Factorization in Formal Languages

Paul C. Bell[1], Daniel Reidenbach[1], and Jeffrey Shallit[2]

[1] Department of Computer Science, Loughborough University, Loughborough, Leicestershire, LE11 3TU, United Kingdom
P.Bell@lboro.ac.uk
D.Reidenbach@lboro.ac.uk

[2] School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada
shallit@cs.uwaterloo.ca

Abstract. We consider several language-theoretic aspects of unique factorization in formal languages. We reprove the familiar fact that the set uf(L) of words having unique factorization into elements of L is regular if L is regular, and from this deduce an quadratic upper and lower bound on the length of the shortest word not in uf(L). We observe that uf(L) need not be context-free if L is context-free.

Next, we consider some variations on unique factorization. We define a notion of "semi-unique" factorization, where every factorization has the same number of terms, and show that, if L is regular or even finite, the set of words having such a factorization need not be context-free. Finally, we consider additional variations, such as unique factorization "up to permutation" and "up to subset". Although all these variations have been considered before, it appears that the languages of words having these properties have not been positioned in the Chomsky hierarchy up to now. We also consider the length of the shortest word *not* having the desired property.

1 Introduction

Let L be a formal language. We say $x \in L^*$ has *unique factorization* if whenever

$$x = y_1 y_2 \cdots y_m = z_1 z_2 \cdots z_n$$

for $y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_n \in L$ then $m = n$ and $y_i = z_i$ for $1 \leq i \leq m$. If every element of L^* has unique factorization into elements of L, then L is called a *code*.

Although codes have been studied extensively (see, for example, [8, 1]), in this paper we look at some novel aspects of unique factorization. Namely, we look at some variations of unique factorization, consider the language of words possessing this type 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.

2 Unique factorizations

Given L, we define $\mathrm{uf}(L)$ to be the set of all elements of L^* having unique factorization into elements of L. So L is a code iff $L^* = \mathrm{uf}(L)$. We recall the following familiar fact:

Proposition 1. *If L is regular, then so is* $\mathrm{uf}(L)$.

Proof. If L contains the empty word ϵ then no elements of L^* have unique factorization, and so $\mathrm{uf}(L) = \emptyset$. So, without loss of generality we can assume $\epsilon \not\in L$.

To prove the result, we show that the relative complement $L^* - \mathrm{uf}(L)$ is regular. 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. Transitions on these "doubled" states 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 $\epsilon \not\in L(M)$, we know $q_0 \not\in F$. We create the machine $M' = (Q', \Sigma, \delta', q_0, F')$ as follows:

$$\delta'(q, a) = \begin{cases} \{\delta(q, a)\}, & \text{if } q \not\in F; \\ \{\delta(q_0, a),\ [\delta(q_0, a), \delta(q, a)]\}, & \text{if } q \in F. \end{cases}$$

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

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

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 because of the transition to $[\delta(q_0, a), \delta(q, a)]$. □

Remark 2. There is a shorter and more transparent proof of this result, 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., [6, Theorem 2.2]). Next, using the Boolean matrix interpretation of finite automata (e.g., [15] and [12, §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 any 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.

Corollary 3. *Suppose L is accepted by a DFA with n states. If L is not a 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 1 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 4. *For all $n \geq 2$, there exists an $O(n)$-state DFA accepting a language L that is not a 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 5. *For all $n \geq 2$, there exists an $O(n)$-state DFA accepting a finite language L that is not a 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 6. 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 7. *If L is a context-free language, then* $\mathrm{uf}(L)$ *need not be context-free.*

Proof. Our example is based on two languages (see, for example, [10]):

(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 $\mathrm{uf}(\texttt{PALSTAR}) =$ `PRIMEPALSTAR`, which was proven in [10] to be non-context-free. □

3 Semi-unique factorizations

We now consider a variation on unique factorization. We say that $x \in L^*$ has *semi-unique factorization* if all factorizations of x into elements of L consist of the same number of factors. More precisely, x has semi-unique factorization if whenever

$$x = y_1 y_2 \cdots y_m = z_1 z_2 \cdots z_n$$

for $y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_n \in L$, then $m = n$.

Given a language L, we define $\mathrm{su}(L)$ to be the set of all elements of L^* having semi-unique factorization over L. This concept was previously studied by Weber and Head [14], where a language L was called *numerically decipherable* if $L = \mathrm{su}(L)$, and an efficient algorithm was proposed for testing this property.

Example 8. Let $L = \{a, ab, aab\}$. Then $\mathrm{su}(L) = (ab)^* a^*$.

Theorem 9. *If L is regular, then* $\mathrm{su}(L)$ *is a co-CFL (and hence a context-sensitive language).*

Proof. To see that $\mathrm{su}(L)$ is a co-CFL, mimic the proof of Proposition 1. We use a stack to keep track of the difference between the number of terms in the two guessed factorizations, and another flag in the state to say which, the "top", or the "bottom" state, has more terms (since the stack can't hold negative counters). We accept if we guess two factorizations having different numbers of terms.

It now follows immediately that $\mathrm{su}(L)$ is a context-sensitive language (CSL), by the Immerman-Szelepcsényi theorem [7, 13]. □

Corollary 10. *Given a regular language L, it is decidable if there exist elements $x \in L^*$ lacking semi-unique factorization.*

Proof. Given L, we can construct a pushdown automaton (PDA) accepting $L^* - \mathrm{su}(L)$. We convert this PDA to a context-free grammar G generating the same language (e.g., [6, Theorem 5.4]). Finally, we use well-known techniques (e.g., [6, Theorem 6.6]) to determine whether $L(G)$ is empty. □

Theorem 11. *If L is regular then* $\mathrm{su}(L)$ *need not be a CFL.*

Proof. Let

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

Consider $\mathrm{su}(L)$ 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 parens to indicate factors)

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

$(a)0 \cdot 0 \cdots 0(b1^jc(23)^k)(23d)$, which has $i+3$ terms if i is the number of 0's; and

$(a0^ib1^jc2)(32)(32)\cdots(32)(3d)$, which has $k+2$ 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

$$\{a0^nb1^nc(23)^{n-1}d : n \geq 1\},$$

which is not a CFL. □

There are even examples, as in Theorem 11, where L is finite. For expository purposes, we give an example over the 21-letter alphabet

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

Theorem 12. *If L is finite, then* $\mathrm{su}(L)$ *need not be a CFL.*

Proof. Define

$$\begin{aligned} L_1 &= \{0ab, cd, ab, cd127, efgh, efgh3, 4ijkl, ijkl, 5, 68\} \\ L_2 &= \{0abc, dabc, d1, 27e, fg, he, h34ij, klij, kl568\} \\ L_3 &= \{0a, bcda, bcd12, 7ef, ghef, gh34i, jk, li, jkl56, 8\} \end{aligned}$$

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

Consider possible factorizations of words of the form

$$0(abcd)^m127(efgh)^n34(ijkl)^p568$$

for some integers $m, n, p \geq 1$. Any 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 $cd127$. 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 $cd127$. Once $cd127$ is picked we must pick a word that begins with e. However, there are only two: $efgh$ and $efgh3$. If we pick $efgh$ we are left in the same situation. Once we pick $efgh3$ we must pick a word starting with 4, but there is only one: $4ijkl$. After this we can either pick 5 and then 68, or we can pick $ijkl$ a number of times, followed by 568.

This gives the factorization

$$(0ab)((cd)(ab))^{m-1}(cd127)(efgh)^{n-1}(efgh3)(4ijkl)(ijkl)^{p-1}(5)(68)$$

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: $27e$. 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 $h34ij$. 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 $kl568$. This gives the factorization

$$(0abc)(dabc)^{m-1}(d1)(27e)((fg)(he))^{n-1}(fg)(h34ij)(klij)^{p-1}(kl568)$$

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 7 so we must choose $7ef$. Now we must choose $ghef$ again and again until we choose $gh34i$. We now choose jk and li alternately until $jkl56$. Finally, we pick 8.

This gives us a factorization

$$(0a)(bcda)^{m-1}(bcd12)(7ef)(ghef)^{n-1}(gh34i)((jk)(li))^{p-1}(jkl56)(8)$$

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

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 $\mathrm{su}(L)$ and intersect with the regular language $0(abcd)^+127(efgh)^+34(ijkl)^+568$ we get

$$\{0(abcd)^n 127(efgh)^n 34(ijkl)^n 568 \ : \ n \geq 1\},$$

which is clearly a non-CFL. □

Remark 13. The previous two examples can be recoded over a binary alphabet, by mapping the i'th letter to the string ba^ib.

4 Permutationally unique factorization

In this section we consider yet another variation on unique factorization, which are factorizations that are unique up to permutations of the factors. This concept was introduced by Lempel [9] under the name "multiset decipherable codes". For other work on these codes, see [11, 5, 2].

Formally, given a language L we say $x \in L^*$ has *permutationally unique factorization* if whenever $x = y_1 y_2 \cdots y_m = z_1 z_2 \cdots z_n$ for

$$y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_n \in L,$$

then $m = n$ and there exists a permutation σ of $\{1, \ldots, n\}$ such that $y_i = z_{\sigma(i)}$ 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 $\mathrm{ufp}(L)$ to be the set of $x \in L^*$ having permutationally unique factorization.

Example 14. Consider $L = \{a^3, a^4\}$. Then

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

Theorem 15. *If L is finite then* $\mathrm{ufp}(L)$ *is a co-CFL and hence a CSL.*

Proof. We sketch the construction of a PDA accepting $\overline{\mathrm{ufp}(L)}$. If a word is in L^* but has two permutationally distinct factorizations, then there has to be some factor appearing in the factorizations a different number of times. Our PDA nondeterministically guesses two different factorizations and a factor $t \in L$ that appears a different number of times in the factorizations, then verifies the factorizations and checks the number. It uses the stack to hold the absolute value of the difference between the number of times t appears in the first factorization and the second. It accepts if both factorizations end properly and the stack is nonempty. □

Theorem 16. *If L is finite then* $\mathrm{ufp}(L)$ *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

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

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

$$AS_1^r S_2^s A T_1^t T_2^q B = aa \cdot (ab)^r \cdot (ac)^s \cdot aa \cdot (ba)^t \cdot (ca)^q \cdot aaa \qquad (1)$$
$$BT_1^r T_2^s S_1^t S_2^q AA = aaa \cdot (ba)^r \cdot (ca)^s \cdot (ab)^t \cdot (ac)^q \cdot aa \cdot aa \qquad (2)$$

It is not difficult to see 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 only non-permutationally equivalent if $r \neq t$ or $s \neq q$. Therefore

$$\begin{aligned} \mathrm{ufp}(L) \cap R &= \{aa \cdot (ab)^r \cdot (ac)^s \cdot aa \cdot (ba)^t \cdot (ca)^q \cdot aaa \mid (r = t) \ \wedge \ (s = q)\} \\ &= \{AS_1^r S_2^s AT_1^r T_2^s B \ : \ r, s \geq 1\}, \end{aligned}$$

which is not a context-free language. □

5 Subset-invariant factorization

In this section we consider yet another variation on unique factorization, previously studied under the name "set-decipherable code" by Blanchet-Sadri and Morgan [2].

We say a word $x \in L^*$ has *subset-invariant factorization* (into elements of L) if there exists a subset $S \subseteq L$ with the property that every factorization of x 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, x has subset-invariant factorization if there exists $S = S(x)$ such that whenever $x = y_1 y_2 \cdots y_m$ with $y_1, y_2, \ldots, y_m \in L$, then $S = \{y_1, y_2, \ldots, y_m\}$. We let $\mathrm{ufs}(L)$ denote the set of those $x \in L^*$ having such a factorization.

Theorem 17. *If L is finite then* $\mathrm{ufs}(L)$ *is regular.*

Proof. The proof is similar to the proof of Theorem 15 above. 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 $L^* - \mathrm{ufs}(L)$. So if L is finite, it follows that $\mathrm{ufs}(L)$ 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:

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

□

The preceding proof also shows that the shortest word failing to have subset-invariant factorization is bounded polynomially:

Corollary 18. *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 $\leq 2m^2n^2$.*

Proof. Let $u \in L^+$ be a minimal length word such that $u \in L^+ - \text{ufs}(L)$. Consider the states of the NFA traversed in processing u. Let $S_0 := [w, w, C(w), x, x, C(x)]$ be the initial state and $S_F := [w_F, \epsilon, v_F, x_F, \epsilon, v'_F]$ the final state, where $v_F \neq v'_F$ and $C(w), C(x)$ are defined as in the proof of Theorem 17. By definition, there must exist some $z \in L$ such that v_F and v'_F differ on $C(z)$, i.e., $v_F^T \cdot C(z) + {v'_F}^T \cdot C(z) = 1$.

Initially the characteristic vectors have a single 1, and once an element is set to 1 in a characteristic vector in the NFA, it is never reset to 0. Thus, there exists some $1 \leq k \leq |u|$ such that $u = u_1 \cdots u_{k-1} \cdot u_k \cdot u_{k+1} \cdots u_{|u|}$ where $S_{k-1} = \delta(S_0, u_1 \cdots u_{k-1})$ has a 0 in the characteristic vectors at position z, and $\delta(S_{k-1}, u_k)$ has a 1 in exactly one of the two characteristic vectors at position z. We shall now prove that $|u_1 \cdots u_{k-1}|, |u_{k+1} \cdots u_{|u|}| \leq m^2n^2$, which proves the result.

We prove the result for the word $v = u_1 \cdots u_{k-1}$; a similar analysis holds for $u_{k+1} \cdots u_{|u|}$. Let $S_0, S_1, \ldots S_{k-1}$ be the states of the NFA visited as we process v. We prove that there does not exist $0 \leq i < j \leq k-1$ such that $S_i = [w_1, s_1, v_1, w_2, s_2, v_2]$ and $S_j = [w_1, s_1, v'_1, w_2, s_2, v'_2]$. We proceed by contradiction. Assume that such an i and j exist. Then $u_{i+1} \cdots u_j$ is such that $\delta(S_i, u_{i+1} \cdots u_j) = S_j$. However, $\delta(S_i, u_{j+1} \cdots u_k)$ and $\delta(S_j, u_{j+1} \cdots u_k)$ can only differ in their binary characteristic vectors, since the transition function does not depend upon the characteristic vectors when we update the words w_1, s_1, w_2, s_2. Thus, we can remove the factor $u_{i+1} \cdots u_j$ from u and still reach a final state of the form $S_{F_2} := [w_F, \epsilon, v_{F_2}, x_F, \epsilon, v'_{F_2}]$, for which we still have that $v_{F_2} \neq v'_{F_2}$, since they differ on element z due to letter u_k. Continuing this idea iteratively, the maximal number of states k is bounded by m^2n^2. Doubling this bound gives the result. □

The next result shows that we can achieve a quadratic lower bound.

Proposition 19. *There exist examples with $|L| = 2n$ and longest word of length n for which the shortest word of L^* failing to have subset-invariant factorization is of length $n(n+1)/2$.*

Proof. We just use the example of Proposition 5. □

Theorem 20. *If L is regular then* $\mathrm{ufs}(L)$ *need not be a CFL.*

Proof. We use a variation of the construction in the proof of Theorem 16. Let $L = (ab)^+(ac)^+aa + (ba)^+(ca)^+ + aa + aaa$. Then (using the notation in the proof of Theorem 16), if

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

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

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

which is not a CFL. □

6 Acknowledgments

The idea of considering semi-unique factorization was inspired by a talk of Nasir Sohail at the University of Waterloo in April 2014.

We are very grateful to the referees for pointing out relevant citations to the literature that we did not know about.

References

1. J. Berstel, D. Perrin, and C. Reutenauer. *Codes and Automata*. Encyclopedia of Mathematics and Its Applications, Vol. 129. Cambridge University Press, 2010.
2. F. Blanchet-Sadri and C. Morgan. Multiset and set decipherable codes. *Computers and Mathematics with Applications* **41** (2001), 1257–1262.
3. F. Burderi and A. Restivo. Coding partitions. *Discrete Mathematics and Theoretical Computer Science* **9** (2007), 227–240.
4. T. Head and A. Weber. Deciding code related properties by means of finite transducers. In R. Capocelli, A. De Santis, and U. Vaccaro, eds., *Sequences II: Methods in Communication, Security, and Computer Science*, Springer-Verlag, 1993, pp. 260–272.
5. T. Head and A. Weber. Deciding multiset decipherability. *IEEE Trans. Info. Theory* **41** (1995), 291–297.

6. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
7. N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.* **17** (1988), 935–938.
8. 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.
9. A. Lempel. On multiset decipherable codes. *IEEE Trans. Info. Theory* **32** (1986), 714–716.
10. N. Rampersad, J. Shallit, and M.-w. Wang. Inverse star, borders, and palstars. *Info. Proc. Letters* **111** (2011), 420–422.
11. A. Restivo. A note on multiset decipherable codes. *IEEE Trans. Info. Theory* **35** (1989), 662–663.
12. J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 2009.
13. R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bull. EATCS* **33** (1987), 96–100.
14. 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.
15. G.-Q. Zhang. Automata, Boolean matrices, and ultimate periodicity. *Inf. Comput.* **152** (1999), 138–154.