Fundamenta Informaticae 91 (2009) 437–461
DOI 10.3233/FI-2009-0051
IOS Press

Associativity of Infinite Synchronized Shuffles and Team Automata

Maurice H. ter Beek*
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR
Via G. Moruzzi 1, 56124 Pisa, Italy
E-mail: maurice.terbeek@isti.cnr.it

Jetty Kleijn†
Leiden Institute of Advanced Computer Science, Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden, The Netherlands
E-mail: kleijn@liacs.nl

Abstract. Motivated by different ways to obtain team automata from synchronizing component automata, we consider various definitions of synchronized shuffles of words. A shuffle of two words is an interleaving of their symbol occurrences which preserves the original order of these occurrences within each of the two words. In a synchronized shuffle, however, also two occurrences of one symbol, each from a different word, may be identified as a single occurrence. In case at least one of the words involved is infinite, a (synchronized) shuffle can also be unfair in the sense that an infinite word may prevail from some point onwards even when the other word still has occurrences to contribute to the shuffle. We prove that for the synchronized shuffle operations under consideration, every (fair or unfair) synchronized shuffle can be obtained as a limit of synchronized shuffles of the finite prefixes of the words involved. In addition, it is shown that with the exception of one, all synchronized shuffle operations that we consider satisfy a natural notion of associativity, also in case of unfairness. Finally, using these results, some compositionality results for team automata are established.

Keywords: Team automata, synchronized shuffling, infinite words, fairness, associativity, compositionality

*Address for correspondence: Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

†Supported by the CNR through a Short-Term Mobility grant.

1. Introduction

Team automata were originally introduced in [17] to model components of groupware systems and their interconnections. They form an automata-based framework for the study of distributed systems consisting of components collaborating through interactions (see, *e.g.*, [3, 4, 5]). A team automaton is a conglomerate of component automata which interact through synchronizations, *i.e.* simultaneous executions of common actions. Composing a team automaton means defining the team's transitions by choosing the actions and synchronizations (combinations of component transitions labeled with the same action) that can take place from the combined states of the components. Each team automaton is thus a product automaton defined over component automata. However, a given fixed collection of component automata does not define a single unique team automaton, but rather an entire range of team automata, one for each choice of the team's transitions (individual or synchronizing transitions from the component automata). This is in contrast with the usual synchronous product construction that defines a unique composite automaton, the transitions of which are exactly those combinations of component transitions that represent a synchronization on a common action by all components sharing that action (see, *e.g.*, [13, 15, 16, 23, 26, 29, 34]). The flexibility of the team automata setup allows one to model different protocols for the collaboration of components in a distributed system.

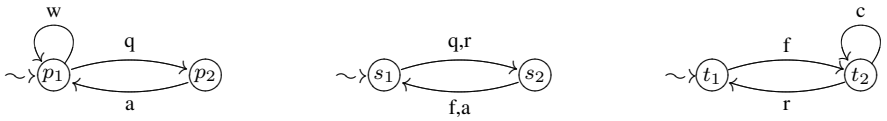


Figure 1. Three component automata: $\mathcal{C}$lient (left), $\mathcal{P}$roxy (middle), and $\mathcal{S}$erver (right).

To illustrate this, we consider various team automata composed of instantiations of the three components Client, Proxy, and Server represented by the automata $\mathcal{C}$, $\mathcal{P}$, and $\mathcal{S}$, respectively, given in Figure 1. A Client is occupied with its **w**ork, but occasionally may need input from outside. In that case it sends a **q**uery. Once it has done so, a Client can proceed only after it has received an **a**nswer. A Server, after receiving a **f**orwarded query, **c**omputes an answer and outputs the **r**esult, after which it is ready for new (forwarded) questions. The Proxy is meant to be a buffer (here with capacity one) between Client(s) and Server(s), modeled with two states. Initially it is ready to receive input (a **q**uery or a **r**esult) and move to its other state, from which it can output a **f**orwarded query or **a**nswer and return to its initial state.

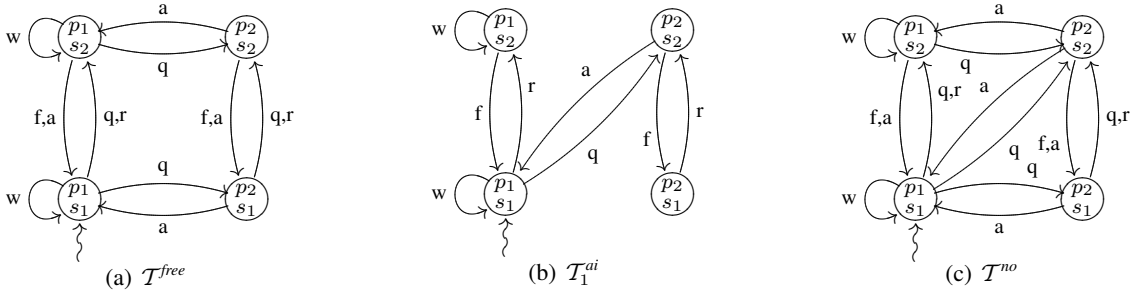
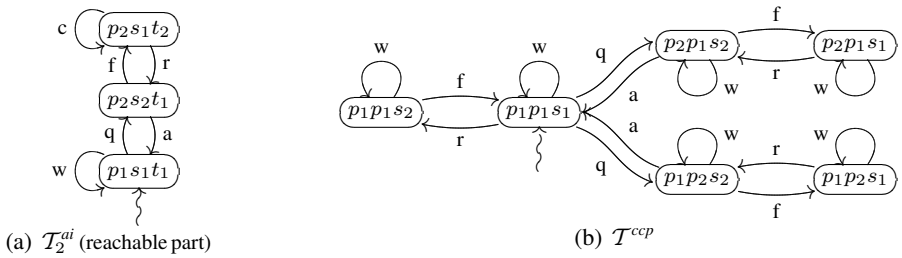


Figure 2. Three team automata: $\mathcal{T}^{free}$, $\mathcal{T}_1^{ai}$ and $\mathcal{T}^{no}$ over $\{\mathcal{C}, \mathcal{P}\}$.

The component automata $\mathcal{C}$, $\mathcal{P}$, and $\mathcal{S}$ can be composed in various ways to yield different team automata. One possibility is to let them exist independently of one another (without any synchronizations or restrictions on the execution of their actions). For the two automata $\mathcal{C}$ and $\mathcal{P}$, this leads to the team automaton $\mathcal{T}^{free}$ depicted in Figure 2(a). From its initial state (p_1, s_1), written as $\frac{p_1}{s_1}$ in the figure, one may now see the sending of a query by the Client followed by the Proxy's reception of a query—possibly even with other actions taking place in between. Hence communication in $\mathcal{T}^{free}$ is asynchronous. Synchronous communication between a Client and the Proxy can be modeled using a composition method which requires the components to synchronize on the relevant common actions, in this case **q** and **a**. The possible occurrences of non-shared actions are not affected and these actions can be executed as before, independently from the other components. For the components Client and Proxy, this leads to the team automaton $\mathcal{T}_1^{ai}$ over $\mathcal{C}$ and $\mathcal{P}$ depicted in Figure 2(b). From the initial state (p_1, s_1), a single transition with label **q** represents the simultaneous sending and receiving of a query by the Client and the Proxy. For three components (Client, Proxy, and Server) this composition method yields the (synchronous product) team automaton $\mathcal{T}_2^{ai}$ over $\mathcal{C}$, $\mathcal{P}$, and $\mathcal{S}$, the reachable part of which is depicted in Figure 3(a). (Note that due to its very simple structure, the Proxy does not necessarily communicate in an alternating fashion with the Client and the Server.) Team automaton $\mathcal{T}^{no}$, depicted in Figure 2(c), combines the two automata $\mathcal{C}$ and $\mathcal{P}$ without imposing any restrictions whatsoever. That is, every action may occur as before in the components to which it belongs without any dependence on the other components, but also (when shared) as a synchronizing action between (some or all) components that share that action. Finally, the team automaton $\mathcal{T}^{ccp}$, depicted in Figure 3(b), models the communication between two different Clients (instantiations of $\mathcal{C}$) and a Proxy. In this case, indiscriminately synchronizing on common actions would lead to two fully synchronized Clients. Instead, in $\mathcal{T}^{ccp}$ only the communication between an individual Client and the Proxy is synchronous. Moreover, as long as the Proxy still has obligations towards one of the Clients (*i.e.* a Client that is in its state p_2) it cannot accomodate a query of the other Client.



(a) $\mathcal{T}_2^{ai}$ (reachable part) (b) $\mathcal{T}^{ccp}$

Figure 3. Two team automata: $\mathcal{T}_2^{ai}$ over $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$ and $\mathcal{T}^{ccp}$ over $\{\mathcal{C}, \mathcal{C}, \mathcal{P}\}$.

Once a team automaton has been composed, it may itself be used as a component in the construction of a higher-level team automaton. For this hierarchical setup to work conveniently, it is important that the order in which components are added does not affect the behavior of the resulting team. In our example, adding a Server to the team $\mathcal{T}_1^{ai}$ consisting of a Client and the Proxy under the assumption that components synchronize on shared actions, should yield a team automaton with a behavior that is indistinguishable from that of $\mathcal{T}_2^{ai}$. Hence the chosen team automaton definition (the strategy followed to obtain a team from component automata) should translate into an associative operation at the behavioral level which implies that team automata obtained in such a way are compositional, *i.e.* their behavior can be obtained from the behaviors of their constituting components [7, 23].

A rather natural and common way of defining the behavior of a (team or component) automaton is to consider all (finite and infinite) sequences of actions that can be executed from its initial state. This leads to prefix-closed (not necessarily finitary) languages as the notion of behavior. From a purely operational point of view we moreover have to take into account that team automata may exhibit unfair behavior, in the sense that some of their components execute *ad infinitum* without giving the other components a fair turn. In our example, the infinite word **www** . . ., which originates from the Client and belongs to the behaviors of both $\mathcal{T}_1^{ai}$ and $\mathcal{T}_2^{ai}$, does not involve the other components of the team automata. Likewise, a word like **wqfrawqawqaw** . . . is a behavior of $\mathcal{T}_2^{ai}$ which from some point onwards no longer relates to the Server. Consequently, the team automata behaviors we are interested in are prefix-closed languages, consisting of potentially unfair combinations of (finite or infinite) words from the behaviors of their individual constituting component automata.

Motivated by basic methods to compose team automata, we investigate the associativity of different (potentially unfair) synchronized shuffle operations. In a synchronized shuffle of two words, rather than just interleaving (shuffling) all occurrences of the symbols of the words, some symbols may now be subject to synchronization. In that case two occurrences of the symbol, each from a different word, are identified as a single occurrence in the shuffle. Synchronized shuffles appear in numerous disguises throughout the literature (see, *e.g.*, [2, 9, 10, 14, 25, 27, 30, 31, 32, 33, 35, 37, 38, 41]). The synchronized shuffle operations we consider in this paper differ in the choice of synchronizing symbols, *i.e.* the set of symbols that may be subject to synchronization. With the standard full synchronous product automaton, comprising all product transitions in which all and only those components sharing an action synchronize on that action (like $\mathcal{T}_1^{ai}$ and $\mathcal{T}_2^{ai}$ in Figures 2(b) and 3(a)), we associate the *full* (called *strong* in [9]) *synchronized shuffle* which, given two alphabets, has their intersection as its set of synchronizing symbols. Then there is the possibility to define a team automaton by imposing no restrictions at all (like $\mathcal{T}^{no}$ in Figure 2(c)). For this case, we have the *arbitrary synchronized shuffle* from [9] that allows, but does not require, synchronization on all symbols from a given alphabet. Thirdly, for the case that one selects common actions on which components have to synchronize while leaving the execution of the other actions to the individual components, there is the *relaxed synchronized shuffle*, which synchronizes on a specific subset of the common symbols of the given alphabets. Whereas the team automaton $\mathcal{T}_1^{ai}$ in Figure 2(b) requires synchronization on all shared symbols, we have as another extreme case team automaton $\mathcal{T}^{free}$ in Figure 2(a), in which there is no synchronization at all. Note that in the latter case (no synchronizing symbols), the synchronized shuffle has actually degenerated to the ordinary (interleaving) shuffle operation that we considered in [8]. To complete the picture, we include in our investigations in this paper also the *weak synchronized shuffle* from [9], which also identifies synchronizing symbols, but allows to skip synchronization for some symbol occurrences. To derive the properties of these four synchronized shuffles we moreover explicitly consider the general *synchronized shuffle*, which lets words synchronize on a given but arbitrary set of synchronizing symbols.

The research presented in this paper is motivated by our ongoing study of team automata and compositionality, and forms a continuation of [4, 5, 7, 8]. It is difficult, if not impossible, to find in the literature explicit results concerning the associativity of (synchronized) shuffle operations in the context of possibly infinite words and unfairness. In [8], we considered the classical shuffle operation (see, *e.g.*, [11, 19, 20, 21, 22, 31, 32, 39]) in the context of possibly infinite words. Rather than trying to adapt existing results for the finitary (fair) case, we first investigated the more general issue of the relationship between shuffles of (finite or infinite) words and the shuffles of their finite prefixes. We showed that the potentially unfair shuffles of two words can be characterized in terms of the shuffles of their prefixes, a

result that does not always hold for fair shuffles. This directly implied the associativity for the general, potentially unfair, shuffle operation of finite and infinite words. Since the five synchronized shuffles we consider in this paper are defined in terms of the shuffle operation, the results obtained in [8] can be used for the investigation of synchronized shuffles. In fact, we establish that also the (fair and unfair) synchronized shuffles of two words are exactly the limits of the synchronized shuffles of their prefixes. Moreover, we prove that each of the synchronized shuffle operations we consider, except for the weak synchronized shuffle, satisfies a natural notion of associativity. In case of the full and relaxed synchronized shuffle, the associativity result does not hold for single words or arbitrary languages. It can however be proven to hold for prefix-closed non-finitary languages (such as the behavior of team automata).

This paper is organized as follows. In the preliminary Section 2, we introduce notations and definitions, and recall relevant results concerning the shuffle operation from [8]. Subsequently, in the main Section 3 of this paper, we define synchronized shuffling, establish some of its basic properties, and characterize synchronized shuffles in terms of the synchronized shuffles of their prefixes. We furthermore introduce four variants of synchronized shuffling and investigate their associativity. In Section 4 we relate various synchronization mechanisms for team automata to shuffle operations and consequently apply our associativity results to show that the corresponding team automata are compositional. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Basic Notations and Definitions

In this section, Δ is a fixed but arbitrary alphabet, *i.e.* a possibly empty, possibly infinite set of symbols. A word over Δ is a (finite or infinite) sequence $a_1a_2\cdots$ of symbols $a_i \in \Delta$. The empty sequence is denoted by λ. The set of all finite words over Δ (including λ) is denoted by Δ^* and $\Delta^+ = \Delta^* \setminus \{\lambda\}$ consists of all nonempty finite words over Δ. The set of all infinite words over Δ is denoted by Δ^ω and $\Delta^\infty = \Delta^* \cup \Delta^\omega$ denotes the set comprising all finite and infinite words over Δ. A language (over Δ) is any subset of Δ^∞. A language L consisting solely of finite words ($L \subseteq \Delta^*$) is called finitary. If L contains no finite words at all ($L \subseteq \Delta^\omega$), then it is referred to as being infinitary. When dealing with a singleton language $L = \{w\}$, we may omit the brackets and write w rather than $\{w\}$.

For a finite word w over Δ, we use $|w|$ to denote its length. Hence $|\lambda| = 0$ and $|a_1a_2\cdots a_n| = n$, if $a_i \in \Delta$ for all $1 \leq i \leq n$. For a word $w \in \Delta^\infty$ and an integer $j \geq 1$, we use $w(j)$ to denote the symbol from Δ which occurs at the jth position in w, provided that either w is infinite or $j \leq |w|$. The set comprising all symbols occurring in w is referred to as the alphabet of w and denoted as $\mathrm{alph}(w)$. Thus $\mathrm{alph}(w) = \{\, a \in \Delta : \exists\, j \in \mathbb{N} \text{ such that } w(j) = a \,\}$. Note that $\mathrm{alph}(\lambda) = \varnothing$ and that $\mathrm{alph}(w)$ may be an infinite set in case Δ is infinite and w is an infinite word. The alphabet of a language L is $\mathrm{alph}(L) = \bigcup\{\, \mathrm{alph}(w) : w \in L \,\}$.

For two words $u, v \in \Delta^\infty$, their concatenation $u \cdot v \in \Delta^\infty$ is defined as follows. If both u and v are finite, then $(u \cdot v)(i) = u(i)$ for $1 \leq i \leq |u|$ and $(u \cdot v)(|u| + i) = v(i)$ for $1 \leq i \leq |v|$. If $u \in \Delta^*$ and $v \in \Delta^\omega$, then $(u \cdot v)(i) = u(i)$ for $1 \leq i \leq |u|$ and $(u \cdot v)(|u| + i) = v(i)$ for $i \geq 1$. If $u \in \Delta^\omega$ and $v \in \Delta^\infty$, then $(u \cdot v)(i) = u(i)$ for all $i \geq 1$. Note that always $u \cdot \lambda = \lambda \cdot u = u$. The concatenation of two languages K and L is the language $K \cdot L = \{\, u \cdot v : u \in K,\ v \in L \,\}$. We mostly write uv and KL rather than $u \cdot v$ and $K \cdot L$, respectively.

A word $u \in \Delta^*$ is a (finite) prefix of a word $w \in \Delta^\infty$, written $u \leq w$, if there exists a $v \in \Delta^\infty$ such

that $w = uv$. The set of all (finite) prefixes of w is $\mathrm{pref}(w) = \{ u \in \Delta^* : u \leq w \}$. For a language L, $\mathrm{pref}(L) = \bigcup\{ \mathrm{pref}(w) : w \in L \}$ consists of all prefixes of all words in L.

Both finite and infinite words can be obtained as the limit of their prefixes. Let $v_1, v_2, \ldots \in \Delta^*$ be an infinite sequence of words such that $v_i \leq v_{i+1}$, for all $i \geq 1$. Then $\lim_{n\to\infty} v_n$ is the unique word $w \in \Delta^\infty$ defined by $w(j) = v_i(j)$, for all $i, j \in \mathbb{N}$ such that $j \leq |v_i|$. Hence $v_i \leq w$ for all $i \geq 1$ and $w = v_k$ whenever there exists a $k \geq 1$ such that $v_n = v_{n+1}$ for all $n \geq k$. For an infinite sequence of finite words $u_1, u_2, \ldots \in \Delta^*$, we use the notation $u_1 u_2 \cdots$ to denote the word $\lim_{n\to\infty} u_1 u_2 \cdots u_n$.

A language $K \subseteq \Delta^\infty$ is said to be limit-closed if $\lim_{n\to\infty} w_n \in K \cup \mathrm{pref}(K)$ whenever $w_1 \leq w_2 \leq \cdots \in \mathrm{pref}(K)$. Limit-closedness guarantees that the finite prefixes of a language determine its infinitary part. This notion has been defined in many disguises throughout the literature on theoretical computer science (see, *e.g.*, [1, 18]). Note that all singleton languages (all finite languages in fact) are limit-closed. In contrast, a^* is not limit-closed because $\lim_{n\to\infty} a^n = a^\omega \notin a^*$.

Let Σ and Γ be two alphabets and let $h : \Sigma \to \Gamma^*$ be a function assigning to each letter of Σ a finite word over Γ. The homomorphic extension of h to Σ^*, also denoted by h, is defined in the usual way by $h(xy) = h(x)h(y)$ for all $x, y \in \Sigma^*$, and we extend h to Σ^∞ by setting $h(\lim_{n\to\infty} v_n) = \lim_{n\to\infty} h(v_n)$, for all $v_1, v_2, \ldots \in \Sigma^*$ such that $v_i \leq v_{i+1}$ for all $i \geq 1$. Note that this is well-defined, since $v_i \leq v_{i+1}$ implies $h(v_i) \leq h(v_{i+1})$. The function $\mathrm{pres}_{\Sigma,\Gamma} : \Sigma \to \Gamma \cup \lambda$ preserves symbols from Γ while erasing all other symbols in Σ. It is defined by $\mathrm{pres}_{\Sigma,\Gamma}(a) = \lambda$ if $a \in \Sigma \setminus \Gamma$ and $\mathrm{pres}_{\Sigma,\Gamma}(a) = a$ if $a \in \Sigma \cap \Gamma$. In the sequel, when the domain alphabet Σ is understood, we may omit the subscript Σ from $\mathrm{pres}_{\Sigma,\Gamma}$.

2.2. Shuffles

Shuffling two words means interleaving their symbol occurrences (while preserving the original order within each word). Consequently, in a shuffle of two words, *i.e.* a word that can be obtained by shuffling these words, one of the two may prevail if it is infinite, with the result that the other word, from a certain point onwards, no longer contributes any symbol occurrences.

Definition 2.1. Let $u, v \in \Delta^\infty$. Then

(1) $w \in \Delta^\infty$ is a *fair shuffle* of u and v if $w = u_1 v_1 u_2 v_2 \cdots$, where $u_i, v_i \in \Delta^*$, for all $i \geq 1$, are such that $u = u_1 u_2 \cdots$ and $v = v_1 v_2 \cdots$, and

(2) $w \in \Delta^\infty$ is a *shuffle* of u and v if either

  (a) w is a fair shuffle of u and v, or

  (b) $w = u_1 v_1 u_2 v_2 \cdots$, with $u_i, v_i \in \Delta^*$ for all $i \geq 1$ and either $u_1 u_2 \cdots \in \mathrm{pref}(u)$ and $v = v_1 v_2 \cdots \in \Delta^\omega$ or $u = u_1 u_2 \cdots \in \Delta^\omega$ and $v_1 v_2 \cdots \in \mathrm{pref}(v)$.

The set of all fair shuffles of two words u and v is denoted by $u \,|||\, v$, and $u \,||\, v$ is the set consisting of all shuffles of u and v. Shuffling two languages is defined element-wise. Hence, the *fair shuffle* of two languages L_1 and L_2 is $L_1 \,|||\, L_2 = \{ w \in u \,|||\, v : u \in L_1,\ v \in L_2 \}$ and, similarly, the *shuffle* of L_1 and L_2 is $L_1 \,||\, L_2 = \{ w \in u \,||\, v : u \in L_1,\ v \in L_2 \}$.

It is immediate that $u \,|||\, v$ and $u \,||\, v$ are never empty: (fairly) shuffling two words always yields at least one word. Furthermore, it easy to see that both fair shuffling and shuffling are commutative

operations. By definition, all shuffles of two finite words are fair: $u \,||\, v = u \,|||\, v$ whenever u and v are finite words. If, on the other hand, at least one of u and v is infinite, then $u \,||\, v = u \,|||\, v$ does not necessarily hold.

Fair shuffles of two words over disjoint alphabets have the following well-known characterization.

Proposition 2.1. Let Δ_1 and Δ_2 be such that $\Delta_1 \cap \Delta_2 = \varnothing$. Let $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$. Then $u \,|||\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \text{pres}_{\Delta_1}(w) = u,\ \text{pres}_{\Delta_2}(w) = v \,\}$.

In [8], it was proven that shuffling the prefixes of two words yields all and only prefixes of the shuffles of these words.

Proposition 2.2. Let $u,v \in \Delta^\infty$. Then $\text{pref}(u) \,||\, \text{pref}(v) = \text{pref}(u \,||\, v) = \text{pref}(u \,|||\, v) = \text{pref}(u) \,|||\, \text{pref}(v)$.

Furthermore, also in [8], it was shown that a word must be a shuffle of two given words whenever all its prefixes are shuffles of prefixes of these two words. This is a property that does not hold when unfair shuffling is not taken into consideration. As formulated next, this property of words can be lifted to limit-closed languages, due to the fact that the infinitary part of such languages is defined by their finite prefixes.

Proposition 2.3. Let $u, v \in \Delta^\infty$, let $K, L \subseteq \Delta^\infty$ be limit-closed and let $w \in \Delta^\omega$. Then $w \in u \,||\, v$ if and only if $\text{pref}(w) \subseteq \text{pref}(u) \,||\, \text{pref}(v)$, and $w \in K \,||\, L$ if and only if $\text{pref}(w) \subseteq \text{pref}(K) \,||\, \text{pref}(L)$.

Using Proposition 2.2 and Proposition 2.3, it was proven in [8] that both fair shuffling and shuffling are associative operations on words.

Proposition 2.4. Let $u, v, w \in \Delta^\infty$. Then $u \,|||\, (v \,|||\, w) = (u \,|||\, v) \,|||\, w$ and $u \,||\, (v \,||\, w) = (u \,||\, v) \,||\, w$.

3. Synchronized Shuffles

As in an ordinary shuffle, the occurrences of symbols in a synchronized shuffle may be interleaving 'original' occurrences from one of the two input words, but they may also be a synchronization of two occurrences of a shared symbol from a designated set of synchronizing symbols. In [4, 7], we coined the term *backbone* for the scattered subwords formed by these synchronizing occurrences: Let Δ and Γ be two alphabets and w a word over Δ. Then $\text{pres}_\Gamma(w)$ is the Γ-*backbone* of w. When Γ is clear from the context we may omit the reference to Γ and speak simply of the *backbone* of w. In this paper we assume, as we did in [4, 7, 9], that these backbones are not defined *a priori*: only their constituents (symbols) without their sequence (order) are known. In [30], a distinction is made between such synchronized shuffling on alphabets and synchronized shuffling on backbones. In the latter case, the set of possible backbones is defined *a priori*: the order that the synchronizing symbols must adhere to is predefined.

The rest of this section is divided into five subsections, each devoted to one particular type of synchronized shuffle. The first of these subsections investigates general properties of synchronized shuffling and as such extends [8] and forms a basis for the other subsections. Until the end of this section, Δ and Γ refer to arbitrary alphabets.

3.1. Synchronized Shuffling: Basic Results

Definition 3.1. Let $u, v \in \Delta^\infty$. Then a word $w \in \Delta^\infty$ is a *synchronized shuffle* (*S-shuffle* for short) *on* Γ of u and v if

(1) either $u = u_1x_1u_2x_2\cdots x_{n-1}u_n$ and $v = v_1x_1v_2x_2\cdots x_{n-1}v_n$, with $u_i, v_i \in (\Delta \setminus \Gamma)^*$ and $x_i \in \Gamma$ for $1 \leq i \leq n-1$, and $u_n, v_n \in (\Delta \setminus \Gamma)^\infty$, in which case $w = w_1x_1w_2x_2\cdots x_{n-1}w_n$ with $w_i \in u_i \,||\, v_i$ for all $1 \leq i \leq n$,

(2) or $u = u_1x_1u_2x_2\cdots$ and $v = v_1x_1v_2x_2\cdots$, with $u_i, v_i \in (\Delta \setminus \Gamma)^*$ and $x_i \in \Gamma$ for all $i \geq 1$, in which case $w = w_1x_1w_2x_2\cdots$ with $w_i \in u_i \,||\, v_i$ for all $i \geq 1$.

In case (2), w is a *fair* S-shuffle of u and v. Moreover, in case (1) w is *fair* whenever $w_n \in (u_n \,|||\, v_n)$.

We use the following notation for S-shuffles. For two words $u, v \in \Delta^\infty$, we let $u \,||^\Gamma v = \{\, w \in \Delta^\infty : w$ is an S-shuffle on Γ of u and $v \,\}$ and $u \,|||^\Gamma v = \{\, w \in \Delta^\infty : w$ is a fair S-shuffle on Γ of u and $v \,\}$. For $L_1, L_2 \subseteq \Delta^\infty$, the *S-shuffle* on Γ of L_1 and L_2 is $L_1 \,||^\Gamma L_2 = \{\, w \in u \,||^\Gamma v : u \in L_1,\ v \in L_2 \,\}$ and $L_1 \,|||^\Gamma L_2 = \{\, w \in u \,|||^\Gamma v : u \in L_1,\ v \in L_2 \,\}$ is the *fair S-shuffle* on Γ of L_1 and L_2.

Example 3.1. $abc \,||^{\{c\}} cd = \{abcd\}$, but $abcd \notin abc \,||\, cd$, and $abc \,||^{\{b,c\}} cd = \varnothing$. Also $a^\omega \,|||^{\{a\}} a = a^\omega \,||^{\{a\}} a = \varnothing$, while $a^\omega \,|||\, a = a^\omega \,||\, a = a^\omega$.

It is easy to see that an S-shuffle of two words exists if and only if these words have the same backbone (implying also that (fair) shuffles of two words always exist).

Lemma 3.1. Let $u, v \in \Delta^\infty$. Then $u \,||^\Gamma v \neq \varnothing$ if and only if $\mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v)$. Moreover, for all $w \in u \,||^\Gamma v$, $\mathrm{pres}_\Gamma(w) = \mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v)$.

Obviously, shuffling is synchronized shuffling on an empty alphabet: $u \,|||\, v = u \,|||^\varnothing v$ and $u \,||\, v = u \,||^\varnothing v$ for all $u, v \in \Delta^\infty$. On the other hand, S-shuffling and fair S-shuffling are defined in terms of (fair) shuffling, which with the commutativity of (fair) shuffling immediately implies that they also are commutative operations.

Similarly, the associativity of (fair) S-shuffling can be proven by referring to the associativity of (fair) shuffling.

Theorem 3.1. Let $u, v, w \in \Delta^\infty$. Then $u \,|||^\Gamma (v \,|||^\Gamma w) = (u \,|||^\Gamma v) \,|||^\Gamma w$ and $u \,||^\Gamma (v \,||^\Gamma w) = (u \,||^\Gamma v) \,||^\Gamma w$.

Proof:
We only prove the first equality. The other proof is analogous. Let $x \in u \,|||^\Gamma (v \,|||^\Gamma w)$. Then by Lemma 3.1, $\mathrm{pres}_\Gamma(x) = \mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v) = \mathrm{pres}_\Gamma(w)$. Now let $y = \mathrm{pres}_\Gamma(x)$. First assume that $y \in \Gamma^*$. Then $y = y_1y_2\cdots y_n$ for some $n \geq 0$ and $y_i \in \Gamma$, for all $1 \leq i \leq n$. Consequently there exist $x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_n \in \Delta^*$ and $x_{n+1}, u_{n+1}, v_{n+1}, w_{n+1} \in \Delta^\infty$ such that $x = x_1y_1x_2y_2\cdots x_ny_nx_{n+1}$, $u = u_1y_1u_2y_2\cdots u_ny_nu_{n+1}$, $v = v_1y_1v_2y_2\cdots v_ny_nv_{n+1}$ and $w = w_1y_1w_2y_2\cdots w_ny_nw_{n+1}$. By Definition 3.1, $x_i \in u_i \,||\, (v_i \,||\, w_i)$, for all $1 \leq i \leq n$, and $x_{n+1} \in u_{n+1} \,|||\, (v_{n+1} \,|||\, w_{n+1})$. Now by Proposition 2.4, $u_i \,||\, (v_i \,||\, w_i) = (u_i \,||\, v_i) \,||\, w_i$, for all $1 \leq i \leq n$, and $u_{n+1} \,|||\, (v_{n+1} \,|||\, w_{n+1}) = (u_{n+1} \,|||\, v_{n+1}) \,|||\, w_{n+1}$. This implies, again by Definition 3.1, that $x \in (u \,|||^\Gamma v) \,|||^\Gamma w$. The case that $y \in \Gamma^\omega$ can be treated analogously. □

The statement of Theorem 3.1 would not hold if the synchronizing symbols were allowed to vary.

Example 3.2. $(ca \,||^{\{a\}}\, ab) \,||^{\{c\}}\, c = cab \,||^{\{c\}}\, c = cab$, while $ca \,||^{\{a\}}\, (ab \,||^{\{c\}}\, c) = ca \,||^{\{a\}}\, \varnothing = \varnothing$.

Also the characterization of fair shuffling in the case of disjoint alphabets given in Proposition 2.1 can be carried over to S-shuffling, now with the proviso that the non-synchronizing symbols are not shared.

Theorem 3.2. Let $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$ be such that $(\Delta_1 \setminus \Gamma) \cap (\Delta_2 \setminus \Gamma) = \varnothing$. Then $u \,|||^\Gamma\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \mathrm{pres}_\Gamma(w) = \mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v),\ \mathrm{pres}_{\Delta_1}(w) = u,\ \mathrm{pres}_{\Delta_2}(w) = v \,\}$.

Proof:
($\subseteq$) Let $w \in u \,|||^\Gamma\, v$. Since by Lemma 3.1 $\mathrm{pres}_\Gamma(w) = \mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v)$, we only have to prove that $\mathrm{pres}_{\Delta_1}(w) = u$ and $\mathrm{pres}_{\Delta_2}(w) = v$. According to Definition 3.1 we can distinguish two cases. First assume that w has a finite backbone $y_1 y_2 \cdots y_n$ for some $n \geq 0$ and $y_1, y_2, \ldots, y_n \in \Gamma$. Then $w = w_1 y_1 w_2 y_2 \cdots y_n w_{n+1}$, with $w_1, w_2, \ldots, w_n \in ((\Delta_1 \cup \Delta_2) \setminus \Gamma)^*$, $w_{n+1} \in ((\Delta_1 \cup \Delta_2) \setminus \Gamma)^\infty$, $u = u_1 y_1 u_2 y_2 \cdots y_n u_{n+1}$, with $u_1, u_2, \ldots, u_n \in (\Delta_1 \setminus \Gamma)^*$ and $u_{n+1} \in (\Delta_1 \setminus \Gamma)^\infty$, and $v = v_1 y_1 v_2 y_2 \cdots y_n v_{n+1}$, with $v_1, v_2, \ldots, v_n \in (\Delta_2 \setminus \Gamma)^*$ and $v_{n+1} \in (\Delta_2 \setminus \Gamma)^\infty$, such that $w_i \in u_i \,|||\, v_i$ for all $1 \leq i \leq n+1$. Since $(\Delta_1 \setminus \Gamma) \cap (\Delta_2 \setminus \Gamma) = \varnothing$, it follows from Proposition 2.1 that $\mathrm{pres}_{\Delta_1}(w_i) = u_i$ and $\mathrm{pres}_{\Delta_2}(w_i) = v_i$ for all $1 \leq i \leq n+1$. Hence $\mathrm{pres}_{\Delta_1}(w) = \mathrm{pres}_{\Delta_1}(w_1)\mathrm{pres}_{\Delta_1}(y_1)\mathrm{pres}_{\Delta_1}(w_2)\mathrm{pres}_{\Delta_1}(y_2) \cdots \mathrm{pres}_{\Delta_1}(y_n)\mathrm{pres}_{\Delta_1}(w_{n+1}) = u_1 y_1 u_2 y_2 \cdots y_n u_{n+1} = u$ and, analogously, $\mathrm{pres}_{\Delta_2}(w) = v_1 y_1 v_2 y_2 \cdots y_n v_{n+1} = v$.

The case of an infinite Γ-backbone can be proven analogously.

($\supseteq$) Let $w \in (\Delta_1 \cup \Delta_2)^\infty$ be such that $\mathrm{pres}_\Gamma(w) = \mathrm{pres}_\Gamma(u) = \mathrm{pres}_\Gamma(v)$, $\mathrm{pres}_{\Delta_1}(w) = u$, and $\mathrm{pres}_{\Delta_2}(w) = v$. Observe that $(\Delta_1 \setminus \Gamma) \cap (\Delta_2 \setminus \Gamma) = \varnothing$ implies that $\Delta_1 \cap \Delta_2 \subseteq \Gamma$. Since the backbone of w consists solely of symbols from $\Delta_1 \cap \Delta_2$, it thus suffices to prove that $w \in u \,|||^{\Delta_1 \cap \Delta_2}\, v$. First assume that w has a finite backbone. Then there exists an $n \geq 1$ such that $w = w_1 y_1 w_2 y_2 \cdots y_n w_{n+1}$, where for all $1 \leq i \leq n$, $w_i \in ((\Delta_1 \setminus \Delta_2) \cup (\Delta_2 \setminus \Delta_1))^*$ and $y_i \in \Delta_1 \cap \Delta_2$, and $w_{n+1} \in ((\Delta_1 \setminus \Delta_2) \cup (\Delta_2 \setminus \Delta_1))^\infty$. Moreover, $\mathrm{pres}_{\Delta_1}(w) = \mathrm{pres}_{\Delta_1}(w_1) y_1 \mathrm{pres}_{\Delta_1}(w_2) y_2 \cdots y_n \mathrm{pres}_{\Delta_1}(w_{n+1}) = u$ and $\mathrm{pres}_{\Delta_2}(w) = \mathrm{pres}_{\Delta_2}(w_1) y_1 \mathrm{pres}_{\Delta_2}(w_2) y_2 \cdots y_n \mathrm{pres}_{\Delta_2}(w_{n+1}) = v$. Hence $u = u_1 y_1 u_2 y_2 \cdots y_n u_{n+1}$, with $u_i = \mathrm{pres}_{\Delta_1 \setminus \Delta_2}(w_i)$, for all $1 \leq i \leq n+1$, and $v = v_1 y_1 v_2 y_2 \cdots y_n v_{n+1}$, with $v_i = \mathrm{pres}_{\Delta_2 \setminus \Delta_1}(w_i)$, for all $1 \leq i \leq n+1$. Since for all $1 \leq i \leq n+1$, $u_i \in (\Delta_1 \setminus \Delta_2)^\infty$, $v_i \in (\Delta_2 \setminus \Delta_1)^\infty$ and $(\Delta_1 \setminus \Delta_2) \cap (\Delta_2 \setminus \Delta_1) = \varnothing$, Proposition 2.1 implies that for all $1 \leq i \leq n$, $w_i \in u_i \,|||\, v_i = u_i \,||\, v_i$, and $w_{n+1} \in u_{n+1} \,|||\, v_{n+1} \subseteq u_{n+1} \,||\, v_{n+1}$. Definition 3.1(1) now implies that $w \in u \,|||^{\Delta_1 \cap \Delta_2}\, v$.

Again, the proof for the case that w has an infinite Γ-backbone is analogous. □

Finally, we investigate the limit behavior of S-shuffles. In particular, we aim to prove that also the S-shuffles of two words can be obtained as limits of the S-shuffles of their prefixes (see Proposition 2.3). First we formulate explicitly how S-shuffles can be expressed in terms of fair S-shuffles of their prefixes.

Lemma 3.2. (1) If $u \in \Delta_1^*$ and $v \in \Delta_2^*$, then $u \,||^\Gamma\, v = u \,|||^\Gamma\, v$.

(2) If $u \in \Delta_1^*$ and $v \in \Delta_2^\omega$, then $u \,||^\Gamma\, v = \{\, w \in u' \,|||^\Gamma\, v : u' \in \mathrm{pref}(u),\ \mathrm{pres}_\Gamma(u') = \mathrm{pres}_\Gamma(u) \,\}$.

(3) If $u \in \Delta_1^\omega$ and $v \in \Delta_2^\omega$, then $u \,||^\Gamma\, v = u \,|||^\Gamma\, v \cup \{\, w \in u' \,|||^\Gamma\, v : u' \in \mathrm{pref}(u),\ \mathrm{pres}_\Gamma(u') = \mathrm{pres}_\Gamma(u) \,\} \cup \{\, w \in u \,|||^\Gamma\, v' : v' \in \mathrm{pref}(v),\ \mathrm{pres}_\Gamma(v') = \mathrm{pres}_\Gamma(v) \,\}$.

Now consider the S-shuffles on Γ of two (possibly infinite) words u and v under the assumption that they have a finite backbone. Hence $u = u_1u_2$ and $v = v_1v_2$, with u_2 and v_2 the maximal suffixes of u and v without symbols from Γ. Then, obviously, the elements of any S-shuffle of u and v consist of a prefix that is an S-shuffle of u_1 and v_1 and a suffix that is a shuffle of u_2 and v_2. Consequently, we obtain the following result.

Lemma 3.3. Let $u_1, v_1 \in ((\Delta\backslash\Gamma)^*\Gamma)^*$ and $u_2, v_2 \in (\Delta\backslash\Gamma)^\infty$. Then $u_1u_2 \,|||^\Gamma\, v_1v_2 = (u_1 \,|||^\Gamma\, v_1)(u_2 \,|||\, v_2)$ and $u_1u_2 \,||^\Gamma\, v_1v_2 = (u_1 \,||^\Gamma\, v_1)(u_2 \,||\, v_2)$.

Using this observation, we can now prove that for the case of finite backbones, the limits of the S-shuffles of the prefixes of two words are S-shuffles of these two words. As a matter of fact, we prove this statement in the more general setting of limit-closed languages.

Lemma 3.4. Let $K, L \subseteq \Delta^\infty$ be two limit-closed languages and let $w \in ((\Delta \setminus \Gamma)^*\Gamma)^*(\Delta \setminus \Gamma)^\infty$ be such that $\text{pref}(w) \subseteq \text{pref}(K) \,||^\Gamma\, \text{pref}(L)$. Then $w \in K \,||^\Gamma\, L$.

Proof:
Let $w_1 \in ((\Delta\backslash\Gamma)^*\Gamma)^*$ and $w_2 \in (\Delta\backslash\Gamma)^\infty$ be such that $w = w_1w_2$. Since $\text{pref}(w) \subseteq \text{pref}(K) \,||^\Gamma\, \text{pref}(L)$, there exist an $n \geq 1$ and words $x_i \in \text{pref}(K)$ and $y_i \in \text{pref}(L)$ such that $w_1 \in x_i \,||^\Gamma\, y_i$ for all $1 \leq i \leq n$. Note that according to Definition 3.1, all $x_i, y_i \in ((\Delta \setminus \Gamma)^*\Gamma)^*$. Let $K_{x_i} = \{\, x \in (\Delta \setminus \Gamma)^\infty : x_ix \in K \,\}$ and $L_{y_i} = \{\, y \in (\Delta \setminus \Gamma)^\infty : y_iy \in L \,\}$, for all $1 \leq i \leq n$.

Let z be a prefix of w_2. Then w_1z is a prefix of w, which implies that $w_1z \in \text{pref}(K) \,||^\Gamma\, \text{pref}(L)$. Hence there exist $x \in \text{pref}(K)$ and $y \in \text{pref}(L)$ such that $w_1z \in x \,||^\Gamma\, y$. Moreover, $x = x'x''$ and $y = y'y''$, with $x', y' \in ((\Delta \setminus \Gamma)^*\Gamma)^*$ and $x'', y'' \in (\Delta \setminus \Gamma)^*$, and $w_1 \in x' \,||^\Gamma\, y'$. Hence $x' = x_i$ and $y' = y_i$ for some $1 \leq i \leq n$. This implies that $w_1z \in x_i\text{pref}(K_{x_i}) \,||^\Gamma\, y_i\text{pref}(K_{y_i})$. Then, by Lemma 3.3, $\text{pref}(w_2) \subseteq \bigcup_{1\leq i\leq n}(\text{pref}(K_{x_i}) \,||\, \text{pref}(L_{y_i}))$.

Thus, in case w_2 is a finite word, there exists a $1 \leq j \leq n$ such that $w_2 \in \text{pref}(K_{x_j}) \,||\, \text{pref}(L_{y_j})$. Hence, with Lemma 3.3, $w = w_1w_2 \in (x_j \,||^\Gamma\, y_j)(K_{x_j} \,||\, L_{y_j}) = x_jK_{x_j} \,||^\Gamma\, y_jL_{y_j} \subseteq K \,||^\Gamma\, L$.

If w_2 is infinite, then—because there are only finitely many pairs x_i, y_i—there exists a $1 \leq j \leq n$ such that for each $z \in \text{pref}(w_2)$ there is a prefix z' of w_2 with $z \leq z'$ and $z \neq z'$, for which $z' \in \text{pref}(K_{x_j}) \,||\, \text{pref}(L_{y_j})$ and thus also $z \in \text{pref}(K_{x_j}) \,||\, \text{pref}(L_{y_j})$. Thus, $\text{pref}(w_2) \subseteq \text{pref}(K_{x_j}) \,||\, \text{pref}(L_{y_j})$ for this j. Since K and L are limit-closed, so are K_{x_j} and L_{y_j}. Proposition 2.3 then implies that $w_2 \in K_{x_j} \,||\, L_{y_j}$. Hence, again with Lemma 3.3, $w = w_1w_2 \in (x_j \,||^\Gamma\, y_j)(K_{x_j} \,||\, L_{y_j}) = x_jK_{x_j} \,||^\Gamma\, y_jL_{y_j} \subseteq K \,||^\Gamma\, L$. □

As in the proof above, also in the case of an infinite backbone, one needs an argument that the S-shuffles of the prefixes of two words can be consistently extended to an (infinite) S-shuffle. Now we rely on König's Lemma and provide a proof along the same lines as the proof of the corresponding lemma for ordinary shuffles (Lemma 17 of [8]). One may observe here that the notion of limit-closedness of a language resembles that of the *adherence* of a language L (the set $\text{adh}(L)$ consisting of all infinite words that can be obtained as a limit of words from L): $L \subseteq \Delta^\infty$ is limit-closed if and only if $\text{adh}(\text{pref}(L)) \subseteq L$. (See [12] and also Chapter 3 of [36] for a clarification of topological notions on infinite words).

Lemma 3.5. Let $K, L \subseteq \Delta^\infty$ be two limit-closed languages and let $w \in ((\Delta \setminus \Gamma)^*\Gamma)^\omega$ be such that $\text{pref}(w) \subseteq \text{pref}(K) \,||^\Gamma\, \text{pref}(L)$. Then $w \in K \,||^\Gamma\, L$.

Proof:
Let $w = w_1x_1w_2x_2\cdots$ with $w_1, w_2, \ldots \in (\Delta \setminus \Gamma)^*$ and $x_1, x_2, \ldots \in \Gamma$. For each $n \geq 1$, we denote the prefix $w_1x_1w_2x_2\cdots w_nx_n$ by $w[n]$. Since $\text{pref}(w) \subseteq \text{pref}(K) \,||^\Gamma \text{pref}(L)$, there exists for each $w[n]$ at least one 'decomposition' sequence $u_1, v_1, u_2, v_2, \ldots, u_n, v_n$ of words from $(\Delta \setminus \Gamma)^*$ such that $u_1x_1u_2x_2\cdots u_nx_n \in \text{pref}(K)$, $v_1x_1v_2x_2\cdots v_nx_n \in \text{pref}(L)$ and $w_i \in (u_i \,||\, v_i)$ for all $1 \leq i \leq n$. That is, $w[n] \in (u_1 \,||\, v_1)x_1(u_2 \,||\, v_2)x_2\cdots(u_n \,||\, v_n)x_n$. With each decomposition sequence $\rho = u_1, v_1, u_2, v_2, \ldots, u_n, v_n$ we associate the word $\rho_\$ = u_1\$v_1\$u_2\$v_2\$\cdots u_n\$v_n$, where \$ is a special marker symbol. Now consider the language $V = \{\, \rho_\$: \rho \text{ is a decomposition sequence for } w[n],\ n \geq 1 \,\}$ consisting of all marked decomposition sequences for the prefixes $w[n]$ of w. Since V is infinite and each $w[n]$ has only finitely many decomposition sequences, we can argue, using König's Lemma or Property 1 from [12], that there exists at least one infinite word z such that $\text{pref}(z) \subseteq V$. Let $z = u_1\$v_1\$u_2\$v_2\$\cdots$. Then $u_1x_1u_2x_2\cdots u_nx_n \in \text{pref}(K)$ and $v_1x_1v_2x_2\cdots v_nx_n \in \text{pref}(L)$. Since K and L are limit-closed this implies that $u = \lim_{n\to\infty} u_1x_1u_2x_2\cdots u_nx_n \in K$ and $v = \lim_{n\to\infty} v_1x_1v_2x_2\cdots v_nx_n \in L$. Hence $w \in u \,||^\Gamma v \subseteq K \,||^\Gamma L$. □

Now we are ready to conclude.

Theorem 3.3. Let $u, v \in \Delta^\infty$, let $K, L \subseteq \Delta^\infty$ be limit-closed and let $w \in \Delta^\omega$. Then $w \in u \,||^\Gamma v$ if and only if $\text{pref}(w) \subseteq \text{pref}(u) \,||^\Gamma \text{pref}(v)$, and $w \in K \,||^\Gamma L$ if and only if $\text{pref}(w) \subseteq \text{pref}(K) \,||^\Gamma \text{pref}(L)$.

Proof:
Since all singleton languages are limit-closed, it suffices to prove the second part of the statement. The if-direction is an immediate consequence of Definition 3.1 and Lemmata 3.4 and 3.5.

To prove the only-if-direction, let $w \in K \,||^\Gamma L$. Then, according to Definition 3.1, one of the following two cases holds. Either $w = (u_1 \,||\, v_1)x_1(u_2 \,||\, v_2)x_2\cdots x_{n-1}(u_n \,||\, v_n)$ for some $n \geq 1$, $u_i, v_i \in (\Delta\setminus\Gamma)^*$ and $x_i \in \Gamma$ for all $1 \leq i \leq n-1$, and $u_n, v_n \in (\Delta\setminus\Gamma)^\infty$ such that $u = u_1x_1u_2x_2\cdots x_{n-1}u_n$ and $v = v_1x_1v_2x_2\cdots x_{n-1}v_n$. Or else $w = (u_1 \,||\, v_1)x_1(u_2 \,||\, v_2)x_2\cdots$ for some $n \geq 1$, $u_i, v_i \in (\Delta \setminus \Gamma)^*$ and $x_i \in \Gamma$ for all $i \geq 1$, such that $u = u_1x_1u_2x_2\cdots$ and $v = v_1x_1v_2x_2\cdots$. Now consider a prefix $y \in \text{pref}(w)$. Then in both cases $y = (u_1 \,||\, v_1)x_1(u_2 \,||\, v_2)x_2\cdots x_{k-1}x$ for some $k \geq 1$ and $x \in \text{pref}(u_k \,||\, v_k)$. Hence $y \in u_1x_1u_2x_2\cdots u_{k-1}x_{k-1}\text{pref}(u_k) \,||^\Gamma v_1x_1v_2x_2\cdots v_{k-1}x_{k-1}\text{pref}(v_k) \subseteq \text{pref}(u) \,||^\Gamma \text{pref}(v)$, by Definition 3.1 and Proposition 2.2, and thus $y \in \text{pref}(K) \,||^\Gamma \text{pref}(L)$. □

3.2. Full Synchronized Shuffling

A *full S-shuffle* of two words is a synchronized shuffle in which all and only the symbols shared by their alphabets act as synchronizing symbols.

Definition 3.2. Let Δ_1 and Δ_2 be two alphabets and let $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in (\Delta_1 \cup \Delta_2)^\infty$. Then w is a *full S-shuffle* (*fS-shuffle* for short) of u and v w.r.t. Δ_1 and Δ_2 if w is an S-shuffle on $\Delta_1 \cap \Delta_2$ of u and v. This fS-shuffle of u and v w.r.t. Δ_1 and Δ_2 is called *fair* if w is a fair S-shuffle on $\Delta_1 \cap \Delta_2$ of u and v.

The full synchronized shuffle operation varies with the specification of the alphabets of the words considered. Given alphabets Δ_1, Δ_2, and words $u \in \Delta_1^\infty$ and $v \in \Delta_2{}^\infty$, we write $u \,{}_{\Delta_1}|||_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : w \text{ is a fair fS-shuffle of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2 \,\}$ and $u \,{}_{\Delta_1}||_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty :$

w is an fS-shuffle of u and v w.r.t. Δ_1 and Δ_2 }. For $L_1 \subseteq \Delta_1^\infty$ and $L_2 \subseteq \Delta_2^\infty$, the *fS-shuffle* of L_1 and L_2 w.r.t. Δ_1 and Δ_2 is $L_1 \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, L_2 = \{\, w \in u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v : u \in L_1,\ v \in L_2 \,\}$ and $L_1 \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, L_2 = \{\, w \in u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v : u \in L_1,\ v \in L_2 \,\}$ is the *fair fS-shuffle* of L_1 and L_2 w.r.t. Δ_1 and Δ_2.

For all pairs of words (languages) over Δ_1, Δ_2, we thus have ${}_{\Delta_1}\underline{|||}_{\Delta_2} = |||^{\Delta_1 \cap \Delta_2}$ and ${}_{\Delta_1}\underline{||}_{\Delta_2} = ||^{\Delta_1 \cap \Delta_2}$. Hence, in case of disjoint alphabets $\Delta_1 \cap \Delta_2 = \varnothing$, the fS-shuffle is the ordinary shuffle for words and languages over Δ_1, Δ_2: ${}_{\Delta_1}\underline{|||}_{\Delta_2} = |||^{\varnothing} = |||$ and ${}_{\Delta_1}\underline{||}_{\Delta_2} = ||^{\varnothing} = ||$.

Example 3.3. If $\Delta = \{a, b, c, d\}$, then $abc \,{}_{\Delta}\underline{||}_{\Delta}\, cd = \varnothing$. With $\Delta_1 = \{a, b, c\}$ and $\Delta_2 = \{c, d\}$, on the other hand, $abc \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, cd = abc \,||^{\{c\}}\, cd = \{abcd\}$. Also, $a^\omega \,{}_{\Delta}\underline{||}_{\Delta}\, b = a^\omega \,{}_{\Delta}\underline{|||}_{\Delta}\, b = \varnothing$, while $a^\omega \,{}_{\{a\}}\underline{|||}_{\{b\}}\, b = a^\omega \,|||\, b = \{\, w \in \{a, b\}^\omega : w$ has one occurrence of $b \,\}$ and $a^\omega \,{}_{\{a\}}\underline{||}_{\{b\}}\, b = a^\omega \,||\, b = (a^\omega \,{}_{\{a\}}\underline{|||}_{\{b\}}\, b) \cup \{a^\omega\}$.

Since the (fair) S-shuffle operation is commutative, it is immediate that also the (fair) fS-shuffle is commutative. Note that now also the alphabets Δ_1 and Δ_2 defining the operation have to be taken into account. Hence, we have $u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v = v \,{}_{\Delta_2}\underline{|||}_{\Delta_1}\, u$ and $u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = v \,{}_{\Delta_2}\underline{||}_{\Delta_1}\, u$.

Also Theorem 3.2, which provides an alternative definition of the fair S-shuffle, can be carried over to the more restricted case of fair fS-shuffles since by definition non-synchronizing symbols are not shared in fS-shuffles. With the additional observation that for every word $w \in (\Delta_1 \cup \Delta_2)^\infty$, the words $\mathrm{pres}_{\Delta_1}(w)$, $\mathrm{pres}_{\Delta_2}(w)$, and w itself always have the same $(\Delta_1 \cap \Delta_2)$-backbone, this yields the following characterization. (Here and in the remainder of this section, we use Δ_1, Δ_2, and Δ_3 to refer to fixed arbitrary alphabets.)

Theorem 3.4. Let $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$. Then $u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \mathrm{pres}_{\Delta_1}(w) = u,\ \mathrm{pres}_{\Delta_2}(w) = v \,\}$.

It is however less obvious how to prove associativity. Moreover, associativity should first be formalized. A natural approach is to consider fS-shuffling associative if $u \,{}_{\Delta_1}\underline{||}_{\Delta_2 \cup \Delta_3}\, (v \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, w)$ equals $(u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v) \,{}_{\Delta_1 \cup \Delta_2}\underline{||}_{\Delta_3}\, w$ for all words $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in \Delta_3^\infty$, and similarly for the fair case. Indeed, fair fS-shuffling is associative in this sense.

Theorem 3.5. Let $u \in \Delta_1^\infty$, let $v \in \Delta_2^\infty$, and let $w \in \Delta_3^\infty$. Then $u \,{}_{\Delta_1}\underline{|||}_{\Delta_2 \cup \Delta_3}\, (v \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, w) = (u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v) \,{}_{\Delta_1 \cup \Delta_2}\underline{|||}_{\Delta_3}\, w$.

Proof:
Let $x \in u \,{}_{\Delta_1}\underline{|||}_{\Delta_2 \cup \Delta_3}\, (v \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, w)$ and let $y \in v \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, w$ be such that $x \in u \,{}_{\Delta_1}\underline{|||}_{\Delta_2 \cup \Delta_3}\, y$. Hence, according to Theorem 3.4, $\mathrm{pres}_{\Delta_1}(x) = u$ and $\mathrm{pres}_{\Delta_2 \cup \Delta_3}(x) = y$. Moreover, $\mathrm{pres}_{\Delta_2}(y) = v$ and $\mathrm{pres}_{\Delta_3}(y) = w$. Now let $z = \mathrm{pres}_{\Delta_1 \cup \Delta_2}(x)$. By repeatedly applying Theorem 3.4 and the properties of preserving homomorphisms, we obtain $\mathrm{pres}_{\Delta_3}(x) = \mathrm{pres}_{\Delta_3}(\mathrm{pres}_{\Delta_2 \cup \Delta_3}(x)) = \mathrm{pres}_{\Delta_3}(y) = w$, and thus $x \in z \,{}_{\Delta_1 \cup \Delta_2}\underline{|||}_{\Delta_3}\, w$. Moreover, $\mathrm{pres}_{\Delta_1}(z) = \mathrm{pres}_{\Delta_1}(\mathrm{pres}_{\Delta_1 \cup \Delta_2}(x)) = \mathrm{pres}_{\Delta_1}(x) = u$ and $\mathrm{pres}_{\Delta_2}(z) = \mathrm{pres}_{\Delta_2}(\mathrm{pres}_{\Delta_1 \cup \Delta_2}(x)) = \mathrm{pres}_{\Delta_2}(x) = \mathrm{pres}_{\Delta_2}(\mathrm{pres}_{\Delta_2 \cup \Delta_3}(x)) = \mathrm{pres}_{\Delta_2}(y) = v$. Hence $z \in u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v$ and thus we have proven that $x \in (u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v) \,{}_{\Delta_1 \cup \Delta_2}\underline{|||}_{\Delta_3}\, w$. To prove the converse inclusion, it suffices to combine the above with the commutativity of fair fS-shuffling. □

General fS-shuffling, however, does not satisfy this notion of associativity since unfair fS-shuffling with an infinite word may ignore words resulting from fS-shuffling with partners that cannot be used in the other fS-shuffling operation (thus destroying associativity).

Example 3.4. Clearly, $a \,{}_{\{a,b\}}\underline{||}_{\{b\}}\, b = \varnothing$ and thus $(a \,{}_{\{a,b\}}\underline{||}_{\{b\}}\, b) \,{}_{\{a,b\}}\underline{||}_{\{c\}}\, c^\omega = \varnothing$. On the other hand, $a \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, (b \,{}_{\{b\}}\underline{||}_{\{c\}}\, c^\omega) = a \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, (b \,{}_{\{b\}}\underline{||}_{\{c\}}\, c^\omega) = a \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, (\{\, c^n bc^\omega : n \geq 0\,\} \cup \{c^\omega\}) = a \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, \{c^\omega\} = \{\, c^n ac^\omega : n \geq 0\,\} \cup \{c^\omega\}$.

At first sight, adding λ to represent the situation that a word may be ignored, appears to be a solution.

Example 3.5. (Example 3.4 cont.) $(\{\lambda, a\} \,{}_{\{a,b\}}\underline{||}_{\{b\}}\, \{\lambda, b\}) \,{}_{\{a,b\}}\underline{||}_{\{c\}}\, \{\lambda, c^\omega\} = \{\lambda, a\} \,{}_{\{a,b\}}\underline{||}_{\{c\}}\, \{\lambda, c^\omega\} = \{\lambda, a, c^\omega\} \cup \{\, c^n ac^\omega : n \geq 0\,\}$, which equals $\{\lambda, a\} \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, (\{\lambda, b\} \,{}_{\{b\}}\underline{||}_{\{c\}}\, \{\lambda, c^\omega\}) = \{\lambda, a\} \,{}_{\{a,b\}}\underline{||}_{\{b,c\}}\, (\{\lambda, b, c^\omega\} \cup \{\, c^n bc^\omega : n \geq 0\,\})$.

In general, however, unfair fS-shuffles of infinite words may use arbitrary prefixes of the words involved, in which case having only λ available is not enough to guarantee associativity.

Example 3.6. $(\{\lambda, a^\omega\} \,{}_{\{a\}}\underline{||}_{\{b\}}\, \{\lambda, b^2\}) \,{}_{\{a,b\}}\underline{||}_{\{b\}}\, \{\lambda, b\} = \{\lambda, a^\omega\} \cup \{\, a^n ba^\omega : n \geq 0\,\}$, while we have $\{\lambda, a^\omega\} \,{}_{\{a\}}\underline{||}_{\{b\}}\, (\{\lambda, b^2\} \,{}_{\{b\}}\underline{||}_{\{b\}}\, \{\lambda, b\}) = \{\lambda, a^\omega\} \,{}_{\{a\}}\underline{||}_{\{b\}}\, \lambda = \{\lambda, a^\omega\}$. This shows that adding only λ is not sufficient since it may be that a partner is ignored only after a certain prefix. Therefore we consider the case that all prefixes of the words involved in the fS-shuffles are present: $((\{\, a^n : n \geq 0\,\} \cup \{a^\omega\}) \,{}_{\{a\}}\underline{||}_{\{b\}}\, \{\lambda, b, b^2\}) \,{}_{\{a,b\}}\underline{||}_{\{b\}}\, \{\lambda, b\} = (\{\, a^n : n \geq 0\,\} \cup \{a^\omega\}) \,{}_{\{a\}}\underline{||}_{\{b\}}\, \{\lambda, b\} = (\{\, a^n : n \geq 0\,\} \cup \{a^\omega\}) \,{}_{\{a\}}\underline{||}_{\{b\}}\, (\{\lambda, b, b^2\} \,{}_{\{b\}}\underline{||}_{\{b\}}\, \{\lambda, b\})$.

This outcome provides the motivation to set out to prove the associativity of (general, unfair) fS-shuffling at the level of prefix-closed languages. To begin with, we combine Theorem 3.4 and Lemma 3.2 to obtain a description of fS-shuffling in terms of prefixes and preserving homomorphisms.

Lemma 3.6. (1) If $u \in \Delta_1^*$ and $v \in \Delta_2^*$, then $u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^* : \mathrm{pres}_{\Delta_1}(w) = u,\ \mathrm{pres}_{\Delta_2}(w) = v\,\}$.

(2) If $u \in \Delta_1^*$ and $v \in \Delta_2^\omega$, then $u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \exists\, u' \in \mathrm{pref}(u)$ such that $\mathrm{pres}_{\Delta_2}(u') = \mathrm{pres}_{\Delta_2}(u),\ \mathrm{pres}_{\Delta_1}(w) = u',\ \mathrm{pres}_{\Delta_2}(w) = v\,\}$.

(3) If $u \in \Delta_1^\omega$ and $v \in \Delta_2^\omega$, then $u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \mathrm{pres}_{\Delta_1}(w) = u,\ \mathrm{pres}_{\Delta_2}(w) = v\,\} \cup \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \exists\, u' \in \mathrm{pref}(u)$ such that $\mathrm{pres}_{\Delta_2}(u') = \mathrm{pres}_{\Delta_2}(u),\ \mathrm{pres}_{\Delta_1}(w) = u',\ \mathrm{pres}_{\Delta_2}(w) = v\,\} \cup \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \exists\, v' \in \mathrm{pref}(v)$ such that $\mathrm{pres}_{\Delta_1}(v') = \mathrm{pres}_{\Delta_1}(v),\ \mathrm{pres}_{\Delta_1}(w) = u,\ \mathrm{pres}_{\Delta_2}(w) = v'\,\}$.

Proof:
By Definition 3.2, $u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = u \,||^{\Delta_1 \cap \Delta_2}\, v$ and $u \,{}_{\Delta_1}\underline{|||}_{\Delta_2}\, v = u \,|||^{\Delta_1 \cap \Delta_2}\, v$, whenever $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$. Moreover, for $x \in \Delta_1^\infty$, $\mathrm{pres}_{\Delta_1 \cap \Delta_2}(x) = \mathrm{pres}_{\Delta_2}(x)$ and, for $x \in \Delta_2^\infty$, $\mathrm{pres}_{\Delta_1 \cap \Delta_2}(x) = \mathrm{pres}_{\Delta_1}(x)$. The statements now follow by combining Theorem 3.4 and Lemma 3.2, with $\Gamma = \Delta_1 \cap \Delta_2$. □

This description enables us to adapt Theorem 3.4, which we used in Theorem 3.5 to prove that fair fS-shuffling is associative, to a characterization of the fS-shuffles—not of pairs of single words, but rather of pairs of words together with their prefixes.

Lemma 3.7. If $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$, then $(\{u\} \cup \mathrm{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \mathrm{pref}(v)) = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \mathrm{pres}_{\Delta_1}(w) \in \{u\} \cup \mathrm{pref}(u),\ \mathrm{pres}_{\Delta_2}(w) \in \{v\} \cup \mathrm{pref}(v)\,\}$.

Proof:
Let $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$. We distinguish the following three cases.

(1) If u and v are finite, then by Lemma 3.6(1), $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = \{\, w \in (\Delta_1 \cup \Delta_2)^* : \text{pres}_{\Delta_1}(w) \in \{u\} \cup \text{pref}(u),\ \text{pres}_{\Delta_2}(w) \in \{v\} \cup \text{pref}(v) \,\}$.

(2) If $u \in \Delta_1^*$ and $v \in \Delta_2^\omega$, then from $u \in \text{pref}(u)$ follows that $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = \text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = (\text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, \text{pref}(v)) \cup (\text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, \{v\})$. Then, by Lemma 3.6(2), $\text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, \{v\} = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \exists\, u' \in \text{pref}(u),\ u'' \in \text{pref}(u') \text{ such that } \text{pres}_{\Delta_2}(u') = \text{pres}_{\Delta_2}(u''), \text{pres}_{\Delta_1}(w) = u'',\ \text{pres}_{\Delta_2}(w) = v \,\} = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \exists\, u'' \in \text{pref}(u) \text{ such that } \text{pres}_{\Delta_1}(w) = u'', \text{pres}_{\Delta_2}(w) = v \,\}$. Combined with (1), we get $\text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \text{pres}_{\Delta_1}(w) \in \text{pref}(u),\ \text{pres}_{\Delta_2}(w) \in \{v\} \cup \text{pref}(v) \,\}$.

(3) If $u \in \Delta_1^\omega$ and $v \in \Delta_2^\omega$, then $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = L_1 \cup L_2 \cup L_3$, with L_1, L_2, and L_3 as follows. First, $L_1 = \text{pref}(u) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \text{pres}_{\Delta_1}(w) \in \text{pref}(u),\ \text{pres}_{\Delta_2}(w) \in \{v\} \cup \text{pref}(v) \,\}$ by (2). Second, $L_2 = (\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, \text{pref}(v) = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \text{pres}_{\Delta_1}(w) \in \{u\} \cup \text{pref}(u),\ \text{pres}_{\Delta_2}(w) \in \text{pref}(v) \,\}$ by (2) and the commutativity of fS-shuffling. Third, $L_3 = u \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\omega : \text{pres}_{\Delta_1}(w) = u,\ \text{pres}_{\Delta_2}(w) = v \,\} \cup L_1' \cup L_2'$, with $L_1' \subseteq L_1$ and $L_2' \subseteq L_2$, by Lemma 3.6(3).

Consequently, $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v)) = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : \text{pres}_{\Delta_1}(w) \in \{u\} \cup \text{pref}(u),\ \text{pres}_{\Delta_2}(w) \in \{v\} \cup \text{pref}(v) \,\}$. □

This result leads to a description of fS-shuffling that is insensitive to the order of application.

Theorem 3.6. Let $u_i \in \Delta_i^\infty$, for all $1 \leq i \leq 3$. Then $(\{u_1\} \cup \text{pref}(u_1)) \,{}_{\Delta_1}\underline{||}_{\Delta_2\cup\Delta_3}\, ((\{u_2\} \cup \text{pref}(u_2)) \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, (\{u_3\} \cup \text{pref}(u_3))) = \{\, w \in (\Delta_1 \cup \Delta_2 \cup \Delta_3)^\infty : \text{pres}_{\Delta_i}(w) \in \{u_i\} \cup \text{pref}(u_i) \text{ for all } 1 \leq i \leq 3 \,\}$.

Proof:
($\subseteq$) Let $w \in (\{u_1\} \cup \text{pref}(u_1)) \,{}_{\Delta_1}\underline{||}_{\Delta_2\cup\Delta_3}\, ((\{u_2\} \cup \text{pref}(u_2)) \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, (\{u_3\} \cup \text{pref}(u_3)))$. By Lemma 3.7, $\text{pres}_{\Delta_1}(w) \in \{u_1\} \cup \text{pref}(u_1)$ and there exists a $y \in (\{u_2\} \cup \text{pref}(u_2)) \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, (\{u_3\} \cup \text{pref}(u_3))$ such that $\text{pres}_{\Delta_2\cup\Delta_3}(w) = y$. Thus, $\text{pres}_{\Delta_2}(w) = \text{pres}_{\Delta_2}(\text{pres}_{\Delta_2\cup\Delta_3}(w)) = \text{pres}_{\Delta_2}(y)$, which by Lemma 3.7 is included in $\{u_2\} \cup \text{pref}(u_2)$. Likewise, $\text{pres}_{\Delta_3}(w) = \text{pres}_{\Delta_3}(\text{pres}_{\Delta_2\cup\Delta_3}(w)) = \text{pres}_{\Delta_3}(y)$ is included in $\{u_3\} \cup \text{pref}(u_3)$.

($\supseteq$) Let $w \in (\Delta_1 \cup \Delta_2 \cup \Delta_3)^\infty$ be such that $\text{pres}_{\Delta_i}(w) \in \{u_i\} \cup \text{pref}(u_i)$, for all $1 \leq i \leq 3$. Now let $z = \text{pres}_{\Delta_2\cup\Delta_3}(w)$. Hence $\text{pres}_{\Delta_2}(z) = \text{pres}_{\Delta_2}(w)$ and $\text{pres}_{\Delta_3}(z) = \text{pres}_{\Delta_3}(w)$. Consequently, according to Theorem 3.4, $z \in \text{pres}_{\Delta_2}(w) \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, \text{pres}_{\Delta_3}(w)$ and $w \in \text{pres}_{\Delta_1}(w) \,{}_{\Delta_1}\underline{|||}_{\Delta_2\cup\Delta_3}\, z \subseteq \text{pres}_{\Delta_1}(w) \,{}_{\Delta_1}\underline{|||}_{\Delta_2\cup\Delta_3}\, (\text{pres}_{\Delta_2}(w) \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, \text{pres}_{\Delta_3}(w)) \subseteq (\{u_1\} \cup \text{pref}(u_1)) \,{}_{\Delta_1}\underline{|||}_{\Delta_2\cup\Delta_3}\, ((\{u_2\} \cup \text{pref}(u_2)) \,{}_{\Delta_2}\underline{|||}_{\Delta_3}\, (\{u_3\} \cup \text{pref}(u_3))) \subseteq (\{u_1\} \cup \text{pref}(u_1)) \,{}_{\Delta_1}\underline{||}_{\Delta_2\cup\Delta_3}\, ((\{u_2\} \cup \text{pref}(u_2)) \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, (\{u_3\} \cup \text{pref}(u_3)))$. □

The associativity of fS-shuffling when applied to prefix-closed languages now follows immediately.

Theorem 3.7. Let $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in \Delta_3^\infty$. Then $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2\cup\Delta_3}\, ((\{v\} \cup \text{pref}(v)) \,{}_{\Delta_2}\underline{||}_{\Delta_3}\, (\{w\} \cup \text{pref}(w))) = ((\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, (\{v\} \cup \text{pref}(v))) \,{}_{\Delta_1\cup\Delta_2}\underline{||}_{\Delta_3}\, (\{w\} \cup \text{pref}(w))$.

3.3. Relaxed Synchronized Shuffling

A *relaxed S-shuffle* of two words is a synchronized shuffle in which only some of the symbols shared by their alphabets are synchronizing symbols.

Definition 3.3. Let Δ_1 and Δ_2 be two alphabets and let $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in (\Delta_1 \cup \Delta_2)^\infty$. Then w is a *relaxed S-shuffle* (*rS-shuffle* for short) *on* Γ of u and v w.r.t. Δ_1 and Δ_2 if w is an S-shuffle on $\Gamma \cap \Delta_1 \cap \Delta_2$ of u and v. This rS-shuffle on Γ of u and v w.r.t. Δ_1 and Δ_2 is called *fair* if w is a fair S-shuffle on $\Gamma \cap \Delta_1 \cap \Delta_2$ of u and v.

The relaxed synchronized shuffle operation is parametrized by the three alphabets Δ_1, Δ_2, and Γ. The latter determines which symbols common to both alphabets are synchronizing symbols. For $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$, we write $u \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : w \text{ is a fair rS-shuffle on } \Gamma \text{ of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2 \,\}$ and $u \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2}\, v = \{\, w \in (\Delta_1 \cup \Delta_2)^\infty : w \text{ is an rS-shuffle on } \Gamma \text{ of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2 \,\}$. For $L_1 \subseteq \Delta_1^\infty$ and $L_2 \subseteq \Delta_2^\infty$, the *fair rS-shuffle* on Γ of L_1 and L_2 w.r.t. Δ_1 and Δ_2 is $L_1 \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2}\, L_2 = \{\, w \in u \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2}\, v : u \in L_1,\ v \in L_2 \,\}$ and $L_1 \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2}\, L_2 = \{\, w \in u \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2}\, v : u \in L_1,\ v \in L_2 \,\}$ is the *rS-shuffle* on Γ of L_1 and L_2 w.r.t. Δ_1 and Δ_2.

Note that the (fair) rS-shuffles thus defined are (fair) fS-shuffles whenever $\Gamma \supseteq \Delta_1 \cap \Delta_2$. In that case, for all pairs of words (languages) over Δ_1, Δ_2, we have ${}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2} = {}_{\Delta_1}\underline{|||}_{\Delta_2} = |||^{\Delta_1 \cap \Delta_2}$ and ${}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2} = {}_{\Delta_1}\underline{||}_{\Delta_2} = ||^{\Delta_1 \cap \Delta_2}$. In case $\Gamma = \varnothing$, then there are no synchronizing symbols and rS-shuffling degenerates to ordinary shuffling. Hence, for all pairs of words (languages) over Δ_1, Δ_2, we have ${}_{\Delta_1}\underline{|||}^{\varnothing}_{\Delta_2} = |||^{\varnothing} = |||$ and ${}_{\Delta_1}\underline{||}^{\varnothing}_{\Delta_2} = ||^{\varnothing} = ||$.

Like the fS-shuffle, also the (fair) rS-shuffle is commutative. That is, $u \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2}\, v = v \,{}_{\Delta_2}\underline{|||}^{\Gamma}_{\Delta_1}\, u$ and $u \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2}\, v = v \,{}_{\Delta_2}\underline{||}^{\Gamma}_{\Delta_1}\, u$ for all $u \in \Delta_1^\infty$ and $v \in \Delta_2^\infty$. For rS-shuffling also a notion of associativity can be upheld, as we show next. We propose to consider rS-shuffling associative if $u \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2 \cup \Delta_3}\, (v \,{}_{\Delta_2}\underline{||}^{\Gamma}_{\Delta_3}\, w)$ equals $(u \,{}_{\Delta_1}\underline{||}^{\Gamma}_{\Delta_2}\, v) \,{}_{\Delta_1 \cup \Delta_2}\underline{||}^{\Gamma}_{\Delta_3}\, w$ for all words $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in \Delta_3^\infty$, and similarly for the fair case. Indeed, at least in the fair case, rS-shuffling is associative in this sense. The proof of this statement uses the definition of rS-shuffling in terms of S-shuffling, and the fact that fair fS-shuffling is associative.

Theorem 3.8. Let $u \in \Delta_1^\infty$, let $v \in \Delta_2^\infty$, and let $w \in \Delta_3^\infty$. Then $u \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2 \cup \Delta_3}\, (v \,{}_{\Delta_2}\underline{|||}^{\Gamma}_{\Delta_3}\, w) = (u \,{}_{\Delta_1}\underline{|||}^{\Gamma}_{\Delta_2}\, v) \,{}_{\Delta_1 \cup \Delta_2}\underline{|||}^{\Gamma}_{\Delta_3}\, w$.

Proof:
It is immediate from the definition of rS-shuffling that proving $\{u\} \,|||^{\Gamma \cap (\Delta_1 \cap (\Delta_2 \cup \Delta_3))}\, (v \,|||^{\Gamma \cap \Delta_2 \cap \Delta_3}\, w) = (u \,|||^{\Gamma \cap \Delta_1 \cap \Delta_2}\, v) \,|||^{\Gamma \cap (\Delta_1 \cup \Delta_2) \cap \Delta_3}\, \{w\}$ suffices. To this aim, we add an index to all occurrences of symbols in the synchronized shuffles which should help to keep track of their roles. Hence, for each $\ell \in \{[123], [12], [13], [23], [1], [2], [3]\}$, we let $\Delta^\ell = \{\, a_\ell : a \in \Delta \,\}$. Consequently, we consider the homomorphism $\varphi : (\Delta_1 \cup \Delta_2 \cup \Delta_3)^\infty \to (\Delta^{[123]} \cup \Delta^{[12]} \cup \Delta^{[13]} \cup \Delta^{[23]} \cup \Delta^{[1]} \cup \Delta^{[2]} \cup \Delta^{[3]})^\infty$, which labels the symbols in the following way:

$$\varphi(a) = \begin{cases} a_{[123]} & \text{if } a \in \Gamma \cap \Delta_1 \cap \Delta_2 \cap \Delta_3, \\ a_{[12]} & \text{if } a \in (\Gamma \cap \Delta_1 \cap \Delta_2) \setminus \Delta_3, \\ a_{[13]} & \text{if } a \in (\Gamma \cap \Delta_1 \cap \Delta_3) \setminus \Delta_2, \\ a_{[23]} & \text{if } a \in (\Gamma \cap \Delta_2 \cap \Delta_3) \setminus \Delta_1, \\ a_{[1]} & \text{if } a \in (\Delta_1 \setminus \Gamma) \cup ((\Gamma \cap \Delta_1) \setminus (\Delta_2 \cup \Delta_3)), \\ a_{[2]} & \text{if } a \in (\Delta_2 \setminus \Gamma) \cup ((\Gamma \cap \Delta_2) \setminus (\Delta_1 \cup \Delta_3)), \text{ and} \\ a_{[3]} & \text{if } a \in (\Delta_3 \setminus \Gamma) \cup ((\Gamma \cap \Delta_3) \setminus (\Delta_1 \cup \Delta_2)). \end{cases}$$

Now let $\hat{\Delta}_1 = \Delta^{[123]} \cup \Delta^{[12]} \cup \Delta^{[13]} \cup \Delta^{[1]}$. Thus $\hat{\Delta}_1$ comprises the indexed versions of all symbols in Δ_1, with index $[1]$ assigned to those symbols in Δ_1 which are not used for synchronization and the other indices specifying in which combinations the original symbols would be synchronizing. Similarly, we define $\hat{\Delta}_2 = \Delta^{[123]} \cup \Delta^{[12]} \cup \Delta^{[23]} \cup \Delta^{[2]}$ and $\hat{\Delta}_3 = \Delta^{[123]} \cup \Delta^{[13]} \cup \Delta^{[23]} \cup \Delta^{[3]}$. Hence $\varphi(u) \in \hat{\Delta}_1^\infty$, $\varphi(v) \in \hat{\Delta}_2^\infty$ and $\varphi(w) \in \hat{\Delta}_3^\infty$. Moreover, $\varphi(a) \in \hat{\Delta}_1 \cap (\hat{\Delta}_2 \cup \hat{\Delta}_3)$ if and only if $a \in \Gamma \cap (\Delta_1 \cap (\Delta_2 \cup \Delta_3))$ and similarly for the other (potential) synchronizing symbols. Since φ is injective, it thus follows that $u \,|||^{\Gamma \cap (\Delta_1 \cap (\Delta_2 \cup \Delta_3))}\, (v \,|||^{\Gamma \cap \Delta_2 \cap \Delta_3}\, w) = \varphi^{-1}(\varphi(u) \,|||^{\hat{\Delta}_1 \cap (\hat{\Delta}_2 \cup \hat{\Delta}_3)}\, (\varphi(v) \,|||^{\hat{\Delta}_2 \cap \hat{\Delta}_3}\, \varphi(w)))$, which by the associativity of Theorem 3.5 is equal to $\varphi^{-1}((\varphi(u) \,|||^{\hat{\Delta}_1 \cap \hat{\Delta}_2}\, \varphi(v)) \,|||^{(\hat{\Delta}_1 \cup \hat{\Delta}_2) \cap \hat{\Delta}_3}\, \varphi(w))$ and this, once again by the labeling of the alphabets, equals $(u \,|||^{\Gamma \cap \Delta_1 \cap \Delta_2}\, v) \,|||^{\Gamma \cap (\Delta_1 \cup \Delta_2) \cap \Delta_3}\, w$. □

The statement of Theorem 3.8 would not hold if the synchronizing symbols were allowed to vary.

Example 3.7. $(a \,{}_{\{a\}}|||^{\{a\}}_{\{b\}}\, b) \,{}_{\{a,b\}}|||^{\{b\}}_{\{a,b\}}\, ab = \{ab, ba\} \,{}_{\{a,b\}}||^{\{b\}}_{\{a,b\}}\, ab = \{aab, aba\}$, which differs from $a \,{}_{\{a\}}|||^{\{a\}}_{\{a,b\}}\, (b \,{}_{\{b\}}|||^{\{b\}}_{\{a,b\}}\, ab) = a \,{}_{\{a\}}|||^{\{a\}}_{\{a,b\}}\, ab = \{ab\}$.

Exactly as in the case of the fS-shuffle, associativity is not a property of (possibly unfairly) rS-shuffling words. This can again be seen from Example 3.4 by noting that rS-shuffles are fS-shuffles whenever Γ includes the alphabets involved. One can establish however that the rS-shuffling of prefix-closed languages is an associative operation, by adapting the proof of Theorem 3.6 which underlies Theorem 3.7 (associativity of fS-shuffling) using the same renaming strategy as applied in the proof of Theorem 3.8.

Theorem 3.9. Let $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$, and $w \in \Delta_3^\infty$. Then $(\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}||^{\Gamma}_{\Delta_2 \cup \Delta_3}\, ((\{v\} \cup \text{pref}(v)) \,{}_{\Delta_2}||^{\Gamma}_{\Delta_3}\, (\{w\} \cup \text{pref}(w))) = ((\{u\} \cup \text{pref}(u)) \,{}_{\Delta_1}||^{\Gamma}_{\Delta_2}\, (\{v\} \cup \text{pref}(v))) \,{}_{\Delta_1 \cup \Delta_2}||^{\Gamma}_{\Delta_3}\, (\{w\} \cup \text{pref}(w))$.

3.4. Arbitrary Synchronized Shuffling

An *arbitrary S-shuffle* of two words is a synchonized shuffle in which synchronization on occurrences of the specified synchronizing symbols is an option rather than a requirement.

Definition 3.4. Let $u, v \in \Delta^\infty$. Then a word $w \in \Delta^\infty$ is an *arbitrary S-shuffle* (*aS-shuffle* for short) *on* Γ of u and v if

(1) either $u = u_1 x_1 u_2 x_2 \cdots x_{n-1} u_n$ and $v = v_1 x_1 v_2 x_2 \cdots x_{n-1} v_n$, with $u_i, v_i \in \Delta^*$ and $x_i \in \Gamma$, for $1 \leq i \leq n-1$, and $u_n, v_n \in \Delta^\infty$, in which case $w = w_1 x_1 w_2 x_2 \cdots x_{n-1} w_n$ with $w_i \in u_i \,||\, v_i$ for all $1 \leq i \leq n$,

(2) or $u = u_1 x_1 u_2 x_2 \cdots$ and $v = v_1 x_1 v_2 x_2 \cdots$, with $u_i, v_i \in \Delta^*$ and $x_i \in \Gamma$, for all $i \geq 1$, in which case $w = w_1 x_1 w_2 x_2 \cdots$ with $w_i \in u_i \,||\, v_i$ for all $i \geq 1$.

In case (2) w is a *fair* aS-shuffle. Moreover, in case (1) w is *fair* whenever $w_n \in (u_n \,|||\, v_n)$.

The arbitrary synchronized shuffle operation varies with the alphabet Γ that determines the synchronizing symbols. Given $u, v \in \Delta^\infty$, we write $u \wr\wr^\Gamma v = \{\, w \in \Delta^\infty : w \text{ is a fair aS-shuffle on } \Gamma \text{ of } u \text{ and } v \,\}$ and $u \wr^\Gamma v = \{\, w \in \Delta^\infty : w \text{ is an aS-shuffle on } \Gamma \text{ of } u \text{ and } v \,\}$. For $L_1, L_2 \subseteq \Delta^\infty$, the *fair aS-shuffle*

on Γ of L_1 and L_2 is $L_1 \,\text{⧢}^{\Gamma}\, L_2 = \{\, w \in u \,\text{⧢}^{\Gamma}\, v : u \in L_1,\ v \in L_2 \,\}$ and $L_1 \,\text{⫴}^{\Gamma}\, L_2 = \{\, w \in u \,\text{⫴}^{\Gamma}\, v : u \in L_1,\ v \in L_2 \,\}$ is the *aS-shuffle* on Γ of L_1 and L_2.

Comparing the above definition with Definition 3.1, we see that the subwords u_i and v_i may have occurrences of synchronizing symbols (from Γ) which are thus not used for synchronization. In particular, $u \,||^{\Gamma}\, v \subseteq u \,\text{⫴}^{\Gamma}\, v$ always holds, but u, v, and their aS-shuffles may have different Γ-backbones and $u \,\text{⧢}^{\Gamma}\, v$ and $u \,\text{⫴}^{\Gamma}\, v$ are never empty. Similar to the S-shuffle, also the aS-shuffle degenerates to the shuffle if there are no symbols to synchronize on: $\text{⧢}^{\varnothing} = |||$ and $\text{⫴}^{\varnothing} = ||$.

As for S-shuffling, it follows from the commutativity of shuffling that (fair) aS-shuffling is also commutative. Moreover (fair) aS-shuffling is associative, as we demonstrate next.

Theorem 3.10. Let $u, v, w \in \Delta^{\infty}$. Then $u \,\text{⧢}^{\Gamma}\, (v \,\text{⧢}^{\Gamma}\, w) = (u \,\text{⧢}^{\Gamma}\, v) \,\text{⧢}^{\Gamma}\, w$ and $u \,\text{⫴}^{\Gamma}\, (v \,\text{⫴}^{\Gamma}\, w) = (u \,\text{⫴}^{\Gamma}\, v) \,\text{⫴}^{\Gamma}\, w$.

Proof:
We only prove the first equality. The other equality can then be proven in the same way.

Similar to the technique applied in the proof of Theorem 3.8, we add markers to symbols in order to keep track of the origin and role of each occurrence in an aS-shuffle. Hence, for each $\ell \in \{[123], [12], [13], [23]\}$, we let $\Gamma^{\ell} = \{\, a_{\ell} : a \in \Gamma \cap \Delta \,\}$ be a new alphabet and we write $\hat{\Gamma} = \Gamma^{[123]} \cup \Gamma^{[12]} \cup \Gamma^{[13]} \cup \Gamma^{[23]}$. Consequently, we consider the homomorphisms

$$\begin{aligned}
\varphi_1 &: (\Delta \cup (\Gamma^{[123]} \cup \Gamma^{[12]} \cup \Gamma^{[13]}))^{\infty} \to \Delta^{\infty}, \\
\varphi_2 &: (\Delta \cup (\Gamma^{[123]} \cup \Gamma^{[12]} \cup \Gamma^{[23]}))^{\infty} \to \Delta^{\infty}, \text{ and} \\
\varphi_3 &: (\Delta \cup (\Gamma^{[123]} \cup \Gamma^{[13]} \cup \Gamma^{[23]}))^{\infty} \to \Delta^{\infty}, \text{ defined by} \\
\varphi_j(a) &= a \text{ for all } a \in \Delta \text{ and } 1 \leq j \leq 3, \text{ and} \\
\varphi_j(a_{\ell}) &= a \text{ for all } a \in \Gamma \cap \Delta,\ \ell \in \{[123], [12], [13], [23]\}, \text{ and } 1 \leq j \leq 3.
\end{aligned}$$

Consider $u \,\text{⧢}^{\Gamma}\, (v \,\text{⧢}^{\Gamma}\, w)$. Then we can use the inverses of φ_1, φ_2, and φ_3 to choose for every occurrence of a symbol from Γ in u, v, and w, respectively, whether it will synchronize with a counterpart from one or both of the other words or not synchronize at all. It is consequently immediate that $u \,\text{⧢}^{\Gamma}\, (v \,\text{⧢}^{\Gamma}\, w) = \varphi(\varphi_1^{-1}(u) \,|||^{\hat{\Gamma}}\, (\varphi_2^{-1}(v) \,|||^{\hat{\Gamma}}\, \varphi_3^{-1}(w)))$, which—by the associativity of Theorem 3.5—is equal to $(\varphi(\varphi_1^{-1}(u) \,|||^{\hat{\Gamma}}\, \varphi_2^{-1}(v)) \,|||^{\hat{\Gamma}}\, \varphi_3^{-1}(w))$, which in its turn is equal to $(u \,\text{⧢}^{\Gamma}\, v) \,\text{⧢}^{\Gamma}\, w$. □

The statement of Theorem 3.10 would not hold if Γ, the set of synchronizing symbols, were not fixed.

Example 3.8. $cab \in ((ca \,\text{⫴}^{\{a\}}\, ab) \,\text{⫴}^{\{c\}}\, c)$, whereas all words in $(ca \,\text{⫴}^{\{a\}}\, (ab \,\text{⫴}^{\{c\}}\, c))$ contain two occurrences of c.

3.5. Weak Synchronized Shuffling

A *weak S-shuffle* of two words is a synchronized shuffle in which synchronization on occurrences of the specified synchronizing symbols is required, but only when it is possible to do so.

Definition 3.5. Let $u, v \in \Delta^{\infty}$. Then a word $w \in \Delta^{\infty}$ is a *weak S-shuffle* (*wS-shuffle* for short) *on* Γ of u and v if

(1) either $u = u_1x_1u_2x_2\cdots x_{n-1}u_n$ and $v = v_1x_1v_2x_2\cdots x_{n-1}v_n$, with $u_i, v_i \in \Delta^*$, $\mathrm{alph}(u_i) \cap \mathrm{alph}(v_i) \cap \Gamma = \varnothing$ and $x_i \in \Gamma$ for $1 \leq i \leq n-1$, and $u_n, v_n \in \Delta^\infty$, in which case $w = w_1x_1w_2x_2\cdots x_{n-1}w_n$ with $w_i \in u_i \,||\, v_i$ for all $1 \leq i \leq n$,

(2) or $u = u_1x_1u_2x_2\cdots$ and $v = v_1x_1v_2x_2\cdots$, with $u_i, v_i \in \Delta^*$, $\mathrm{alph}(u_i) \cap \mathrm{alph}(v_i) \cap \Gamma = \varnothing$ and $x_i \in \Gamma$, for all $i \geq 1$, in which case $w = w_1x_1w_2x_2\cdots$ with $w_i \in u_i \,||\, v_i$ for all $i \geq 1$.

In case (2) w is a *fair* wS-shuffle. Moreover, in case (1) w is *fair* whenever $w_n \in (u_n \,|||\, v_n)$.

The weak synchronized shuffle operation is again parametrized by the alphabet Γ that determines the synchronizing symbols. Given $u, v \in \Delta^\infty$, we write $u \underset{\sim}{||}^\Gamma v = \{\, w \in \Delta^\infty : w \text{ is a wS-shuffle on } \Gamma \text{ of } u \text{ and } v \,\}$ and $u \underset{\sim}{|||}^\Gamma v = \{\, w \in \Delta^\infty : w \text{ is a fair wS-shuffle on } \Gamma \text{ of } u \text{ and } v \,\}$. For $L_1, L_2 \subseteq \Delta^\infty$, the *fair wS-shuffle* on Γ of L_1 and L_2 is $L_1 \underset{\sim}{|||}^\Gamma L_2 = \{\, w \in u \underset{\sim}{|||}^\Gamma v : u \in L_1,\ v \in L_2 \,\}$ and $L_1 \underset{\sim}{||}^\Gamma L_2 = \{\, w \in u \underset{\sim}{||}^\Gamma v : u \in L_1,\ v \in L_2 \,\}$ is the *wS-shuffle* on Γ of L_1 and L_2.

Comparing the above definition with Definitions 3.1 and 3.4, we see that not all occurrences of synchronizing symbols (from Γ) in the subwords u_i and v_i have to be used for synchronization. In particular, $u \,||^\Gamma v \subseteq u \underset{\sim}{||}^\Gamma v \subseteq u \wr\!\wr^\Gamma v$ always holds, but u, v, and their wS-shuffles may have different Γ-backbones and $u \underset{\sim}{|||}^\Gamma v$ and $u \underset{\sim}{||}^\Gamma v$ are never empty. Similar to the S-shuffle, also the wS-shuffle degenerates to the shuffle in the absence of symbols to synchronize on: $\underset{\sim}{|||}^\varnothing = |||$ and $\underset{\sim}{||}^\varnothing = ||$.

Example 3.9. $abc \underset{\sim}{||}^{\{c\}} cd = \{abcd\}$. Moreover, $a^\omega \underset{\sim}{||}^{\{a\}} a = a^\omega \underset{\sim}{|||}^{\{a\}} a = a^\omega \,||\, a = a^\omega \,|||\, a = a^\omega$, while $a^\omega \,||^{\{a\}} a = a^\omega \,|||^{\{a\}} a = \varnothing$.

As for S-shuffling, it follows from the commutativity of shuffling that (fair) wS-shuffling is also commutative. However, contrary to the previously considered shuffles, (fair) wS-shuffling is *not* associative, as is demonstrated by the following example.

Example 3.10. $(ab \underset{\sim}{||}^{\{a,b\}} (ba \underset{\sim}{||}^{\{a,b\}} ba)) = ab \underset{\sim}{||}^{\{a,b\}} ba = \{aba, bab\}$ and $((ab \underset{\sim}{||}^{\{a,b\}} ba) \underset{\sim}{||}^{\{a,b\}} ba) = \{aba, bab\} \underset{\sim}{||}^{\{a,b\}} ba = \{aba, bab, baba\}$.

To conclude this section, as the next example shows the synchronized shuffle operation and its four variants considered in this section may yield languages lying strictly within each other.

Example 3.11. Let $\Delta_1 = \{a, b\}$, $\Delta_2 = \{a, b, c\}$, $\Gamma = \{a, c\}$, $K = \{aba\}$, and $L = \{aa, aac, aba\}$. Then $K \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, L = \{aba\}$, $K \,||^\Gamma L = (K \,{}_{\Delta_1}\underline{||}_{\Delta_2}\, L) \cup \{abba\}$, $K \,{}_{\Delta_1}\underline{||}^\Gamma_{\Delta_2}\, L = (K \,||^\Gamma L) \cup \{abac\}$, $K \underset{\sim}{||}^\Gamma L = (K \,{}_{\Delta_1}\underline{||}^\Gamma_{\Delta_2}\, L) \cup \{abaa, aaba, abaac, aabac, ababa\}$ and $K \wr\!\wr^\Gamma L = (K \underset{\sim}{||}^\Gamma L) \cup (K \,||\, L) \cup \{abbaa, aabba, abaca, aacba, aabca\}$, where $K \,||\, L = \{abaaa, aabaa, aaaba, abaaac, abaaca, aabaca, aabaac, aaabac, aaabca, aaacba, aacaba, aababa, aabbaa, abaaba, ababaa\}$.

4. Team Automata

In this section we will apply the associativity results obtained in the previous section to team automata. In particular, we consider the strategies to compose a team automaton from component automata which led us to consider the different synchronized shuffle operations just discussed.

Definition 4.1. A *(component) automaton* $\mathcal{C}$ is a quadruple (Q, Σ, δ, I), with finite set Q of *states*, set Σ of *actions* (such that $Q \cap \Sigma = \varnothing$), set $\delta \subseteq Q \times \Sigma \times Q$ of *transitions*, and set $I \subseteq Q$ of *initial states*.

Let a be an action of $\mathcal{C}$ specified as above. Each pair (q, q') such that $(q, a, q') \in \delta$ is an *a-transition* of $\mathcal{C}$ and an a-transition of the form (q, q) is a *loop* (on a). Furthermore, $\delta_a = \{ (q, q') : (q, a, q') \in \delta \}$ is the set of all a-transitions of $\mathcal{C}$.

In contrast to the usual definition [4, 5], in this definition no distinction is made between different types of actions (input, output, internal) as that is irrelevant to our discussion here. Component automata do not have separately specified final or accepting states; their computations are viewed as ongoing runs.

With each automaton $\mathcal{C} = (Q, \Sigma, \delta, I)$ we associate sequential *computations* defined as those finite or infinite sequences $q_0 a_1 q_1 a_2 \cdots$ such that $q_0 \in I$ and $(q_{i-1}, a_i, q_i) \in \delta$ for all $i \geq 1$. We write $\mathbf{C}_\mathcal{C}$ to denote the set of all computations of $\mathcal{C}$. The *behavior* of $\mathcal{C}$, denoted by $\mathbf{B}^\Sigma_\mathcal{C}$, is defined as comprising all sequences of actions that can be executed starting from the initial state. Formally, $\mathbf{B}^\Sigma_\mathcal{C} = \mathrm{pres}_\Sigma(\mathbf{C}_\mathcal{C})$.

Team automata are composed from component automata. They have as their actions the actions of these component automata and their states are the Cartesian products of the states of the components.

We use the following notation: Let $\mathcal{I} \subseteq \mathbb{N}$ be a finite set of indices $\{i_1, i_2, \ldots, i_n\}$ with $i_j < i_\ell$ if $1 \leq j < \ell \leq n$ and let V_i be a set, for each $i \in \mathcal{I}$. Then $\prod_{i\in\mathcal{I}} V_i$ denotes the Cartesian product $\{ (v_{i_1}, v_{i_2}, \ldots, v_{i_n}) : v_{i_j} \in V_{i_j},\ 1 \leq j \leq n \}$. If $\mathcal{I} = \varnothing$, then $\prod_{i\in\mathcal{I}} V_i = \varnothing$. For $j \in \mathcal{I}$ we have $\mathrm{proj}_{\mathcal{I},j} : \prod_{i\in\mathcal{I}} V_i \to V_j$ as the projection defined by $\mathrm{proj}_{\mathcal{I},j}((a_{i_1}, a_{i_2}, \ldots, a_{i_n})) = a_j$. In addition, $\mathrm{proj}_{\mathcal{I},j}{}^{[2]}(a, b) = (\mathrm{proj}_{\mathcal{I},j}(a), \mathrm{proj}_{\mathcal{I},j}(b))$ for all $a, b \in \prod_{i\in\mathcal{I}} V_i$. We may omit the subscript $\mathcal{I}$ whenever it is clear from the context.

The set of transitions of a team automaton is not completely determined by its components. It is defined (one could say user-defined) by choosing certain synchronizations of component transitions, while excluding others. For the sequel, we let $\mathcal{S} = \{ \mathcal{C}_i : i \in \mathcal{I} \}$ be an arbitrary but fixed set of component automata specified as $\mathcal{C}_i = (Q_i, \Sigma_i, \delta_i, I_i)$ and we let $\Sigma = \bigcup_{i\in\mathcal{I}} \Sigma_i$.

Definition 4.2. Let $a \in \Sigma$. The set $\Delta_a(\mathcal{S})$ of *synchronizations* on a is defined as $\Delta_a(\mathcal{S}) = \{ (q, q') \in \prod_{i\in\mathcal{I}} Q_i \times \prod_{i\in\mathcal{I}} Q_i : (\exists\, j \in \mathcal{I} : \mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}) \wedge (\forall\, i \in \mathcal{I} : (\mathrm{proj}_i{}^{[2]}(q, q') \in \delta_{i,a}) \vee (\mathrm{proj}_i(q) = \mathrm{proj}_i(q'))) \}$.

Hence $\Delta_a(\mathcal{S})$ contains *all* possible combinations of a-transitions of the components constituting $\mathcal{S}$, with all non-participating components remaining idle. It is explicitly required that in every synchronization at least one component participates. When defining a team automaton over $\mathcal{S}$, one chooses, for each action a, a subset of $\Delta_a(\mathcal{S})$.

Definition 4.3. A *team automaton* $\mathcal{T}$ over $\mathcal{S}$ is a quadruple (Q, Σ, δ, I), with $Q = \prod_{i\in\mathcal{I}} Q_i$, $\Sigma = \bigcup_{i\in\mathcal{I}} \Sigma_i$, $\delta \subseteq Q \times \Sigma \times Q$ such that $\delta_a = \{ (q, q') : (q, a, q') \in \delta \} \subseteq \Delta_a(\mathcal{S})$ for all $a \in \Sigma$, and $I = \prod_{i\in\mathcal{I}} I_i$.

Note that every team automaton is again a component automaton that in its turn can be used in the definition of a higher-level team automaton.

4.1. Synchronization Patterns

According to Definition 4.3, the synchronizations taking place among the components of a team automaton can be defined explicitly. One may also follow a fixed strategy or pattern (for each action) to

determine the cooperation among components. Such a pattern could specify—per action a—conditions that determine a unique subset of $\Delta_a(\mathcal{S})$ as the set of a-transitions of the team.

Definition 4.4. Let $\mathcal{R}_a(\mathcal{S}) \subseteq \Delta_a(\mathcal{S})$, for all $a \in \Sigma$, and let $\mathcal{R}_\Sigma = \{\, \mathcal{R}_a(\mathcal{S}) \mid a \in \Sigma \,\}$. Then (Q, Σ, δ, q_0) is the $\mathcal{R}_\Sigma$*-team automaton* over $\mathcal{S}$ if $\delta_a = \mathcal{R}_a(\mathcal{S})$ for all $a \in \Sigma$. We may omit Σ if no confusion can arise.

We now formalize four fixed synchronization patterns, based on the way in which execution of an action is supposed to be shared among the components. An action a is *free* in team automaton $\mathcal{T}$ if none of its a-transitions is brought about by a synchronization on a by two or more components from $\mathcal{S}$, action a is *ai* (short for action-indispensable) in $\mathcal{T}$ if all its a-transitions are brought about by a synchronization of all components from $\mathcal{S}$ sharing a, and, finally, action a is *si* (short for state-indispensable) in $\mathcal{T}$ if all its a-transitions are brought about by a synchronization of all components from $\mathcal{S}$ in which a is currently enabled. (An action a is said to be *enabled* at state q in an automaton $\mathcal{C}$, denoted by $a \ \mathrm{en}_{\mathcal{C}}\ q$, if there exists a state q' such that (q, q') is an a-transition of $\mathcal{C}$.) Note that actions that are not shared among (at least two) component automata, are always *free* and *ai* and *si* in any team automaton over these component automata. In addition, we may impose no restrictions whatsoever on the synchronizations on an action. In the next definition, for each of these four types of requirements, we specify the corresponding *synchronization patterns* $\mathcal{R}^{free}$, $\mathcal{R}^{ai}$, $\mathcal{R}^{si}$, and $\mathcal{R}^{no}$ for each action a as subsets of $\Delta_a(\mathcal{S})$.

Definition 4.5. Let $a \in \Sigma$.

- $\mathcal{R}_a^{free}(\mathcal{S}) = \{\, (q, q') \in \Delta_a(\mathcal{S}) : \#\{\, i \in \mathcal{I} : a \in \Sigma_i \wedge \mathrm{proj}_i{}^{[2]}(q, q') \in \delta_{i,a} \,\} = 1 \,\}$,[1]
- $\mathcal{R}_a^{ai}(\mathcal{S}) = \{\, (q, q') \in \Delta_a(\mathcal{S}) : a \in \Sigma_i \Rightarrow \mathrm{proj}_i{}^{[2]}(q, q') \in \delta_{i,a} \text{ for all } i \in \mathcal{I} \,\}$,
- $\mathcal{R}_a^{si}(\mathcal{S}) = \{\, (q, q') \in \Delta_a(\mathcal{S}) : (a \in \Sigma_i \wedge a \ \mathrm{en}_{\mathcal{C}_i}\ q) \Rightarrow \mathrm{proj}_i{}^{[2]}(q, q') \in \delta_{i,a} \text{ for all } i \in \mathcal{I} \,\}$, and
- $\mathcal{R}_a^{no}(\mathcal{S}) = \Delta_a(\mathcal{S})$.

Example 4.1. Consider the component automata $\mathcal{C}$, $\mathcal{P}$, and $\mathcal{S}$, depicted in Figure 1.

Automaton $\mathcal{T}^{free}$ in Figure 2(a) is the $\mathcal{R}^{free}$-team automaton over $\{\mathcal{C}, \mathcal{P}\}$. Note that $\mathcal{T}^{free}$ is also the $(\mathcal{R}^{free}_{\{\mathrm{q,r,f,a}\}} \cup \mathcal{R}^{no}_{\{\mathrm{w}\}})$-team automaton over $\{\mathcal{C}, \mathcal{P}\}$. Automaton $\mathcal{T}_1^{ai}$ in Figure 2(b) is the $\mathcal{R}^{ai}$-team automaton over $\{\mathcal{C}, \mathcal{P}\}$ and also the $(\mathcal{R}^{ai}_{\{\mathrm{q,a}\}} \cup \mathcal{R}^{free}_{\{\mathrm{w,f,r}\}})$-team automaton over $\{\mathcal{C}, \mathcal{P}\}$. Automaton $\mathcal{T}^{no}$ in Figure 2(c) is the $\mathcal{R}^{no}$-team automaton over $\{\mathcal{C}, \mathcal{P}\}$.

Figure 3(a) depicts the reachable part of the $\mathcal{R}^{ai}$-team automaton $\mathcal{T}^{ai}$ over $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$, which is the same as the $(\mathcal{R}^{ai}_{\{\mathrm{q,a,f,r}\}} \cup \mathcal{R}^{free}_{\{\mathrm{w,c}\}})$-team automaton over $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$.

Finally, automaton $\mathcal{T}^{ccp}$ over $\{\mathcal{C}, \mathcal{C}, \mathcal{P}\}$ depicted in Figure 3(b) is an example of a team automaton that cannot be defined as a combination of the synchronization patterns defined in Definition 4.5: While actions w, f, and r are *si* in $\mathcal{T}^{ccp}$, the actions q and a are neither *free* nor *ai* nor *si* in $\mathcal{T}^{ccp}$. Moreover, both the set of q-transitions in $\mathcal{T}^{ccp}$ and the set of a-transitions in $\mathcal{T}^{ccp}$ are constrained (in the sense that more transitions could have been included).

[1] We denote the cardinality of a finite set V by $\#V$.

4.2. Compositionality

A team automaton over $\mathcal{S}$ is said to *satisfy compositionality* if its behavior can be described in terms of that of its component automata: There exists a set-theoretic operation that when applied to the behaviors of the component automata in $\mathcal{S}$, yields the behavior of the team automaton. On a more general level, compositionality is a property related to the strategy underlying the construction of the team automaton, in the sense that the chosen strategy corresponds to an *associative* language-theoretic operation which admits to obtain the behavior of each team from the behaviors of its components.

In [6], it is proven that $\mathcal{R}^{si}$-team automata do not satisfy compositionality.

Proposition 4.1. [6] Let $\mathcal{I} = \{1, 2\}$ and let $\mathcal{T}$ be the $\mathcal{R}^{si}$-team automaton over $\mathcal{S}$. Then there exists no set-theoretic operation $|$ on languages such that $\mathbf{B}^{\Sigma}_{\mathcal{T}} = \mathbf{B}^{\Sigma_1}_{\mathcal{C}_1} \mid \mathbf{B}^{\Sigma_2}_{\mathcal{C}_2}$.

In [7], it is shown that when the behavior is restricted to finite sequences, constructing team automata according to the synchronization patterns $\mathcal{R}^{free}$ and $\mathcal{R}^{ai}$ guarantees compositionality. The results of the previous sections on the associativity of infinite synchronized shuffling make it possible to extend these results to the full (non-finitary) behaviors of team automata.

Before we do so, we have to clarify one feature of team automata which may lead to ambiguities in how we should view their behavior.

Definition 4.6. Let Θ be an alphabet disjoint from $\bigcup_{i\in\mathcal{I}} Q_i$. Then $\mathcal{S}$ is Θ-*loop-limited* if for all $i \in \mathcal{I}$ and for all $a \in \Theta \cap \Sigma_i$, whenever $(q, q) \in \delta_{i,a}$ for some $q \in Q_i$, then for all $\ell \in (\mathcal{I} \setminus \{i\})$, $\delta_{\ell,a} = \varnothing$.

$\mathcal{S}$ is also said to be loop-limited if it is Σ-loop-limited.

Example 4.2. Consider again the component automata $\mathcal{C}$, $\mathcal{P}$, and $\mathcal{S}$, depicted in Figure 1. The set $\{\mathcal{C}, \mathcal{P}\}$ is loop-limited: The only loop in these two automata is (p_1, w, p_1) in $\mathcal{C}$ and indeed the set of w-transitions of $\mathcal{P}$ is empty. The set $\{\mathcal{C}, \mathcal{C}, \mathcal{P}\}$ is not loop-limited. As a consequence, we do not know which of the $\mathcal{C}$lients is actually at **w**ork in the computation $(p_1, p_1, s_1)\mathrm{w}(p_1, p_1, s_1)$ of team automaton $\mathcal{T}^{ccp}$ in Figure 3(b): Only one or both of them?

If the set $\mathcal{S}$ is not loop-limited for some actions, then it may happen that there are team transitions in which it is not clear how many and which of the components participate. This clearly is important for the definition of the operations combining component behaviors (see [4, 7] for more technical details). Now we are ready to state the results relating synchronization patterns and synchronized shuffles. For the sequel, let $\mathcal{I} = \{1, 2, \ldots, n\}$.

Since shuffling is associative (Proposition 2.4), we can omit parentheses. Moreover, for languages L_i, with $i \in \mathcal{I}$, we set $||_{i\in\mathcal{I}}\, L_i = L_1 \,||\, L_2 \,||\, \cdots \,||\, L_n$.

Theorem 4.1. Let $\mathcal{T}$ be the $\mathcal{R}^{free}$-team automaton over $\mathcal{S}$. If $\mathcal{S}$ is loop-limited, then $\mathbf{B}^{\Sigma}_{\mathcal{T}} = ||_{i\in\mathcal{I}}\, \mathbf{B}^{\Sigma_i}_{\mathcal{C}_i}$.

Example 4.3. (Examples 4.1 and 4.2 cont.) $\{\mathcal{C}, \mathcal{P}\}$ is loop-limited and indeed behavior $\mathbf{B}^{\{\mathrm{w,q,r,f,a}\}}_{\mathcal{T}^{free}}$ of the $\mathcal{R}^{free}$-team automaton $\mathcal{T}^{free}$ over $\{\mathcal{C},\mathcal{P}\}$ equals shuffling its components' behaviors: $\mathbf{B}^{\{\mathrm{w,q,a}\}}_{\mathcal{C}} \,||\, \mathbf{B}^{\{\mathrm{q,r,f,a}\}}_{\mathcal{P}}$.

Since the full synchronized shuffle is an associative operation on prefix-closed languages (Theorem 3.7), we can adopt a left-associative normal form. Moreover, for languages L_i, with $i \in \mathcal{I}$, we

write $\underline{||}_{\{\Sigma_i : i \in \mathcal{I}\}} L_i$ instead of $(\underline{||}_{\{\Sigma_i : i \in (\mathcal{I} \setminus \{n\})\}} L_i)$ $\underline{||}_{\cup_{i \in (\mathcal{I} \setminus \{n\})} \Sigma_i \cap \Sigma_n} L_n$. Since the $\mathcal{R}^{ai}$ synchronization pattern requires participation of all components sharing the action considered, it is clear that loops have no awkward consequences for the description of the behavior.

Theorem 4.2. Let $\mathcal{T}$ be the $\mathcal{R}^{ai}$-team automaton over $\mathcal{S}$. Then $\mathbf{B}^{\Sigma}_{\mathcal{T}} = \underline{||}_{\{\Sigma_i : i \in \mathcal{I}\}} \mathbf{B}^{\Sigma_i}_{\mathcal{C}_i}$.

Example 4.4. (Example 4.1 cont.) Behavior $\mathbf{B}^{\{\mathrm{w,q,r,f,a,c}\}}_{\mathcal{T}_2^{ai}}$ of the $\mathcal{R}^{ai}$-team automaton $\mathcal{T}_2^{ai}$ over $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$ equals fS-shuffling its components' behaviors: $(\mathbf{B}^{\{\mathrm{w,q,a}\}}_{\mathcal{C}} \underline{||}_{\{\mathrm{w,q,a}\}} \mathbf{B}^{\{\mathrm{q,r,f,a}\}}_{\mathcal{P}})$ $\underline{||}_{\{\mathrm{w,q,r,f,a}\}} \mathbf{B}^{\{\mathrm{f,r,c}\}}_{\mathcal{S}}$.

Just as we did above for the full synchronized shuffle, also in case of the relaxed synchronized shuffle (but now using Theorem 3.9) we can write $\underline{||}^{\Gamma}_{\{\Sigma_i : i \in \mathcal{I}\}} L_i$ for languages L_i, with $i \in \mathcal{I}$, instead of $(\underline{||}^{\Gamma}_{\{\Sigma_i : i \in (\mathcal{I} \setminus \{n\})\}} L_i)$ $\underline{||}^{\Gamma}_{\cup_{i \in (\mathcal{I} \setminus \{n\})} \Sigma_i \cap \Sigma_n} L_n$.

Theorem 4.3. Let $\mathcal{T}$ be the $(\mathcal{R}^{ai}_{\Gamma} \cup \mathcal{R}^{free}_{\Sigma \setminus \Gamma})$-team automaton over $\mathcal{S}$. If $\mathcal{S}$ is $(\Sigma \setminus \Gamma)$-loop-limited, then $\mathbf{B}^{\Sigma}_{\mathcal{T}} = \underline{||}^{\Gamma}_{\{\Sigma_i : i \in \mathcal{I}\}} \mathbf{B}^{\Sigma_i}_{\mathcal{C}_i}$.

Example 4.5. (Example 4.1 cont.) It is immediate that $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$ is $\{\mathrm{w,c}\}$-loop-limited and indeed behavior $\mathbf{B}^{\{\mathrm{w,q,r,f,a,c}\}}_{\mathcal{T}_2^{ai}}$ of the $(\mathcal{R}^{ai}_{\{\mathrm{q,r,f,a}\}} \cup \mathcal{R}^{free}_{\{\mathrm{w,c}\}})$-team automaton $\mathcal{T}_2^{ai}$ over $\{\mathcal{C}, \mathcal{P}, \mathcal{S}\}$ equals rS-shuffling its components' behaviors: $(\mathbf{B}^{\{\mathrm{w,q,a}\}}_{\mathcal{C}} \underline{||}^{\{\mathrm{q,r,f,a}\}}_{\{\mathrm{w,q,a}\}} \mathbf{B}^{\{\mathrm{q,r,f,a}\}}_{\mathcal{P}})$ $\underline{||}^{\{\mathrm{q,r,f,a}\}}_{\{\mathrm{w,q,r,f,a}\}} \mathbf{B}^{\{\mathrm{f,r,c}\}}_{\mathcal{S}}$.

Finally, since the arbitrary synchronized shuffle operation allows—but does not require!—all symbols of a specified set to be synchronized, it is clear that it can be used to prove that also $\mathcal{R}^{no}$-team automata satisfy compositionality: It suffices to include all actions of the component automata in the specified set. With a reasoning like above for the ordinary shuffle, but now referring to Theorem 3.10, we can write $\wr\wr^{\Gamma}_{i \in \mathcal{I}} L_i$ for $L_1 \wr\wr^{\Gamma} L_2 \wr\wr^{\Gamma} \cdots \wr\wr^{\Gamma} L_n$ as the aS-shuffle on Γ of languages L_i, with $i \in \mathcal{I}$.

Theorem 4.4. Let $\mathcal{T}$ be the $\mathcal{R}^{no}$-team automaton over $\mathcal{S}$. Then $\mathbf{B}^{\Sigma}_{\mathcal{T}} = \wr\wr^{\Sigma}_{i \in \mathcal{I}} \mathbf{B}^{\Sigma_i}_{\mathcal{C}_i}$.

Example 4.6. (Example 4.1 cont.) Behavior $\mathbf{B}^{\{\mathrm{w,q,r,f,a}\}}_{\mathcal{T}^{no}}$ of the $\mathcal{R}^{no}$-team automaton $\mathcal{T}^{no}$ over $\{\mathcal{C}, \mathcal{P}\}$ equals aS-shuffling its components' behaviors: $\mathbf{B}^{\{\mathrm{w,q,a}\}}_{\mathcal{C}} \wr\wr^{\{\mathrm{w,q,r,f,a}\}} \mathbf{B}^{\{\mathrm{q,r,f,a}\}}_{\mathcal{P}}$.

5. Conclusion

Shuffling and synchronized shuffling are rather well-known operations that in many variants have been extensively studied, sometimes out of plain mathematical interest, other times because of their role in defining a semantics for distributed systems. When such systems may be composed iteratively, the modularity of the chosen semantics is important. For team automata, this means that the corresponding synchronized shuffle operation on words and languages should be associative. Moreover, since team automata are systems exhibiting ongoing, infinite behavior, the composition operation should also be 'sound' in the sense that all prefixes of a (finite or infinite) synchronized shuffle of two words should be synchronized shuffles of finite prefixes of these two words. While it is in general not difficult to prove associativity of shuffling operations in case only finite words are involved (see, *e.g.*, [4, 7, 8, 11,

19, 22, 32, 38, 39, 40, 41]), this changes when infinite words are allowed, or when certain variants of synchronized shuffling are considered (see, *e.g.*, [8, 14, 27, 35, 37]). In fact, there exist variants of synchronized shuffling for which associativity does not hold—sometimes contrary to one's intuition (see, *e.g.*, [9, 14, 27, 30, 31, 32] and Example 3.10).

Motivated by certain natural synchronization patterns applied in the composition of team automata, we set out to investigate the properties of certain variants of the synchronized shuffle operation. First we considered the general synchronized shuffle operation itself and established its soundness and associativity. This allowed us to prove that also the full, the relaxed, and the arbitrary synchronized shuffle operations satisfy some notion of associativity (Theorems 3.7, 3.9, and 3.10). Together with Proposition 2.4, these results form the basis of the compositionality results, presented in Section 4, for team automata constructed according to (combinations of) the synchronization patterns $\mathcal{R}^{free}$, $\mathcal{R}^{ai}$, and $\mathcal{R}^{no}$. In this way we have extended the results from [7], where compositionality of team automata has been considered only in the context of finitary behaviors. The last variant, the weak synchronized shuffle (first introduced and studied in [9] out of plain mathematical interest) has no satisfactory operational interpretation in terms of synchronization patterns of team automata. This is confirmed by the observation that it is not associative and consequently does not belong to a synchronization pattern which would define team automata satisfying compositionality.

Acknowledgment

The authors thank the anonymous referees for suggestions that have improved the paper's presentation.

References

[1] Abrahamson, K.: *Decidability and Expresiveness of Logics of Processes*, Ph.D. Thesis, University of Washington, Seattle, 1980.

[2] Bailly, A., Clerbout, M., Simplot-Ryl, I.: Component Composition Preserving Behavioural Contracts Based on Communication Traces, *Proceedings of the Tenth International Conference on Implementation and Application of Automata (CIAA'05), Sophia Antipolis, France* (J. Farré, I. Litovsky, S. Schmitz, Eds.), Lecture Notes in Computer Science 3845, Springer-Verlag, Berlin, 2006, 55–66.

[3] ter Beek, M. H. (webmaster): Team Automata webpage, http://fmt.isti.cnr.it/~mtbeek/TA.html

[4] ter Beek, M. H.: *Team Automata—A Formal Approach to the Modeling of Collaboration Between System Components*, Ph.D. Thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2003.

[5] ter Beek, M. H., Ellis, C. A., Kleijn, J., Rozenberg, G.: Synchronizations in team automata for groupware systems, *Computer Supported Cooperative Work—The Journal of Collaborative Computing*, **12**(1), 2003, 21–69.

[6] ter Beek, M. H., Gadducci, F., Janssens, D.: A calculus for team automata, *Proceedings of the Ninth Brazilian Symposium on Formal Methods (SBMF'06), Natal, Rio Grande do Norte, Brazil* (A. Martins Moreira, L. Ribeiro, Eds.), Electronic Notes in Theoretical Computer Science 195, Elsevier Science Publishers, Amsterdam, 2007, 41–55.

[7] ter Beek, M. H., Kleijn, J.: Team Automata Satisfying Compositionality, *Proceedings of FME 2003: Formal Methods— The Twelfth International Symposium of Formal Methods Europe, Pisa, Italy* (K. Araki, S. Gnesi, D. Mandrioli, Eds.), Lecture Notes in Computer Science 2805, Springer-Verlag, Berlin, 2003, 381–400.

[8] ter Beek, M. H., Kleijn, J.: Infinite Unfair Shuffles and Associativity, *Theoretical Computer Science*, **380**(3), 2007, 401–410.

[9] ter Beek, M. H., Martín-Vide, C., Mitrana, V.: Synchronized Shuffles, *Theoretical Computer Science*, **341**(1–3), 2005, 263–275.

[10] Bergstra, J. A., Ponse, A., Smolka, S. A. (Eds.): *Handbook of Process Algebra*, Elsevier Science Publishers, Amsterdam, 2001.

[11] Bloom, S. L., Ésik, Z.: Free Shuffle Algebras in Language Varieties, *Theoretical Computer Science*, **163**(1–2), 1996, 55–98.

[12] Boasson, L., Nivat, M.: Adherences of Languages, *Journal of Computer and System Sciences*, **20**(3), 1980, 285–309.

[13] Carmona, J., Cortadella, J.: Input/Output Compatibility of Reactive Systems, *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD'02), Portland, Oregon* (M. D. Aagaard, J. W. O'Leary, Eds.), Lecture Notes in Computer Science 2517, Springer-Verlag, Berlin, 2002, 360–377.

[14] De Simone, R.: Langages Infinitaires et Produit de Mixage, *Theoretical Computer Science*, **31**, 1984, 83–100.

[15] Drusinsky, D., Harel, D.: On the Power of Bounded Concurrency I—Finite Automata, *Journal of the ACM*, **41**(3), 1994, 517–539.

[16] Duboc, C.: Mixed Product and Asynchronous Automata, *Theoretical Computer Science*, **48**(3), 1986, 183–199.

[17] Ellis, C. A.: Team Automata for Groupware Systems, *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work—The Integration Challenge (GROUP'97), Phoenix, Arizona* (S. C. Hayne, W. Prinz, Eds), ACM Press, New York, 1997, 415–424.

[18] Emerson, E. A.: Alternative Semantics for Temporal Logics, *Theoretical Computer Science*, **26**(1–2), 1983, 121–130. (See also Technical Report TR-182, Department of Computer Sciences, University of Texas, Austin, 1981.)

[19] Ginsburg, S.: *Algebraic and Automata-Theoretic Properties of Formal Languages*, Fundamental Studies in Computer Science 2, North-Holland Publishing Company, Amsterdam, 1975.

[20] Ginsburg, S., Spanier, E. H.: Mappings of Languages by Two-Tape Devices, *Journal of the ACM*, **12**(3), 1965, 423–434.

[21] Gischer, J. L.: Shuffle Languages, Petri Nets, and Context Sensitive Grammars, *Communications of the ACM*, **24**, 1981, 597–605.

[22] Jantzen, M.: The Power of Synchronizing Operations on Strings, *Theoretical Computer Science*, **14**, 1981, 127–154.

[23] Jonsson, B.: Compositional Specification and Verification of Distributed Systems, *ACM Transactions on Programming Languages and Systems*, **16**(2), 1994, 259–303.

[24] Karhumäki, J., Maurer, H. A., Păun, Gh., Rozenberg, G. (Eds.): *Jewels are Forever—Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, Springer-Verlag, Berlin, 1999.

[25] Kimura, T.: An Algebraic System for Process Structuring and Interprocess Communication, *Proceedings of the Eighth ACM SIGACT Symposium on Theory of Computing (STOC'76), Hershey, Pennsylvania*, ACM Press, New York, 1976, 92–100.

[26] Lanotte, R., Maggiolo-Schettini, A., Peron, A.: Timed Cooperating Automata, *Fundamenta Informaticae*, **42**(1–4), 2000, 1–21.

[27] Latteux, M., Roos, Y.: Synchronized Shuffle and Regular Languages, In [24], 1999, 35–44.

[28] Lothaire, M.: *Combinatorics on Words*, Encyclopedia of Mathematics and its Applications 17, Cambridge University Press, Cambridge, 1983.

[29] Lynch, N. A., Tuttle, M. R.: An Introduction to Input/Output Automata, *CWI Quarterly*, **2**(3), 1989, 219–246. (Also appeared as Technical Memo MIT/LCS/TM-373, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1988.)

[30] Manea, F., Mitrana, V., Voinescu, D.-C.: Synchronized Shuffle on Backbones, *Fundamenta Informaticae*, **73**(1–2), 2006, 191–202.

[31] Mateescu, A., Mateescu, G. D.: Fair and Associative Infinite Trajectories, In [24], 1999, 327–338.

[32] Mateescu, A., Rozenberg, G., Salomaa, A.: Shuffle on Trajectories: Syntactic Constraints, *Theoretical Computer Science*, **197**(1–2), 1998, 1–56.

[33] Ogden, W. F., Riddle, W. E., Rounds, W. C.: Complexity of Expressions Allowing Concurrency, *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages (POPL'78), Tucson, Arizona*, ACM Press, New York, 1978, 185–194.

[34] von Oheimb, D., Lotz, V.: Formal Security Analysis with Interacting State Machines, *Proceedings of the Seventh European Symposium on Research in Computer Security (ESORICS'02), Zürich, Switzerland* (D. Gollmann, G. Karjoth, M. Waidner, Eds.), Lecture Notes in Computer Science 2502, Springer-Verlag, 2002, 212–228.

[35] Park, D.: On the semantics of fair parallelism, *Proceedings of the Copenhagen Winter School on Abstract Software Specifications* (D. Bjørner, Ed.), Lecture Notes in Computer Science 86, Springer-Verlag, Berlin, 1979, 504–526.

[36] Perrin, D., Pin, J. - E.: *Infinite Words—Automata, Semigroups, Logic and Games*, Pure and Applied Mathematics 141, Elsevier Science Publishers, Amsterdam, 2004.

[37] Redziejowski, R. R.: Associative Omega-products of Traces, *Fundamenta Informaticae*, **67**(1–3), 2005, 175–185.

[38] Roscoe, A. W.: *The Theory and Practice of Concurrency*, Prentice Hall, London, 1997.

[39] Sakarovitch, J., Simon, I.: Subwords, Chapter 6 in [28], 1983, 105–142.

[40] Shaw, A. C.: Software Descriptions with Flow Expressions, *IEEE Transactions on Software Engineering*, **4**(3), 1978, 242–254.

[41] van de Snepscheut, J. L. A.: *Trace Theory and VLSI Design*, Lecture Notes in Computer Science 200, Springer-Verlag, Berlin, 1985.