Shuffling posets on trajectories (technical report)

Luc Edixhoven[0000−0002−6011−9535]

[1] Open University of the Netherlands
[2] Centrum Wiskunde & Informatica (CWI)
`luc.edixhoven@ou.nl`

Abstract. Choreographies describe possible sequences of interactions among a set of agents. We aim to join two lines of research on choreographies: the use of the shuffle on trajectories operator to design more expressive choreographic languages, and the use of models featuring partial orders, to compactly represent concurrency between agents. Specifically, in this paper, we explore the application of the shuffle on trajectories operator to individual posets, and we give a characterisation of shuffles of posets which again yield an individual poset.

Keywords: Posets · Shuffle on trajectories · Concurrency

Technical report of a paper to be published in the proceedings of iFM 2023.

1 Introduction

Distributed systems are becoming ever more important. However, designing and implementing them is difficult. The complexity resulting from concurrency and dependencies among agents makes the process error-prone and debugging non-trivial. As a consequence, much research has been dedicated to analysing communication patterns, or protocols, among sets of agents in distributed systems. Examples of such research goals are to show the presence or absence of certain safety properties in a given system, to automate such analysis, and to guarantee the presence of desirable properties by construction.

Part of this research deals with *choreographies*. Choreographies can be used as global specifications for asynchronously communicating agents, and contain certain safety properties by construction. As a drawback, choreographic languages typically have limitations on their expressiveness, since they rely on grammatical constructs for their safety properties, which exclude some communication patterns. We have recently shown that the *shuffle on trajectories* operator can be used to specify choreographies without compromising expressiveness [2]. Consequently, it could serve as a basis for more expressive choreographic languages.

Other recent work on choreographies includes the use of models featuring partial orders, such as event structures [1] and pomsets [6,3], to represent and

analyse the behaviour of choreographies. By using a partial order to explicitly capture causal dependencies between pairs of actions, these models avoid the exponential blowup from, e.g., parallel composition of finite state machines.

We aim to join these two lines of research by extending the shuffle on trajectories operator from words, i.e., totally ordered traces, and languages to partially ordered traces and sets thereof. In this paper, as a first step, we explore the application of the shuffle on trajectories operator to individual partially ordered sets, or posets. The main challenge is that the resulting behaviour cannot always be represented as one poset and may require a set of them. In particular, we give a characterisation of shuffles of posets which again yield an individual poset.

Outline We recall the concept and definition of the shuffle on trajectories operator in Section 2. We briefly discuss posets in Section 3. In Section 4 we discuss how to apply the shuffle on trajectories operator to posets, and specifically which shuffles of posets will yield an individual poset as a result. Finally, we briefly discuss future work in Section 5.

The proofs of Proposition 1 and Lemma 1 can be found in the appendix.

2   Shuffle on trajectories

We recall the basic definitions from [2]. The shuffle on trajectories operator is a powerful variation of the traditional shuffle operator[3], which adds a control trajectory (or a set thereof) to restrict the permitted orders of interleaving. This allows for fine-grained control over orderings when shuffling words or languages. The binary operator was defined — and its properties thoroughly studied — by Mateescu et al. [4]; a multiary variant was introduced slightly later [5].

When defined on words, the shuffle on trajectories takes n words and a *trajectory*, which is a word over the alphabet $\{1, \ldots, n\}$. This trajectory specifies the exact order of interleaving of the shuffled words: in Figure 1, the trajectory 1221112112 specifies that the result should first take a symbol from the first word, then from the second, then again from the second and so on.

Formally, let $w_1, \ldots, w_n$ be finite words over some alphabet and let t be a finite word over the alphabet $\{1, \ldots, n\}$. Let ε be the empty word. Then:

$$\sqcup\!\sqcup_t^n (w_1, \ldots, w_n) = \begin{cases} a \sqcup\!\sqcup_{t'}^n (w_1, \ldots, w_i', \ldots, w_n) & \text{if } t = it' \text{ and } w_i = aw_i' \\ \varepsilon & \text{if } t = w_1 = \ldots = w_n = \varepsilon \end{cases}$$

We note that $\sqcup\!\sqcup_t^n(w_1, \ldots, w_n)$ is only defined if the number of occurrences of i in t precisely matches the length of w_i for every i. We then say that t *fits* w_i.

Example 1.

– $\sqcup\!\sqcup_{121332}^3 (ab, cd, ef) = acbefd$, since 121332 fits every word.

---

[3] In concurrency theory, the shuffle operator is also known as free interleaving, non-communication merge, or parallel composition.
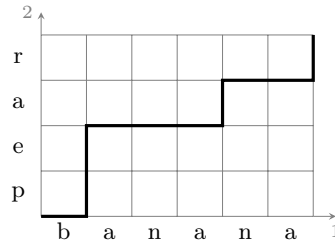
Fig. 1. The shuffle of 'banana' and 'pear' over a trajectory 1221112112: 'bpeanaanar'.

- $\sqcup\!\sqcup^2_{121}(ab, cd)$ is undefined, since 121 does not fit cd.

The shuffle on trajectories operator naturally generalises to languages: the shuffle of a number of languages on a set (i.e., a language) of trajectories is defined as the set of all valid shuffles of words in the languages for which the trajectory fits all the words. Formally:

$$\sqcup\!\sqcup^n_T(L_1, \ldots, L_n) = \{\sqcup\!\sqcup^n_t(w_1, \ldots, w_n) \mid t \in T, w_1 \in L_1, \ldots, w_n \in L_n\}$$

As the operator's arity is clear from its operands, we typically omit it.

3   Posets

Partially ordered sets, or posets for short, consist of a set of nodes E (events), and a partial order[4] $\leq$ defining dependencies between pairs of events — i.e., an event can only fire if all events preceding it in the partial order have already fired. We write $a < b$ to denote that $a \leq b$ and $a \neq b$. We write $a \geq b$ resp. $a > b$ to denote that $b \leq a$ resp. $b < a$. We write $a \not\geqq b$ to denote that $a \not\leq b$ and $b \not\leq a$; we then say that a and b are *concurrent*. We occasionally write $E_P, \leq_P, <_P, \geq_P, >_P$ and $\not\geqq_P$ to specify that the set of events or relation belongs to poset P, but where this is clear from context we typically omit the subscript.

The behaviour (or language) of a poset P, written $L(P)$, is the set of all maximal traces, i.e., maximal sequences of its events, that abide by $\leq$. In this sense, posets can be considered a generalisation of words with concurrency: they feature a fixed set of symbols (events)[5], but they can allow multiple orderings of them instead of only a single one. Concurrent events can happen in any order. Consequently, all traces obtained from a trace in $L(P)$ by swapping adjacent concurrent events must also be in $L(P)$. In fact, any trace in $L(P)$ can be obtained from any other trace in $L(P)$ in this fashion.

---

[4] Recall that a partial order is reflexive, transitive and antisymmetric.

[5] There is a difference between words and posets in the sense that the events in a poset must be unique, whereas a word may contain duplicate symbols. It would be more accurate to say that words generalise to *labelled* posets, or lposets, and from there to partially ordered *multisets*, or pomsets. However, in this paper we focus on posets, where all symbols are thus unique.
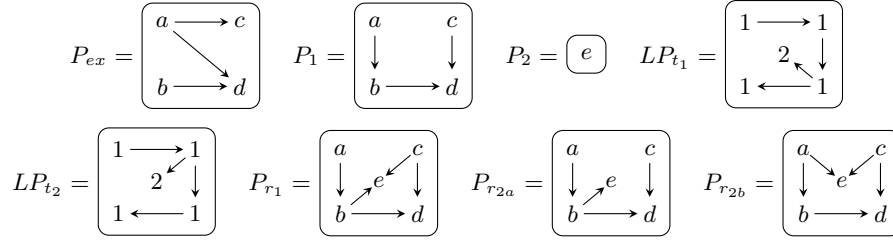
Fig. 2. Graphical representation of a number of posets and lposets, where an arrow from a to b should be read as $a \leq b$. The partial order is the reflexive and transitive closure of the dependencies depicted by the arrows. For the lposets, the labels are shown rather than their events.

Example 2. For poset P_{ex} in Figure 2, $E = \{a, b, c, d\}$ and the partial order consists of $a \leq a$, $a \leq c$, $a \leq d$, $b \leq b$, $b \leq d$, $c \leq c$ and $d \leq d$. Its language $L(P_{ex})$ consists of the traces *abcd*, *abdc*, *acbd*, *bacd*, and *badc*.

We note that the dependencies in a poset can also be observed in its set of traces. For example, if $a < b$ then a will precede b in every trace, and if $a \ngeq b$ then there will both be traces where a precedes b and traces where b precedes a. Formally, we can extract the following relation $\leq_L$ from a set of traces $L \subseteq E^*$:

$$\frac{\exists x, y, z \in E^* : xaybz \in L \quad \forall \hat{x}, \hat{y}, \hat{z} \in E^* : \hat{x}b\hat{y}a\hat{z} \notin L}{a \leq_L b} \qquad \frac{}{a \leq_L a} \qquad \frac{a \leq_L b \leq_L c}{a \leq_L c}$$

We then propose the following:

Proposition 1. *Let $P = \langle E_P, \leq_P \rangle$ be a poset. Then $\leq_{L(P)} = \leq_P$.*

To model trajectories, which require duplicate symbols, we must also introduce *labelled* posets, or *lposets*. In these, every event is assigned a label, which is not necessarily unique. Its traces then use these labels instead of the events.

4   Shuffling posets

As a first step towards shuffling posets, we first reinterpret shuffles on words as posets. In other words: we consider the case where all posets, including the trajectory, are totally ordered and thus consist of a single trace. This is shown in Figure 3, which features the shuffle from Figure 1 interpreted as a poset. The traces 'banana' and 'pear' are present as totally ordered parts of the poset, and the trajectory adds additional dependencies between the two, as shown by the vertical (and diagonal) arrows.

Generalising this to arbitrary posets and lposets is not trivial, but we have some knowledge to assist us. Crucially, since we can determine the language of a
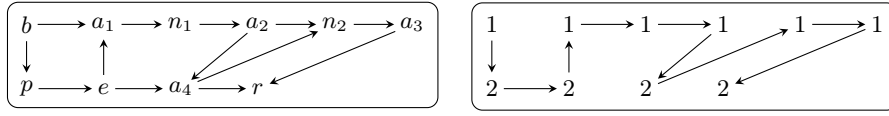
Fig. 3. The figure on the left shows the shuffle from Figure 1 interpreted as a shuffle of posets. Indices have been added to duplicate symbols to make them unique. Some of the arrows are redundant but are kept to illustrate the general idea. The figure on the right shows the trajectory, 1221112112, as an lposet.

poset, it must be so that the result of shuffling posets yields the same language as the shuffle of the languages of these posets, which is defined in Section 2:

$$L(\sqcup_{P_t}(P_1, \ldots, P_n)) = \sqcup_{L(P_t)}(L(P_1), \ldots, L(P_n))$$

If the result is an individual poset, by Proposition 1 it must then be:

$$\sqcup_{P_t}(P_1, \ldots, P_n) = \langle E_{P_1} \cup \ldots \cup E_{P_n}, \leq_{\sqcup_{L(P_t)}(L(P_1), \ldots, L(P_n))} \rangle$$

For example, consider $\sqcup_{LP_{t_1}}(P_1, P_2)$, with LP_{t_1}, P_1 and P_2 as in Figure 2. LP_{t_1} has traces 1121 and 1112, P_1 has traces $abcd$, $acbd$ and $cabd$, and P_2 has a single trace e. Shuffling these languages yields $L_1 = \{abced, acbed, cabed, abcde, acbde, cabde\}$. From this we extract $\leq_{L_1}$, which contains all the dependencies present in P_1 and P_2 and, additionally, $a \leq_{L_1} e$, $b \leq_{L_1} e$ and $c \leq_{L_1} e$. This corresponds to poset P_{r_1} in Figure 2, which indeed yields the language L_1.

However, now consider $\sqcup_{LP_{t_2}}(P_1, P_2)$, again as in Figure 2. LP_{t_2} has traces 1211, 1121 and 1112, which yields $L_2 = L_1 \cup \{abecd, acebd, caebd\}$. From this we extract $\leq_{L_2}$, which still contains all the dependencies in P_1 and P_2, but otherwise only $a \leq_{L_2} e$: the traces $abecd$ and $acebd$ imply that b and c are concurrent with e. However, then the trace $aebcd$ should also be in L_2, which it is not. We can then conclude from Proposition 1 that there exists no poset P such that $L(P) = L_2$. In fact, L_2 corresponds to a set of two posets, namely $P_{r_{2a}}$ and $P_{r_{2b}}$ in Figure 2.

We proceed by giving a characterisation of shuffles of posets for which the result corresponds to an individual poset. A key insight is that, if the result must correspond to an individual poset, then any two events which are concurrent in one of the operands of the shuffle must, in the resulting poset, have the same relation ($<$, $>$ or $\not\gtrless$) to any third event originating from another operand:

Lemma 1. *Let LP_t be an lposet and $P_1, \ldots, P_n, P$ posets such that $L(\sqcup_{LP_t}(P_1, \ldots, P_n)) = L(P)$ and $L(P) \neq \emptyset$. If $a, b \in E_{P_i}$ such that $a \not\gtrless_{P_i} b$ and $c \in E_{P_j}$ with $i \neq j$, then either $a, b <_P c$, or $a, b >_P c$, or $a, b \not\gtrless_P c$.*

We can then group the events in every P_i according to the reflexive and transitive closure of the concurrency relation $\not\gtrless_{P_i}$; two events which are related in this closure then belong to the same group. Note that, while the events in a group are partially ordered, the groups of every P_i are, by construction, totally ordered. It follows from Lemma 1 that two events in the same group, even when not concurrent, must have the same relation to any event outside of their group
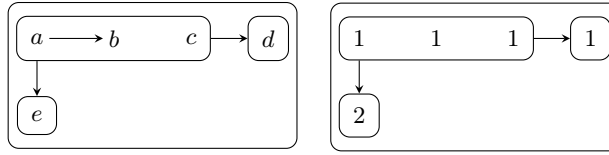
Fig. 4. The figure on the left shows P_{r_1}, corresponding to $\sqcup\!\sqcup_{LP_{t_1}}(P_1, P_2)$ (see Figure 2), restructured to show the groups of P_1 and P_2. An arrow from one group of events to another should be read as an arrow from all events in the originating group to all events in the target group. The figure on the right shows the restructured LP_{t_1}; the dependencies within groups in LP_{t_1} are irrelevant for the resulting traces.

in P. This in turn implies a similar condition on the trajectory lposet: any two i-labelled events in LP_t that can match two events from the same group of P_i must have the same relation to any j-labelled event in LP_t (where j is not necessarily unequal to i) that can match an event outside of their group.

   Figure 4 shows P_{r_1}, corresponding to $\sqcup\!\sqcup_{LP_{t_1}}(P_1, P_2)$, and LP_{t_1} (from Figure 2), both restructured to show the groups of P_1 and P_2. This demonstrates an interesting parallel with Figure 3: both feature horizontal traces with additional arrows specifying dependencies between components of these traces. However, in Figure 3 the components consist of individual events, whereas in Figure 4 the components consist of posets. In this sense, shuffles resulting in individual posets generalise shuffles on traces.

   Concluding, we can then characterise shuffles on posets which result in individual posets as those where the trajectory lposet is structured along the operand posets' groups, as in Figure 4, possibly with dependencies between different operands' groups.

5   Future work

Now that we have studied shuffles of posets resulting in individual posets, there are two evident avenues for future work: (1) shuffles of lposets, where one label may occur multiple times rather than just considering orderings of unique events and (2) shuffles of posets resulting in sets of posets and shuffles of sets of posets, where the main challenge may be to minimise the resulting number of posets.

References

1. Castellani, I., Dezani-Ciancaglini, M., Giannini, P.: Event structure semantics for multiparty sessions. J. Log. Algebraic Methods Program. **131**, 100844 (2023)
2. Edixhoven, L., Jongmans, S.: Balanced-by-construction regular and ω-regular languages. Int. J. Found. Comput. Sci. **34**(2&3), 117–144 (2023)
3. Edixhoven, L., Jongmans, S., Proença, J., Cledou, G.: Branching pomsets for choreographies. In: ICE. EPTCS, vol. 365, pp. 37–52 (2022)
4. Mateescu, A., Rozenberg, G., Salomaa, A.: Shuffle on trajectories: Syntactic constraints. Theor. Comput. Sci. **197**(1-2), 1–56 (1998)

5. Mateescu, A., Salomaa, K., Yu, S.: On fairness of many-dimensional trajectories. J. Autom. Lang. Comb. **5**(2), 145–157 (2000)
6. Tuosto, E., Guanciale, R.: Semantics of global view of choreographies. J. Log. Algebraic Methods Program. **95**, 17–40 (2018)

A   Proofs from the paper

Proposition 1. *Let $P = \langle E_P, \leq_P \rangle$ be a poset. Then $\leq_{L(P)} = \leq_P$.*

Proof.

- Suppose that $a \leq_P b$ for some $a, b \in E_P$. Then, since $\leq_P$ is a partial order, either:
  - $a = b$, in which case also $a \leq_{L(P)} b$ since $\leq_{L(P)}$ is reflexive; or
  - $a \leq_P b$ but $a \neq b$, in which case a will precede b in every (maximal) trace of P. Furthermore, $L(P)$ will not be empty, and then $a \leq_{L(P)} b$ by its first rule.
- Suppose that $a \leq_{L(P)} b$ for some $a, b \in E_P$. Then, by definition of $\leq_{L(P)}$, either:
  - $a = b$, in which case also $a \leq_P b$ since $\leq_P$ is reflexive; or
  - (1) $\exists x, y, z \in E_P^* : xaybz \in L(P)$ and (2) $\forall \hat{x}, \hat{y}, \hat{z} \in E_P^* : \hat{x}b\hat{y}a\hat{z} \notin L(P)$. Suppose, for the sake of contradiction, that $a \not\leq_P b$. Then either $a >_P b$, which contradicts (1), or $a \not\geq_P b$, which contradicts (2) since there should then also exist some trace in $L(P)$ in which b precedes a. We can thus conclude that $a \leq_P b$.

  We note that the transitive rule for $\leq_{L(P)}$ is subsumed by the other two and does not need to be considered separately. Nevertheless, a straightforward inductive argument suffices to cover this.

Lemma 1. *Let LP_t be an lposet and $P_1, \ldots, P_n, P$ posets such that $L(\sqcup_{LP_t}(P_1, \ldots, P_n)) = L(P)$ and $L(P) \neq \emptyset$. If $a, b \in E_{P_i}$ such that $a \not\geq_{P_i} b$ and $c \in E_{P_j}$ with $i \neq j$, then either $a, b <_P c$, or $a, b >_P c$, or $a, b \not\geq_P c$.*

Proof. If no subscript is given, in this proof, $<$ and $\not\geq$ refer to $<_P$ and $\not\geq_P$. We start by making three observations:

(1) As noted in Section 3, if $a \not\geq_{P_i} b$, then there must exist traces in $L(P_i)$ such that a occurs before b in one, and after b in another. Furthermore, since all traces of a poset can be obtained from an arbitrary one by swapping adjacent concurrent events, there must also exist traces such that a and b are adjacent to each other, in both orders, e.g., $vabw$ and $vbaw$ for some v, w.

(2) The causal relation between two events in P_i remains the same in P. The shuffle on trajectories operator does not change the internal order of the traces of its operands, meaning that, for example if d occurs before e in every trace of P_i, then it will also occur before e in every trace of P. Similarly, if P_i both contains traces in which d occurs before e and traces in which d occurs after e, then so will P. In particular, this means that, since $a \not\geq_{P_i} b$, also $a \not\geq b$.

(3) If two events are concurrent in P_i and can thus be swapped in traces of P_i when adjacent, then they can also be swapped in traces of P when there are no other events from P_i in between them. The shuffle on trajectories operator does not make any distinction between the corresponding traces from P_i. Additionally, we note that this implies that the two events are concurrent with all events (from other operands) in between.

Consider the relation between a, b and c in P. As observed in (2), $a \not\geq b$.

– Suppose that $a < c$ and $c < b$. It then follows by transitivity (recall that $\leq$ is a partial order) that $a < b$, which contradicts $a \not\geq b$. Analogously, we can exclude the case where $b < c$ and $c < a$.
– Suppose that $a < c$ and $b \not\geq c$. Then, there must exist a trace $uavcwbx$ in $L(P)$ for some u, v, w, x, i.e., a trace where a occurs before c and where b occurs after c. If vw contains no events from P_i, then it follows from $a \not\geq b$ and (3) that $L(P)$ must also contain $ubvcwax$, which contradicts $a < c$. Otherwise, we systematically remove the events from P_i in vw:
  • Let d be the leftmost event from P_i in vw such that $d < b$. Then, since we must be able to swap adjacent concurrent events in traces of P_i until a and b are adjacent (as noted in (1)), d must be concurrent with all events from P_i to the left of it in avw (including a). We can thus swap d with its left neighbour from P_i until it has been swapped with a, thus reducing the number of events from P_i in vw by 1. We repeat this until there are no events left in the remainder of vw that fit this description.
  • Analogously, let e be the rightmost event from P_i in the remainder of vw such that $a < e$. Then e must be concurrent with all events from P_i to the right of it until at least b, so we swap it with its right neighbour from RPi until it has been swapped with b. Again, we repeat this until there are no events left that fit this description.
  • All events from P_i remaining in vw must now be concurrent with both a and b. We can thus keep swapping a with its right neighbour (or, alternatively, b with its left neighbour) from P_i until there are no events from P_i remaining between a and b.
  If, at some point, a has passed c and is now to the right of it, then this contradicts $a < c$. If a remains to the left of c and b to its right, then we can swap a and b to contradict $a < c$. Otherwise, if b has passed c and is now to the left of it, then b must be concurrent with all events to the right of it at least until c (as noted in (3)). We can then swap it with its right neighbour (from P) until it has been swapped with c, after which we can swap it with a to contradict $a < c$.
  As all cases lead to a contradiction, we can thus conclude that it is not possible that $a < c$ and $b \not\geq c$. Analogously, we can also exclude the cases where $c < a$ and $b \not\geq c$, where $b < c$ and $a \not\geq c$, and where $c < b$ and $a \not\geq c$.

As we have excluded all other cases, this proves that either $a, b < c$, or $c < a, b$, or $a, b \not\geq c$. □