On the Self Shuffle Language

Pamela Fleischmann\textsuperscript{1}, Tero Harju\textsuperscript{2}, Lukas Haschke\textsuperscript{1}, Jonas Höfer\textsuperscript{1}, and Dirk Nowotka\textsuperscript{1}

\textsuperscript{1} Kiel University, Kiel, Germany, fpa@informatik.uni-kiel.de
\textsuperscript{2} University of Turku, Turku, Finland, harju@utu.fi

Abstract. The shuffle product $u \sqcup v$ of two words $u$ and $v$ is the set of all words which can be obtained by interleaving $u$ and $v$. Motivated by the paper \textit{The Shuffle Product: New Research Directions} by Restivo (2015) we investigate a special case of the shuffle product. In this work we consider the shuffle of a word with itself called the \textit{self shuffle} or \textit{shuffle square}, showing first that the self shuffle language and the shuffle of the language are in general different sets. We prove that the language of all words arising as a self shuffle of some word is context sensitive but not context free. Furthermore, we show that the self shuffle $w \sqcup w$ uniquely determines $w$.

1 Introduction

The most common operation on words is the concatenation, i.e., appending a word $u$ to a word $v$ leads to the word $vu$. With the operation and the empty word as neutral element, the set of all words over a given alphabet is a free monoid. More complicated is the \textit{shuffle (product)} of two words. Given $u$ and $v$, the shuffle product $u \sqcup v$ is a set containing all words $w$ such that deleting $u$ from $w$ results in $v$ and deleting $v$ from $w$ results in $u$. For instance, given $lop$ and $apt$, we have $laptop \in lop \sqcup apt$: deleting the first and the two last letters results in $apt$ while deleting the second to the fourth letter results in $lop$. The definition of the shuffle of two words can be generalised to languages: given two languages $L_1$ and $L_2$, the shuffle $L_1 \sqcup L_2$ is defined as the union of all $u \sqcup v$ for $u \in L_1$ and $v \in L_2$.

The study of the shuffle product on formal languages is, for instance, motivated by its applications in programme verification. As a standard tool for modelling process algebras \cite{1}, the shuffle product applied on regular languages has been studied in detail (see cf. \cite{11, 12}). First, \cite{11} showed that literal morphisms on languages can be modelled by the shuffle operation and 2-testable languages, i.e., no variety but the variety of all regular languages is closed under shuffle. Since then, there have been many results concerning the closure properties of the shuffle operator on formal languages. In \cite{2}, the class of intermixed languages was studied, i.e., the subclass which first contains the singletons and second is closed under Boolean operations, the product, and the shuffle product. Moreover, the authors showed very important results on the expressive power of the shuffle product. A related problem is the shuffle decomposition of formal languages. For
certain subclasses of regular languages the shuffle decomposition is known to be decidable, but undecidable for context free languages [5]. Partially commutative context free languages have been studied in [7]. The subproblem of the perfect shuffle, i.e., the letters have to be taken alternating from each word, has been investigated in [8,10]. Notice that in contrast to the general problem were the shuffle of two words is a set, the perfect shuffle of two words is just a word, e.g., the perfect shuffle of banana and aaa gives banana.

Another subproblem is the self shuffle (also known as shuffle square). Here, only the shuffle of a word with itself, is investigated. Thus, given \( w = \text{abc} \), we get \( w \shuffle w = \{ \text{abcabc, ababc, abacbc, aabcbc, aabbcc} \} \). The word \( w \) is called the \textit{shuffle base} (also known as shuffle root) of \( w \shuffle w \). Raising the definition to languages, one obtains \( L \shuffle w \) as the union of all \( w \shuffle w \) for \( w \in L \subseteq \Sigma^* \). Thus, given a language \( L \), the self shuffle \( L \shuffle \) is a subset of the shuffle product \( L \shuffle L \). The notion of the self shuffle has been investigated lately in several papers (see, e.g., [2,8]). Here, the shuffle of square free words, i.e., words that have no factor of the form \( vv \) for some non empty word \( v \), was studied. The decision problem whether a word is a self shuffle has been independently shown to be \( \text{NP}\)-complete by [4] and [15], although the alphabets were restricted. Recently, [3] showed the result for binary alphabets in general. For a collection of the recent developments and open problems regarding self shuffle, we refer to [14].

\textbf{Own Contribution.} In this work, we investigate the complexity of the self shuffle of languages. First, we show that the investigation of the self shuffle language \( L \shuffle w \) for a given language \( L \) over \( \Sigma \) has a motivation on its own by proving that \( L \shuffle w \) is in general a proper subset of \( L \shuffle L \). Afterwards, we prove that \( \Sigma \shuffle w \) is not context free but context sensitive, i.e., regular and context free languages are not closed under the self shuffle on the one hand and on the other hand the word problem for \( \Sigma \shuffle w \) is in \( \text{MLINSPACE} \). Thus, we are proving that in contrast to the shuffle product, \( L \shuffle w \) does not have to be regular even if \( L \) is a regular language. Finally, we show that given a set \( w \shuffle w \), the shuffle base \( w \) is uniquely determined.

\textbf{Structure of the Work.} In Section 2 we introduce all the necessary definitions for the shuffle and the self shuffle as well as three minor auxiliaries results. In Section 3 we show that \( L \shuffle w \) and \( L \shuffle L \) are equal if and only if \( L \) contains at most one element. Afterwards we place \( \Sigma \shuffle w \) in the Chomsky hierarchy. We finish this section with the well known associated statements about the general shuffle product in order to compare the self shuffle with the shuffle product. In the last section we conclude our results and present some further research directions.

2 Preliminaries

Let \( \mathbb{N} = \{ 1, 2, \ldots \} \) denote the natural numbers and set \( \mathbb{N}_0 = \mathbb{N} \cup \{ 0 \} \) as well as \( [n] = \{ 1, \ldots , n \} \) and \( [n]_0 = [n] \cup \{ 0 \} \) for all \( n \in \mathbb{N} \). For a given \( n \in \mathbb{N} \) set \( S_n = \{ (i_1, \ldots , i_n) \in \mathbb{N}^n | \forall j \in [n-1] : i_j < i_{j+1} \} \). For \( s_1 = (i_1, \ldots , i_n) \in S_n \) and \( s_2 = (j_1, \ldots , j_m) \in S_m \), for some \( n, m \in \mathbb{N} \), we define \( s_1 \cap s_2 \) by \( \{ i_1, \ldots , i_n \} \cap \{ j_1, \ldots , j_m \} \). Given a tuple \( s, \pi_i(s) \) is the projection of \( s \) onto its \( i^{th} \) component.
An alphabet $\Sigma$ is a finite set of symbols, called letters. A word $w$ is a finite sequence of letters from a given alphabet and its length $|w|$ is the number of $w$’s letters. For $i \in [|w|]$ let $w[i]$ denote $w$’s $i$th letter. The set of all finite words over the alphabet $\Sigma$, denoted by $\Sigma^*$, is the free monoid generated by $\Sigma$ with concatenation as operation and the neutral element is the empty word $\varepsilon$, i.e., the word of length 0. Set $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ and $\Sigma^k = \{w \in \Sigma^+ \mid |w| = k\}$ for some $k \in \mathbb{N}_0$. Set $\text{alph}(w) = \{a \in \Sigma \mid \exists i \in [|w|] : w[i] = a\}$ as $w$’s alphabet and for each $a \in \Sigma$ set $|w[a]| = \{|i \in [|w|] \mid w[i] = a\}$. For $u, w \in \Sigma^*$, $u$ is called a factor of $w$, if $w = xuy$ for some words $x, y \in \Sigma^*$. If $x = \varepsilon$ (resp., $y = \varepsilon$) then $u$ is called a prefix (resp., suffix) of $w$. Let $\text{Pref}_i(w)$ denote $w$’s prefix of length $i \in [|w|]_0$.

A factor (resp. prefix, suffix) $u$ of $w$ is called a proper factor (resp. prefix, suffix), if $u \notin \{\varepsilon, w\}$. For $w \in \Sigma^*$ and $s \in S_{|w|}$ define $w[s] = w[i_1] \cdots w[i_{|w|}]$ if $s = (i_1, \ldots, i_{|w|})$. For $1 \leq i \leq j \leq |w|$ define the factor from $w$’s $i$th letter to the $j$th letter by $w[i..j] = w[i] \cdots w[j]$. For $w \in \Sigma^*$ and $n \in \mathbb{N}_0$ define inductively $w^0 = \varepsilon$ and $w^n = w w^{n-1}$. Moreover, set $w^R = w [|w|][|w| - 1] \cdots w[1]$ for $w \in \Sigma^*$.

A function $f : \Sigma^* \to \Sigma^*$ is called morphic if $f(xy) = f(x)f(y)$ holds for all $x, y \in \Sigma^*$. Notice that for a morphic function it suffices to define $f$ on $\Sigma$. Define the deletion of a letter $a \in \Sigma$ from a word by the morphism $\delta_a : \Sigma^* \to \Sigma^*$ with $\delta(a) = \varepsilon$ and $\delta(b) = b$ for all $b \in \Sigma \setminus \{a\}$.

In the following we define the notion of the shuffle and the self shuffle of words. Since one definition is based on scattered factors, we introduce them first.

**Definition 1.** A word $w \in \Sigma^*$ is called a scattered factor (also known as (scattered) substring or subsequence) of $w \in \Sigma^*$ if there exist $s \in S_{|w|} \cap [|w|][|w|]$ with $u[s] = u$. We call $s$ an occurrence of $u$ in $w$. Let $\Omega_u(w)$ be the set of all occurrences of $u$ in $w$ and $\text{ScatFact}(w)$ the set of all of $w$’s scattered factors.

**Definition 2.** Let $w \in \Sigma^*$ and $u, v \in \text{ScatFact}(w)$. We call $o_1 \in \Omega_u(w)$, $o_2 \in \Omega_v(w)$ non overlapping if they are disjoint, i.e., $o_1 \cap o_2 = \emptyset$. Two non overlapping occurrences $o_1 \in \Omega_u(w)$ and $o_2 \in \Omega_v(w)$ are called exhausting if $|w| = |uw| \mid w| \mid w|_{|w|}$ holds; denote that $o_1$ and $o_2$ are non overlapping and exhausting by $o_1 \cup_o o_2$.

The word $\text{pa}$ has five occurrences in $\text{papaya}$, namely $(1, 2)$, $(1, 4)$, $(1, 6)$, $(3, 4)$, and $(3, 6)$. Since the sets of indices of $(1, 2)$ and $(3, 4)$, $(1, 2)$ and $(3, 6)$, and $(1, 4)$ and $(3, 6)$, as well as $(1, 6)$ and $(3, 4)$ are disjoint these occurrences are non overlapping. On the other hand, $\text{ana}$ has the occurrences $(2, 3, 4)$, $(2, 3, 6)$, $(2, 5, 6)$, and $(4, 5, 6)$ in $\text{banana}$ and, thus, there are no non overlapping occurrences of $\text{ana}$ in $\text{banana}$. Moreover, no tuple of occurrences of $\text{pa}$ or $\text{ana}$, respectively, is exhausting for $\text{papaya}$ or $\text{banana}$. But $(1, 3, 5) \in \Omega_{\text{papaya}}(\text{ppy})$ and $(2, 4, 6) \in \Omega_{\text{papaya}}(\text{a}^3)$ are exhausting.

The shuffle (product) (somehow the complementary operation to building scattered factors) was introduced first by Eilenberg and Mac Lane in 1953 [9].

**Definition 3.** The shuffle of $u \in \Sigma^m$ and $v \in \Sigma^n$, for $m, n \in \mathbb{N}_0$ is defined inductively by $u \shuffle \varepsilon = \varepsilon \shuffle u = \{u\}$ and $u[1..m - 1] u[m] \shuffle v[1..n - 1] v[n] = (u[1..m - 1] \shuffle v) u[m] \cup (u \shuffle v[1..n - 1]) v[n]$. 
Since in some proofs it is easier to argue with scattered factors, we present
the following equivalent definition (cf. [13]).

**Definition 4.** For \( u, v \in \Sigma^* \) define the shuffle of \( u \) and \( v \) by \( u \shuffle v = \{ w \in \Sigma^{[u]+[v]} : \exists o_1 \in \Omega_w(u) \exists o_2 \in \Omega_w(v) : o_1 \cup o_2 \} \).

Let \( u = ab \) and \( v = ca \). Thus, the shuffle \( u \shuffle v \) contains words of length four. Each combination of two positions in \( S_2 \) works as an occurrence of \( u \) in the resulting words. The positions of \( v \)'s letters are uniquely determined. Thus, \(|u \shuffle v| \leq \binom{|u|+|v|}{2}\) which is in this case six. Notice that by \( u = ab \) and \( v = ca \) we obtain \( u \shuffle v = \{abca, acab, acba, caba, caab\} \), since \((2,4), (3,4) \in \Omega_w(u)\) lead to the same word.

The definition of the shuffle can be extended to languages \( L_1, L_2 \subseteq \Sigma^* \) which results in an associative, commutative operation with \( \{\varepsilon\} \) as neutral element.

**Definition 5.** For \( L_1, L_2 \subseteq \Sigma^* \) define \( L_1 \shuffle L_2 = \bigcup_{u \in L_1, v \in L_2} u \shuffle v \).

Now, we define the special case of the self shuffle of one word, i.e., we only consider \( u \shuffle u \) for some \( u \in \Sigma^* \).

**Definition 6.** For \( u \in \Sigma^* \) define the self shuffle of \( u \) by \( u \shuffle u \). Given a word \( w \in \Sigma^* \), we call \( u \in \Sigma^* \) with \( w \in u \shuffle u \) a shuffle base of \( w \). We extend this notion to sets: \( u \in \Sigma^* \) is a shuffle base of \( X \subseteq \Sigma^* \) if \( X \subseteq u \shuffle u \).

**Remark 7.** For each \( w \in v \shuffle v \) for some \( v \in \Sigma^* \), we have \(|w|_a \equiv 2 \) for all \( a \in \Sigma \). Moreover, \( w \in v \shuffle u \) implies immediately \( w^R \in v^R \shuffle u^R \) since \( \text{ScatFact}(x^R) = \{y^R \in \Sigma^*: y \in \text{ScatFact}(x)\} \) for all \( w, v, u, x \in \Sigma^* \).

**Remark 8.** Given the self shuffle \( u \shuffle u \), the shuffle base is in general not unique. Consider, for instance, \( aabaabaa \in (aaba \shuffle aaba) \cap (abaab \shuffle abaa) \). Thus, \( abaa \) and \( aaba \) are both shuffle bases of \( a^2 ba^2 ba^2 \).

Again, we raise the definition of the self shuffle of a word to languages, namely the self shuffle language \( L_{\shuffle} \).

**Definition 9.** For \( L \subseteq \Sigma^* \) set \( L_{\shuffle} = \bigcup_{w \in L} w \shuffle w \). If \(|L| = 1\) we simply write \( v_{\shuffle} \) instead of \( \{v\}_{\shuffle} \). We call \( \Sigma^*_{\shuffle} \) the self shuffle language for a given alphabet \( \Sigma \).

We conclude this section with some helpful lemmata which all follow immediately from the definitions. Using the notation of occurrences, we get \( u \shuffle u = \{ w \in \Sigma^{2[u]} : o_1 \in \Omega_w(u) \exists o_2 \in \Omega_w(u) : o_1 \cup o_2 \} \) for a word \( u \in \Sigma^* \), i.e., each word \( w \in u \shuffle u \) has two non overlapping and exhausting occurrences of \( u \). These occurrences can be ordered.

**Lemma 10.** For \( v \in \Sigma^* \) and given \( w \in v_{\shuffle} \) there exist \( o_1, o_2 \in \Omega_w(v) \) with \( o_1 \cup o_2 \) such that \( \pi_k(o_1) < \pi_k(o_2) \) for all \( k \in [|v]| \). We call \( o_1 \) the first and \( o_2 \) the second occurrence of \( v \) in \( w \).
Proof. Since \( w \in v \cup v \) there exist two non overlapping occurrences of \( v \) in \( w \). Assume w.l.o.g. that \( \pi_1(o_1) < \pi_1(o_2) \).

Suppose there exists \( k \in [[v]] \) with \( \pi_k(o_1) > \pi_k(o_2) \). Choose \( k \) minimal. Because \( o_1 \) and \( o_2 \) are occurrences of the same word, their \( i \)th projections refer to the same letter for all \( i \in [[v]] \), in particular \( v[\pi_k(o_1)] = v[\pi_k(o_2)] \). Define \( o_1', o_2' \in \Omega_w(v) \) with \( o_1(i) = o_1'(i) \) for \( i \neq k \) and \( o_1(k) = o_2(k) \) as well as \( o_2(i) = o_2'(i) \) for \( i \neq k \) and \( o_2(k) = o_1(k) \). The claim follows inductively. \( \square \)

The following two lemmata relate the prefixes of the elements of a self shuffle with the prefixes of the corresponding shuffle base.

**Lemma 11.** For \( v \in \Sigma^* \) and \( w \in v_{\cup} \), we have \( \text{alph}(\text{Pref}_i(w)) \subseteq \text{alph}(\text{Pref}_i(v)) \) for all \( i \in [[v]]_0 \).

**Proof.** For all \( i \in [[v]]_0 \) exists some \( j \leq i \) such that \( \text{Pref}_i(w) \subseteq \text{Pref}_{i-j}(v) \) by Definition 4. This implies \( \text{alph}(\text{Pref}_i(w)) = \text{alph}(\text{Pref}_i(v)) \cup \text{alph}(\text{Pref}_{i-j}(v)) \subseteq \text{alph}(\text{Pref}_i(v)) \). \( \square \)

**Lemma 12.** Let \( v \in \Sigma^* \) and \( w \in v_{\cup} \). Then for all \( \ell \in [[w]] \) we have some \( i \in [[v]]_0 \) such that for all \( a \in \Sigma \) we have \( |\text{Pref}_\ell(w)|_a = |\text{Pref}_\ell(v)|_a + |\text{Pref}_{\ell-i}(v)|_a \).

**Proof.** Follows directly from Definition 4. \( \square \)

**Remark 13.** Notice that the results of Lemma 11 and Lemma 12 also hold for suffixes by Remark 4.

### 3 The Self Shuffle Language

In this section we investigate the shuffle operator applied to the same word, i.e., for \( w \in \Sigma^* \) we are interested in the set \( w \cup w \) as well as in \( L_w \) for a given \( L \subseteq \Sigma^* \). Our first results gives an upper bound for the self shuffle of a word, connecting the self shuffle with the Catalan numbers and the Dyck words.

**Lemma 14.** For \( v \in \Sigma^* \) we have \( |v_{\cup}| \leq C_n \) where \( C_n \) denotes the \( n \)th Catalan number. The bound is tight if \( v[i] \neq v[j] \) for all \( i \neq j \).

**Proof.** Let \( v \in \Sigma^n \) for \( n \in \mathbb{N} \). Recall that a word \( w \in \{(, \})^n \) is Dyck word of length \( n \) if \( |w| = |v| \) and \( |\text{Pref}_i(v)| \geq |\text{Pref}_i(v)|_i \). It is folklore that there are \( C_n \) Dyck words of length \( 2n \) where \( C \) denotes the \( \ell \)th Catalan number. For each Dyck word of length \( 2n \), each \( i \)th occurrence of ( and ) by \( v[i] \) each, for all \( i \in [n] \). This is well-defined by the definition of the Dyck words. Let \( D_{2n} \) be the set of Dyck words of length \( 2n \) and \( f : D_{2n} \rightarrow v_{\cup} \) be the mapping defined as described. For each \( w \in v_{\cup} \), we have by Lemma 10 non overlapping and exhausting occurrences \( o_1, o_2 \) of \( v \) in \( w \) with \( o_1 < o_2 \) (component wise). The unique word \( \bar{w} \in \{(, \})^{2n} \) with \( o_1, o_2 \) occurrence of \((, \})^n \) respectively is a Dyck word and the preimage of \( w \) under \( f \). Thus, \( f \) is surjective and we have \( C_n = |D_{2n}| \geq v_{\cup} \). Assume \( v \in \Sigma^* \) now with \( v[i] \neq v[j] \) for all \( i \neq j \). The occurrences \( o_1 < o_2 \) and their associated Dyck words are unique for all \( w \in v_{\cup} \). Therefore, in this case, \( f \) is injective and the bound is tight. \( \square \)
Our first main results shows that for a given language $L$, the sets $L_u$ and $L \cup L$ are only identical iff $|L| \leq 1$ holds. Otherwise, we have $L_u \subset L \cup L$, which motivates the further investigation of $L_u$.

**Theorem 15.** Let $L \subseteq \Sigma^*$, then $L_u = L \cup L$ if and only if $|L| \leq 1$.

**Proof.** Let $L \subseteq \Sigma^*$. The claim obviously holds for $|L| \leq 1$. Thus, assume $|L| \geq 2$. Suppose $L_u = L \cup L$. Suppose there exist $x, y \in L$ with $|x| \neq |y|$ and w.l.o.g. $|x| < |y|$. Choose $x$ of minimal length in $L$ and $y$ of second minimal length in $L$. Then $xy \in L \cup L = L_u$ and thus there exists $v \in L$ with $xy \in v \cup v$. This implies $|v| = \frac{|x| + |y|}{2} < |y|$ which is a contradiction to the choice of $y$. Thus, we have that for all $x, y \in L, |x| = |y|$ holds. Let $n \in \mathbb{N}$ be the length of $L$'s element.

Let $a \in \Sigma$ and $|x|_a = k, |y|_a = \ell$. By $xy \in L_u$, there exists $w_1 \in L$ with $|w_1|_a = \frac{k+\ell}{2}$ which implies $k \equiv \ell$. Moreover, we have $w_1 \neq x, y$ and therefore $xw_1, yw_1 \in L \cup L$. Hence, there exists $w_2 \in L$ with $|w_2|_a = \frac{3k+\ell}{4}$ and therefore $k \equiv 4 \ell$. Inductively, we can build $xw_i$ with $|w_i|_a = \frac{(2^i-1)k+\ell}{2^i}$ for all $i \in \mathbb{N}$ which implies $k \equiv 2^i \ell$ for all $i \in \mathbb{N}$. This implies $k = \ell$. Since this holds for all letters $a \in \Sigma$, we know for all $x, y \in L$ that $|x|_a = |y|_a$ and thus all words in $L$ are permutations of each other, i.e. $|L| \leq n!$. Assume w.l.o.g. $\Sigma = \text{alph}(w)$ for some $w \in L$.

By the assumption, we consider only languages $L$ with words of length $\geq 2$. As induction basis, consider $L$ with $|w| = 2$ for $w \in L$. Thus, $L = \{ab, ba\}$ and we have $L \cup L \neq L_u$.

Define $L_a = \{\delta_a(w) | w \in L\}$ for all $a \in \Sigma$. Let $v \in L_a$. If $|v| = 0$ then the preimage of $v$ in $L$ is unary and we would have $|L| = 1$. If $|v| = 1$, then the preimage of $v$ is binary and we have the above contradiction.

**case 1:** $\exists a \in \Sigma : |L_a| > 1$

Then by induction hypothesis there exists $w' \in (L_a \cup L_a) \setminus (L_a)_u$, i.e., there exist $x', y' \in L_a$ with $w' \in x' \cup y'$ and $x' \neq y'$. Thus, there exist $x, y \in L$ with $\delta_a(x) = x'$ and $\delta_a(y) = y'$. Moreover, there exists $w \in x \cup y$ with $\delta_a(w) = w'$. By $L \cup L = L_u$, there exists $v \in L$ with $w \in v \cup v$, then we get $\delta_a(w) \in \delta_a(v) \cup \delta_a(v)$. Thus, we have $w \in (L \cup L) \setminus L_u$ - a contradiction.

**case 2:** $\forall a \in \Sigma : |L_a| = 1$

If $|\Sigma| > 2$, we get $|L| = 1$ which is a contradiction to the general supposition. Thus, assume $|\Sigma| = 2$. Choose $x, y \in L$ with

\[
x = a^sab^sx_1 \quad \text{and} \quad y = a^tbb^tx_2
\]

such that $x \neq y$ and $r \in \mathbb{N}_0$ maximal for appropriate $s, t \in \mathbb{N}_0$ chosen maximal w.r.t. $r, x_1, x_2 \in \Sigma^*$. Set

\[
w = a^{2r+1}b^{s+t+1}x_1x_2 \in x \cup y.
\]

By $L \cup L = L_u$ there exists $v \in L$ with $w \in v \cup v$.

**case 2.1:** $x \neq v$

Thus, we have $v[1..r+1] = a^{r+1}$. This is a contradiction to the choice of $y$.  

**case 2.2:** $x = v$
By Lemma 16 we have \( x[1..s + t + r + 2] = a^{r+1}b^{s+t+1} \). This is a contradiction to the choice of \( s \).

Thus, we have also a contradiction in case 2 and by this a contradiction to the general supposition.

Before we show that the set of all self shuffles \( \Sigma_{\omega}^* \) is not context free but context sensitive, we prove two auxiliary results. The first one shows that whenever we have a word \( w \in \omega \), starting with an odd number of the same letter followed by a unary factor \( x \), this is a factor of \( v \) (cf. Figure 1). The second one shows that if \( w \in \omega \) with a unary scattered factor \( u \) of odd length then every even power of \( u \) is not a scattered of any element of \( v \).

**Lemma 16.** Let \( v \in \Sigma^* \). If \( a^rb^s \in v \cup v \) for \( a, b \in \Sigma \) different, \( u \in \Sigma^* \), \( r \in \mathbb{N} \) odd, \( s \in \mathbb{N} \) then there exists \( r' \) with \( \frac{r}{2} < r' \leq r \) such that \( a'^rb^s \) is a prefix of \( v \).

**Proof.** Set \( w = a^rb^s \). By \( w \in v \cup v \) there exists \( s_1, s_2 \in \Omega_w(v) \) with \( s_1 \cup s_2 \). Let \( s_1 = (i_1, \ldots, i_{|w|}) \) and \( s_2 = (j_1, \ldots, j_{|w|}) \). By \( w[1..r] = a^r \) there exists \( r_1, r_2 \in \mathbb{N}_0 \) with \( r_1 + r_2 = r \), \( v[1..r_1] = a^{r_1} \), and \( v[1..r_2] = a^{r_2} \), i.e. \( \{i_1, \ldots, i_{r_1}, j_1, \ldots, j_{r_2}\} = \{r\} \). Since \( r \) is odd, either \( r_1 \) or \( r_2 \) is odd, or in particular they are not equal. W.l.o.g. assume \( r_1 < r_2 \). Because \( r = r_1 + r_2 \), we have \( \frac{r}{2} < r_2 \). Since \( a^{|\max(r_1, r_2)|} \) is a prefix of \( v \), we have \( v[r_1 + 1] = a \) and thus \( w[i_{r_1 + 1}] = a \). Since \( w[r_1 \ldots r_2] = b^s \), we get \( v[r_2 + 1 \ldots r_2 + s] = b^s \). This proves the claim. \( \square \)

**Figure 1.** Factorisation into prefixes of \( v \), the longer one marked in black

**Lemma 17.** Let \( w \in v \cup v \) for some \( v \in \Sigma^* \), and let \( u \in \text{ScatFact}(w) \) with \( |u|_{a} \) odd for some \( a \in \Sigma \). Choose \( u_1, \ldots, u_n \in \Sigma^+ \) and \( y_0, \ldots, y_n \in \Sigma^* \) arbitrarily such that \( u = u_1 \cdots u_n \) and \( w = y_0u_1y_1 \cdots u_ny_n \). Then \( y_0u_1y_1 \cdots u_ny_n \notin v \cup v \) for any even \( s \in \mathbb{N}_0 \) and any \( x \in \Sigma^* \).

**Proof.** Since \( w \in v \cup v \), we have \( |w|_{a} = 2 \cdot |v|_{a} \) is even. Furthermore, \( |w|_{a} = r + |y|_{a} \) for \( r := |y_0 \cdots y_n| \in \mathbb{N}_0 \). Since \( |u|_{a} \) is odd and \( |v|_{a} \) is even, \( r \) is also odd. Let \( \tilde{w} = y_0u_1y_1 \cdots u_ny_n \) for \( s \in 2\mathbb{N}_0 \) and consider \( |\tilde{w}|_{a} \). We have

\[
|\tilde{w}|_{a} = |y_0 \cdots y_n|_{a} + s \cdot |u_1 \cdots u_n|_{a} = r + s \cdot |u|_{a} \equiv_{2} 1 + 0 \equiv_{2} 1.
\]

Thus, there does not exist \( x \in \Sigma^* \) with \( \tilde{w} \in v \cup v \).

Based on these two lemmata, we show that the self shuffle language is not context free.

**Theorem 18.** The self shuffle language \( \Sigma_{\omega}^* \) for \( |\Sigma| \geq 2 \) is not context free.
Corollary 19. The regular and context-free languages are not closed under the operation given by \( L \mapsto L_{wi} \).

Proof. Follows directly by Theorem 18. \( \square \)

The following theorem shows that \( \Sigma_{wi}^* \) is a Type-1 language in the Chomsky hierarchy, i.e., it is context sensitive.

Theorem 20. The self shuffle language \( \Sigma_{wi}^* \) is context sensitive.

Proof. Consider the following grammar \( G \).

\[
\begin{align*}
S &\rightarrow \varepsilon \mid \hat{S} \quad (1) \\
\hat{S} &\rightarrow L_a R_a \hat{S} \mid L_a R_a & \text{for } a \in \Sigma \quad (2) \\
R_a L_b &\rightarrow L_b R_a & \text{for } a, b \in \Sigma \quad (3) \\
X_a &\rightarrow a & \text{for } X \in \{L, R\} \text{ and } a \in \Sigma \quad (4)
\end{align*}
\]

Notice, that by replacing Rule 3 with four rules, realising the derivation sequence \( R_a L_b \rightarrow R_a H_b \rightarrow H_a H_b \rightarrow H_a R_b \rightarrow L_a R_b \), the grammar is context sensitive and
equivalent to the given one. Furthermore, it still generates the same language because the only derivation sequence involving the new rules is the given one. For proving \( \Sigma^*_w = L(G) \) observe first that we can identify non-terminals \( X_w \) with terminals \( x \) for all \( x \in \Gamma \) and \( X \in \{L, R\} \) because they are only eliminated by Rule 4. Furthermore, \( L_x \) and \( R_x \) separate all letters into two non-overlapping and exhausting occurrences satisfying Lemma 10.

Consider first \( w \in L(G) \). Rules 1 and 2 generate two non overlapping and exhausting occurrences of the same word. In particular we get the occurrences \( o_1 = (1, 3, \ldots, |w| - 1) \) and \( o_2 = (2, 4, \ldots, |w|) \) since each letter is appended twice (once by \( L_a \) and once by \( R_a \)) consecutively. The resulting word (under application of Rule 3) is therefore in \( \Sigma^*_w \). Rule 3 only swaps letters of different occurrences. Thus, the order of letters in one occurrence does not change and we get \( w \in \Sigma^*_w \).

Consider now \( w \in \Sigma^*_w \) and let \( v \in \Sigma^* \) a shuffle base of \( w \). We have to show that there exists a derivation of \( w \) in \( G \). By Lemma 10 there exist non overlapping and exhausting, first and second occurrences \( o_1, o_2 \in \Omega_w(v) \) of \( v \) in \( w \). Furthermore, we have \( \pi_1(o_1) < \pi_1(o_2) \) for all \( i \in |v| \). Using Rule 4 we map the first occurrences to \( L \) and the second to \( R \) marked non-terminals. Because \( \pi_1(o_1) < \pi_1(o_2) < \pi_i(o_2) \) for all \( i \in \{2, \ldots, |v|\} \), we know that \( w[2\ldots \pi_1(o_2) - 1] \) consists just of \( L \) marked non-terminals. Using Rule 3 we can move \( w[\pi_1(o_2)] \) to the second index of \( w \):

\[
w = L_{o_1}L_{o_2} \cdots L_{o_n}R_{o_2} \cdots \rightarrow_{\pi}^* L_{o_1}L_{o_2}L_{o_3} \cdots L_{o_n} \cdots .
\]

Notice that in \( L_{o_1}L_{o_2} \cdots \) the associated \( R \) occurs after \( R_{o_2} \). Denote this new word by \( \tilde{w} \). This induces new occurrences \( \tilde{o}_1, \tilde{o}_2 \in \Omega_w(v) \) of \( v \) in \( \tilde{w} \) with \( \pi_1(\tilde{o}_1) = 1 \) and \( \pi_2(\tilde{o}_2) = 2 \). Now \( \pi_{2\ldots |v|}(\tilde{o}_1) \) and \( \pi_{2\ldots |v|}(\tilde{o}_2) \) are non overlapping and exhausting occurrences of \( v[2\ldots |v|] \) in \( \tilde{w}[3\ldots |\tilde{w}|] \). These occurrences are still first and second occurrences in the sense of Lemma 10 because for all \( i \in \{2, \ldots, |v|\} \) with \( \pi_1(o_1) < \pi_1(o_2) \) we know that

\[
\pi_1(\tilde{o}_1) = 1 + \pi_1(o_1) \leq \pi_1(o_2) < \pi_i(o_2) = \pi_i(\tilde{o}_2).
\]

For all other \( i \) the occurrences are identical to \( o_1 \) and \( o_2 \) and therefore the invariant still holds. By an inductive argument, we can therefore sort the word into one that can be generated using Rule 1 and Rule 2.

**Corollary 21.** The word problem for \( \Sigma^*_w \) is in NLINSPACE.

**Proof.** Follows directly from Theorem 20 because context sensitive languages are in NLINSPACE. \( \square \)

**Remark 22.** By the same construction of Theorem 20 one can obtain that \( L_1 \cup L_2 \) is context sensitive for given context sensitive languages \( L_1, L_2 \subseteq \Sigma^* \) replace each terminal symbol by a fresh non terminal symbol, insert a new rule \( S \rightarrow S_1S_2 \) for the old starting symbols \( S_1 \) and \( S_2 \) resp., and rules allowing the interleaving of the new non terminal symbols.
Our third result shows that even though the shuffle base of a word may not be unique, each set $w \shuffle w$ for a given $w \in \Sigma^*$ is unique in the sense, that no other word $v \in \Sigma^*$ has the same self shuffle set. Before we can prove that the mapping $w \mapsto w \shuffle w$ is indeed injective, we need again two auxiliary lemmata.

Lemma 23 (*). Let $u \in \Sigma^*$, $a, b \in \Sigma$ and $n \in \mathbb{N}$, then

$$\bigcup_{i=1}^{n}(uba^i \shuffle ub)a^{n-i} = \bigcup_{i=0}^{n}(uba^i \shuffle u)ba^{n-i}.$$ 

Proof. For $n = 1$ the statement holds trivially. Furthermore, we have

$$\bigcup_{i=1}^{n+1}(uba^i \shuffle ub)a^{n+1-i} = \bigcup_{i=1}^{n+1}\left((uba^{i-1} \shuffle ub)a^{(n+1)-i+1} \cup (uba^i \shuffle u)ba^{(n+1)-i}\right)$$

$$= \bigcup_{i=1}^{n+1}(uba^{i-1} \shuffle ub)a^{(n+1)-i+1} \cup \bigcup_{i=1}^{n+1}(uba^i \shuffle u)ba^{(n+1)-i}.$$ 

For the first set in the union we have by induction

$$\bigcup_{i=1}^{n+1}(uba^{i-1} \shuffle ub)a^{(n+1)-i+1} = \left(\bigcup_{i=1}^{n+1}(uba^{i-1} \shuffle ub)a^{(n+1)-i}\right)a$$

$$= \left(\bigcup_{i=0}^{n}(uba^i \shuffle ub)a^{n-i}\right)a$$

$$= (ub \shuffle ub)a^n \cup \bigcup_{i=1}^{n}(uba^i \shuffle ub)a^{n-i}a$$

$$= (ub \shuffle u)ba^n \cup \bigcup_{i=0}^{n}(uba^i \shuffle u)ba^{n-i}a$$

$$= \bigcup_{i=0}^{n}(uba^i \shuffle u)ba^{n+1-i}.$$ 

Thus, we conclude

$$\bigcup_{i=0}^{n}(uba^i \shuffle u)ba^{n+1-i} \cup \bigcup_{i=1}^{n+1}(uba^i \shuffle u)ba^{n+1-i} = \bigcup_{i=0}^{n+1}(uba^i \shuffle u)ba^{n+1-i}. \quad \square$$

Lemma 24 (*). Let $n \in \mathbb{N}$, $u, v \in \Sigma^*$ and $a \in \Sigma$. If

$$\bigcup_{i=1}^{n}(ua^i \shuffle u)a^{n-i} = \bigcup_{i=1}^{n}(va^i \shuffle v)a^{n-i}$$

then there exists some $\ell \in [|u|]$ such that $u = \text{Pref}_\ell(u)a^{[u]|-\ell}$, $u = \text{Pref}_\ell(v)a^{[v]|-\ell}$ and $\text{Pref}_\ell(u)_\omega = \text{Pref}_\ell(v)_\omega$. 

Proof. We always have $|u| = |v|$. If $|u| = |v| = 0$ we have $u = \varepsilon = v$.

Let $b, c \in \Sigma$, $u, v \in \Sigma^*$ and assume the assumption holds for $ub$ and $vc$. Using Lemma 23 for $n = 1$ we have

$$
\bigcup_{i=0}^{n} (uba^i \cup u) ba^{n-i} = \bigcup_{i=1}^{n} (uba^i \cup ub) a^{n-i} = \bigcup_{i=1}^{n} (vca^i \cup vc) a^{n-i} = \bigcup_{i=0}^{n} (vca^i \cup v) ca^{n-i}.
$$

Therefore, $b = c$ holds because the shuffle sets are always non-empty.

Assume $a \neq b$. Therefore, the $n + 1$ suffixes are unique and we obtain the following disjoint union

$$
\bigcup_{i=0}^{n} (uba^i \cup u) ba^{n-i} = \bigcup_{i=0}^{n} (vba^i \cup v) ba^{n-i}.
$$

Using the $i = 0$ equality we obtain $(ub \cup u) ba^n = (vba \cup v) ba^n$ and thus

$$
ub \cup ub = (u \cup u)b = (vb \cup v)b = vb \cup vb.
$$

Assume $a = b$. We obtain

$$
\bigcup_{i=0}^{n} (ua^i \cup u) aa^{n-i} = \bigcup_{i=0}^{n} (ua^{i+1} \cup u) a^{(n+1)-i} = \bigcup_{i=1}^{n+1} (ua^i \cup u) a^{(n+1)-i+1},
$$

and thus, using an analogous claim for $v$,

$$
\bigcup_{i=1}^{n+1} (ua^i \cup u) a^{(n+1)-i+1} = \bigcup_{i=1}^{n+1} (va^i \cup v) a^{(n+1)-i+1}.
$$

The claim follows by injectivity and induction. \qed

Theorem 25. The mapping $w \mapsto w_{||}$ is injective.

Proof. We have $|u| = |v|$. We prove the claim by induction on the length of $u$. If $|u| = |v| = 0$ the claim follows trivially.

Let $a, b \in \Sigma$ and assume $a \cup a = v \cup v$. By the commutativity and the inductive definition of the shuffle we have

$$(ua \cup u)a = ua \cup ua = vb \cup vb = (vb \cup v)b.$$

Therefore, $a = b$ and $a \cup u = v \cup v$. The claim follows by Lemma 24 for $n = 1$ and induction. \qed

Remark 26. Thus, given a finite set $S \subset \Sigma^{2n}$, we know that $S$ is only a self shuffle language if there exists a unique $w \in \Sigma^n$ with $S = w \cup w$. Moreover, we know by $ww \in w \cup w$ that $w$ has to be the first half of one of $S$'s squares.
We finish this section by giving two results stated as well-known in the literature. Since it fits well to illustrate the differences between $L \sqcup L$ and $L_\sqcup$, and the proofs in the literature are often left to the reader, we present them here. Since the self shuffle operation does not map regular languages to regular languages, we weaken the statement to the shuffle of two regular languages. And indeed, in this case the shuffle is again regular. The idea is to build the product automaton and using the first or the second component non deterministically.

**Proposition 27.** Regular languages are closed under shuffle, i.e., given $L_1, L_2 \subseteq \Sigma^*$ regular, $L_1 \sqcup L_2$ is also regular.

**Proof.** Let $L_1, L_2 \subseteq \Sigma^*$ regular which are accepted by DFA $A_1$ and $A_2$ with $A_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ for $i \in [2]$. Let $Q := Q_1 \times Q_2$,

$$\delta : \Sigma \times Q \to 2^Q, (a, (q, q')) \mapsto \{(\delta_1(a, q), q'), (q, \delta_2(a, q'))\},$$

$F = F_1 \times F_2$ and $A := (Q, \Sigma, \delta, (q_1, q_2), F)$. By construction, $L(A) = L_1 \sqcup L_2$ because every accepted word consists of two interleaved words accepted by $A_1$ and $A_2$ respectively. \hfill \Box

**Remark 28.** Given $L \subseteq \Sigma^*$, Proposition 27 shows that $L \sqcup L$ is regular, whereas Corollary 19 proves that $L_\sqcup$ is not necessarily regular. This result cannot be generalised to context free languages. In this class, $L_1 \sqcup L_2$ does not have to be context free even if $L_1$ and $L_2$ are context free. Proposition 29 and Lemma 30 are analogous to the results for intersections of context free languages with context free languages and regular languages respectively.

**Proposition 29.** Context free languages are not closed under shuffle, i.e., given $L_1, L_2 \subseteq \Sigma^*$ context free, $L_1 \sqcup L_2$ is not necessarily context free.

**Proof.** The languages $\{a^nca^n \mid n \in \mathbb{N}\}$ and $\{b^mcb^m \mid m \in \mathbb{N}\}$ are context free. If their shuffle were context free, the language

$$\{a^nca^n \mid n \in \mathbb{N}\} \sqcup \{b^mcb^m \mid m \in \mathbb{N}\} \cap L(a^*b^*c^2a^*b^*)$$

would be context free since context free languages are closed under intersection with regular languages. Since this intersection is the language $\{a^nb^mc^2a^n b^m \mid n, m \in \mathbb{N}\}$ we have a contradiction. \hfill \Box

We can obtain the following weaker result.

**Corollary 30.** Context free languages are closed under shuffle with regular languages.

**Proof.** Analogous to the proof of Proposition 27 since only one automaton uses the stack. \hfill \Box
4 Conclusion

In this work, we investigated the self shuffle of a word and the self shuffle language. Even though $L_{w}$ seems to be easier than $L \shuffle L$, since we only allow to shuffle each word from $L$ with itself, for languages with at least two elements, we have $L_{w} \subset L \shuffle L$. We proved that $L_{w}$ is not necessarily regular whereas $L \shuffle L$ is always regular. Notice that of course $L_{w}$ is regular if $L$ is finite since then $L_{w}$ is also finite. We showed that $\Sigma_{w}$ is context sensitive but not context free. Thus, one cannot gain an advantage by just allowing self shuffles instead of shuffling all words from a given language. On the other hand, we showed that the mapping $w \mapsto w_{w}$ is injective, i.e., each word determines its own self shuffle. Since the shuffle base of a word is not necessarily unique, we believe that further investigations of the shuffle base of words as well as of $w_{w}$ may lead to interesting results. Moreover, since the problem to decide whether a given $w \in \Sigma^{*}$ is in the self shuffle of some word $v$ is known to be $\text{NP}$-complete, it could be worth investigating the problem restricted to $w \in L$ for some given $L \subseteq \Sigma^{*}$.

References

1. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
2. J. Berstel, L. Boasson, O. Carton, J.-E. Pin, and A. Restivo. The expressive power of the shuffle product. *Inf. Comput.*, 208(11):1258–1272, 2010.
3. L. Bulteau and S. Vialette. Recognizing binary shuffle squares is $\text{NP}$-hard. *Theor. Comput. Sci.*, 806:116–132, 2020.
4. S.R. Buss and M. Soltys. Unshuffling a square is $\text{NP}$-hard. *J. Comput. Syst. Sci.*, 80:766–776, 2014.
5. C. Câmpeanu, K. Salomaa, and S. Vágvölgyi. Shuffle decompositions of regular languages. *Int. J. Found. Comput. Sci.*, 13(6):799–816, 2002.
6. J. Currie and K. Saari. Square-free words with square-free self-shuffles. *The Electronic Journal of Combinatorics*, 21, 01 2014.
7. W. Czerwinski and S. Lasota. Partially-commutative context-free languages. In *EXPRESS/SOS 2012, Newcastle upon Tyne, UK, September 3, 2012*, volume 89 of *EPTCS*, pages 35–48, 2012.
8. P. Diaconis, R.L. Graham, and W.M. Kantor. The mathematics of perfect shuffles. *Advances in Applied Mathematics*, 4(2):175–196, 1983.
9. S. Eilenberg and S. Mac Lane. On the groups $\pi_{n}(\Pi, n)$. I. *Annals of Mathematics*, 58(1):55–106, 1953.
10. J.A. Ellis, H. Fan, and J. Shallit. The cycles of the multiway perfect shuffle permutation. *Discret. Math. Theor. Comput. Sci.*, 5:169–180, 2002.
11. Z. Ésik and I. Simon. Modeling literal morphisms by shuffle. *Semigroup Forum*, 56:225–227, 1998.
12. T. Harju and M. Müller. Square-free shuffles of words. *Theor. Comput. Sci.*, 601:29–38, 2015.
13. M. Lothaire. *Combinatorics on words, Second Edition*. Cambridge mathematical library. Cambridge University Press, 1997.
14. A. Restivo. The shuffle product: New research directions. In *Language and Automata Theory and Applications*, pages 70–81, Cham, 2015. Springer International Publishing.
15. R. Rizzi and S. Vialette. On recognizing words that are squares for the shuffle product. In *Computer Science – Theory and Applications*, pages 235–245. Springer Berlin Heidelberg, 2013.