we investigate the language of words $\mathrm{UD}(L)$ in L^* that have unique factorization over L. We also consider similar notions for weaker forms of unique decipherability, such as *numerically decipherable* words $\mathrm{ND}(L)$, *multiset decipherable* words $\mathrm{MSD}(L)$ and *set decipherable* words $\mathrm{SD}(L)$. Although these notions of unique factorization have been considered before, it appears that the languages of words having these properties have not been positioned in the Chomsky hierarchy up until now.

We show that $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ and $\mathrm{SD}(L)$ need not be context-free if L is context-free. In fact $\mathrm{ND}(L)$ and $\mathrm{MSD}(L)$ need not be context-free even if L is finite, although $\mathrm{UD}(L)$ and $\mathrm{SD}(L)$ are regular in this case. We show that if L is context-sensitive, then so are $\mathrm{UD}(L)$, $\mathrm{ND}(L)$, $\mathrm{MSD}(L)$ and $\mathrm{SD}(L)$. We also prove that the membership problem (resp., emptiness problem) for these classes is PSPACE-complete (resp., undecidable). We finally determine upper and lower bounds on the length of the shortest word of L^* not having the various forms of unique decipherability into elements of L.

Keywords: codes, unique decipherability, numerical decipherability, set decipherability, multiset decipherability

1. Introduction

Let L be a formal language over an alphabet Σ. We say that $w \in L^*$ is *uniquely decipherable* (or has *unique factorization*) if there exists a unique factorization of w into elements of L (formal definitions of this, and other notions of decipherability, are given in Section 2.2). If every element of L^* has unique

*Corresponding author
Email addresses: p.c.bell@ljmu.ac.uk (Paul C. Bell), D.Reidenbach@lboro.ac.uk (Daniel Reidenbach), shallit@uwaterloo.ca (Jeffrey O. Shallit)
[1] Supported by NSERC grant no. 105829/2013

factorization into elements of L, then L is called a *uniquely decipherable* (UD) code. Every source message encoded by a UD code can therefore be decoded in its entirety. Codes are important in a number of contexts — for example in the transmission of messages, where we may attempt to minimize data traffic if the items of data to be encoded have an *a priori* known frequency distribution.

The situation may arise, however, where we only require the facility to extract some *partial* information from an encoded message, without necessarily being able to retrieve the entire message. This naturally leads to various weaker notions of unique factorization of encoded messages. Three of the most common, and those studied in this paper, are *numerically decipherable* (ND) codes, *multiset decipherable* (MSD) codes and *set decipherable* (SD) codes.

The concept of a *numerically decipherable* (ND) code was introduced by Weber and Head [15], where a language L is called an ND code if every element $w \in L^*$ has the property that all possible factorizations of w over L use exactly the same number of factors, but the actual factors themselves are unimportant. A motivating example given by Weber and Head is that of sending a stock inventory between two parties, where the number of items is important but the actual items themselves can be ignored. They showed that determining whether a given finite set L of n words with a total length of t over an alphabet Σ is an ND code can be decided in time $O(nt)$ and in $O((n + |\Sigma|)t)$ space [15]. In fact, the algorithm can also be used to determine if L is a UD and SD code (defined below), with the same space and time bounds.

The notion of a *multiset decipherable* (MSD) code was introduced by Lempel [10]. An MSD code L allows unique factorization of words in L^* if the order of transmission of the codewords is ignored; thus each factorization of an element of L^* uses the same *multiset* of elements and they are permutations of each other. Lempel gave motivating examples of online compilations of inventories, construction of histograms and updating of relative frequencies where the order of elements is not important, only the multiplicity of code words.

An MSD code is called *proper* if it is an MSD code, but not a UD code (a similar terminology can be defined for SD and ND codes). Lempel proved that for $n \geq 4$ and $|\Sigma| \geq 2$, there exists a finite set L of n elements over alphabet Σ such that L is a proper MSD code, but every MSD code of just two words is a UD code [10].

Another variation on unique factorization named *set decipherable* (SD) codes was introduced by Guzmán [5], and was also explored by Blanchet-Sadri and Morgan [3]. A language L is an SD code if every factorization of an element of L^* over L uses the same *set* of elements. Thus, a message encoded by an SD code can be uniquely deciphered up to both commutativity of elements and the number of occurrences of elements. A motivating example here, given in [5], is that of two rival parties, willing to reveal to each other the presence or absence of certain items without revealing the exact count.

Let CO denote the set of all codes. Guzmán showed [5] that UD $\subset$ MSD $\subset$ SD $\subset$ CO, where we note that all inclusions are strict and thus there exist proper MSD and SD codes. On the other hand, certainly UD $\subseteq$ SD $\subseteq$ ND and there do exist ND codes that are not SD codes (see Example 13), so these

containments are proper. These containments are shown in Figure 1. Also note that ND and SD are incomparable, since Example 13 exhibits a finite language L that is an ND code but not an SD code, and Example 20 (from [5]) exhibits a language L' that is an SD code but not an ND code.

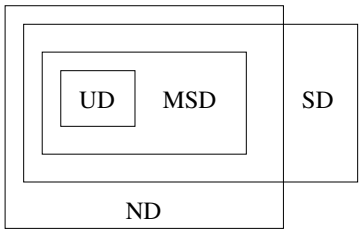


Figure 1: Proper containments of classes UD, ND, MSD and SD.

Another aspect of codes that has been studied in the literature concerns the minimal length of codewords. The McMillan sum $\mu : 2^{\Sigma^*} \to \mathbb{Q}$ of a finite code C over an alphabet Σ is defined as follows:

$$\mu(C) = \sum_{w \in C} |\Sigma|^{-|w|}.$$

It is known that every finite UD code L satisfies $\mu(L) \leq 1$; see [4]. This gives some indication that code words cannot become "too short". A UD code L is called *full* if $\mu(L) = 1$. Blanchet-Sadri and Morgan showed [3] that no SD code contains a full UD code as a proper subcode. Lempel conjectured [10] that MSD codes satisfy Kraft's inequality, but this was later disproved by Restivo [12]. Kraft's inequality is not further considered in this paper.

Although codes and their properties have been studied extensively (see, for example, [2, 9]), in this paper we look at some novel aspects of unique factorization. Namely, we consider the language of words possessing the various types of unique factorization, and position the resulting language in the Chomsky hierarchy. We also consider the length of the shortest word *not* having the desired property, if it exists.

A preliminary version of this paper appeared in [1]. In the present version, we improve the upper bound on the maximal length of a minimal word having subset-invariant factorization, if any such word exists, from $O(m^2n^2)$ to $O(m^2n)$ and study various complexity issues in Section 7.

2. Notation and preliminaries

2.1. Basic notation

As usual, we let Σ denote a finite alphabet, Σ^* denote the set of all *words* over Σ, and Σ^+ denote the set of all nonempty words over Σ. A subset $L \subseteq \Sigma^*$ is called a *language*. We denote by L^* the set $\{w_1 w_2 \cdots w_k | k \geq 0$ with each $w_i \in L\}$ and by L^+ the set $\{w_1 w_2 \cdots w_k | k \geq 1$ with each $w_i \in L\}$. A *code* is a

nonempty subset $C \subseteq \Sigma^+$. Each element of C is called a *codeword*. A *message* over C is a concatenation of the codewords of C. Given a word $w \in \Sigma^*$, we let $\text{suff}(w)$ denote the set of all suffixes of w. The reader unfamiliar with these notions can consult [2].

2.2. Variants of unique factorization

We now formally define *uniquely decipherable* (UD), *numerically decipherable* (ND), *multiset decipherable* (MSD) and *set decipherable* (SD) codes.

Definition 1 (UD code)**.** Given L, we say that $w \in L^*$ is *uniquely decipherable* if w has a unique factorization over elements of L. More precisely, w is uniquely decipherable if whenever

$$w = u_1 u_2 \cdots u_m = v_1 v_2 \cdots v_n$$

for $u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n \in L$, then $m = n$ and $u_i = v_i$ for all $1 \leq i \leq n$. Given a language L, we define $\text{UD}(L)$ to be the set of all elements of L^* that are uniquely decipherable over L. If $L^* = \text{UD}(L)$, then L is called a UD code.

Note that UD codes are the standard type of code studied in the theory of codes, see [2]. We now define some additional variants of such codes.

Definition 2 (MSD code)**.** Given a language L we say $w \in L^*$ is *multiset decipherable* if whenever $w = u_1 u_2 \cdots u_m = v_1 v_2 \cdots v_n$ for

$$u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n \in L,$$

then $m = n$ and there exists a permutation σ of $\{1, \ldots, n\}$ such that $u_i = v_{\sigma(i)}$ for $1 \leq i \leq n$. In other words, we consider two factorizations that differ only in the order of the factors to be the same. We define $\text{MSD}(L)$ to be the set of $w \in L^*$ that are multiset decipherable. We may equivalently say that w is multiset decipherable if all factorizations of w over L have the same *multiset*, hence the terminology. If $L^* = \text{MSD}(L)$, then L is called an MSD code.

Definition 3 (ND code)**.** Given L, we say that $w \in L^*$ is *numerically decipherable* if all factorizations of w into elements of L consist of the same number of factors. More precisely, w is numerically decipherable if whenever

$$w = u_1 u_2 \cdots u_m = v_1 v_2 \cdots v_n$$

for $u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n \in L$, then $m = n$. Given a language L, we define $\text{ND}(L)$ to be the set of all elements of L^* that are numerically decipherable over L. If $L^* = \text{ND}(L)$, then L is called an ND code.

Definition 4 (SD code)**.** Given a language L, we say $w \in L^*$ is *set decipherable* (or has *subset-invariant factorization*) over L if there exists a subset $S \subseteq L$ such that every factorization of w into elements of L uses exactly the elements of S — no more, no less — although each element may be used a different number of times. More precisely, w has subset-invariant factorization if there exists $S = S(w) \subseteq L$ such that whenever $w = w_1 w_2 \cdots w_k$ with $w_1, w_2, \ldots, w_k \in L$, then $S = \{w_1, w_2, \ldots, w_k\}$. We let $\text{SD}(L)$ denote the set of $w \in L^*$ that are set decipherable. If $L^* = \text{SD}(L)$, then L is called an SD code.

We call a UD code L a *prefix (suffix) code* if there does not exist $w_1, w_2 \in L$ with $w_1 \neq w_2$ such that w_1 is a prefix (suffix) of w_2. It is clear that for every prefix or suffix code L, we have $L^* = \mathrm{UD}(L)$.

The paper is split into several sections. We classify the various types of unique factorization according to the position of L in the Chomsky hierarchy in Section 3 for *unique decipherability*, in Section 4 for *numerically decipherability*, in Section 5 for *multiset decipherability*, and in Section 6 for *set decipherability*. In Section 7 we consider the complexity of membership and emptiness for these types of factorizations. We show that if L is a context-free language, then $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L), \mathrm{SD}(L)$ are all context-sensitive languages. Furthermore, given a context-free grammar (CFG) G for L, we can constructively produce a context-sensitive grammar (CSG) for each of $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ and $\mathrm{SD}(L)$, and therefore the corresponding membership problem is decidable.

We also show that if L is context-free then the emptiness problem for $\mathrm{UD}(L)$ is undecidable, whereas if L is context-sensitive, then the membership problem for $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ and $\mathrm{SD}(L)$ is PSPACE-complete.

3. Uniquely decipherable (UD) codes

We begin with the following proposition, which is folklore.

Proposition 5. *If L is regular, then so is* $\mathrm{UD}(L)$.

Proof. If L contains the empty word ϵ then no elements of L^* are uniquely decipherable, and so $\mathrm{UD}(L) = \emptyset$. So, without loss of generality we can assume $\epsilon \notin L$.

To prove the result, we show that the relative complement $L^* - \mathrm{UD}(L)$ is regular (regular languages themselves being closed under complement). Let L be accepted by a deterministic finite automaton (DFA) M. On input $x \in L^*$, we build a nondeterministic finite automaton (NFA) M' to guess two different factorizations of x and verify they are different. The machine M' maintains the single state of the DFA M for L as it scans the elements of x, until M' reaches a final state q. At this point M' moves, via an ϵ-transition, to a new kind of state that records pairs of states of M. Transitions on these "doubled" states (denoted $[p, q]$ for states $p, q \in Q$) still follow M's transition function in both coordinates, with the exception that if either state is in F, we allow a "reset" implicitly to q_0. Each implicit return to q_0 marks, in a factorization, the end of a term. The final states of M' are the "doubled" states with both elements in F.

More precisely, assume $M = (Q, \Sigma, \delta, q_0, F)$. Since $\epsilon \notin L(M)$, we know $q_0 \notin F$. We create the machine $M' = (Q', \Sigma, \delta', q_0, F')$ as follows: we let $Q' = Q \cup Q \times Q$ and

$$\delta'(q, a) = \begin{cases} \{\delta(q, a)\}, & \text{for all } q \in Q; \\ \{\delta(q_0, a),\ [\delta(q_0, a), \delta(q, a)]\}, & \text{if } q \in F. \end{cases}$$

Writing $r = \delta(p, a)$, $s = \delta(q, a)$, $t = \delta(q_0, a)$, we also set

$$\delta'([p, q], a) = \begin{cases} \{[r, s]\}, & \text{if } p \notin F,\ q \notin F; \\ \{[r, s], [t, s]\}, & \text{if } p \in F,\ q \notin F; \\ \{[r, s], [r, t]\}, & \text{if } p \notin F,\ q \in F; \\ \{[r, s], [t, s], [r, t], [t, t]\}, & \text{if } p \in F,\ q \in F. \end{cases}$$

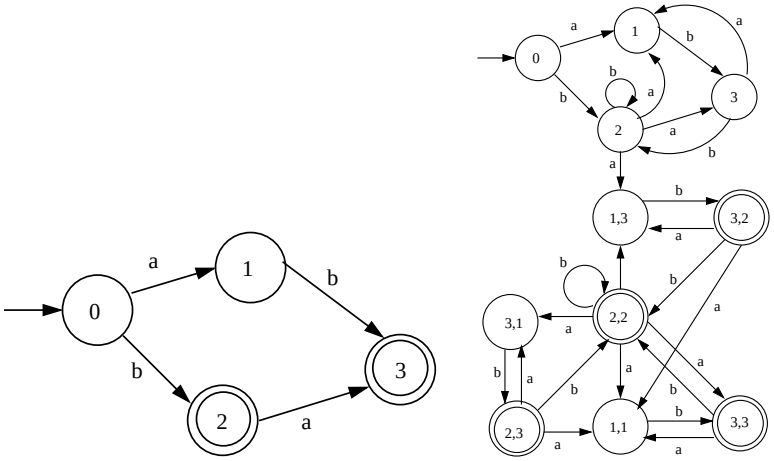
Finally, we set $F' = F \times F$. To see that the construction works, suppose that $x \in L^*$ has two different factorizations

$$x = y_1 y_2 \cdots y_j y_{j+1} \cdots y_k = y_1 y_2 \cdots y_j z_{j+1} \cdots z_\ell$$

with y_{j+1} a proper prefix of z_{j+1}. Then an accepting path starts with singleton sets until the end of y_j. The next transition goes to a pair having first element $\delta(q_0, a)$ with a the first letter of y_{j+1}. Subsequent transitions eventually lead to a pair in $F \times F$.

On the other hand, if x is accepted, then two different factorizations are traced out by the accepting computation in each coordinate. The factorizations are guaranteed to be different by the transition to $[\delta(q_0, a), \delta(q, a)]$. □

Example 6. Let $L = \{b, ab, ba\}$. A DFA recognising L is simple to describe; see Fig. 2a. (We have omitted the single dead state.) Applying the procedure of Proposition 5 allows us to construct an NFA to recognise the relative complement $L^* - \mathrm{UD}(L)$, as shown in Fig. 2b.



(a) DFA for L (b) NFA for $L^* - \mathrm{UD}(L)$

Figure 2: Language L and $L^* - \mathrm{UD}(L)$

Remark 7. There is a shorter and more transparent proof of Proposition 5, as follows. Given a DFA for L, create an NFA A for L^* by adding ϵ-transitions from every final state back to the initial state, and then removing the ϵ-transitions

using the familiar method (e.g., [7, Theorem 2.22]). Next, using the Boolean matrix interpretation of finite automata (e.g., [16] and [13, §3.8]), we can associate an adjacency matrix M_a with the transitions of A on the letter a. Then, on input $x = a_1 a_2 \cdots a_i$, a DFA can compute the matrix $M_x = M_{a_1} M_{a_2} \cdots M_{a_i}$ using ordinary integer matrix multiplication, with the proviso that every entry that is 2 or more is changed to 2 after each matrix multiplication. This can be done by a DFA since the number of such matrices is at most 3^{n^2} where n is the number of states of M. Then, accepting if and only if the entry in the row and column corresponding to the initial state of A is 1, we get a DFA accepting exactly those x having unique factorization into elements of L. While this proof is much simpler, the state bound it provides is quite extravagant compared to our previous proof.

Proposition 8. *Suppose L is accepted by a DFA with n states. If L is not a UD code, then there exists a word $x \in L^*$ with at least two distinct factorizations into elements of L, with $|x| < n^2 + n$.*

Proof. Our construction in the proof of Proposition 5 gives an NFA M' accepting all words with at least two different factorizations, and it has $n^2 + n$ states. If M' accepts anything at all, it accepts a word of length at most $n^2 + n - 1$. □

Proposition 9. *For all $n \geq 2$, there exists an $O(n)$-state DFA accepting a language L that is not a UD code, such that the shortest word in L^* having two factorizations into elements of L is of length $\Omega(n^2)$.*

Proof. Consider the language $L_n = b(a^n)^* \cup (a^{n+1})^* b$. It is easy to see that L_n can be accepted by a DFA with $2n + 5$ states, but the shortest word in L_n^* having two distinct factorizations into elements of L_n is $b\, a^{n(n+1)}\, b$, of length $n^2 + n + 2$. □

In fact, there are even examples of finite languages with the same property.

Proposition 10. *For all $n \geq 2$, there exists an $O(n)$-state DFA accepting a finite language L that is not a UD code, such that the shortest word in L^* having two factorizations is of length $\Omega(n^2)$.*

Proof. Let $\Sigma = \{b, a_1, a_2, \ldots, a_n\}$ be an alphabet of size $n + 1$, and let L_n be the language of $2n$ words

$$\{a_1, a_n\} \cup \{b^i a_{i+1} \ : \ 1 \leq i < n\} \cup \{a_i b^i \ : \ 1 \leq i < n\}$$

defined over Σ.

Then it is easy to see that L_n can be accepted with a DFA of $2n + 2$ states, while the shortest word having two distinct factorizations is

$$a_1 b a_2 b^2 a_3 b^3 \cdots a_{n-1} b^{n-1} a_n,$$

which is of length $n(n + 1)/2$. □

Remark 11. The previous example can be recoded over a three-letter alphabet by mapping each a_i to the base-2 representation of i, padded, if necessary, to make it of length ℓ, where $\ell = \lceil \log_2 n \rceil$. With some reasonably obvious reuse of states this can still be accepted by a DFA using $O(n)$ states, and the shortest word with two distinct factorizations is still of length $\Omega(n^2)$.

Theorem 12. *If L is a context-free language, then* $\mathrm{UD}(L)$ *need not be context-free.*

Proof. Our example is based on two languages (see, for example, [11]):

(a) `PALSTAR`, the set of all strings over the alphabet $\Sigma = \{0, 1\}$ that are the concatenation of one or more even-length palindromes; and

(b) `PRIMEPALSTAR`, the set of all elements of `PALSTAR` that *cannot* be written as the concatenation of two or more elements of `PALSTAR`.

Clearly `PALSTAR` is a context-free language (CFL). We see that $\mathrm{UD}($`PALSTAR`$) =$ `PRIMEPALSTAR`, which was proven in [11] to be non-context-free. □

4. Numerically decipherable (ND) codes

We now investigate numerically decipherable codes, where uniqueness is defined for a word if all factorizations of that word over a given language have the same *number* of factors (see Definition 3 for formal details). We start with a motivating example.

Example 13. Let $L = \{ab, bab, abb\}$. Then $\mathrm{ND}(L) = L^*$. This follows since every $w \in L^*$ with more than one factorization over L must contain a word of the form $ab(bab)^k bab = (abb)^{k+1}ab$ for $k \geq 0$, with the number of factors in each case being $k+2$. Note that L is not a UD code, however, since $ab \cdot bab = abb \cdot ab$ has two factorizations. Therefore L is a proper ND code.

We now prove an analogous result to Proposition 5 for ND codes, noting that if L is regular, then $\mathrm{ND}(L)$ sits higher in the Chomsky hierarchy than $\mathrm{UD}(L)$.

Theorem 14. *If L is regular then* $\mathrm{ND}(L)$ *need not be a CFL.*

Proof. We define regular language L by the following rational expression:

$$L = a0^+b + 1 + c(23)^+ + 23d + a + 0 + b1^+c(23)^+ + a0^+b1^+c2 + 32 + 3d.$$

Consider $\mathrm{ND}(L)$ and intersect with the regular language $a0^+b1^+c(23)^+d$.

Then there are only three possible factorizations for a given word here. They look like (using parentheses to indicate factors)

- $(a0^ib)1 \cdot 1 \cdot 1 \cdots 1(c(23)^k)(23d)$, having $j+3$ terms if j is the number of 1's;

- $(a)0 \cdot 0 \cdots 0(b1^j c(23)^k)(23d)$, having $i+3$ terms if i is the number of 0's; and
- $(a0^i b1^j c2)(32)(32)\cdots(32)(3d)$, having $k+2$ terms, if k is the number of (32)'s.

So if all three factorizations have the same number of terms we must have $i = j = k-1$, giving us

$$\{a0^n b1^n c(23)^{n-1} d : n \geq 1\},$$

which is not a CFL. □

With some more work, we may even find examples as in Theorem 14 where L is finite, as we now show.

Theorem 15. *If L is finite, then* $\mathrm{ND}(L)$ *need not be a CFL.*

Proof. For expository purposes, we give an example over the 21-letter alphabet

$$\Sigma = \{0,1,2,3,4,5,6,7,8,a,b,c,d,e,f,g,h,i,j,k,l\}.$$

Define

$$\begin{aligned} L_1 &= \{0ab, cd, ab, cd123, efgh, efgh4, 5ijkl, ijkl, 6, 78\} \\ L_2 &= \{0abc, dabc, d1, 23e, fg, he, h45ij, klij, kl678\} \\ L_3 &= \{0a, bcda, bcd12, 3ef, ghef, gh45i, jk, li, jkl67, 8\}, \end{aligned}$$

and set $L := L_1 \cup L_2 \cup L_3$.

Consider possible factorizations of words of the form

$$0(abcd)^m 123(efgh)^n 45(ijkl)^p 678$$

for some integers $m, n, p \geq 1$. Every factorization of such a word into elements of L must begin with either $0ab$, $0abc$, or $0a$. There are three cases to consider:

Case 1: the first word is $0ab$. Then the next word must begin with c, and there are only two possible choices: cd and $cd123$. If the next word is cd then since no word begins with 1 the only choice is to pick a word starting with a, and there is only one: ab. After picking this, we are back in the same situation, and can only choose between cd followed by ab, or $cd123$. Once $cd123$ is picked we must pick a word that begins with e. However, there are only two: $efgh$ and $efgh4$. If we pick $efgh$ we are left in the same situation. Once we pick $efgh4$ we must pick a word starting with 5, but there is only one: $5ijkl$. After this we can either pick 6 and then 78, or we can pick $ijkl$ a number of times, followed by 678.

This gives the factorization

$$(0ab)((cd)(ab))^{m-1}(cd123)(efgh)^{n-1}(efgh4)(5ijkl)(ijkl)^{p-1}(6)(78)$$

having $1 + 2(m-1) + 1 + (n-1) + 1 + 1 + (p-1) + 1 + 1 = 2m + n + p + 2$ terms.

Case 2: the first word is $0abc$. Then the next word must begin with d, and there are only two choices: $dabc$ and $d1$. If we pick $dabc$ we are back in the same situation. If we pick $d1$ then the next word must begin with 2, but there is only one such word: $23e$. Then the next word must begin with f, but there is only one: fg. Then the next word must begin with h, but there are only two: he and $h45ij$. If we pick he we are back in the same situation. Otherwise we must have a word beginning with k, but there are only two: $klij$ and $kl678$. This gives the factorization

$$(0abc)(dabc)^{m-1}(d1)(23e)((fg)(he))^{n-1}(fg)(h45ij)(klij)^{p-1}(kl678)$$

having $1 + (m-1) + 2 + 2(n-1) + 1 + 1 + (p-1) + 1 = m + 2n + p + 2$ terms.

Case 3: the first word is $0a$. Then only $bcda$ and $bcd12$ start with b, so we must choose $bcda$ over and over until we choose $bcd12$. Only one word starts with 3 so we must choose $3ef$. Now we must choose $ghef$ again and again until we choose $gh45i$. We now choose jk and li alternately until $jkl67$. Finally, we pick 8.

This gives us a factorization

$$(0a)(bcda)^{m-1}(bcd12)(3ef)(ghef)^{n-1}(gh45i)((jk)(li))^{p-1}(jkl67)(8)$$

with $1 + (m-1) + 2 + (n-1) + 1 + 2(p-1) + 2 = m + n + 2p + 2$ terms.

So for all these three factorizations to have the same number of terms, we must have

$$2m + n + p + 2 = m + 2n + p + 2 = m + n + 2p + 2.$$

Eliminating variables we get that $m = n = p$. So when we compute $\mathrm{ND}(L)$ and intersect with the regular language $0(abcd)^+123(efgh)^+45(ijkl)^+678$ we get

$$\{0(abcd)^n123(efgh)^n45(ijkl)^n678 \ : \ n \geq 1\},$$

which is clearly a non-CFL. □

Remark 16. The previous two examples can be recoded over a binary alphabet, by mapping the i'th letter to the string ba^ib.

5. Multiset decipherable (MSD) codes

We now consider multiset decipherable codes, where uniqueness is defined for a word if all factorizations of that word over a given language have the same *multiset* of factors (see Definition 2 for formal details). We again start with a motivating example.

Example 17. Consider $L = \{a^3, a^4\}$. Then

$$\mathrm{MSD}(L) = \{a^3, a^4, a^6, a^7, a^8, a^9, a^{10}, a^{11}, a^{13}, a^{14}, a^{17}\}.$$

We can also find examples of finite languages which are *proper* MSD codes, as the following example from [10] shows.

Example 18. [10] Let $L = \{aab, aba, aabaa, baaababa\}$. Then $\mathrm{MSD}(L) = L^*$; thus L is an MSD code, but L is not a UD code since

$$aab \cdot aabaa \cdot aba \cdot baaababa = aabaa \cdot baaababa \cdot aab \cdot aba.$$

As was the case for numerical decipherability, if L is a finite language, then $\mathrm{MSD}(L)$ need not be context-free.

Theorem 19. *If L is finite then* $\mathrm{MSD}(L)$ *need not be a CFL.*

Proof. Let $\Sigma = \{a, b, c\}$. Define $L = \{A, B, S_1, S_2, T_1, T_2\} \subseteq \Sigma^+$ as follows:

$$A = aa,\ B = aaa,\ S_1 = ab,\ S_2 = ac,\ T_1 = ba,\ T_2 = ca.$$

Let $R = aa(ab)^+(ac)^+aa(ba)^+(ca)^+aaa$, and consider words of the form

$$w = aa(ab)^r(ac)^saa(ba)^t(ca)^qaaa \in L^* \cap R$$

with $r, s, t, q \geq 1$ and the following two factorizations of w:

$$AS_1^rS_2^sAT_1^tT_2^qB = aa \cdot (ab)^r \cdot (ac)^s \cdot aa \cdot (ba)^t \cdot (ca)^q \cdot aaa \quad (1)$$

$$BT_1^rT_2^sS_1^tS_2^qAA = aaa \cdot (ba)^r \cdot (ca)^s \cdot (ab)^t \cdot (ac)^q \cdot aa \cdot aa \quad (2)$$

We now show that w must be of one of these two forms. Since w has prefix $aaab$, it must start with either AS_1 or BT_1. If it starts with $AS_1 = aa \cdot ab$, the next factors must be S_1^{r-1} to match $(ab)^r$, so we have AS_1^r. We then see $(ac)^s$, which can only match with S_2^s. Next, we see $aaba$; thus we must choose $AT_1 = aa \cdot ba$. We then have $(ba)^{t-1}$, which can only match with T_1^{t-1}, and then $(ca)^q$, matching only with T_2^q. Finally, the suffix is aaa which can only match with B as required.

If w starts with $BT_1 = aaa \cdot ba$, the next part is $(ba)^{r-1}$, which only matches with T_1^{r-1}. Then we see $(ca)^s$, so we must use factors T_2^s. We then see $(ab)^t$ and $(ac)^q$, matching with S_1^t and S_2^q respectively. Finally we have $aaaa$ matching only with AA as required.

If $r = t$ and $s = q$, then the number of each factor $(A, B, S_1, S_2, T_1, T_2)$ in factorizations (1) and (2) is identical. Therefore, w always has more than one factorization (of type (1) or (2)); however, that factorization is not unique up to multiset if $r \neq t$ or $s \neq q$. Therefore

$$\begin{aligned} \mathrm{MSD}(L) \cap R &= \{aa(ab)^r(ac)^saa(ba)^t(ca)^qaaa \mid (r = t) \ \wedge \ (s = q)\} \\ &= \{AS_1^rS_2^sAT_1^rT_2^sB \ : \ r, s \geq 1\}, \end{aligned}$$

which is not a context-free language. Thus, since R is regular, we see that $\mathrm{MSD}(L)$ is not context-free. □

6. Set decipherable (SD) codes

We finally consider the notion of *set decipherability*, whereby an element of L^* is set decipherable if all its factorizations use the same *set* of elements (see Definition 4 for formal details). We begin with a motivating example from [5].

Example 20. [5] Let $L = \{x_1, x_2, x_3, x_4\} \subseteq \{a, b\}^*$ where $x_1 = a, x_2 = aba, x_3 = bbabb, x_4 = babbab$. Guzmán [5] proved that $L^* = \mathrm{SD}(L)$; thus L is a SD code, since every word $w \in L^*$ with non-unique factorization must use all four factors of L. On the other hand, we see that

$$\begin{aligned} x_1x_4x_4x_3x_2 &= a \cdot babbab \cdot babbab \cdot bbabb \cdot aba \\ &= aba \cdot bbabb \cdot a \cdot bbabb \cdot babbab \\ &= x_2x_3x_1x_3x_4x_1, \end{aligned}$$

and thus L is not a UD, ND or MSD code.

We now note that unlike for numerical and multiset decipherability, when L is a finite language then the set $\mathrm{SD}(L)$ of words of L^* which are set decipherable is in fact regular.

Theorem 21. *If L is finite then $\mathrm{SD}(L)$ is regular.*

Proof. On input x we nondeterministically attempt to construct two different factorizations into elements of L, recording which elements of L we have seen so far. We accept if we are successful in constructing two different factorizations (which will be different if and only if some element was chosen in one factorization but not the other). This NFA accepts $L^* - \mathrm{SD}(L)$. If L is finite, it follows that $\mathrm{SD}(L)$ is regular.

In more detail, here is the construction. States of our NFA are 6-tuples of the form $[w_1, s_1, v_1, w_2, s_2, v_2]$ where w_1, w_2 are the words of L we are currently trying to match; s_1, s_2 are, respectively, the suffixes of w_1, w_2 we have yet to see, and v_1, v_2 are binary characteristic vectors of length $|L|$, specifying which elements of L have been seen in the factorization so far (including w_1 and w_2, although technically they may not have been seen yet). Letting $C(z)$ denote the vector with all 0's except a 1 in the position corresponding to the word $z \in L$, the initial states are $[w, w, C(w), x, x, C(x)]$ for all words $w, x \in L$. The final states are of the form $[w, \epsilon, v_1, x, \epsilon, v_2]$ where $v_1 \neq v_2$. Transitions on a letter a look like $\delta([w_1, as_1, v_1, w_2, as_2, v_2], a) = [w_1, s_1, v_1, w_2, s_2, v_2]$. In addition there are ϵ-transitions that update the corresponding vectors if s_1 or s_2 equals ϵ, and that "reload" the new w_1 and w_2 we are expecting to see:

$$\begin{aligned} \delta([w_1, \epsilon, v_1, w_2, s_2, v_2], \epsilon) &= \{[w, w, v_1 \vee C(w), w_2, s_2, v_2] \ : \ w \in L\}, \\ \delta([w_1, s_1, v_1, w_2, \epsilon, v_2], \epsilon) &= \{[w_1, s_1, v_1, w, w, v_2 \vee C(w)] \ : \ w \in L\}, \end{aligned}$$

where $\vee$ here denotes the binary 'or' operator applied componentwise to binary vectors. □

A modification to the previous construction also shows that the shortest word failing to have subset-invariant factorization is bounded polynomially:

Proposition 22. *Suppose $|L| = n$ and the length of the longest word of L is m. Then if some word of L^* fails to have subset-invariant factorization, there is a word with this property of length $O(m^2 n)$.*

Proof. We give an NFA $\mathcal{N}$ recognising $L^+ - \mathrm{SD}(L)$. Let $u \in L^+$ be a minimal length word in $L^+ - \mathrm{SD}(L)$. By the definition of subset-invariant factorization, there exists a $w_i \in L$ such that $u = a_1 a_2 \cdots a_{k_1} = b_1 \cdots b_{k_2'} w_i b_{k_2'+1} \cdots b_{k_2}$, where $a_i \in L - \{w_i\}$ and $b_j \in L$ for each $1 \leq i \leq k_1$ and $1 \leq j \leq k_2$. In other words, u has a factorization where w_i is used, and another factorization where it is not used.

We define n superstates S_i of $\mathcal{N}$ for $1 \leq i \leq n$. Superstate S_i tries to find two factorizations of an input word, one of which uses w_i and one of which does not. We keep track of two separate factorizations of the input word, which we denote v_1 and v_2, where $v_1 \in (L - \{w_i\})^*$ and $v_2 \in L^*$. The NFA has ϵ transitions to the start state of each S_i. We now describe the states of S_i.

States in S_i are of the form (v_1, v_2, k), where $k \in \{1, 2\}$, at least one of v_1 or v_2 equals ε, $v_1 \in \mathrm{suff}(w_j)$ for some $w_j \in L - \{w_i\}$ and $v_2 \in \mathrm{suff}(w_{j'})$ for some $w_{j'} \in L$. The start states of S_i are $\{(w_j, \varepsilon, 1) | w_j \in L - \{w_i\}\}$ and $\{(\varepsilon, w_{j'}, 1) | w_{j'} \in L\}$. We define the following transitions in S_i, where $\xrightarrow{u}$ denotes a transition labelled by a *word* u and $k \in \{1, 2\}$:

$$
\begin{aligned}
(v_1, \varepsilon, k) &\xrightarrow{v_1} (\varepsilon, v_2', k) && \text{if } \exists v_2' \in \Sigma^* \text{ where } v_1 v_2' \in L;\\
(v_1, \varepsilon, k) &\xrightarrow{v_1'} (v_1'', \varepsilon, k) && \text{if } \exists v_1', v_1'' \in \Sigma^* \text{ where } v_1 = v_1' v_1'' \text{ and } v_1' \in L;\\
(\varepsilon, v_2, k) &\xrightarrow{v_2} (v_1', \varepsilon, k) && \text{if } \exists v_1' \in \Sigma^* \text{ where } v_2 v_1' \in L - \{w_i\};\\
(\varepsilon, v_2, k) &\xrightarrow{v_2'} (\varepsilon, v_2'', k) && \text{if } \exists v_2', v_2'' \in \Sigma^* \text{ where } v_2 = v_2' v_2'' \text{ and } v_2' \in L - \{w_i\};\\
(v_1, \varepsilon, 1) &\xrightarrow{v_1} (\varepsilon, v_2', 2) && \text{if } \exists v_2' \in \Sigma^* \text{ where } v_1 v_2' = w_i;\\
(v_1, \varepsilon, 1) &\xrightarrow{v_1'} (v_1'', \varepsilon, 2) && \text{if } \exists v_1', v_1'' \in \Sigma^* \text{ where } v_1 = v_1' v_1'' \text{ and } v_1' = w_i.
\end{aligned}
$$

The final state of $\mathcal{N}$ is $(\varepsilon, \varepsilon, 2)$. We also add ε transitions from $(\varepsilon, \varepsilon, 2)$ to $\{(w_j, \varepsilon, 2) | 1 \leq j \leq n \text{ and } j \neq i\}$ and $\{(\varepsilon, w_{j'}, 2) | 1 \leq j' \leq n\}$ in order that, having read a word without subset-invariant factorization, we can recognise the extensions of this word that still belong to L^* and for which the first component continues to omit w_i from its factorization.

In superstate S_i, we thus see that we have $4(m+1)n$ states. If there exists a word $u \in L^+ - \mathrm{SD}(L)$, then there is some $w_i \in L$ for which u has a factorization containing/not containing w_i. After reading in word $b_1 \cdots b_{k_2'}$, we can reach state $(v_1, \varepsilon, 1)$ where $v_1 \in \mathrm{suff}(w_p)$ for some $w_p \in L - \{w_i\}$. This is since $b_1 \cdots b_{k_2'}$ can be factored over L exactly (thus we have ε in the second component) and the first component is the suffix of some element of $L - \{w_i\}$ that is yet to be matched. At this point, when we read v_1, we can transition to $u'' = (\varepsilon, v_2', 2)$ if there exists $v_2' \in \Sigma^*$ where $v_1 v_2' = w_i$. Otherwise, we transition from $(v_1, \varepsilon, 1)$

to $(v_1'', \varepsilon, 2)$ after reading v_1', if there exists $v_1', v_1'' \in \Sigma^*$ where $v_1 = v_1' v_1''$ and $v_1' = w_i$. After this point, we know we have read w_i in exactly one factorization (denoted by the element 2 of the state). Thus, reaching $(\varepsilon, \varepsilon, 2)$ implies the read word has two factorizations, the second of which contains w_i.

If we visit the same state in S_i more than once, then a shorter word $u' \in L^+ - \mathrm{SD}(L)$ exists (which is a contradiction since u is assumed shortest). Since states in S_i are connected by words of length no more than m, and we have $4(m+1)n$ states in S_i, then a minimal solution word is no longer than $O(m^2 n)$. □

To further illustrate the method shown in the previous proof, we introduce the following example.

Example 23. Let $L = \{1, 4, b2, b^2 3, b^3 4, 1b, 2b^2, 3b^3, 4b^4\}$. It is clear that we have the following two factorizations of $u = 1b2b^2 3b^3 4$ over L:

$$1 \cdot b2 \cdot b^2 3 \cdot b^3 4$$
$$1b \cdot 2b^2 \cdot 3b^3 \cdot 4$$

Note that $w_i = 2b^2 \in L$ is a factor appearing in one factorization, but not the other. Then we find the following path in the corresponding NFA (in superstate S_i) defined in Corollary 22:

$$(1, \varepsilon, 1) \xrightarrow{1} (\varepsilon, b, 1) \xrightarrow{b} (2, \varepsilon, 1) \xrightarrow{2} (\varepsilon, b^2, 2) \xrightarrow{b^2} (3, \varepsilon, 2) \xrightarrow{3} (\varepsilon, b^3, 2) \xrightarrow{b^3} (4, \varepsilon, 2) \xrightarrow{4} (\varepsilon, \varepsilon, 2)$$

The next result shows that we can achieve a quadratic lower bound for the shortest word in L^* lacking set-decipherability.

Proposition 24. *There exist examples with $|L| = 2$ and longest word of length m for which the shortest word of L^* failing to have subset-invariant factorization is of length $m(m-1)$.*

Proof. Consider the finite language $L_m = \{a^{m-1}, a^m\} \subseteq \{a\}^*$ for some $m > 1$. The shortest word of L_m^* lacking subset-invariant factorization is $a^{(m-1)m}$, since $a^{(m-1)m} = (a^{m-1})^m = (a^m)^{m-1}$. No shorter subset-invariant factorization of $a^{(m-1)m}$ over L_m can exist since m and $m+1$ are relatively prime. □

Proposition 24 thus defines a class of languages L_m for $m \geq 1$ consisting of two words of length $\Theta(m)$ for which the shortest word lacking subset-invariant factorization has length $\Theta(m^2)$.

In contrast to *unique decipherability*, if L is a regular language, then $\mathrm{SD}(L)$ need not be context-free, as we now show.

Theorem 25. *If L is regular then $\mathrm{SD}(L)$ need not be a CFL.*

Proof. We use a variation of the construction in the proof of Theorem 19. Let $L = (ab)^+(ac)^+aa + (ba)^+(ca)^+ + aa + aaa$. Then (using the notation in the proof of Theorem 19), if

$$w = aa(ab)^r(ac)^s aa(ba)^t(ca)^q aaa \in \mathrm{SD}(L) \cap R$$

with $r, s, t, q \geq 1$ then there are two different factorizations of w:

$$\begin{aligned} w &= aa \cdot (ab)^r (ac)^s aa \cdot (ba)^t (ca)^q \cdot aaa \\ &= aaa \cdot (ba)^r (ca)^s \cdot (ab)^t (ac)^q aa \cdot aa, \end{aligned}$$

which are subset-invariant if and only if $r = t$ and $s = q$. So

$$\mathrm{SD}(L) \ \cap \ R = \{aa(ab)^r(ac)^s aa(ba)^r(ca)^s aaa \ : \ r, s \geq 1\},$$

which is not a CFL, and thus neither is $\mathrm{SD}(L)$. □

7. Complexity results

We have so far studied classifications of $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ and $\mathrm{SD}(L)$ for given language classes L. We now investigate a variety of *upper bounds* for these forms of unique decipherability.

Theorem 26. *Let L be a context-sensitive language. Then the following languages are all (constructively) context-sensitive:* $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ *and* $\mathrm{SD}(L)$.

Proof. Since L is a CSL, there exists a linear bounded automaton (LBA) M_0 that decides it. We construct a nondeterministic LBA M to decide $L^* \cap \overline{\mathrm{XD}(L)}$, where $\mathrm{XD} \in \{\mathrm{UD}, \mathrm{ND}, \mathrm{MSD}, \mathrm{SD}\}$, i.e. the relative complement of XD is decided by M. By the Immerman-Szelepcsényi theorem [8, 14], the class of context-sensitive languages is (constructively) closed under complement, and therefore $\mathrm{XD}(L)$ is also context-sensitive.

Let the input alphabet of M_0 be Σ. The LBA M accepts a word $w \in \Sigma^*$ iff $w \in L^* \cap \overline{\mathrm{XD}(L)}$. Assume that M has multiple work tapes at its disposal, by the standard method of increasing the tape alphabet size by a constant factor. The LBA M works in three stages:

Stage 1. *Verify that $w \in L^*$.* We modify M_0 to create a machine M_0' with a new (possibly nondeterministic) ϵ edge from each final state to each initial state, so M_0' decides L^*. If $w \notin L^*$ we reject. Otherwise, we proceed to Stage 2.

Stage 2. *Guess two factorizations of w.* We use two (initially empty) work tapes τ and ρ to store two guesses at independent factorizations of w. Let $\circ$ be a symbol such that $\circ \notin \Sigma$. The LBA reads each letter of w sequentially and appends it to both τ and ρ. After appending each letter except the final letter of w, the machine may nondeterministically append a $\circ$ symbol to τ and/or to ρ. Thus the word w is written on τ and ρ with $\circ$ spacer symbols.

Stage 3. *Verify unique factorization of w is violated by τ, ρ.* The machine first ensures that the word on τ is not equal to that on ρ (this is trivial, so we give no details). Then, let τ also denote the word written on work tape τ and write $\tau = \tau_1 \circ \tau_2 \circ \cdots \circ \tau_{k_1}$ such that $\tau_i \in \Sigma^*$ for $1 \leq i \leq k_1$. Define $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_{k_2}$ such that $\rho_j \in \Sigma^*$ for $1 \leq j \leq k_2$.

We now independently apply M_0 to each τ_i and ρ_j for $1 \leq i \leq k_1$ and $1 \leq j \leq k_2$ to ensure that each such word belongs to L (otherwise, the LBA rejects). If each $\tau_i \in L$ and $\rho_j \in L$, then ρ and τ store two distinct factorizations of w over L, with factors split by the $\circ$ symbol and thus w is not *uniquely decipherable* so the machine accepts. If in addition $k_1 \neq k_2$, then w is not *numerically decipherable*. To determine if w is *set* or *multiset decipherable*, we can use two additional work tapes to write the set of factors used in ρ and τ, or to keep track of the number of times each factor is used (respectively). □

Example 27. Let $L = \{ab, bab, abb\}$ be the language of Example 13. Given $w = abbabbabbab \in L^* \cap \overline{\mathrm{UD}(L)}$, let us consider how w may be parsed by the LBA M of Theorem 26 at the end of Stage 2:

Input :	$abbabbabbab$
τ :	$ab \circ bab \circ bab \circ bab$
ρ :	$abb \circ abb \circ abb \circ ab$

Since $\tau \neq \rho$, and each factor belongs to L, then $w \notin \mathrm{UD}(L)$. We also note that $w \notin \mathrm{MSD}(L)$ and $w \notin \mathrm{SD}(L)$ is shown by the same factorization.

If L is a context-sensitive language specified by a given LBA, Theorem 26 shows that the membership problem for each of $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$, and $\mathrm{SD}(L)$ lies in PSPACE. In the next proposition, we strengthen this to show that membership in these classes is actually PSPACE-complete.

Proposition 28. *Suppose L is a CSL, given by an LBA M. Then the membership problem for each of the classes $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$ and $\mathrm{SD}(L)$ is* PSPACE-complete.

Proof. Theorem 26 shows that if L is a context-sensitive language, then so are $\{\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L), \mathrm{SD}(L)\}$, and thus membership is in PSPACE.

For the hardness result, note that the membership problem for an arbitrary CSL is PSPACE-complete, but this does not immediately imply that membership in $\mathrm{UD}(L), \mathrm{ND}(L), \mathrm{MSD}(L)$, or $\mathrm{SD}(L)$ is PSPACE-hard to determine.

Let $\#$ be some new symbol such that $\# \notin \Sigma$, where Σ is the alphabet of L. Let $L\#$ denote (by abuse of notation) the concatenation of the languages L and $\{\#\}$. Note that $\mathrm{UD}(L\#) = (L\#)^*$ since $L\#$ is a prefix code (since $\# \notin \Sigma$), and thus $L\#$ is uniquely decipherable. This implies that $L\#$ is also *numerically, multiset* and *set decipherable* and therefore $\mathrm{ND}(L\#) = \mathrm{MSD}(L\#) = \mathrm{SD}(L\#) = (L\#)^*$. Now, a word $w \in L$ if and only if $w\# \in \mathrm{UD}(L\#)$ (and similarly for $\mathrm{ND}(L\#), \mathrm{MSD}(L\#), \mathrm{SD}(L\#)$). □

Theorem 29. *Let L be a CFL given by a CFG G. Then determining if any of $\mathrm{UD}(L)$, $\mathrm{ND}(L)$, $\mathrm{MSD}(L)$ or $\mathrm{SD}(L)$ is empty is undecidable.*

Proof. We will reduce from the Post correspondence problem (PCP). The result follows from a modification of the proof of [7, Theorem 9.20], where it is proven that the ambiguity problem for CFLs is undecidable. Given a context-free

language L, the ambiguity problem is to determine if there exists some word $w \in L$ with more than one leftmost derivation for the CFG that generates L.

Let $\Sigma_k = \{a_1, a_2, \ldots, a_k\}$ and $\Sigma_2 = \{0, 1\}$. Given an instance of PCP $h, g : \Sigma_k^* \to \Sigma_2^*$, we form a CFG, G as follows: the start variable is S, and the rules are

$$\begin{aligned}
&S \to A \mid B \mid \# \\
&A \to h(a_1)Aa_1\# \\
&B \to g(a_1)Ba_1 \\
&A \to h(a_i)Aa_i \mid h(a_i)a_i \\
&B \to g(a_i)Ba_i \mid g(a_i)a_i
\end{aligned}$$

for $2 \leq i \leq k$. Without loss of generality, we may assume that a solution to this instance starts with the element a_1, and this element is used only once (by assuming that it is a 'Claus instance' of PCP [6]).

($\Rightarrow$) We must prove that each element in $L(G)^*$ has a unique factorization if and only if there does not exist a solution to the PCP instance. Assume that there exists a solution to the PCP instance, $w = a_1 w_2 w_3 \cdots w_n \in \Sigma_k^n$ where $w_2, w_3, \ldots, w_n \in \{a_2, \ldots, a_k\}$. Then the following word is derivable in $L(G)^*$ in the following two ways:

$$[h(a_1)h(w_2)\cdots h(w_n)w_n \cdots w_2 a_1 \#] = [g(a_1)g(w_2)\cdots g(w_n)w_n \cdots w_2 a_1][\#],$$

where we use [ ] to denote a factor.

In the first derivation, we use exactly one element of $L(G)$, and in the second derivation we have two elements of $L(G)$. This proves that if there exists a solution to the PCP instance, then some element of $L(G)^*$ has more than one factorization over $L(G)$ (under all four notions of factorization that we consider).

($\Leftarrow$) If no solution to the PCP instance exists, then every element of $L(G)^*$ has a unique factorization into members of $L(G)$, which is easy to see [7]. The modification we made to rules $S \to A \mid B \mid \#$ and $A \to h(a_1)Aa_1\#$ do not change this. □

8. Conclusions

In this paper we studied various notions of factorizations for formal languages. Table 1 summarizes the results regarding the position of each of the various types of factorization in the Chomsky hierarchy with respect to the language L chosen. By CSL-CFL we mean that the resulting language is a CSL, and need not be a CFL.

We are also interested in determining, where possible, good upper and lower bounds for the length of the shortest word that violates particular types of factorization for given language classes. In Table 2 we summarize our results for

L	UD(L)	ND(L)	MSD(L)	SD(L)
Finite	Regular	CSL-CFL	CSL-CFL	Regular
Regular	Regular	CSL-CFL	CSL-CFL	CSL-CFL
CFL	CSL-CFL	CSL-CFL	CSL-CFL	CSL-CFL
CSL	CSL-CFL	CSL-CFL	CSL-CFL	CSL-CFL

Table 1: Comparison of various factorizations

L	UD(L)		SD(L)	
	Lower	Upper	Lower	Upper
Finite	$\Omega(k^2)$	$O(n^2)$	$\Omega(km^2)$	$O(km^2)$
Regular	$\Omega(n^2)$	$O(n^2)$	-	-

Table 2: Bounds of the shortest length words violating UD(L) and SD(L)

upper and lower bounds on the length of the shortest word not satisfying UD(L) or SD(L). In this table, n denotes the number of states in a DFA recognising L, if L is regular, and m denotes the longest word in L and $k = |L|$, if L is a finite set of words. If L is regular, then SD(L) need not even be context-free, as we show in Theorem 25, and therefore we do not specify bounds in this case.

Acknowledgments

The idea of considering numerically decipherable codes was inspired by a talk of Nasir Sohail at the University of Waterloo in April 2014.

[1] P. C. Bell, D. Reidenbach, and J. Shallit. Factorizations in formal languages. In I. Potapov, ed., *Proc. 19th International Conference on Developments in Language Theory, DLT'15*, Lecture Notes in Computer Science, Vol. 9168, Springer, 2015, pp. 97–107.

[2] J. Berstel, D. Perrin, and C. Reutenauer. *Codes and Automata*. Encyclopedia of Mathematics and Its Applications, Vol. 129. Cambridge University Press, 2010.

[3] F. Blanchet-Sadri and C. Morgan. Multiset and set decipherable codes. *Computers and Mathematics with Applications* **41** (2001), 1257–1262.

[4] R. G. Galagher. *Information Theory and Reliable Communications*. Wiley, New York, 1968.

[5] F. Guzmán. Decipherability of codes. *Journal of Pure and Applied Algebra*, **141(51)**, (1999), 13–35.

[6] V. Halava, T. Harju, and M. Hirvensalo. Undecidability bounds for integer matrices using Claus instances. *International Journal of Foundations of Computer Science* **18** (5) (2007), 931–948.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, 2nd edition. Addison-Wesley, 2001.

[8] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing* **17** (1988), 935–938.

[9] H. Jürgensen and S. Konstantinidis. Codes. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, Vol. 1: Word, Language, Grammar, Springer-Verlag, 1991, pp. 511–607.

[10] A. Lempel. On multiset decipherable codes. *IEEE Transactions on Information Theory* **32** (1986), 714–716.

[11] N. Rampersad, J. Shallit, and M.-w. Wang. Inverse star, borders, and palstars. *Information Processing Letters* **111** (2011), 420–422.

[12] A. Restivo. A note on multiset decipherable codes. *IEEE Transactions on Information Theory* **35** (1989), 662–663.

[13] J. Shallit. *A Second Course in Formal Languages and Automata Theory.* Cambridge University Press, 2009.

[14] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bulletin of the EATCS* **33** (1987), 96–100.

[15] A. Weber and T. Head. The finest homophonic partition and related code concepts. In I. Prívara, B. Rovan, and P. Ruzicka, eds., *Proc. 19th International Symposium on Mathematical Foundations of Computer Science, MFCS'94*, Lecture Notes in Computer Science, Vol. 841, Springer, 1994, pp. 618–628.

[16] G.-Q. Zhang. Automata, Boolean matrices, and ultimate periodicity. *Information and Computation* **152** (1999), 138–154.