The n-ary Initial Literal and Literal Shuffle

Stefan Hoffmann

Informatikwissenschaften, FB IV, Universität Trier
Universit atsring 15, 54296 Trier, Germany
hoffmanns@informatik.uni-trier.de

Abstract. The literal and the initial literal shuffle have been introduced to model the behavior of two synchronized processes. However, it is not possible to describe the synchronization of multiple processes. Furthermore, both restricted forms of shuffling are not associative. Here, we extend the literal shuffle and the initial literal shuffle to multiple arguments. We also introduce iterated versions, much different from the iterated ones previously introduced for the binary literal and initial literal shuffle. We investigate formal properties, and show that in terms of expressive power, in a full trio, they coincide with the general shuffle. Furthermore, we look at closure properties with respect to the regular, context-free, context-sensitive, recursive and recursively enumerable languages for all operations introduced. Then, we investigate various decision problems motivated by analogous problems for the (ordinary) shuffle operation. Most problems we look at are tractable, but we also identify one intractable decision problem.

Keywords: shuffle, literal shuffle, initial literal shuffle, formal language theory

1 Motivation and Contribution

In [2, 3], the *initial literal shuffle* and the *literal shuffle* were introduced, by giving the following natural motivation (taken from [2, 3]):

> [...] The shuffle operation naturally appears in several problems, like concurrency of processes [13, 21, 25] or multipoint communication, where all stations share a single bus [13]. That is one of the reasons of the large theoretical literature about this operation (see, for instance [1, 9, 13, 14, 15, 20]). In the latter example [of midpoint communication], the general shuffle operation models the asynchronous case, where each transmitter uses asynchronously the single communication channel. If the hypothesis of synchronism is made (step-lock transmission), the situation is modelled by what can be named 'literal' shuffle. Each transmitter emits, in turn, one elementary signal. The same remark holds for concurrency, where general shuffle corresponds to asynchronism and literal shuffle to synchronism. [...]

So, the shuffle operation corresponds to the parallel composition of words, which model instructions or event sequences of processes, i.e., *sequentialized execution histories of concurrent processes.*

In this framework, the initial literal shuffle is motivated by modelling the synchronous operation of two processes that start at the same point in time, whereas the literal shuffle could model the synchronous operation if started at different points in time. However, both restricted shuffle variants are only binary operations, which are not associative. Hence, actually only the case of *two* processes synchronized to each other is modelled, i.e., the (initial) literal shuffling applied multiple times, in any order, does not model adequately the synchronous operation of multiple processes. So, the iterative versions, as introduced in [2, 3], are not adequate to model multiple processes, and, because of the lack of associativity, the bracketing is essential, which is, from a mathematical point of view, somewhat unsatisfying.

Here, we built up on [2, 3] by extending both restricted shuffle variants to multiple arguments, which do not arise as the combination of binary operations. So, technically, for each n we have a different operation taking n arguments. With these operations, we derive iterated variants in a uniform manner. We introduce two variants:

(1) the *n-ary initial literal shuffle*, motivated by modelling n synchronous processes started at the same point in time;
(2) the *n-ary literal shuffle*, motivated by modelling n synchronous processes started at different points in time.

Additionally, in Section 7, we also introduce two additional variants for which the results are independent of the order of the arguments. Hence, our variants might be used when a more precise approach is necessary than the general shuffle can provide.

We study the above mentioned operations and their iterative variants, their relations to each other and their expressive power. We also study their closure properties with respect to the classical families of the Chomsky hierarchy [12] and the recursive languages. We also show that, when adding the full trio operations, the expressive power of each shuffle variant is as powerful as the general shuffle operation. In terms of computational complexity, most problem we consider, which are motivated from related decision problems for the (general) shuffle operation, are tractable. However, we also identify a decision problem for the second variant that is NP-complete.

The goal of the present work is to give an analysis of these operations from the point of view of *formal language theory.*

2 The Shuffle Operation in Formal Language Theory

Beside [2, 3], we briefly review other work related to the shuffle operation. We focus on research in formal language theory and computational complexity.

The shuffle and iterated shuffle have been introduced and studied to understand the semantics of parallel programs. This was undertaken, as it appears to be, independently by Campbell and Habermann [7], by Mazurkiewicz [23] and by Shaw [28]. They introduced *flow expressions*, which allow for sequential operators (catenation and iterated catenation) as well as for parallel operators (shuffle and iterated shuffle). See also [26, 27] for an approach using only the ordinary shuffle, but not the iterated shuffle. Starting from this, various subclasses of the flow expressions were investigated [4, 6, 15, 16, 17, 18, 19, 31].

Beside the literal shuffle [2, 3], and the variants introduced in this work, other variants and generalizations of the shuffle product were introduced. Maybe the most versatile is *shuffle by trajectories* [22], whereby the selection of the letters from two input words is controlled by a given *language of trajectories* that indicates, as a binary language, at which positions letters from the first or second word are allowed. This framework entails numerous existing operations, from concatenation up to the general shuffle, and in [22] the authors related algebraic properties and decision procedures of resulting operations to properties of the trajectory language. In [30], with a similar motivation as ours, different notions of *synchronized shuffles* were introduced. But in this approach, two words have to “link”, or synchronize, at a specified subword drawn from a subalphabet (the letters, or actions, that should be synchronized), which the authors termed the *backbone*. Hence, their approach differs in that the synchronization appears letter-wise, whereas here we synchronize position-wise, i.e., at specific points in time the actions occur together in steps, and are not merged as in [30].

3 Preliminaries and Definitions

By $\mathbb{N}_0$ we denote the *natural numbers* including zero. The *symmetric group*, i.e., the set of all permutations with function composition as operation, is $\mathcal{S}_n = \{f : \{1,\ldots,n\} \to \{1,\ldots,n\} \mid f \text{ bijective}\}$.

By Σ we denote a finite set of symbols, called an *alphabet*. The set Σ^* denotes the set of all finite sequences, i.e., of all words with the concatenation operation. The finite sequence of length zero, or the *empty word*, is denoted by ε. Subsets of Σ^* are called *languages*. For a given word, we denote by $|w|$ its length, and for $a \in \Sigma$ by $|w|_a$ the number of occurrences of the symbol a in w. For a word $w = u_1 \cdots u_n$ with $u_i \in \Sigma$, $i \in \{1,\ldots,n\}$, we write $w^R = u_n \cdots u_1$ for the *mirror operation*. For $L \subseteq \Sigma^*$ we set $L^+ = \bigcup_{i=1}^{\infty} L^i$ and $L^* = L^+ \cup \{\varepsilon\}$, where we set $L^1 = L$ and $L^{i+1} = \{uv \mid u \in L^i, v \in L\}$ for $i \geq 1$.

A *finite deterministic and complete automaton* will be denoted by $\mathcal{A} = (\Sigma, S, \delta, s_0, F)$ with $\delta : S \times \Sigma \to S$ the state transition function, S a finite set of states, $s_0 \in S$ the start state and $F \subseteq S$ the set of final states. The properties of being deterministic and complete are implied by the definition of δ as a total function. The transition function $\delta : S \times \Sigma \to S$ could be extended to a transition function on words $\delta^* : S \times \Sigma^* \to S$ by setting $\delta^*(s, \varepsilon) := s$ and $\delta^*(s, wa) := \delta(\delta^*(s, w), a)$ for $s \in S$, $a \in \Sigma$ and $w \in \Sigma^*$. In the remainder we drop the distinction between both functions and also denote this extension by δ. The *language accepted* by an automaton $\mathcal{A} = (\Sigma, S, \delta, s_0, F)$ is $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(s_0, w) \in F\}$. A language $L \subseteq \Sigma^*$ is called *regular* if $L = L(\mathcal{A})$ for some finite automaton.

Definition 1. *The* shuffle operation*, denoted by* $\sqcup\!\sqcup$*, is defined by*

$$\begin{aligned} u \sqcup\!\sqcup v = \{w \in \Sigma^* \mid\ & w = x_1 y_1 x_2 y_2 \cdots x_n y_n \text{ for some words} \\ & x_1, \ldots, x_n, y_1, \ldots, y_n \in \Sigma^* \text{ such that } u = x_1 x_2 \cdots x_n \text{ and } v = y_1 y_2 \cdots y_n\}, \end{aligned}$$

for $u, v \in \Sigma^*$ *and* $L_1 \sqcup\!\sqcup L_2 := \bigcup_{x \in L_1, y \in L_2} (x \sqcup\!\sqcup y)$ *for* $L_1, L_2 \subseteq \Sigma^*$.

Example 2. $\{ab\} \sqcup\!\sqcup \{cd\} = \{abcd, acbd, acdb, cadb, cdab, cabd\}$

The shuffle operation is commutative, associative and distributive over union. We will use these properties without further mention. In writing formulas without brackets we suppose that the shuffle operation binds stronger than the set operations, and the concatenation operator has the strongest binding. For $L \subseteq \Sigma^*$ the *iterated shuffle* is $L^{\sqcup\!\sqcup,*} = \bigcup_{i=0}^{\infty} L^{\sqcup\!\sqcup,i}$ with $L^{\sqcup\!\sqcup,0} = \{\varepsilon\}$ and $L^{\sqcup\!\sqcup,i+1} = L \sqcup\!\sqcup L^{\sqcup\!\sqcup,i}$. The *positive iterated shuffle* is $L^{\sqcup\!\sqcup,+} = \bigcup_{i=1}^{\infty} L^{\sqcup\!\sqcup,i}$.

A *full trio* [10] is a family of languages closed under homomorphisms, inverse homomorphisms and intersections with regular sets. A full trio is closed under arbitrary intersection if and only if it is closed under shuffle [9]. Also, by a theorem of Nivat [24], a family of languages forms a full trio if and only if it is closed under *generalized sequential machine mappings (gsm mappings)*, also called *finite state transductions*. For the definition of gsm mappings, as well as of *context-sensitive* and *recursively enumerable languages*, we refer to the literature, for example [12]. For two arguments, the *interleaving operation* (or *perfect shuffle* [11]) was introduced in [2, 3]. Here, we give a straightforward generalization for multiple, equal-length, input words.

Definition 3 (n-ary interleaving operation). *Let $n \geq 1$, $k \geq 0$, $u_1, \ldots, u_n \in \Sigma^k$. If $k > 0$, write $u_i = x_1^{(i)} \cdots x_k^{(i)}$, $x_j^{(i)} \in \Sigma$ for $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, n\}$. Then we define $I : (\Sigma^k)^n \to \Sigma^{nk}$ by*

$$I(u_1, \ldots, u_n) = x_1^{(1)} \cdots x_1^{(n)} x_2^{(1)} \cdots x_2^{(n)} \cdot \ldots \cdot x_k^{(1)} \cdots x_k^{(n)}.$$

If $k = 0$, then $I(\varepsilon, \ldots, \varepsilon) = \varepsilon$.

Example 4. $I(aab, bbb, aaa) = abaababba$.

If we interleave all equal-length words in given regular languages, the resulting language is still regular.

Proposition 5. *Let $L_1, \ldots, L_n \subseteq \Sigma^*$ be regular. Then $\{I(u_1, \ldots, u_n) \mid \exists m \geq 0\ \forall i \in \{1, \ldots, n\} : u_i \in L_i \cap \Sigma^m\}$ is regular.*

In [2, 3], the *initial literal shuffle* and the *literal shuffle* were introduced.

Definition 6 ([2, 3]). *Let $U, V \subseteq \Sigma^*$. The* initial literal shuffle *of U and V is*

$$U ⧢_1 V = \{I(u,v)w \mid u, v, w \in \Sigma^*, |u| = |v|, (uw \in U, v \in V) \text{ or } (u \in U, vw \in V)\}.$$

and the literal shuffle *is*

$$\begin{aligned} U ⧢_2 V = \{w_1 I(u,v) w_2 \mid{}& w_1, u, v, w_2 \in \Sigma^*, |u| = |v|, \\ & (w_1 u w_2 \in U, v \in V) \text{ or } (u \in U, w_1 v w_2 \in V) \text{ or} \\ & (w_1 u \in U, v w_2 \in V) \text{ or } (u w_2 \in U, w_1 v \in V)\}. \end{aligned}$$

Example 7. $\{abc\} ⧢_1 \{de\} = \{adbec\}$, $\{abc\} ⧢_2 \{de\} = \{abcde, abdce, adbec, daebc, deabc\}$.

The following iterative variants were introduced in [2, 3].

Definition 8 ([2, 3]). *Let $L \subseteq \Sigma^*$. For $i \in \{1, 2\}$, set*

$$L^{⧢_i^*} = \bigcup_{n \geq 0} L_n, \text{ where } L_0 = \{\varepsilon\} \text{ and } L_{n+1} = L_n ⧢_i L.$$

The next results are stated in [2, 3].

Proposition 9 ([2, 3]). *Let $\mathcal{L}$ be a full trio. The following are equivalent:*

1. *$\mathcal{L}$ is closed under shuffle.*
2. *$\mathcal{L}$ is closed under literal shuffle.*
3. *$\mathcal{L}$ is closed under initial literal shuffle.*

Proposition 10 ([2, 3]). *Let $F \subseteq \Sigma^*$ be finite. Then $F^{⧢_1^*}$ is regular.*

4 The n-ary Initial Literal and Literal Shuffle

Here, for any number of arguments, we introduce both shuffle variants, define iterated versions and state basic properties.

Definition 11. *Let $u_1, \ldots, u_n \in \Sigma^*$ and $N = \max\{|u_i| \mid i \in \{1, \ldots, n\}\}$. Set*

1. $\shuffle_1^n(u_1, \ldots, u_n) = h(I(u_1\$^{N-|u_1|}, \ldots, u_n\$^{N-|u_n|}))$ *and*
2. $\shuffle_2^n(u_1, \ldots, u_n) = \{h(I(v_1, \ldots, v_n)) \mid v_i \in U_i, i \in \{1, \ldots, n\}\}$,

where $U_i = \{\$^k u_i \$^{r-k} \mid 0 \le k \le r$ with $r = n \cdot N - |u_i|\}$ for $i \in \{1, \ldots, n\}$ and $h : (\Sigma \cup \{\$\})^* \to \Sigma^*$ is the homomorphism given by $h(\$) = \varepsilon$ and $h(x) = x$ for $x \in \Sigma$.

Note that writing the number of arguments in the upper index should pose no problem or confusion with the power operator on functions, as the n-ary shuffle variants are only of interest for $n \ge 2$, and raising a function to a power only makes sense for function with a single argument.

For languages $L_1, \ldots, L_n \subseteq \Sigma^*$, we set

$$\shuffle_1^n(L_1, \ldots, L_n) = \bigcup_{u_1 \in L_1, \ldots u_n \in L_n} \{\shuffle_1^n(u_1, \ldots, u_n)\}$$
$$\shuffle_2^n(L_1, \ldots, L_n) = \bigcup_{u_1 \in L_1, \ldots u_n \in L_n} \shuffle_2^n(u_1, \ldots, u_n).$$

Example 12. Let $u = a, v = bb, w = c$. Then $\shuffle_1^3(u, v, w) = abcb$ and

$$\shuffle_2^3(u, v, w) = \{bbac, babc, abcb, acbb, abbc, bbca, bcba, bcab, cbab, cabb, cbba\}$$
$$\shuffle_2^3(v, u, w) = \{bbac, babc, bacb, abcb, abbc, acbb, cbab, cbba, cabb, bcba, bbca\}$$

We see $bacb \notin \shuffle_2^3(u, v, w)$, but $bacb \in \shuffle_2^3(v, u, w)$.

Example 13. Please see Figure 1 for a graphical depiction of the word

$$a_1^{(1)} a_2^{(1)} a_3^{(1)} a_4^{(1)} a_1^{(2)} a_5^{(1)} a_2^{(2)} a_6^{(1)} a_3^{(2)} a_1^{(3)} a_7^{(1)} a_4^{(2)} a_2^{(3)} a_5^{(2)} a_3^{(3)} a_6^{(2)} a_7^{(2)} a_8^{(2)} a_9^{(2)}$$

from $\shuffle_2^3(u, v, w)$ with

$$u = a_1^{(1)} a_2^{(1)} a_3^{(1)} a_4^{(1)} a_5^{(1)} a_6^{(1)} a_7^{(1)},$$
$$v = a_1^{(2)} a_2^{(2)} a_3^{(2)} a_4^{(2)} a_5^{(2)} a_6^{(2)} a_7^{(2)} a_8^{(2)} a_9^{(2)},$$
$$w = a_1^{(3)} a_2^{(3)} a_3^{(3)}.$$

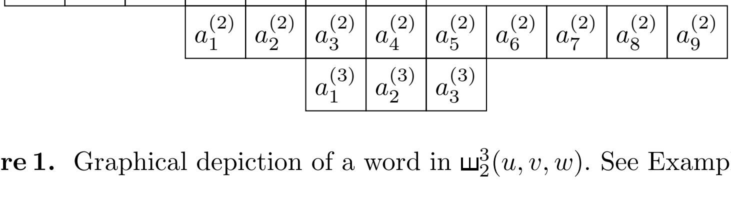
$a_1^{(1)}$ $a_2^{(1)}$ $a_3^{(1)}$ $a_4^{(1)}$ $a_5^{(1)}$ $a_6^{(1)}$ $a_7^{(1)}$
$a_1^{(2)}$ $a_2^{(2)}$ $a_3^{(2)}$ $a_4^{(2)}$ $a_5^{(2)}$ $a_6^{(2)}$ $a_7^{(2)}$ $a_8^{(2)}$ $a_9^{(2)}$
$a_1^{(3)}$ $a_2^{(3)}$ $a_3^{(3)}$

Figure 1. Graphical depiction of a word in $\shuffle_2^3(u, v, w)$. See Example 13.

Now, we can show that the two introduced n-ary literal shuffle variants generalize the initial literal shuffle and the literal shuffle from [2, 3].

Lemma 14. *Let $U, V \subseteq \Sigma^*$ be two languages. Then*

$$\shuffle_1(U,V) = \shuffle_1^2(U,V) \quad \text{and} \quad \shuffle_2\,(U,V) = \shuffle_2^2(U,V).$$

We can also write the second n-ary variant in terms of the first and the mirror operation, as stated in the next lemma.

Lemma 15. *Let $u_1, \ldots, u_n \in \Sigma^*$. Then*

$$\shuffle_2^n(u_1, \ldots, u_n) = \bigcup_{\substack{x_1,\ldots,x_n \in \Sigma^* \\ y_1,\ldots,y_n \in \Sigma^* \\ u_i = x_i y_i}} \{(\shuffle_1^n(x_1^R, \ldots, x_n^R))^R \cdot \shuffle_1^n(y_1, \ldots, y_n)\}$$

With these multiple-argument versions, we define an "iterated" version, where iteration is not meant in the usual sense because of the lack of associativity for the binary argument variants.

Definition 16. *Let $L \subseteq \Sigma^*$ be a language. Then, for $i \in \{1,2\}$, define*

$$L^{\shuffle_i,\circledast} = \{\varepsilon\} \cup \bigcup_{n \geq 1} \shuffle_i^n(L, \ldots, L).$$

For any $i \in \{1,2\}$, as $\shuffle_i^1(L) = L$, we have $L \subseteq L^{\shuffle_i,\circledast}$. Now, let us investigate some properties of the operations under consideration.

Proposition 17. *Let $L_1, \ldots, L_n \subseteq \Sigma^*$ and $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ a permutation. Then*

1. *$L_{\pi(1)} \cdots L_{\pi(n)} \subseteq \shuffle_2^n(L_1, \ldots, L_n)$.*
2. *Let $k \in \mathbb{N}_0$. Then $\shuffle_2^n(L_1, \ldots, L_n) = \shuffle_2^n(L_{((1+k-1) \bmod n)+1}, \ldots, L_{((n+k-1) \bmod n)+1})$.*
3. *$\shuffle_1^n(L_1, \ldots, L_n) \subseteq \shuffle_2^n(L_1, \ldots, L_n) \subseteq L_1 \shuffle \cdots \shuffle L_n$;*
4. *$L_1^* \subseteq L_1^{\shuffle_2,\circledast}$;*
5. *$\Sigma^* = \Sigma^{\shuffle_i,\circledast}$ for $i \in \{1,2\}$;*
6. *$L_1^{\shuffle_1,\circledast} \subseteq L_1^{\shuffle_2,\circledast} \subseteq L_1^{\shuffle,*}$;*
7. *for $u_1, \ldots, u_n, u \in \Sigma^*$, if $u \in \shuffle_i^n(\{u_1\}, \ldots, \{u_n\})$, then $|u| = |u_1| + \cdots + |u_n|$.*

Proof (sketch). We only give a rough outline for Property 2. Let $x_{i,j} \in \Sigma$ for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$. Then, the main idea is to use the equations

$$\begin{aligned}
&h(I(x_{1,1} \cdots x_{1,m}, x_{2,1} \cdots x_{2,m}, \ldots, x_{n,1} \cdots x_{n,m})) \\
&= h((x_{1,1} \cdots x_{n,1})(x_{1,2} \cdots x_{n,2}) \cdots (x_{1,m} \cdots x_{n,m})) \\
&= h(\$^n(x_{1,1} \cdots x_{n,1})(x_{1,2} \cdots x_{n,2}) \cdots (x_{1,m} \cdots x_{n,m})) \\
&= h((\$^{n-1} x_{1,1})(x_{2,1} \cdots x_{n,1} x_{1,2}) \cdots (x_{2,m} \cdots x_{n,m}\$)) \\
&= h(I(\$x_{2,1} \cdots x_{2,m}, \$x_{3,1} \cdots x_{3,m}, \ldots, \$x_{n,1} \cdots x_{n,m}, x_{1,1} \cdots x_{1,m}\$))
\end{aligned}$$

and $\shuffle_2^n(u_1, \ldots, u_n) = h(\{I(v_1, \ldots, v_n) \mid \exists m \forall i \in \{1, \ldots, n\} : v_i \in \$^* u_i \$^* \cap \Sigma^m\})$ with $h : (\Sigma \cup \{\$\})^* \to \Sigma^*$ as in Definition 11. □

Remark 18. By the first two properties, all permutations of concatenations of arguments are in $\shuffle_2^n$, but this shuffle variant itself is only invariant under a cyclic permutation of its arguments. Note that Example 12 shows that it is not invariant under arbitrary permutations of its arguments. For $n = 2$, where it equals the literal shuffle by Lemma 14, as interchanging is the only non-trivial permutation, which is a cyclic one, this product is commutative, as was noted in [2, 3]. But as shown above, this property only extends to cyclic permutation for more than two arguments.

With both iterated variants we can describe languages that are not context-free, as shown by the next proposition. Comparing Proposition 19 with Proposition 10, we see that these operations are more powerful than the iterated initial literal shuffle from [2, 3], in the sense that we can leave the family of regular languages for finite input languages.

Proposition 19. $(abc)^{\shuffle_1,\circledast} = (abc)^{\shuffle_2,\circledast} \cap a^*b^*c^* = \{a^m b^m c^m \mid m \geq 0\}$.

5 Closure Properties

We first show that, when adding the full trio operations, the iterated version of our shuffle variants are as powerful as the shuffle, or as powerful as the iterated variants of the binary versions of the initial literal and literal shuffle introduced in [2, 3] by Proposition 9. But before, and for establishing our closure properties, we state the next result.

Lemma 20. *Let $\mathcal{L}$ be a family of languages. If $\mathcal{L}$ is closed under intersection with regular languages, isomorphic mappings and (general) shuffle, then $\mathcal{L}$ is closed under each shuffle variant $\shuffle_i^n$ for $i \in \{1, 2\}$.*

With the previous lemma, we can derive the next result.

Proposition 21. *Let $\mathcal{L}$ be a full trio. The following properties are equivalent:*

1. *$\mathcal{L}$ is closed under shuffle.*
2. *$\mathcal{L}$ is closed under $\shuffle_i^n$ for some $i \in \{1, 2\}$ and $n \geq 2$.*

In a full trio, we can express the iterated shuffle with our iterated versions of the n-ary shuffle variants.

Proposition 22. *Let $L \subseteq \Sigma^*$ be a language, $\$ \notin \Sigma$, and $h : (\Sigma \cup \$)^* \to \Sigma^*$ the homomorphism defined by: $h(x) = x$ if $x \in \Sigma$, $h(\$) = \varepsilon$. Then, for $i \in \{1, 2\}$,*

$$L^{\shuffle,*} = h(h^{-1}(L)^{\shuffle_i,\circledast}).$$

It is well-known that the regular languages are closed under shuffle [5]. Also, the context-sensitive languages are closed under shuffle [17]. A full trio is closed under intersection if and only if it is closed under shuffle [9]. As the recursively enumerable languages are a full trio that is closed under intersection, this family of languages is also closed under shuffle. As the context-free languages are a full trio that is not closed under intersection [12], it is also not closed under shuffle. The last result is also implied by Proposition 19 and Proposition 21. The recursive languages are only closed under non-erasing homomorphisms, so we could not reason similarly. Nevertheless, this family of languages is closed under shuffle.

Proposition 23. *The family of recursive languages is closed under shuffle.*

We now state the closure properties of the families of regular, context-sensitive, recursive and recursively enumerable languages.

Proposition 24. *The families of regular, context-sensitive, recursive and recursively enumerable languages are closed under* $⧢_i^n$ *for* $i \in \{1, 2\}$*. Furthermore, the families of context-sensitive, recursive and recursively enumerable languages are closed under the iterated versions, i.e., if* L *is context-sensitive, recursive or recursively enumerable, then* $L^{⧢_i,\circledast}$*,* $i \in \{1, 2\}$*, is context-sensitive, recursive or recursively enumerable, respectively.*

Proof (sketch). The closure of all mentioned language families under $⧢_i^n$ with $i \in \{1, 2\}$ is implied by Lemma 20, as they are all closed under intersection with regular languages and shuffle by Proposition 23 and the considerations before this statement. Now, we give a sketch for the iterated variant $L^{⧢_1,\circledast}$.

Let $M = (\Sigma, \Gamma, Q, \delta, s_0, F)$ be a Turing machine for L. The following construction will work for all language classes. More specifically, if given a context-sensitive, recursive or recursively enumerable language L with an appropriate machine M, it could be modified to give a machine that describes a language in the corresponding class, but the basic idea is the same in all three cases. Recall that the context-sensitive languages could be characterized by linear bounded automata [12].

We construct a 3-tape Turing machine, with one input tape, that simulates M and has three working tapes. Intuitively,

1. the input tape stores the input u;
2. on the first working tape, the input is written in a decomposed way, and on certain parts, the machine M is simulated;
3. on the second working tape, for each simulation run of M, a state of M is saved;
4. the last working tape is used to guess and store a number $0 < n \le |u|$.

We sketch the working of the machine. First, it non-deterministically guesses a number $0 < n \le |u|$ and stores it on the last tape. Then, it parses the input u in several passes, each pass takes $0 < k \le n$ symbols from the front of u and puts them in an ordered way on the second working tape, and, non-deterministically, decreases k or does not decrease k. More specifically, on the second working tape, the machine writes a word, with a special separation sign #,

$$\#u_1\#u_2\#\cdots\#u_n\#$$

where $⧢_1^n(u_1, \ldots, u_n)$ equals the input parsed so far. When the input word is completey parsed, it simulates M to check if each word u_i on the second working tape is contained in L. □

Lastly, we can characterize the family of non-empty finite languages using $⧢_1^n$.

Proposition 25. *The family of non-empty finite languages is the smallest family of languages* $\mathcal{L}$ *such that*

1. $\{w\} \in \mathcal{L}$ *for some word* $w \neq \varepsilon$ *with all symbols in* w *distinct, i.e.,* $w = a_1 \cdots a_m$ *with* $a_i \neq a_j$ *for* $1 \le i \neq j \le m$ *and* $a_i \in \Sigma$ *for* $i \in \{1, \ldots, m\}$*,*
2. *closed under union,*
3. *closed under homomorphisms* $h : \Sigma^* \to \Gamma^*$ *such that* $|h(x)| \le 1$ *for* $x \in \Sigma$*,*

4. *closed under $\sqcup\!\sqcup_1^n$ for some $n \geq 2$.*

And, without closure under any homomorphic mappings.

Proposition 26. *The family of non-empty finite languages is the smallest family of languages $\mathcal{L}$ such that (1) $\{\{\varepsilon\}\} \cup \{\{a\} \mid a \in \Sigma\} \subseteq \mathcal{L}$ and which is (2) closed under union and $\sqcup\!\sqcup_1^n$ for some $n \geq 2$.*

6 Computational Complexity

Here, we consider various decision problems for both shuffle variants motivated by similar problems for the ordinary shuffle operation [4, 25, 29, 31]. It could be noted that all problems considered are tractable when considered for $\sqcup\!\sqcup_1$. However, for $\sqcup\!\sqcup_2$, most problems considered are tractable except one that is NP-complete. Hence, the ability to vary the starting positions of different words when interlacing them consecutively, in an alternating fashion, seems to introduce computional hardness. For $\sqcup\!\sqcup_1$, we find the following:

Proposition 27. *Given $L \subseteq \Sigma^*$ represented by a non-deterministic*[1] *automaton and words $w_1, \ldots, w_n \in \Sigma^*$, it is decidable in polynomial time if $\sqcup\!\sqcup_1^n(w_1, \ldots, w_n) \in L$.*

Proposition 28. *Given words $w \in \Sigma^*$ and $v \in \Sigma^*$, it is decidable in polynomial time if $w \in \{v\}^{\sqcup\!\sqcup_1, \circledast}$.*

The non-uniform membership for a languages $L \subseteq \Sigma^*$ is the computational problem to decide for a given word $w \in \Sigma^*$ if $w \in L$. In [25] it was shown that the shuffle of two deterministic context-free languages can yield a language that has an NP-complete non-uniform membership problem. This result was improved in [4] by showing that there even exist linear deterministic context-free languages whose shuffle gives an intractable non-uniform membership problem.

Next, we show that for the initial literal and the literal shuffle, this could not happen if the original languages have a tractable membership problem, which is the case for context-free languages [12].

Proposition 29. *Let $U, V \subseteq \Sigma^*$ be languages whose membership problem is solvable in polynomial time. Then, also the membership problems for $U \sqcup\!\sqcup_1 V$ and $U \sqcup\!\sqcup_2 V$ are solvable in polynomial time.*

Proof. Let w be a given word and write $w = w_1 \cdots w_n$ with $w_i \in \Sigma$ for $i \in \{1, \ldots, n\}$.

Then, to check if $w \in U \sqcup\!\sqcup_2 V$, we try all decompositions $w = xyz$ with $x, y, z \in \Sigma^*$ and $|y|$ even. For $w = xyz$, write $y = y_1 \cdots y_{2n}$ with $y_i \in \Sigma$ and $n \geq 0$. Then test if $xy_1y_3 \cdots y_{2n-1} \in U$ and $y_2 \cdots y_{2n}z \in V$, or $y_1y_3 \cdots y_{2n-1}z \in U$ and $xy_2 \cdots y_{2n} \in V$, or $xy_1y_3 \cdots y_{2n-1}z \in U$ and $y_2 \cdots y_{2n} \in V$, or $y_1y_3 \cdots y_{2n-1} \in U$ and $xy_2 \cdots y_{2n}z \in V$ As $U \sqcup\!\sqcup_2 V = V \sqcup\!\sqcup_2 U$ this is sufficient to find out if $w \in U \sqcup\!\sqcup_2 V$.

For $U \sqcup\!\sqcup_1 V$, first check if $w \in U$ and $\varepsilon \in V$, or if $\varepsilon \in U$ and $w \in V$. If neither of the previous checks give a YES-answer, then try all decompositions $w = yz$ with $y = y_1 \cdots y_{2n}$ for $y_i \in \Sigma$ and $n > 0$. Then, test if $y_1y_3 \cdots y_{2n-1}z \in U$ and $y_2 \cdots y_{2n} \in V$, or if $y_1y_3 \cdots y_{2n-1}z \in V$ and $y_2 \cdots y_{2n} \in U$. If at least one of these tests gives a YES-answer, we have $w \in U \sqcup\!\sqcup_1 V$, otherwise $w \notin U \sqcup\!\sqcup_1 V$.

In all cases, only polynomially many tests were necessary. □

[1] In a non-deterministic automaton the transitions are represented by a relation instead of a function, see [12].

A similar procedure could be given for any fixed number n and $L_1, \ldots, L_n \subseteq \Sigma^*$ to decide the membership problem for $\sqcup\!\sqcup_i^n(L_1, \ldots, L_n)$, $i \in \{1, 2\}$ in polynomial time.

Lastly, the following is an intractable problem for the second shuffle variant.

Proposition 30. *Suppose $|\Sigma| \geq 3$. Given a finite language $L \subseteq \Sigma^*$ represented by a deterministic automaton and words $w_1, \ldots, w_n \in \Sigma^*$, it is* NP*-complete to decide if $\sqcup\!\sqcup_2^n(w_1, \ldots, w_n) \cap L \neq \emptyset$.*

Proof (sketch). We give the basic idea for the hardness proof. Similarly as in [31] for the corresponding problem in case of the ordinary shuffle and a single word $L = \{w\}$ as input, we can use a reduction from 3-Partition. This problem is known to be strongly NP-complete, i.e., it is NP-complete even when the input numbers are encoded in unary [8].

> 3-Partition
> *Input:* A sequence of natural numbers $S = \{n_1, \ldots, n_{3m}\}$ such that $B = (\sum_{i=1}^{3m} n_i)/m \in \mathbb{N}_0$ and for each i, $1 \leq i \leq 3m$, $B/4 < n_i < B/2$.
> *Question:* Can S be partitioned into m disjoint subsequences $S_1, \ldots, S_m$ such that for each k, $1 \leq k \leq m$, S_k has exactly three elements and $\sum_{n \in S_k} n = B$.

Let $S = \{n_1, \ldots, n_{3m}\}$ be an instance of 3-Partition. Set

$$L = \{aaauc \in \{a, b, c\}^* \mid |u|_b = B, |u|_a = 0, |u|_c = 2\}^m.$$

We can construct a deterministic automaton for L in polynomial time. Then, the given instance of 3-Partition has a solution if and only if

$$L \cap \sqcup\!\sqcup_2^n(ab^{n_1}c, ab^{n_2}c, \ldots, ab^{n_{3m}}c) \neq \emptyset. \quad \square$$

Lastly, as the constructions in the proof of Lemma 20 are all effective, and the inclusion problem for regular languages is decidable [12], we can decide if a given regular language is preserved under any of the shuffle variants.

As the inclusion problem is undecidable even for context-free languages [12], we cannot derive an analogous result for the other families of languages in the same way.

Proposition 31. *For every regular language $L \subseteq \Sigma^*$ and $i \in \{1, 2\}$, we can decide whether L is closed under $\sqcup\!\sqcup_i^n$, i.e, if $\sqcup\!\sqcup_i^n(L, \ldots, L) \subseteq L$ holds.*

7 Permuting Arguments

If we permute the arguments of the first shuffle variant $\sqcup\!\sqcup_1^n$ we may get different results. Also, for the second variant, see Proposition 2, only permuting the arguments cyclically does not change the result, but permuting the arguments arbitrarily might change the result, see Example 12.

Here, we introduce two variants of $\sqcup\!\sqcup_1$ that are indifferent to the order of the arguments, i.e., permuting the arguments does not change the result, by considering all possibilities in which the strings could be interlaced. A similar definition is possible for the second variant.

Definition 32 (n-ary symmetric initial literal shuffle). *Let $u_1, \ldots, u_n \in \Sigma^*$ and $x_1, \ldots, x_n \in \Sigma$. Then the function $\sqcup\!\sqcup_3^n : (\Sigma^*)^n \to \mathcal{P}(\Sigma^*)$ is given by*

$$\sqcup\!\sqcup_3^n(u_1, \ldots, u_n) = \bigcup_{\pi \in \mathcal{S}_n} \{\sqcup\!\sqcup_1^n(u_{\pi(1)}, \ldots, u_{\pi(n)})\}.$$

An even stronger form as the previous definition do we get, if we do not care in what order we put the letters at each step.

Definition 33 (n-ary non-ordered initial literal shuffle). *Let $u_1, \ldots, u_n \in \Sigma^*$ and $x_1, \ldots, x_n \in \Sigma$. Then define*

$$\begin{aligned} ⧢_4^n(x_1u_1, \ldots, x_nu_n) &= \bigcup_{\pi \in \mathcal{S}_n} x_{\pi(1)} \cdots x_{\pi(n)} ⧢_4^n (u_1, \ldots, u_n) \\ ⧢_4^n(u_1, \ldots, u_{j-1}, \varepsilon, u_{j+1}, \ldots, u_n) &= ⧢_4^{n-1}(u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n) \\ ⧢_4^1(u_1) &= \{u_1\}. \end{aligned}$$

Similarly as in Definition 16 we can define iterated versions $L^{⧢_3,\circledast}$ and $L^{⧢_4,\circledast}$ for $L \subseteq \Sigma^*$. The following properties follow readily.

Proposition 34. *Let $L_1, \ldots, L_n \subseteq \Sigma^*$, $\pi \in \mathcal{S}_n$ and $i \in \{3, 4\}$. Then*

1. $⧢_3^n(L_1, \ldots, L_n) = ⧢_3^n(L_{\pi(1)}, \ldots, L_{\pi(n)})$;
2. $⧢_4^n(L_1, \ldots, L_n) = ⧢_4^n(L_{\pi(1)}, \ldots, L_{\pi(n)})$;
3. $\{⧢_1^n(L_1, \ldots, L_n)\} \subseteq ⧢_3^n(L_1, \ldots, L_n)\} \subseteq ⧢_4^n(L_1, \ldots, L_n)$;
4. $⧢_i^n(L_1, \ldots, L_n) \subseteq L_1 ⧢ \cdots ⧢ L_n$;
5. $\Sigma^* = \Sigma^{⧢_i,\circledast}$ *for* $i \in \{3, 4\}$;
6. $L_1^{⧢_1,\circledast} \subseteq L_1^{⧢_3,\circledast} \subseteq L_1^{⧢_4,\circledast} \subseteq L_1^{⧢,*}$;
7. *for* $u_1, \ldots, u_n, u \in \Sigma^*$, *if* $u \in ⧢_i^n(\{u_1\}, \ldots, \{u_n\})$, *then* $|u| = |u_1| + \cdots + |u_n|$.

With these properties, we find that for the iteration the first and third shuffle variants give the same language operator.

Lemma 35. *For languages $L \subseteq \Sigma^*$ we have $⧢_1^n(L, \ldots, L) = ⧢_3^n(L, \ldots, L)$.*

For the iterated version, this gives that the first and third variant are equal.

Corollary 36. *Let $L \subseteq \Sigma^*$ be a language. Then $L^{⧢_1,\circledast} = L^{⧢_3,\circledast}$.*

Hence, with Proposition 19, we can deduce that $(abc)^{⧢_3,\circledast} = (abc)^{⧢_4,\circledast} \cap a^*b^*c^* = \{a^mb^mc^m \mid m \geq 0\}$ and so even for finite languages, the iterated shuffles yield languages that are not context-free.

We find that Lemma 20, Proposition 21, Proposition 22 and Proposition 24 also hold for the third and fourth shuffle variant. To summarize:

Proposition 37. *Let $i \in \{3, 4\}$. Then:*

1. *If $\mathcal{L}$ is a family of languages closed under intersection with regular languages, isomorphic mappings and (general) shuffle, then $\mathcal{L}$ is closed under $⧢_i^n$ for $i \in \{3, 4\}$ and each $n \geq 1$.*
2. *If $\mathcal{L}$ is a full trio, then $\mathcal{L}$ is closed under shuffle if and only if it is closed under $⧢_i^n$.*
3. *For regular $L \subseteq \Sigma^*$, it is decidable if $⧢_i^n(L, \ldots, L) \subseteq L$.*
4. *For $L \subseteq \Sigma^*$ and the homomorphism $h : (\Sigma \cup \$)^* \to \Sigma^*$ given by $h(x) = x$ for $x \in \Sigma$ and $h(\$) = \varepsilon$, we have $L^{⧢,*} = h(h^{-1}(L)^{⧢_i,\circledast})$.*
5. *The families of regular, context-sensitive, recursive and recursively enumerable languages are closed under $⧢_i^n$ and the families of context-sensitive, recursive and recursively enumerable languages are closed for $L^{⧢_i,\circledast}$.*

Lastly, we give two examples.

Example 38. Set $L = \{ab, ba\}$. Define the homomorphism $g : \{a, b\}^* \to \{a, b\}^*$ by $g(a) = b$ and $g(b) = a$, i.e., interchanging the symbols.

1. $L^{ш_1,\circledast} = L^{ш_3,\circledast} = \{ug(u) \mid u \in \{a, b\}^*\}$.

   Proof. Let $u_1, \ldots, u_n \in L$ and $w = ш_1^n(u_1, \ldots, u_n)$. Then $w = I(u_1, \ldots, u_n)$. As $|u_1| = \ldots = |u_n| = 2$, we can write $w = x_1 \cdots x_{2n}$ with $x_i \in \{a, b\}$ for $i \in \{1, \ldots, 2n\}$. By Definition 3 of the I-operator, we have, for $i \in \{1, \ldots, n\}$, $u_i = x_i x_{i+n}$. So, if $u_i = ab$, then $x_i = a$ and $x_{i+n} = b$, and if $u_i = ba$, then $x_i = b$ and $x_{i+n} = a$. By Corollary 36, $L^{ш_1,\circledast} = L^{ш_3,\circledast}$.

2. $L^{ш_4,\circledast} = \{uv \mid u, v \in \{a, b\}^*, |u|_a = |v|_b, |u|_b = |v|_a\}$.

   Proof. Let $u_1, \ldots, u_n \in L$ and $w = ш_4^n(u_1, \ldots, u_n)$. Then, using the inductive Definition 33 twice, we find $w = uv$, where u contains all the first symbols of the arguments $u_1, \ldots, u_n$ in some order, and v all the second symbols. Hence, for each a in u, we must have a b in v and vice versa. So, w is contained in the set on the right hand side. Conversely, suppose $w = uv$ with $|u|_a = |v|_b$ and $|u|_b = |v|_a$. Then set $n = |u| = |v|$, $u = x_1 \cdots x_n$, $v = y_1 \cdots y_n$ with $x_i, y_i \in \Sigma$ for $i \in \{1, \ldots, n\}$. We can reorder the letters to match up, i.e., an a with a b and vice versa. More specifically, we find a permutation $\pi \in \mathcal{S}_n$ such that $w = I(x_{\pi(1)}y_1, \ldots, x_{\pi(n)}y_n) \in ш_4^n(x_1y_1, \ldots, x_ny_n) \in L^{ш_4,\circledast}$.

8 Conclusion and Summary

The literal and the initial literal shuffle were introduced with the idea to describe the execution histories of step-wise synchronized processes. However, a closer investigation revealed that they only achieve this for two processes, and, mathematically, the lack of associativity of these operations prevented usage for more than two processes. Also, iterated variants derived from this non-associative binary operations depend on a fixed bracketing and does not reflect the synchronization of n processes. The author of the original papers [2, 3] does not discuss this issue, but is more concerned with the formal properties themselves. Here, we have introduced two operations that lift the binary variant to an arbitrary number of arguments, hence allowing simultaneous step-wise synchronization of an arbitrary number of processes. We have also introduced iterative variants, which are more natural than the previous ones for the non-associative binary operations. In summary, we have

1. investigated the formal properties and relations between our shuffle variant operations,
2. we have found out that some properties are preserved in analogy to the binary case, but others are not, for example commutativity, see Proposition 17;
3. we have shown various closure or non-closure properties for the family of languages from the Chomsky hierarchy and the recursive languages;
4. used one shuffle variant to characterize the family of finite languages;
5. in case of a full trio, we have shown that their expressive power coincides with the general shuffle, and, by results from [2, 3], with the initial literal and literal shuffle;
6. we have investigated various decision problems, some of them are tractable even if an analogous decision problem with the general shuffle operation is intractable. However, we have also identified an intractable decision problem for our second n-ary shuffle variant.

As seen in Proposition 30 for the NP-complete decision problem, we needed an alphabet of size at least three. For an alphabet of size one, i.e., an unary alphabet, the n-ary shuffle for any of the variants considered reduces to the (n-times) concatenation of the arguments, which is easily computable. Then, deciding if the result of this concatenation is contained in a given regular language could be done in polynomial time. So, a natural question is if the problem formulated in Proposition 30 remains NP-complete for binary alphabets only. Also, it is unknown what happens if we alter the problem by not allowing a finite language represented by a deterministic automaton as input, but only a single word, i.e., $|L| = 1$. For the general shuffle, this problem is NP-complete, but it is open what happens if we use the second n-ary variant. Also, it is unknown if the problem remains NP-complete if we represent the input not by an automaton, but by a finite list of words.

Acknowledgement
I thank anonymous reviewers of a previous version for feedback and remarks that helped to improve the presentation. I also thank the reviewers of the current version for careful reading and pointing out typos and some unclear formulations. Due to the strict page limit, I cannot put all proofs into the paper. I have supplied proof sketches where possible. Also, an extended version with all the missing proofs is in preparation.

Bibliography

[1] T. Araki and N. Tokura: *Flow languages equal recursively enumerable languages.* Acta Informatica, 15 1981, pp. 209–217.

[2] B. Bérard: *Formal properties of literal shuffle.* Acta Cyb., 8(1) 1987, pp. 27–39.

[3] B. Bérard: *Literal shuffle.* Theor. Comput. Sci., 51 1987, pp. 281–299.

[4] M. Berglund, H. Björklund, and J. Björklund: *Shuffled languages - representation and recognition.* Theor. Comput. Sci., 489-490 2013, pp. 1–20.

[5] J. A. Brzozowski, G. Jirásková, B. Liu, A. Rajasekaran, and M. Szykuła: *On the state complexity of the shuffle of regular languages*, in Descrip. Compl. of Formal Systems - 18th IFIP WG 1.2 International Conference, DCFS 2016, Bucharest, Romania, July 5-8, 2016. Proceedings, C. Câmpeanu, F. Manea, and J. Shallit, eds., vol. 9777 of Lecture Notes in Computer Science, Springer, 2016, pp. 73–86.

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

[7] R. H. Campbell and A. N. Habermann: *The specification of process synchronization by path expressions*, in Operating Systems OS, E. Gelenbe and C. Kaiser, eds., vol. 16 of LNCS, Springer, 1974, pp. 89–102.

[8] M. R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*, W. H. Freeman, first edition ed., 1979.

[9] S. Ginsburg: *Algebraic and Automata-Theoretic Properties of Formal Languages*, Elsevier Science Inc., USA, 1975.

[10] S. Ginsburg and S. Greibach: *Abstract families of languages*, in 8th Annual Symposium on Switching and Automata Theory (SWAT 1967), 1967, pp. 128–139.

[11] D. Henshall, N. Rampersad, and J. O. Shallit: *Shuffling and unshuffling.* Bull. EATCS, 107 2012, pp. 131–142.

[12] J. E. Hopcroft and J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, 1979.

[13] K. Iwama: *Unique decomposability of shuffled strings: A formal treatment of asynchronous time-multiplexed communication*, in Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, STOC '83, New York, NY, USA, 1983, Association for Computing Machinery, p. 374–381.

[14] K. Iwama: *The universe problem for unrestricted flow languages.* Acta Informatica, 19(1) Apr. 1983, pp. 85–96.

[15] M. Jantzen: *The power of synchronizing operations on strings.* Theor. Comput. Sci., 14 1981, pp. 127–154.

[16] M. Jantzen: *Extending regular expressions with iterated shuffle.* Theor. Comput. Sci., 38 1985, pp. 223–247.

[17] J. Jedrzejowicz: *On the enlargement of the class of regular languages by the shuffle closure.* Inf. Process. Lett., 16(2) 1983, pp. 51–54.

[18] J. Jedrzejowicz and A. Szepietowski: *Shuffle languages are in P.* Theor. Comput. Sci., 250(1-2) 2001, pp. 31–53.

[19] M. Kudlek and N. E. Flick: *Properties of languages with catenation and shuffle.* Fundam. Inform., 129(1-2) 2014, pp. 117–132.

[20] M. Latteux: *Cônes rationnels commutatifs.* J. Comp. Sy. Sc., 18(3) 1979, pp. 307–333.

[21] M. Latteux: *Behaviors of processes and synchronized systems of processes*, in Theoretical Foundations of Programming Methodology, S. G. Broy M., ed., vol. 91 of NATO Advanced Study Institutes Series (Series C — Mathematical and Physical Sciences), Springer, Dordrecht, 1982, pp. 473–551.

[22] A. Mateescu, G. Rozenberg, and A. Salomaa: *Shuffle on trajectories: Syntactic constraints.* Theor. Comput. Sci., 197(1-2) 1998, pp. 1–56.

[23] A. W. Mazurkiewicz: *Parallel recursive program schemes*, in Mathematical Foundations of Computer Science 1975, 4th Symposium, Mariánské Lázne, Czechoslovakia, September 1-5, 1975, Proceedings, J. Becvár, ed., vol. 32 of Lecture Notes in Computer Science, Springer, 1975, pp. 75–87.

[24] M. Nivat: *Transductions des langages de chomsky.* Annales de l'Institut Fourier, 18(1) 1968, pp. 339–455.

[25] W. F. Ogden, W. E. Riddle, and W. C. Round: *Complexity of expressions allowing concurrency*, in Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL '78, New York, NY, USA, 1978, Association for Computing Machinery, p. 185–194.

[26] W. E. Riddle: *An approach to software system behavior description.* Comput. Lang., 4(1) 1979, pp. 29–47.

[27] W. E. Riddle: *An approach to software system modelling and analysis.* Comput. Lang., 4(1) 1979, pp. 49–66.

[28] A. C. Shaw: *Software descriptions with flow expressions.* IEEE Trans. Softw. Eng., 4 1978, pp. 242–254.

[29] L. J. Stockmeyer and A. R. Meyer: *Word problems requiring exponential time (preliminary report)*, in Proceedings of the fifth annual ACM Symposium on Theory of Computing, STOC, ACM, 1973, pp. 1–9.

[30] M. H. ter Beek, C. Martín-Vide, and V. Mitrana: *Synchronized shuffles.* Theor. Comput. Sci., 341(1-3) 2005, pp. 263–275.

[31] M. K. Warmuth and D. Haussler: *On the complexity of iterated shuffle.* J. Comput. Syst. Sci., 28(3) 1984, pp. 345–358.