Decidability results for choreography realization

Niels Lohmann and Karsten Wolf

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{niels.lohmann, karsten.wolf}@uni-rostock.de

Abstract. A service choreography defines a set of permitted sequences of message events as a specification for the interaction of services. Realizability is a fundamental sanity check for choreographies comparable to the notion of soundness for workflows.
We study several notions of realizability: partial, distributed, and complete realizability. They establish increasingly strict conditions on realizing services. We investigate decidability issues under the synchronous and asynchronous communication models. For partial realizability, we show undecidability whereas the other two problems are decidable with reasonable complexity.

1 Introduction

A *choreography* describes the interaction of services. In the literature on services, this term has been used for representing the behavior of a system composed of services ("interconnected models") or for the restriction of that behavior to the communication events ("interaction model"). In this paper, we follow the second interpretation. To be more precise, a choreography is typically understood as a *specification* of interaction that can be used as a contract between organizations. This specification is later compared to those interactions that implement the specification. If the implementation produces those interactions which are specified in the choreography, this implementation *realizes* the choreography. Consequently, the question of *realizability* is a fundamental sanity property for choreographies.

The realizability problem has several dimensions. The first dimension is concerned with the notation in which the choreography is given. Several languages have been proposed for choreography description, including WS-CDL [8], Let's Dance [20], UML collaboration diagrams [3], and BPMN 2.0 [14]. They all have in common that they permit the specification of a regular set of sequences of message events. For covering all these languages, we abstract from the syntactic sugar of these languages and assume a choreography to be given in the shape of a finite automaton.

The second dimension for the realizability problem is the communication model assumed. In this paper, we consider synchronous as well as asynchronous communication. In the asynchronous case, we do not assume that messages arrive in the same order in which they have been sent. In the spectrum of reasonable communication models (cf. [10] for a survey), we thus consider the models with the tightest, respectively loosest coupling between sender and receiver of a message. We do not consider FIFO based models.

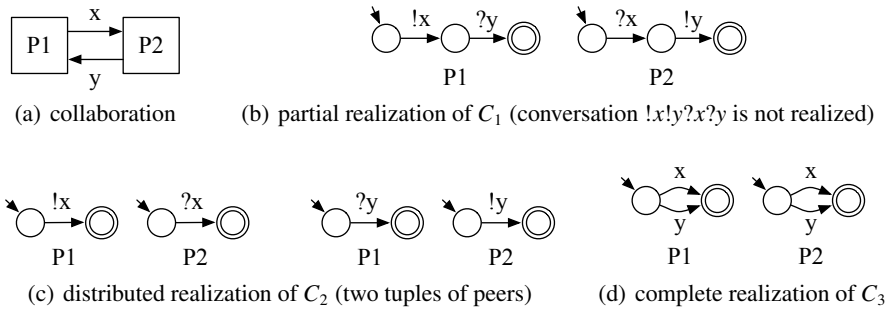

Fig. 1. Collaboration (a) and peer implementations for the partially realizable choreography $C_1 = \{!x!y?x?y,\ !x?x!y?y\}$ (b), the distributedly realizable choreography $C_2 = \{!x?x,\ !y?y\}$ (c), and the completely realizable choreography $C_3 = \{x,\ y\}$ (d)

In the third dimension of the realizability problem, we need to determine what it exactly means for an implementation to conform to a choreography. Following earlier considerations [12], we study three concepts: partial, distributed, and complete realization. In a partial realization, the implementation produces some, but not necessarily all sequences of message events specified in the choreography, cf. Fig. 1(b). Here, the choreography is seen as a space of opportunities which need not be exhausted by the implementation. Distributed choreography follows the same intention, but assures that the choreography does not contain junk sequences which cannot be contained in any realization. Hence, a choreography is distributedly realizable if there is a (possibly infinite) family of implementations such that each specified sequence of the choreography is realized in at least one of them, cf. Fig. 1(c). Complete realizability, in turn, requires that all sequences specified in the choreography can be produced in a single implementation, cf. Fig. 1(d). The three concepts form a hierarchy; that is, complete realizability implies distributed, and distributed implies partial realizability.

Contribution. We show that, for both considered communication models, partial realizability is undecidable whereas distributed and complete realizability are decidable. Our undecidability results depend on a reduction of the famous undecidable Post correspondence problem (PCP). The decision procedures for distributed and complete realizability depend on standard language theoretic constructions such as projection, checking language equivalence, and minimization of automata. Thus, despite exponential worst case complexity, we may assume mature algorithms with reasonable run times.

Organization. After giving the formal definitions of our concepts (Sect. 2), we study first partial (Sect. 3), then distributed (Sect. 4), and finally complete realizability (Sect. 5). In each of the sections, we first present our results for synchronous communication in full detail. Then, for space reasons, we just briefly discuss how these arguments need to be modified in the asynchronous case. In Sect. 6 we discuss related work before Sect.7 concludes the paper and lists open problems.

2 Basic definitions

2.1 Interconnected models and interaction models

Throughout this paper, fix a finite set of message channels M that is partitioned into asynchronous message channels M_A and synchronous message channels M_S. From M, derive a set of message events $E := !E \cup ?E \cup M_S$, consisting of asynchronous send events $!E := \{!x \mid x \in M_A\}$, asynchronous receive events $?E := \{?x \mid x \in M_A\}$, and synchronization events. Furthermore, we distinguish a non-communicating event $\tau \notin E$. For an event $x \in E$, define $channel(x) = a$ if $x = a$, $x = ?a$, or $x = !a$.

Definition 1 (Peer, collaboration). *A* peer $P = [I, O]$ *consists of a set of input message channels $I \subseteq M$ and a set of output message channels $O \subseteq M$, $I \cap O = \emptyset$. A* collaboration *is a set $\{[I_1, O_1], \ldots, [I_n, O_n]\}$ of peers such that $I_i \cap I_j = \emptyset$ and $O_i \cap O_j = \emptyset$ for all $i \neq j$, and $\bigcup_{i=1}^{n} I_i = \bigcup_{i=1}^{n} O_i$.*

A peer and a collaboration (cf. Fig. 1(a)) can be seen as a syntactic signature of a service and a composition, respectively. The behavior itself (i.e., the order in which messages are exchanged and when a peer terminates) is modeled by *peer automata*. A peer automaton is a state machine whose transitions are labeled by message events or τ.

Definition 2 (Peer automaton). *A* peer automaton $A = [Q, \delta, q_0, F, \mathcal{P}]$ *is a tuple such that Q is a set of states, $\delta \subseteq Q \times (E_I \cup E_O \cup \{\tau\}) \times Q$ is a transition relation, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of final states, and $\mathcal{P} = \{[I_1, O_1], \ldots, [I_n, O_n]\}$ is a nonempty set of peers. Thereby, $E_I := \{?x \mid x \in M_A \cap \bigcup_{i=1}^{n} I_i\} \cup (M_S \cap \bigcup_{i=1}^{n} I_i)$ are the input events of A and $E_O := \{!x \mid x \in M_A \cap \bigcup_{i=1}^{n} O_i\} \cup (M_S \cap \bigcup_{i=1}^{n} O_i)$ are output events of A.*

A implements *the peers $\mathcal{P}$, and for $[q, x, q'] \in \delta$, we also write $q \xrightarrow{x} q'$. A is called a* single-peer automaton*, if $|\mathcal{P}| = 1$. A is called a* multi-peer automaton*, if $|\mathcal{P}| > 1$ and $\mathcal{P}$ is a collaboration. A is called* τ-free *if $q \xrightarrow{x} q'$ implies $x \neq \tau$ for all $q, q' \in Q$. A is called* deterministic *if A is τ-free and $q \xrightarrow{x} q'$ and $q \xrightarrow{x} q''$ imply $q' = q''$. A is called* finite *if the number of states reachable from q_0 is finite. An* accepting run *of A is a sequence of events $x_1 \cdots x_m$ such that $q_0 \xrightarrow{x_1} \cdots \xrightarrow{x_m} q_f$ with $q_f \in F$.*

The interplay of peers is modeled by their *composition*. In case of asynchronous communication, pending messages are represented by a multiset. Denote the set of all multisets over M_A with $Bags(M_A)$, the empty multiset with $[\,]$, and the multiset containing only one instance of $x \in M_A$ with $[x]$. Addition of multisets is defined pointwise.

Definition 3 (Composition of single-peer automata). *Let $A_1, \ldots, A_n$ be finite single-peer automata ($A_i = [Q_i, \delta_i, q_{0_i}, F_i, \{P_i\}]$ for $i = 1, \ldots, n$) such that their peers form a collaboration. Define the* composition *$A_1 \oplus \cdots \oplus A_n$ as the multi-peer automaton $[Q, \delta, q_0, F, \{P_1, \ldots, P_n\}]$ with $Q := Q_1 \times \cdots \times Q_n \times Bags(M_A)$, $q_0 := [q_{0_1}, \ldots, q_{0_n}, [\,]]$, $F := F_1 \times \cdots \times F_n \times \{[\,]\}$, and, for all $i \neq j$ and $B \in Bags(M_A)$ the transition relation δ contains exactly the following elements:*

- $[q_1, \ldots, q_i, \ldots, q_n, B] \xrightarrow{\tau} [q_1, \ldots, q_i', \ldots, q_n, B]$*, if and only if* $[q_i, \tau, q_i'] \in \delta_i$ *(internal move by A_i),*

- $[q_1, \ldots, q_i, \ldots, q_n, B] \xrightarrow{!x} [q_1, \ldots, q'_i, \ldots, q_n, B + [x]]$, *if and only if* $x \in M_A$ *and* $[q_i, !x, q'_i] \in \delta_i$ *(asynchronous send by* A_i*),*
- $[q_1, \ldots, q_i, \ldots, q_n, B + [x]] \xrightarrow{?x} [q_1, \ldots, q'_i, \ldots, q_n, B]$, *if and only if* $x \in M_A$ *and* $[q_i, ?x, q'_i] \in \delta_i$ *(asynchronous receive by* A_i*), and*
- $[q_1, \ldots, q_i, \ldots, q_j, \ldots, q_n, B] \xrightarrow{x} [q_1, \ldots, q'_i, \ldots, q'_j, \ldots, q_n, B]$, *if and only if* $x \in M_S$, $[q_i, x, q'_i] \in \delta_i$, *and* $[q_j, x, q'_j] \in \delta_j$ *(synchronization between* A_i *and* A_j*).*

The composition of two finite service automata may have an infinite number of states, because we consider arbitrary multisets of asynchronous messages. In the remainder, we only consider finite compositions.

2.2 Languages and traces

The results in the next sections heavily rely on concepts of regular languages and traces.

Definition 4 (Automaton versus language). *Let* $A = [Q, \delta, q_0, F, \mathcal{P}]$ *be a finite peer automaton. For an accepting run* ρ*, define the* event sequence *of* ρ *as* $\rho_{|E}$ *(i.e.,* ρ *without* τ*-steps). The* language *of A, denoted* $\mathcal{L}(A)$*, is the set of the event sequences of all accepting runs of A. The other way round, if* $\mathcal{P}$ *is a set of peers and L is a regular language over the alphabet* $\bigcup_{[I,O]\in\mathcal{P}} I \cup O$ *then* $\mathcal{A}(L)$ *is the minimal (regarding size of Q) finite, deterministic, and* τ*-free peer-automaton that implements* $\mathcal{P}$ *and has* $\mathcal{L}(\mathcal{A}(L)) = L$*.*

Formal languages theory asserts that, for every nonempty regular language L, an automaton $\mathcal{A}(L)$ exists and is unique up to isomorphism. Throughout this paper, we only consider regular languages and choreographies. This accords to many industrial and academic choreography specification languages.

Definition 5 (Choreography). *Let* $\mathcal{P} = \{[I_1, O_1], \ldots, [I_n, O_n]\}$ *be a collaboration. A* conversation *of* $\mathcal{P}$ *is a word over the events of* $\mathcal{P}$*. A* choreography *for* $\mathcal{P}$ *is a nonempty regular set of conversations of* $\mathcal{P}$*.*

The individual realizability notions differ in the amount of conversations which must be realized by the peers. They all have in common that no new conversation must be introduced. Hence, the projected peers need to be *coordinated* at design time such that they do not produce unspecified conversations. The example choreographies in Fig. 1 show that this coordination can already be impossible even if two peers share message channels. To characterize possible and impossible coordination, we first introduce *distant* message events. We call two message events distant if there exists no peer which can observe both, for instance $!x$ and $!y$ in Fig. 1(b):

Definition 6 (Distant message events). *Let* $\mathcal{P}$ *be a collaboration. Two message events* $a, b \in E$ *are* distant *if and only if there exist no peer* $[I, O] \in \mathcal{P}$ *such that* $\{channel(a), channel(b)\} \subseteq (I \cup O)$*.*

Several results in this article shall depend on the observation that no composition of peer automata is able to enforce any order on concurrently activated distant events. That is, if distant events subsequently occur in one order, they can also occur in the reverse

order. An exception are related asynchronous send and receive events. They are distant according to our definition but the send event always precedes the corresponding receive event as long as no other message of this kind is pending. The following definition formalizes this observation. Whether there are pending asynchronous message, can be determined by a simple counting on the prefix of the sequence.

Definition 7 (Message counting, trace). *For $x \in M_A$ and an event sequence w, define $\hat{x}(w)$ by the following induction scheme:*

Base: *For the empty sequence ε, let $\hat{x}(\varepsilon) = 0$.*
Step: *Let $\hat{x}(wa) = \hat{x}(w) + 1$, if $a = !x$, $\hat{x}(wa) = \hat{x}(w) - 1$, if $a = ?x$, $\hat{x}(wa) = \hat{x}(w)$, for all other events a.*

Let $\mathcal{P}$ be a collaboration. For a word w over the alphabet E, define the trace *of w, $\langle w \rangle_{\mathcal{P}}$, by the following induction scheme:*

Base: *Let $w \in \langle w \rangle_{\mathcal{P}}$.*
Step: *For all distant events a, b such that there is no $x \in M$ with $a = !x$ and $b = ?x$, $w_1abw_2 \in \langle w \rangle_{\mathcal{P}}$ implies $w_1baw_2 \in \langle w \rangle_{\mathcal{P}}$ and, if $\hat{x}(w_1) > 0$, $w_1!x?xw_2 \in \langle w \rangle_{\mathcal{P}}$ implies $w_1?x!xw_2 \in \langle w \rangle_{\mathcal{P}}$.*

If $\mathcal{P}$ is clear from the context,we simply write $\langle w \rangle$ instead of $\langle w \rangle_{\mathcal{P}}$.

This concept is very similar to local traces [11]. If $M_a = \emptyset$, it coincides with Mazurkiewicz traces [13] which have been intensely investigated [5]. By our definition of composition, the message count functions $\hat{x}$ will always return 0 for event sequences of terminating runs and values greater than or equal to 0 for prefixes of terminating runs.

Proposition 1 (Notation for languages). *Let L_1 and L_2 be regular languages. Then (1) the concatenation of L_1 and L_2, L_1L_2, (2) the complement of L_1, $\overline{L_1}$, (3) the union of L_1 and L_2, $L_1 \cup L_2$, (4) the difference of L_1 and L_2, $L_1 \setminus L_2$, (5) iteration/Kleene star, L_1^*, (6) nonempty iteration, $L_1^+ = L_1L_1^*$, (7) projection to the letters appearing in the events of $\mathcal{P}$, $L_1|_{\mathcal{P}}$, and (8) the shuffle product of L_1 and L_2, $L_1 \,||\, L_2$, are regular.*

3 Partial realizability

Definition 8 (Partial realizability). *Let C be a choreography for a collaboration $\{P_1, \ldots, P_n\}$. The finite single-peer automata $A_1, \ldots, A_n$* partially realize *C if, for all i, A_i implements $\{P_i\}$ and $\emptyset \neq \mathcal{L}(A_1 \oplus \cdots \oplus A_n) \subseteq C$.*

Example. The choreography C_1 in Fig. 1 is only partially realizable (e.g., by the peer automata depicted in Fig. 1(b)), because the conversation $!x!y?x?y$ cannot be implemented by peers without also producing the unspecified conversations $!y!x?x?y$ or $!y!x?y?x$. Only the conversation $!x?x!y?y$ is realized.

3.1 The synchronous case

In this subsection, we assume $M_A = \emptyset$. We shall show that partial decidability is undecidable for $n \geq 4$ and trivial for $n \leq 3$. We start with some simple observations about partial realizability in presence of distant events.

Proposition 2. *Let $A_1, \ldots, A_n$ be finite single-peer automata whose collection of peers forms a collaboration. Then for all sequences w over E, $w \in \mathcal{L}(A_1 \oplus \cdots \oplus A_n)$ implies $\langle w \rangle \subseteq \mathcal{L}(A_1 \oplus \cdots \oplus A_n)$.*

Proof. Follows directly from the definition of composition and distant events. □

That is, realization cannot be finer than the level of granularity of traces. The other way round, we can realize a single conversation w provided that the choreography contains its whole trace $\langle w \rangle$. This can be done by letting each peer automaton execute, in sequence, those letters of w which occur in its set of peers. Formally:

Proposition 3. *Let w be a sequence over E. Then $\mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(\{w\}|_{\mathcal{P}_i})) = \langle w \rangle$.*

Proof. Follows directly from the definition of composition and distant events. □

Joining these two observations, we obtain a characterization of partial realizability that we shall use throughout the remainder of this section.

Lemma 1. *A choreography C for a collaboration $\mathcal{P}$ is partially realizable if and only if it contains a conversation w with $\langle w \rangle \subseteq C$.*

Proof. Implication: Assume C is partially realizable. Then exists at least one conversation w that is realized by peers and Proposition 2 states that $\langle w \rangle \subseteq C$.
Replication: Assume there exists a conversation w with $\langle w \rangle \subseteq C$. Then Proposition 3 states that the single-peer automata $\mathcal{A}(\{w\}|_{\mathcal{P}_1}), \ldots, \mathcal{A}(\{w\}|_{\mathcal{P}_n})$ realize $\langle w \rangle$ and, as $\emptyset \neq \langle w \rangle \subseteq C$, also partially C. □

We are now ready to consider the case of at most three peers.

Theorem 1. *Let C be a choreography for a collaboration with at most three peers. Then C is partially realizable if and only if $C \neq \emptyset$.*

Proof. Take an arbitrary conversation w in C and apply the construction of Prop. 3. Observe further that there cannot be distant events, because each event is shared by two peers. This means that the realized language is $\{w\}$ which is clearly a nonempty subset of C. □

For the case of four or more peers, we show undecidability.

Theorem 2. *Partial realizability is undecidable for choreographies that involve at least four peers.*

Undecidability is shown by reduction of the famous *Post correspondence problem* using a proof pattern that is inspired by a proof in [15].

Definition 9 (Post correspondence problem (PCP)). *A* Post system *over alphabet X is a finite set $P = \{[u_1, v_1], \ldots, [u_k, v_k]\}$ of ordered pairs of words $u_i, v_i \in X^*$. A* candidate *is a nonempty finite sequence $i_1 \cdots i_n$ of indices $i_j \in \{1, \ldots, k\}$. Candidate $i_1 \cdots i_n$ is a* solution *of Post system P if $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$. The* Post correspondence problem *is to decide for a given Post system P whether it has a solution.*

In other words, the question is whether we can arrange the pairs (in arbitrary copies) such that the concatenation of the left elements yields the same sequence as the concatenation of the right elements. Undecidability of PCP is a classical result in the theory of computable functions.

In the sequel, we show that decidability of partial realizability would imply decidability of PCP. Consequently, we start with a Post system P and construct a choreography C for a collaboration such that P has a solution if and only if C is partially realizable; that is, C includes $\langle w \rangle$ for at least one conversation $w \in C$.

Message channels. Let X be the alphabet used in P and assume that $X' := \{x' \mid x \in X\}$ is another, disjoint alphabet of same size. For a sequence $x_1 \cdots x_m$ in X let $(x_1 \cdots x_m)' = x'_1 \cdots x'_m$. Let k be the number of pairs in P and assume further, without loss of generality, that none of X and X' contain elements from $\{1, \ldots, k\}$. Set $M_S = X \cup X' \cup \{1, \ldots, k\}$ and $M_A = \emptyset$.

Collaboration. We translate P into a collaboration with four peers $P_1 = [I_1, O_1], \ldots, P_4 = [I_4, O_4]$. We set $I_1 = O_2 = X \cup \{1, \ldots, k\}$, $I_3 = O_4 = X'$, and $O_1 = I_2 = O_3 = I_4 = \emptyset$. This means that two messages are distant if and only if one of them is in $X \cup \{1, \ldots, k\}$ and the other is in X'.

Encoding of candidates. Consider the following encoding of an arbitrary candidate $i_1 \cdots i_n$ of P: $w(i_1 \cdots i_n) = i_1 u_{i_1} v'_{i_1} \cdots i_n u_{i_n} v'_{i_n}$. That is, we have a sequence of blocks where each block consists of a pair number, the left side of the pair, and the primed version of the right side of the pair. If $i_1 \cdots i_n$ is a solution, the projection of $w(i_1 \cdots i_n)$ to X yields the same sequence as its projection to X' (up to the "priming" of the letters in X'). Letters in X do not commute, so the projection to X is the same for all members of the trace $\langle w(i_1 \ldots i_n) \rangle$. The same is true for the projection to X'. On the other hand, letters of X' commute arbitrarily with letters in X and in $\{1, \ldots, k\}$. This leads us to the core observation for our construction.

Proposition 4. *The sequence $i_1 \cdots i_n$ is a solution of the Post system P if and only if the trace $\langle w(i_1 \cdots i_n) \rangle$ contains a word of the language defined by the regular expression $(x_1 x'_1 \mid \cdots \mid x_n x'_n \mid 1 \mid \cdots \mid k)^*$.*

In other words, the letters of X and X' can be adjusted such that they can be compared letter by letter (and the pair numbers occur somewhere in between).

Example. As an example, consider the Post system $P = \{[a, baa], [ab, aa], [bba, bb]\}$ with $k = 3$ pairs over the alphabet $X = \{a, b\}$. Define $X' = \{a', b'\}$ and the peers $P_1 = [\{a, b, 1, 2, 3\}, \emptyset]$, $P_2 = [\emptyset, \{a, b, 1, 2, 3\}]$, $P_3 = [\{a', b'\}, \emptyset]$, and $P_4 = [\emptyset, \{a', b'\}]$. Consider the candidate 3 2 3 1 and define the word $w(3\,2\,3\,1) = 3bbab'b'\ 2aba'a'\ 3bbab'b'\ 1ab'a'a'$. The letters in this word can be reordered, and in the trace $\langle w(3\,2\,3\,1) \rangle$ we can find the word $3bb'bb'aa'2aa'bb'3bb'bb'aa'1aa'$. By Prop. 4, we can conclude that the candidate 3 2 3 1 is a solution, and indeed *bba ab bba a = bb aa bb baa*.

Choreography. We want the choreography C to contain all sequences w on $X \cup X' \cup \{1, \ldots, k\}$ except

(1) at least one sequence of the trace $\langle w \rangle$ if the word w cannot be reshuffled to the encoding of some candidate of P and
(2) at least one sequence of the trace $\langle w \rangle$ if the word projections of w to X and X' lead to different sequences.

At the same time we need to assure that,

(3) for any solution $i_1 \cdots i_n$ of P, the trace of its encoding, $\langle w(i_1 \ldots i_n) \rangle$, is indeed fully contained in C.

The recognition of the faulty sequence w is facilitated by the fact that the respective trace $\langle w \rangle$ contains all possible reshufflings of w. These reshufflings contain normal forms for which the characterization of fault sequences is straightforward.

It is easy to see that a choreography that satisfies (1), (2), and (3) is indeed partially realizable if and only if P has a solution. Hence, it remains to show that there are regular languages L_1 and L_2 such that

- L_1 contains at least one sequence of $\langle w \rangle$ if w cannot be reshuffled to the encoding of some candidate of P (1), but no sequence of $\langle w(i_1 \cdots i_n) \rangle$, for any solution $i_1 \cdots i_n$ of P (3) and
- L_2 contains at least one sequence of $\langle w \rangle$ if the projections of w to X and X' lead to different sequences (2), but no sequence of $\langle w(i_1 \cdots i_n) \rangle$, for any solution $i_1 \cdots i_n$ of P (3).

We present L_1 and L_2 as expressions using the operations mentioned in Def. 1 which proves regularity. Concerning L_1, there can be two reasons for the incapability to reshuffle a conversation to the encoding of any candidate. First, the projection of a word to $X \cup \{1, \ldots, k\}$ may not correspond to a sequence of pair numbers and corresponding left elements of pairs. As this projection is invariant under reshuffling (no pair of letters in $X \cup \{1, \ldots, k\}$ is distant to each other), removal of such words cannot compromise condition (3). This is reflected in

$$L_1 = L_{11} \cup L_{12} \quad \text{with} \quad L_{11} = \overline{(1u_1 \mid \cdots \mid ku_k)^*} \,\|\, X'^*.$$

Second, the projection to X' may not deliver the (unique) sequence that fits to the projection to $X \cup \{1, \ldots, k\}$. This, in turn, may be caused by (a) excess letters from X' after having served all pairs or (b) the incapability to complement some iu_i with the unique fitting v'_i. We model L_{12} such that we detect the problem immediately subsequent to the largest prefix that can be shuffled into the correct encoding. Consequently, let

$$L_{12} = (1u_1v'_1 \mid \cdots \mid ku_kv'_k)^* \, (L_a \cup L_b).$$

The two languages in the tail of this expression correspond to the mentioned problems. Thus,

$$L_a = X'^+ \quad \text{and} \quad L_b = \bigcup_{j=1}^{k} \Big(ju_j((X'^* \setminus v'_jX'^*) \,\|\, (X \cup \{1, \ldots, k\})^*) \Big).$$

For language L_2, we only need to detect a single mismatch or excess letters in either subalphabet. Let $X = \{x_1, \ldots, x_m\}$. Then we set

$$L_2 = (x_1 x_1' \mid \cdots \mid x_m x_m' \mid 1 \mid \cdots \mid k)^* \, (X^+ \mid X'^+ \mid \bigm|_{i \neq j} x_i x_j' (X \cup X' \cup \{1, \ldots, k\})^*).$$

For both languages L_1 and L_2, the construction transparently shows that they satisfy the specified conditions. Hence, we may come to our final conclusion that

$$C := (X \cup X' \cup \{1, \ldots, k\})^* \setminus (L_1 \cup L_2)$$

contains a word w with $\langle w \rangle \subseteq L$ if and only if the Post system P has a solution. This concludes the proof of Theorem 2.

Example (cont.). For the example Post system, the following words are examples for the defined languages:

- $1aba'\,2ab'a' \in L_{11}$ — this word uses pairs $[ab, a], [a, ba] \notin P$.
- $3bbab'b'a' \in L_{12}$ — the letter a' after pair $[bba, bb]$ is too much.
- $2aba'b' \in L_{12}$ — this word uses a pair $[ab, ab] \notin P$.
- $3bb'bb'aa'\,1ab'a'a' \in L_2$ — no solution, because one a is not matched.

3.2 The asynchronous case

Assume now $M_S = \emptyset$. We show that the arguments used in the synchronous case extend to asynchronous communication. First, Propositions 2 and 3 hold in the asynchronous case as well. For the latter proposition, observe in particular that events $!x$ and $?x$ commute only if a message x is pending before the execution of the considered event $!x$. This is reflected both in the definition of composition and the definition of traces.

For the PCP reduction of a PCP instance P with k pairs, a topology with three peers P_1, P_2, and P_3. is sufficient. Having distinct events for sending and receiving, we can use $M_A = X \cup \{1, \ldots, k\}$. While the sending events take the role of X in the previous subsection, the corresponding receiving events replace the primed letters above. Then $O_1 = !X \cup \{!1, \ldots, !k\}$, $I_2 = ?X$, $I_3 = \{?1, \ldots, ?k\}$, and $I_1 = O_2 = O_3 = \emptyset$. We use the same choreography as above, except for the fact that we assume P_3 to receive messages arbitrarily; that is, we shuffle the choreography used above with $(\{?1, \ldots, ?k\})^*$. Send and corresponding receive events are distant except for the case that no message of shape x is pending. However, we can exploit that already a "monotonous" version of PCP is undecidable:

> Given a Post system $\{[u_1, v_1], \ldots, [u_k, v_k]\}$, is there a candidate $i_1 \cdots i_n$ such that $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$ and, for all $j < n$, $v_{i_1} \cdots v_{i_j} \sqsubseteq u_{i_1} \cdots u_{i_j}$.

Thereby, $\sqsubseteq$ denotes the prefix operator: the left pairs are always a prefix of the right pairs. Undecidability can be observed from the standard reduction of the halting problem for Turing machines to PCP. In this proof, the difference between the u-sequence and the v-sequence is used for coding configurations of the Turing machine. That is, the u-sequence is always ahead of the v-sequence which can only catch up after having passed a terminating configuration of the machine.

In the monotonous setting, the coding of a PCP solution satisfies the condition that every receive event is preceded by sufficiently many send events. The reshuffling to a form where send and corresponding receive events are immediate neighbors is also not blocked by inactivated receive events, Thus, the argument used in the synchronous case extends to the asynchronous case.

Corollary 1. *Partial realizability under the asynchronous communication model is undecidable if at least three peers are involved.*

4 Distributed realizability

Definition 10 (Distributed realizability). *Let C be a choreography for a collaboration $\{P_1, \ldots, P_n\}$. The set of tuples of finite single-peer automata $\{[A_{1j}, \ldots, A_{nj}] \mid j \in \mathbb{N}^+\}$ is* distributedly realize *C if, for $i = 1, \ldots, n$ and all j, (i) A_{ij} implements $\{P_i\}$, (ii) $\emptyset \neq \mathcal{L}(A_{1j} \oplus \cdots \oplus A_{nj}) \subseteq C$, and (iii) $\bigcup_j \mathcal{L}(A_{1j} \oplus \cdots \oplus A_{nj}) = C$.*

Example. The choreography C_2 in Fig. 1 is distributedly realizable: There exist two tuples of peers (cf. Fig. 1(c)) such that every conversation of the choreography is implemented. As the distant events $!x$ and $!y$ cannot be coordinated, C_2 is not completely realizable.

4.1 The synchronous case

Again assume $M_A = \emptyset$. Distributed realizability can be rephrased using traces.

Theorem 3. *A choreography C for a collaboration $\mathcal{P}$ is distributedly realizable if and only if $C = \bigcup_{w \in C} \langle w \rangle$.*

Proof. If $C = \bigcup_{w \in C} \langle w \rangle$, Prop. 3 proves distributed realizability. The other way round, if there is some $w \in C$ with $\langle w \rangle \not\subseteq C$, Prop. 2 shows that w cannot be covered by any realization. □

It remains to find an effective way to check whether $C = \bigcup_{w \in C} \langle w \rangle$. This problem has, however, already been solved in trace theory [5]:

Proposition 5. *Let C be a choreography for a collaboration $\mathcal{P}$ and $A = \mathcal{A}(C)$ the minimal deterministic automaton that accepts the language of C. $C = \bigcup_{w \in C} \langle w \rangle$ if and only if, for all states q_1, q_2 of A and all distant events x and y, $q_1 \xrightarrow{xy} q_2$ implies $q_1 \xrightarrow{yx} q_2$.* □

Although we imported the result, we present the sketch of the proof for reasons of self-containedness. Minimal deterministic automata are linked to the Nerode relation $\sim_L$. For a language L, let $w_1 \sim_L w_2$ if, for all w, it holds that $w_1 w \in L$ if and only if $w_2 w \in L$. The main observation on the Nerode relation is that, in any automaton accepting L, $q_0 \xrightarrow{w_1} q$ and $q_0 \xrightarrow{w_2} q$ implies $w_1 \sim_L w_2$. For the minimal deterministic automaton accepting L, the reverse holds as well: If $w_1 \sim_L w_2$, $q_0 \xrightarrow{w_1} q_1$ and $q_0 \xrightarrow{w_2} q_2$ then $q_1 = q_2$. Applying this observation to our problem, we see that, for all sequences w and distant events a and b, we have $wab \sim_C wba$ thus proving the above result.

4.2 The asynchronous case

As Propositions 2 and 3 extend to the asynchronous case (i. e., $M_S = \emptyset$), so does Thm. 3.

Corollary 2. *A choreography C for a collaboration $\mathcal{P}$ that uses only asynchronous communication is distributedly realizable if and only if $C = \bigcup_{w \in C} \langle w \rangle$.*

The actual decision procedure requires some additional considerations, though. We start with reminding that no messages are pending after termination. That is, for all terminating runs w and all messages x, $\hat{x}(w) = 0$. This observation can be used for extending the $\hat{}$-notation to states of any automaton A that accepts C.

Lemma 2. *Let C be a distributedly realizable choreography. Let A be an automaton that accepts C. Assume that A does not have trap states; that is, states from which no final state of A is reachable. For all sequences w_1 and w_2, if $q_0 \xrightarrow{w_1} q$ and $q_0 \xrightarrow{w_2} q$ then, for all $x \in M_A$, $\hat{x}(w_1) = \hat{x}(w_2)$.*

Proof. Assume the contrary. As a final state is reachable from q, say by executing w, both $w_1 w$ and $w_2 w$ are accepted in A. One of these sequences has an unbalanced number of send and receive events for some x, violating the termination condition for compositions and thus contradicting distributed realizability of C. □

This observation yields a simple necessary condition for distributed realizability:

Corollary 3. *Let C be a choreography and A an automaton that has no trap states and accepts C. Then C is realizable only if the following system of equations has a unique and nonnegative solution. In the system, for each state q of A and $x \in M_A$, $\hat{q}(x)$ is a distinct variable and we impose the following equations.*

- $\hat{q}(x) = 0$, *for all $x \in M_A$ and all $q \in \{q_0\} \cup F$;*
- $\hat{q}(x) + 1 = \hat{q}'(x)$, *if $q \xrightarrow{!x} q'$;*
- $\hat{q}(x) - 1 = \hat{q}'(x)$, *if $q \xrightarrow{?x} q'$;*
- $\hat{q}(x) = \hat{q}'(x)$, *if $q \xrightarrow{y} q'$, $y \neq\, !x$, and $y \neq\, ?x$.*

In the following considerations, we assume that C passed this sanity check and thus employ the solution $\hat{q}(x)$ of the presented system of equations. Reflecting the restrictions for commutation of $!x$ and $?x$, we propose the following modification of Prop. 5.

Lemma 3. *Let C be a choreography for a collaboration $\mathcal{P}$ using asynchronous communication satisfying the condition established in Cor. 3 and let $A = \mathcal{A}(C)$ the minimal deterministic automaton that accepts the language of C. $C = \bigcup_{w \in C} \langle w \rangle$ if and only if, for all states q_1, q_2 of A and all distant events a and b:*

- *If there is no message x with $a = !x$ and $b = ?x$ then $q_1 \xrightarrow{ab} q_2$ implies $q_1 \xrightarrow{ba} q_2$*
- *If, for some message x, $a = !x$ and $b = ?x$, and $\hat{q_1}(x) > 0$ then $q_1 \xrightarrow{ab} q_2$ implies $q_1 \xrightarrow{ba} q_2$*

Proof. Again, the proof relies on the relation between the Nerode equivalence and the minimal deterministic automaton accepting C. Indeed, in all situations where the conditions for a and b are satisfied, we have $wab \sim_C wba$ thus justifying the stated diamond property in the automaton. In particular, condition $\hat{q_1}(x) > 0$ asserts that $!x$ and $?x$ commute. □

4.3 Complexity

Complexity depends on the assumptions to be imposed on the original representation of C. From most relevant choreography description languages we are aware of, it is easy to derive a finite automaton model for the choreography. Thus, we assume such an automaton for C to be given. On the other hand, we do not assume this automaton to be deterministic, let alone minimal. Thus, the costs for checking distributed realizability comprise the efforts for:

- transforming the given automaton into a minimal deterministic one. This involves the well known power set construction for transforming a nondeterministic automaton into a deterministic one which may cause exponential blow-up in the number of states;
- checking the diamond property of Prop. 5 or Lemma 3 which can be done in linear time with respect to the number of states of the automaton.

In the asynchronous case, we additionally need to solve the linear system of equations of Cor. 3 which requires, for tis particular system of equations, only linear time as well.

As the most costly step, transformation into a deterministic automaton, is well studied in the area of compiler construction, we believe that existing standard solutions will be sufficiently efficient for practice.

5 Complete realizability

Definition 11 (Complete realizability). *Let C be a choreography for a collaboration $\{P_1, \ldots, P_n\}$. The finite single-peer automata $A_1, \ldots, A_n$* completely realize *C if, for all i, A_i implements $\{P_i\}$ and $\mathcal{L}(A_1 \oplus \cdots \oplus A_n) = C$.*

Example. The choreography C_3 in Fig. 1 is completely realizable: Every specified conversation is implemented by the peers in Fig. 1(d).

5.1 The synchronous case

Assume $M_A = \emptyset$.

Theorem 4. *A choreography C for a collaboration $\{P_1, \ldots, P_n\}$ is completely realizable if and only if it is completely realized by $\mathcal{A}(C|_{P_1}), \ldots, \mathcal{A}(C|_{P_n})$,*

Proof. If the automata $\mathcal{A}(C|_{P_i})$ $(1 \leq i \leq n)$ completely realize C, nothing remains to be shown. So assume C is completely realizable, say, by finite single-peer automata $B_1, \ldots, B_n$. We show that C is also realized by the $\mathcal{A}(C|_{P_i})$ $(i = 1, \ldots, n)$.

We show first $C \subseteq \mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$. Let $w \in C$. Every automaton $\mathcal{A}(C|_{P_i})$ has $w|_{P_i}$ as one of its accepting runs. Consequently, w can be realized using a suitable scheduling of the events in $\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i})$.

Next, we show $C \supseteq \mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$. Let $w \in \mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$. When realizing w, automaton $\mathcal{A}(C|_{P_i})$ executes $w|_{P_i}$ $(i = 1, \ldots, n)$. By construction of these automata, this is only possible if there are conversations $w_i \in C$ $(i = 1, \ldots, n)$ such that $w_i|_{P_i} = w|_{P_i}$.

As the composition of the B_i realizes at least the conversations in C, they realize all the words w_i ($i = 1, \ldots, n$). In a run that produces w_i, automaton B_i executes the event sequence $w_i|_{P_i} = w|_{P_i}$. Consider now a run where each of the B_i executes the event sequence $w|_{P_i}$ in the order given by w. Globally, this run produces w. As the composition of the B_i realizes at most the conversations specified in C, we finally conclude $w \in C$. □

In contrast to the simplistic automata used in the previous sections, the composition of automata $\mathcal{A}(C|_{P_i})$ may contain deadlocks; that is, runs which cannot be extended in a nonfinal state. We show, however, that this is the case only if all complete realizations contain deadlocks.

Theorem 5. *A choreography C for the collaboration $\{P_1, \ldots, P_n\}$ is completely and deadlock freely realizable if and only if it is completely and deadlock freely realized by $\mathcal{A}(C|_{P_1}), \ldots, \mathcal{A}(C|_{P_n})$,*

Proof. In addition to the arguments in Thm. 4, it remains to be shown that every partial run in $\mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$ can be extended to a terminating run if that is possible in any complete realization $B_1 \oplus \cdots \oplus B_n$ of C. Let w_1 be a partial run in $\mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$ that does not end in a final state, thus $w_1 \notin C$. Using the same argument as for Thm. 4, we can show that w_1 is also the event sequence produced by some run in $B_1 \oplus \cdots \oplus B_n$ which cannot be a terminating run, because $w_1 \notin C$. Thus, $B_1 \oplus \cdots \oplus B_n$ is able to extend the run to a terminating run by executing an additional event sequence w_2 (i.e., $w_1 w_2 \in C$). As $\mathcal{L}(\bigoplus_{i=1}^{n} \mathcal{A}(C|_{P_i}))$ completely realizes C, $w_1 w_2$ is also executable here. Since all the $\mathcal{A}(C|_{P_i})$ are deterministic by definition, there is only one state that can be reached after having executed w_1. Hence, the unique state reached by the partial run w_1 enables the continuation w_2 and thus cannot be a deadlock. □

5.2 The asynchronous case

Under the asynchronous communication model (i. e., $M_S = \emptyset$), it is clear that, as for distributed realizability, the number of send and receive events must be balanced in terminating runs. Hence, we may import Cor. 3 from the previous section as a necessary condition for complete realizability. Assuming a choreography that meets this condition, the partial synchronization between send and corresponding receive events is fully reflected in the choreography. This means that, repeating the arguments for Thm. 4, a receive event is always activated in the composition of automata if that is locally the case. Other than this, there are no significant differences in the argument, and we may state:

Corollary 4. *A choreography C for a collaboration $\{P_1, \ldots, P_n\}$ using asynchronous communication is completely realizable if and only if the conditions established in Cor. 3 is satisfied and it is completely realized by $\mathcal{A}(C|_{P_1}), \ldots, \mathcal{A}(C|_{P_n})$,*

The same is true for the case of deadlock free realizability:

Corollary 5. *A choreography C for the collaboration $\{P_1, \ldots, P_n\}$ using asynchronous communication is completely and deadlock freely realizable if and only if he condition established in Cor. 3 is satisfied and it is completely and deadlock freely realized by $\mathcal{A}(C|_{P_1}), \ldots, \mathcal{A}(C|_{P_n})$,*

5.3 Complexity

Checking the condition of Cor. 3 can be done in linear time on any automaton representing C. The projection of C to an individual peer P_i amounts to replacing all events distant to P_i by τ and requires linear time for each P_i. The size of the composition is at most the product of the sizes of the components. Checking language equivalence is PSPACE-complete [7]. We have to leave open whether language equivalence can be done more efficiently in the case where C is checked against the composition of its projections.

For the deadlock free case, the resulting components must be determinized and minimized, with potential exponential blow-up, and the resulting composition must be checked for deadlock freedom which requires linear time in the size of the composition.

6 Related work

Realizability received much attention in recent literature, see [17] for a survey.

Complete realizability. Alur et al. [1] present necessary and sufficient criteria to deadlock freely realize a choreography specified by a set of message sequence charts (MSCs). Both synchronous and asynchronous communication is supported. Their proposed algorithms are very efficient, but are limited to acyclic choreography specifications, because the used MSC model does not support arbitrary iteration. Salaün and Bultan [16] investigate complete realizability of choreographies specified by collaboration diagrams. The authors express the realizability problem in terms of LOTOS and present a case study conducted with a LOTOS verification tool. Their approach tackles both synchronous and asynchronous communication (using bounded FIFO queues). Collaboration diagrams, however, provide only limited support for repetitive behavior (only single events can be iterated) and choices (events can be skipped, but complex decisions cannot be modeled). Hence, the reduction of the PCP is not applicable. These restrictions also apply to the results of Bultan and Fu [3] in which sufficient conditions for complete realizability of collaboration diagrams are elaborated. A tool to check the sufficient criteria of [3,6] is presented by Bultan et al. [2]. Using this tool, the authors showed that many collaboration diagrams in literature are not completely realizable. In fact, most of these models are, however, distributedly realizable. Realizability of conversation protocols by asynchronously communicating Büchi automata is examined by Fu et al. [6]. The authors define a necessary condition for complete realizability. One of the prerequisites, *synchronous compatibility*, heavily restricts asynchronous communication. Kazhamiakin and Pistore [9] study a variety of communication models and their impact on realizability. They provide an algorithm that finds the "simplest" communication model under which a given choreography can be completely realized.

Other realizability notions. Decker and Weske [4] study realizability of interaction Petri nets. To the best of our knowledge, it is the only approach in which (complete and partial) realizability is not defined in terms of languages. Instead, the authors require the peer implementations and the choreography to be branching bisimilar. This results in a stronger realizability notion which needs further investigations with respect to decidability issues.

We defined distributed realizability in an earlier paper [12], and to the best of our knowledge, this notion was not yet subject of other work. In the same paper, we showed that complete, distributed, and partial realizability can be approached using an algorithm to check for *distributed controllability* [18]. However, undecidability has been shown for this problem recently [19]. This result as such did, however, not directly imply undecidability of partial realizability.

7 Conclusion and open problems

We showed that partial realizability is undecidable if at least four (synchronous communication), respectively three (asynchronous communication) peers are involved. The result relies on the capability of expressing arbitrarily large chunks of distant, noninterfering events. This observation could lead, in future work, to decidable subproblems. Furthermore, the case of only two asynchronously communicating peers is left open. Also the case of mixed communication models requires further investigation.

Distributed realizability is decidable. Realizability only depends on the question whether the choreography is closed under the commutation of distant events. An apparent follow-up question would be whether it is possible to cover all specified sequences with *finitely many* implementations.

For complete realizability, we found the choreography projections to the respective peers to be a canonical realization. If that projection does not realize, no one else does. If the projections are transformed into deterministic automata, this result extends to the problem of deadlock free complete realizability.

The decision procedures suggested by our arguments depend on automata minimization, checking language equivalence, and other, trivially implementable checks. Hence, we assume that the decision procedures can be turned into tools with acceptable behavior on relevant instances.

Acknowledgment. We would like to thank Dietrich Kuske for a very helpful briefing in trace theory.

References

1. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence charts. IEEE Trans. Software Eng. 29(7), 623–633 (2003)
2. Bultan, T., Ferguson, C., Fu, X.: A tool for choreography analysis using collaboration diagrams. In: ICWS 2009. pp. 856–863. IEEE (2009)
3. Bultan, T., Fu, X.: Specification of realizable service conversations using collaboration diagrams. SOCA 2(1), 27–39 (2008)
4. Decker, G., Weske, M.: Local enforceability in interaction Petri nets. In: BPM 2007. pp. 305–319. LNCS 4714, Springer (2007)
5. Diekert, V.: The Book of Traces. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1995)
6. Fu, X., Bultan, T., Su, J.: Conversation protocols: a formalism for specification and verification of reactive electronic services. Theor. Comput. Sci. 328(1-2), 19–37 (2004)

7. Kanellakis, P.C., Smolka, S.A.: CCS expressions, finite state processes, and three problems of equivalence. Inf. Comput. 86(1), 43–68 (1990)
8. Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y.: Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation, W3C (Nov 2005), http://www.w3.org/TR/ws-cdl-10
9. Kazhamiakin, R., Pistore, M.: Analysis of realizability conditions for Web service choreographies. In: FORTE 2006. pp. 61–76. LNCS 4229, Springer (2006)
10. Kazhamiakin, R., Pistore, M., Santuari, L.: Analysis of communication models in Web service compositions. In: WWW 2006. pp. 267–276. ACM (2006)
11. Kleijn, H.C.M., Morin, R., Rozoy, B.: Event structures for local traces. Electr. Notes Theor. Comput. Sci. 16(2) (1998)
12. Lohmann, N., Wolf, K.: Realizability is controllability. In: WS-FM 2009. pp. 110–127. LNCS 6194, Springer (2010)
13. Mazurkiewicz, A.W.: Trace theory. In: Advances in Petri Nets. pp. 279–324. LNCS 255, Springer (1986)
14. OMG: Business Process Model and Notation (BPMN). FTF Beta 1 for Version 2.0, Object Management Group (2009), http://www.omg.org/spec/BPMN/2.0
15. Sakarovitch, J.: The "last" decision problem for rational trace languages. In: LATIN 1992. pp. 460–473. LNCS 583, Springer (1992)
16. Salaün, G., Bultan, T.: Realizability of choreographies using process algebra encodings. In: IFM 2009. pp. 167–182. LNCS 5423, Springer (2009)
17. Su, J., Bultan, T., Fu, X., Zhao, X.: Towards a theory of Web service choreographies. In: WS-FM 2007. pp. 1–16. LNCS 4937, Springer (2008)
18. Wolf, K.: Does my service have partners? LNCS ToPNoC II(5460), 152–171 (2009)
19. Wolf, K.: Decidability issues for decentralized controllability of open nets. In: AWPN 2010. pp. 124–129. CEUR Workshop Proceedings Vol. 643, CEUR-WS.org (2010)
20. Zaha, J.M., Barros, A.P., Dumas, M., Hofstede, A.H.M.t.: Let's dance: A language for service behavior modeling. In: OTM 2006. pp. 145–162. LNCS 4275, Springer (2006)