Realizability of pomsets via communicating automata

Roberto Guanciale
KTH Royal Institute of Technology, Sweden (robertog@kth.se)

Emilio Tuosto
Department of Informatics, University of Leicester, UK (emilio@le.ac.uk)

Pomsets are a model of concurrent computations introduced by Pratt. They can provide a syntax-oblivious description of semantics of coordination models based on asynchronous message-passing, such as Message Sequence Charts (MSCs). In this paper, we study conditions that ensure a specification expressed as a set of pomsets can be faithfully realized via communicating automata.

Our main contributions are (i) the definition of a realizability condition accounting for flexible termination, (ii) verification conditions for global specifications with "multi-threaded" participants, and (iii) the definition of realizability conditions that can be decided directly over pomsets. A positive by-product of our approach is the efficiency gain obtained when restricting to specific classes of choreographies characterisable in term of behavioural types.

1 Introduction

Asynchronous message-passing is a widely used paradigm to specify, design, and implement communication-centred applications or systems. This paradigm has been used at different abstraction levels, including formal models (e.g. π-calculus [21, 17] and communicating automata [6]), specification languages (e.g. *message-sequence charts* (MSCs) [22]), choreography languages (e.g. global calculus [7] and WS-CDL [24]), programming languages (e.g. actor models for Erlang, Scala, and Go).

Choreographic approaches are gaining momentum to handle the complexity of distributed systems [14]. These frameworks envisage two views: a global specification and a local one. The former defines the order and constraints under which messages are sent and received, while the local view defines the behavior of each participant. The composition of local participants should respect the global specification. In this setting, the *realizability* of the global specifications becomes a concern since there could be some specifications that are impossible to implement using the local views in a given communication model.

We propose a general semantic representation based on partially ordered multisets (pomsets) [19], capable of specifying global behaviors and analyze their realizability in terms of asynchronous message-passing. Our framework assumes asynchronous point-to-point communications and features a notion of realizability that

1. rules out systems where some participants cannot ascertain termination
2. admits multi-threaded participants
3. allows us to define syntax-oblivious conditions
4. can be decided by an analysis of the partial orders of communication events.

These features have several practical advantages. Indeed, by (1), we admit systems where participants may get stuck on some messages, only if that is specified in the global model. The use of multi-threaded participants (2) makes our framework more expressive than existing ones. Syntax independent conditions (3) are applicable to different global models. Finally, (4) enables the identification of design errors in global models rather than in execution traces where they are harder to analyse.

Submitted to:
ICE 2018

Outline Section 2 gives the basic definitions. Section 3 introduces the problems of realisability and sound termination and provides verification conditions in the style of [1]. Section 4 presents the sufficient conditions for realizability and sound termination that can be tested over partial orders. Section 5 discusses the complexity of the new verification conditions. Finally, Section 6 discusses related work and Section 7 draws some conclusions.

2 Pomsets and message-sequence charts

We collect the main definitions needed in the rest of the paper. The material of this section is not an original contribution[1] and it is presented only to make the paper self-contained borrowing and combining definitions and notations from [8, 1, 13, 6].

Let $\mathcal{P}$ be a set of *participants* (ranged over by A, B, etc.), $\mathcal{M}$ a set (of types) of *messages* (ranged over by m, x, etc.). We take $\mathcal{P}$ and $\mathcal{M}$ disjoint. Participants coordinate with each other by exchanging messages over *communication channels*, that are elements of the set $\mathcal{C} = (\mathcal{P} \times \mathcal{P}) \setminus \{(\mathsf{A},\mathsf{A}) \mid \mathsf{A} \in \mathcal{P}\}$ and we abbreviate $(\mathsf{A},\mathsf{B}) \in \mathcal{C}$ as $\mathsf{A}\,\mathsf{B}$. The set of *labels* $\mathcal{L}$ is defined by

$$\mathcal{L} = \mathcal{L}^! \cup \mathcal{L}^? \qquad \text{where} \qquad \mathcal{L}^! = \mathcal{C} \times \{!\} \times \mathcal{M} \qquad \text{and} \qquad \mathcal{L}^? = \mathcal{C} \times \{?\} \times \mathcal{M}$$

The elements of $\mathcal{L}^!$ and $\mathcal{L}^?$, outputs and inputs, respectively represent *sending* and *receiving* actions; we shorten $(\mathsf{A}\,\mathsf{B},!,\mathsf{m})$ as $\mathsf{A}\,\mathsf{B}!\mathsf{m}$ and $(\mathsf{A}\,\mathsf{B},?,\mathsf{m})$ as $\mathsf{A}\,\mathsf{B}?\mathsf{m}$ and let l, l', ... range over $\mathcal{L}$. The *subject* of an action is defined by

$$\mathsf{sbj}\big(\mathsf{A}\,\mathsf{B}!\mathsf{m}\big) = \mathsf{A} \quad \text{(the sender)} \qquad \text{and} \qquad \mathsf{sbj}\big(\mathsf{A}\,\mathsf{B}?\mathsf{m}\big) = \mathsf{B} \quad \text{(the receiver)}$$

We borrow the formalisation of partially-ordered multi-set of [8].

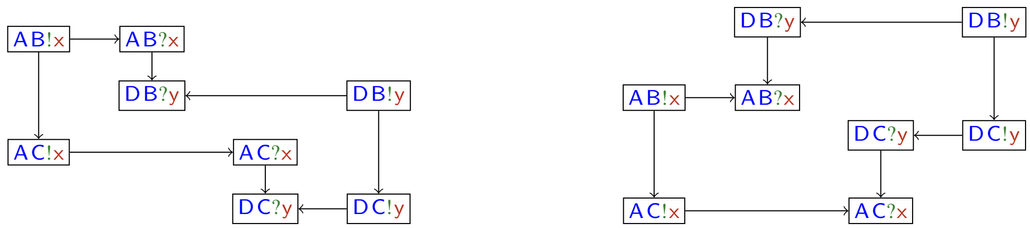
Definition 1 (Lposets)**.** *A labelled partially-ordered set (lposet) is a triple* $(\mathcal{E}, \leq, \lambda)$*, with* $\mathcal{E}$ *a set of events,* $\leq \subseteq \mathcal{E} \times \mathcal{E}$ *a reflexive, anti-symmetric, and transitive relation on* $\mathcal{E}$*, and* $\lambda : \mathcal{E} \to \mathcal{L}$ *a labelling function.*

Intuitively, $\leq$ represents causality; for $e \neq e'$, if $e \leq e'$ and both events occur then e' is caused by e. Note that λ is not required to be injective: for $e \neq e' \in \mathcal{E}$, $\lambda(e) = \lambda(e')$ means that e and e' model different occurrences of the same action.

Definition 2 (Pomsets)**.** *Two lposets* $(\mathcal{E}, \leq, \lambda)$ *and* $(\mathcal{E}', \leq', \lambda')$ *are* isomorphic *if there is a bijection* $\phi : \mathcal{E} \to \mathcal{E}'$ *such that* $e \leq e' \iff \phi(e) \leq' \phi(e')$ *and* $\lambda = \lambda' \circ \phi$*. A partially-ordered multi-set (of actions), pomset for short, is an isomorphism class of lposets.*

Using pomsets in place of lposets allows us to abstract away from the names of events in $\mathcal{E}$. In the following, $[\mathcal{E}, \leq, \lambda]$ denotes the isomorphism class of $(\mathcal{E}, \leq, \lambda)$, symbols $r, r', \ldots$ (resp. $R, R', \ldots$) range over (resp. sets of) pomsets, and we assume that any r contains at least one lposet which will possibly be referred to as $(\mathcal{E}_r, \leq_r, \lambda_r)$. An event e is an *immediate predecessor* of an event e' in a pomset r if $e \neq e'$, $e \leq_r e'$, and for all $e'' \in \mathcal{E}_r$ such that $e \leq_r e'' \leq_r e'$ either $e = e''$ or $e' = e''$. If e is an immediate predecessor of e' in r then e' is an *immediate successor* of e in r. We will represent pomsets as the Hasse diagram of the immediate predecessor relation as done in the examples of Fig. 1. For instance, in the pomset $r_{(1)_a}$ the input event of B from A immediately precedes the input of B from D while the events with those labels are in the reversed order in $r_{(1)_b}$.

[1] Except for the different definition of accepting states of communicating automata.



(a) $r_{(1)_a}$ (b) $r_{(1)_b}$

Figure 1: Two pomsets

Definition 3 (Projection of pomsets)**.** *The* projection $r\!\downharpoonright_{\mathsf{A}}$ of a pomset r on a participant $\mathsf{A} \in \mathcal{P}$ *is obtained by restricting r to the events having subject* A*: formally* $r\!\downharpoonright_{\mathsf{A}} = [\mathcal{E}_{r,\mathsf{A}},\ \leq_r \cap (\mathcal{E}_{r,\mathsf{A}} \times \mathcal{E}_{r,\mathsf{A}}),\ \lambda_r|_{\mathcal{E}_{r,\mathsf{A}}}]$ *where* $\mathcal{E}_{r,\mathsf{A}} = \{e \in \mathcal{E}_r \mid \mathit{sbj}(\lambda_r(e)) = \mathsf{A}\}$.

Pomsets are a quite expressive model of global views of choreographies [23]; in fact, MSCs[2] can be defined as a subclass of pomsets.

Definition 4 (Well-formedness, completeness, and MSCs)**.** *A pomset r over $\mathcal{L}$ is* well-formed *if for every event $e \in \mathcal{E}_r$*

1. *if $\lambda_r(e) = \mathsf{A\,B!m}$, there is at most one $e' \in \mathcal{E}_r$ immediate successor of e in r with $\lambda_r(e') = \mathsf{A\,B?m}$ (and, if such e' exists, we say that e and e'* match *each other)*
2. *if $\lambda_r(e) = \mathsf{A\,B?m}$, there exists exactly one $e' \in \mathcal{E}_r$ immediate predecessor of e in r with $\lambda_r(e') = \mathsf{A\,B!m}$*
3. *for each $e' \in \mathcal{E}_r$, if e is an immediate predecessor of e' and $\mathit{sbj}(\lambda_r(e)) \neq \mathit{sbj}(\lambda_r(e'))$ then e and e' are matching output and input events respectively*
4. *for each $e' \neq e \in \mathcal{E}_r$ with $\lambda_r(e) = \lambda_r(e') = \mathsf{A\,B!m}$, and for all $\bar{e}, \bar{e}' \in \mathcal{E}_r$ immediate successors in r of e and of e' respectively if $\lambda_r(\bar{e}) = \lambda_r(\bar{e}') = \mathsf{A\,B?m}$ and $e \leq_r e'$ then $\bar{e}' \not\leq_r \bar{e}$*

Pomset r is complete *if there is no send event in $\mathcal{E}_r$ without a matching receive event.*

A message-sequence chart *is a well-formed and complete pomset r such that $\leq_{r\downharpoonright_{\mathsf{A}}}$ is a total order, for every $\mathsf{A} \in \mathcal{P}$.*

Well-formed pomsets permit to represent inter-participant concurrency by not imposing orders of events belonging to different participants that are not matching communications. Also, well-formed pomsets allow intra-participant concurrency (i.e. multi-threaded participants) since they do not require $\leq_{r\downharpoonright_{\mathsf{A}}}$ to be totally ordered. MSCs are obtained by restricting participants to be single-threaded. The pomsets in Fig. 1 are indeed MSCs describing different orders of the same set of events. The total orders yielded by the vertical arrows correspond to the *projections* of the pomsets on participants; these projections are obtained by restricting $r_{(1)_a}$ and $r_{(1)_b}$ to the events having the same subject. More precisely, the projection on one of the participants consists of the i-th vertical arrow where i is the alphabetical order of the participant (e.g., the projection of C is the third arrow). The behaviour of A (and D) is the same in both MSCs: A (resp. D) first sends message x (resp. y) to B and then to C. The behaviour of B (and C) differs: in $r_{(1)_a}$, B first receives the message from A then the one from D, in $r_{(1)_b}$, B has the same interactions but in opposite order. Likewise for C.

[2] Pomsets can also be used to give semantics to the composition of MSCs; see [13].

To handle choices we use sets of pomsets R so that each possible branch is represented by a pomset $r \in R$ representing the causal dependencies of the communication actions of the branch. For instance, the set $R_{(1)} = \{r_{(1)_a}, r_{(1)_b}\}$ represents a choice between the fact that B may receive messages x and y in any order.

A natural question to ask is:

"is it possible to realize $R_{(1)}$ with asynchronously communicating local views?"

The next section answers this question for pomsets similarly to what done in [1] where closure conditions for MSCs where identified.

3 Realizability and termination soundness of pomsets

Hereafter we assume all structures, including languages, words and pomsets, to be finite. Given a pomset r, a *linearization* of r is a string in $\mathcal{L}^\star$ obtained by considering a total ordering of the events $\mathcal{E}_r$ that is consistent with the partial order $\leq_r$, and then replacing each event by its label. More precisely, let $|\ \mathcal{E}_r\ |$ be the cardinality of $\mathcal{E}_r$, a word $w = \lambda_r(e_1)\dots\lambda_r(e_{|\mathcal{E}_r|})$ is a *linearization* of a pomset r if $e_1 \dots e_{|\mathcal{E}_r|}$ is a permutation that totally orders the events in $\mathcal{E}_r$ so that if $e_i \leq_r e_j$ then $i \leq j$. For a pomset r, define $\mathbb{L}(r)$ to be the set of all linearizations of r. A word w over $\mathcal{L}$ is *well-formed* (resp. *complete*) if it is the linearization of a well-formed (resp. *complete*) pomset. Hereafter, for a word $w \in \mathcal{L}^\star$, $w\downharpoonright_{\mathsf{A}}$ denotes the projection of w that retains only those events where participant $\mathsf{A} \in \mathcal{P}$ is the subject. Operation $\_\downharpoonright_{\mathsf{A}}$ acts element-wise on languages over $\mathcal{L}$. The *language* of a set of pomsets R is simply defined as $\mathbb{L}(R) = \bigcup_{r \in R} \mathbb{L}(r)$.

Local views are often conveniently modelled in terms of *communicating automata* of some sort. An A*-communicating finite state machine (*A*-CFSM)* $M = (Q, q_0, F, \to)$ is a finite-state automaton on the alphabet $\mathcal{L}$ such that, $q_0 \in Q$ is the initial state, $F \subseteq Q$ are the accepting states, and for each $q \xrightarrow{l} q'$ holds $\mathsf{sbj}(l) = \mathsf{A}$. A *(communicating) system* is a map $S = (M_\mathsf{A})_{\mathsf{A} \in \mathcal{P}}$ assigning an A-CFSM M_A to each participant $\mathsf{A} \in \mathcal{P}$. For all $\mathsf{A} \neq \mathsf{B} \in \mathcal{P}$, we shall use an unbounded multiset $b_{\mathsf{A}\,\mathsf{B}}$ where M_A puts the message to M_B and from which M_B consumes the messages from M_A.

The semantics of communicating systems is defined in terms of transition relations between *configurations* which keep track of the state of each machine and the content of each buffer. Let $S = (M_\mathsf{A})_{\mathsf{A} \in \mathcal{P}}$ be a communicating system. A *configuration* of S is a pair $s = \langle \vec{q}\ ;\ \vec{b} \rangle$ where $\vec{q} = (q_\mathsf{A})_{\mathsf{A} \in \mathcal{P}}$ maps each participant A to its local state $q_\mathsf{A} \in Q_\mathsf{A}$ and $\vec{b} = (b_{\mathsf{A}\,\mathsf{B}})_{\mathsf{A}\,\mathsf{B} \in C}$ where the buffer $b_{\mathsf{A}\,\mathsf{B}} : \mathcal{M} \to \mathbb{N}$ is a map assigning the number of occurrences of each message; state q_A keeps track of the state of the automaton M_A and buffer $b_{\mathsf{A}\,\mathsf{B}}$ keeps track of the messages sent from A to B. The *initial* configuration s_0 is the one where, for all $\mathsf{A} \in \mathcal{P}$, q_A is the initial state of the corresponding CFSM and all buffers are empty. Given two configurations $s = \langle \vec{q}\ ;\ \vec{b} \rangle$ and $s' = \langle \vec{q'}\ ;\ \vec{b'} \rangle$, relation $s \xRightarrow{l} s'$ holds if there is a message $\mathsf{m} \in \mathcal{M}$ such that either (1) or (2) below holds:

1. $l = \mathsf{A}\,\mathsf{B}!\mathsf{m}$ and $q_\mathsf{A} \xrightarrow{l}_\mathsf{A} q'_\mathsf{A}$ and
   a. $q'_\mathsf{C} = q_\mathsf{C}$ for all $\mathsf{C} \neq \mathsf{A} \in \mathcal{P}$ and
   b. $b'_{\mathsf{A}\,\mathsf{B}} = b_{\mathsf{A}\,\mathsf{B}}[\mathsf{m} \mapsto b_{\mathsf{A}\,\mathsf{B}}(\mathsf{m}) + 1]$
2. $l = \mathsf{A}\,\mathsf{B}?\mathsf{m}$ and $q_\mathsf{B} \xrightarrow{l}_\mathsf{B} q'_\mathsf{B}$ and
   a. $q'_\mathsf{C} = q_\mathsf{C}$ for all $\mathsf{C} \neq \mathsf{B} \in \mathcal{P}$ and
   b. $b_{\mathsf{A}\,\mathsf{B}}(\mathsf{m}) > 0$ and $b'_{\mathsf{A}\,\mathsf{B}} = b_{\mathsf{A}\,\mathsf{B}}[\mathsf{m} \mapsto b_{\mathsf{A}\,\mathsf{B}}(\mathsf{m}) - 1]$

where, $f[x \mapsto y]$ is the usual notation for the updating of a function f in a point x of its domain with a value y. Condition (1) puts m on channel $\mathsf{A}\,\mathsf{B}$, while (2) gets m from channel $\mathsf{A}\,\mathsf{B}$ by simply updating the number of occurrences of m in the buffer $b_{\mathsf{A}\,\mathsf{B}}$. In both cases, any machine or buffer not involved in the transition is left unchanged in the new configuration s'.

The automata model adopted in [1] is a slight variant of *communicating-finite state machines* (CFSMs) [6]. The two models have the same definition of automata; they differ in how communication is attained, but are equivalent up to internal transitions (which in [1] have been used to simplify proofs). We used the definition of CFSMs in [6] to encompass accepting states (necessary to define our more flexible notion of correct termination). Another minor deviation from the definition of CFSMs introduced in [6] is that buffers become multisets in [1] while in [6] they follow a FIFO policy.

Given a communicating system S, a configuration $s = \langle \vec{q} \,;\, \vec{b} \rangle$ of S is (*i*) *accepting* if all buffers in $\vec{b}$ are empty and the local state $\vec{q}(\mathsf{A})$ of each participant A is accepting while (*ii*) s is a *deadlock* if no accepting configuration is reachable from s. We can then define the *language of S* as the set $\mathbb{L}(S) \in \mathcal{L}^\star$ of sequences $l_0 \dots l_{n-1}$ such that $s_0 \xRightarrow{l_0} \dots \xRightarrow{l_{n-1}} s_n$ and s_n is an accepting configuration.

The notion of *realizability* and *sound termination* (cf. Definitions 5 and 6 below) are given in terms of the relation between the *language* of the global view and the one of a system of local views "implementing" it. Our notion of realizability considers languages over $\mathcal{L}$ as sets of traces of the distributed executions of some CFSMs, analogously to [1].

Definition 5 (Realizability)**.** *A language* $\mathbb{L} \subseteq \mathcal{L}^\star$ *is* weakly realizable *if there is a communicating system* S *such that* $\mathbb{L} = \mathbb{L}(S)$*; when* S *is deadlock-free we say that* $\mathbb{L}$ *is* safely realizable*. A set of pomsets* R *is* weakly *(resp.* safely*)* realizable *if* $\mathbb{L}(R)$ *is weakly (resp. safely) realizable.*

The notion of realizability is meaningful when pomsets are *well-formed* and *complete*, namely when they yield a proper match among receive and send events.

In general, safe realizability is not enough to rule out undesirable designs. In fact, it admits systems where participants cannot ascertain termination and may be left waiting forever for some messages. This may lead non-terminating participants to unnecessarily lock resources once the coordination is completed. We explain this considering Fig. 2 which can be interpreted as follows. Participant A starts a transaction with B by sending message x. Pomset r_{2_a} represents a scenario where the transaction is left open. Pomset r_{2_b} represents a scenario where the transaction is opened and eventually committed. Yet, B is uncertain whether message y is going to be sent or not while C may be left waiting for message z when B does not receive y. However, depending on the application requirements, it may be the case that termination awareness is important for B and not for C because e.g., either C is not "wasting" resources or it is immaterial that such resources are left locked. To handle this limitation we introduce a novel termination condition, which allows to specify the subset of participants that should be able to identify when no further message can be exchanged.

Definition 6 (Termination soundness)**.** *A participant* $\mathsf{A} \in \mathcal{P}$ *is* termination-unaware *in a system* S *if there exists an accepting configuration* $\langle \vec{q} \,;\, \vec{b} \rangle$ *reachable in* S *having a transition departing from* $\vec{q}(\mathsf{A})$ *that is labelled in* $\mathcal{L}^?$.

A set of participants $\mathcal{P}' \subseteq \mathcal{P}$ *is* termination-aware *in a system* S *there is no* $\mathsf{A} \in \mathcal{P}$ *that is termination-unaware in* S*. A language* $\mathbb{L}$ *over* $\mathcal{L}$ *is* termination-sound *for* $\mathcal{P}' \subseteq \mathcal{P}$ *if* $\mathbb{L}$ *is safely realizable by a system for which* $\mathcal{P}'$ *is termination-aware. A set of pomsets* R *is* termination-sound *for* $\mathcal{P}'$ *if* $\mathbb{L}(R)$ *is termination-sound for* $\mathcal{P}'$.

Realizability and termination soundness can be established by analyzing verification conditions of the language. In [1] two closure conditions are introduced that entail weak and safe realizability. A word w over $\mathcal{L}$ is *$\mathcal{P}$-feasible* for $\mathbb{L} \subseteq \mathcal{L}^\star$ if $\forall \mathsf{A} \in \mathcal{P} : \exists w' \in \mathbb{L} : w\!\downharpoonright_{\mathsf{A}} = w'\!\downharpoonright_{\mathsf{A}}$. A language $\mathbb{L}$ over the alphabet $\mathcal{L}$ has[3] the closure condition **CC2** when

$$\mathbb{L} \supseteq \{w \in \mathcal{L}^\star \mid w \text{ well-formed, complete, and } \mathcal{P}\text{-feasible for } \mathbb{L}\}$$

[3] We stick with the terminology in [1] where closure conditions are not given specific names.

Intuitively, **CC2** entails that $\mathbb{L}$ is realizable by the set of participants performing the actions in $\mathcal{L}$: if each participant cannot tell apart a trace w with one of its expected executions (i.e., those in $\mathbb{L}$) then w must be in $\mathbb{L}$ or, in the terminology of [1], w is *implied*. Closure condition **CC2** characterises the class of weakly realizable languages over $\mathcal{L}$.

Theorem 1 ([1])**.** *A language $\mathbb{L}$ is weakly realizable if, and only if, $\mathbb{L}$ contains only well-formed and complete words and satisfies **CC2**.*

The language of the set of pomsets $\{r_{(1)_a}, r_{(1)_b}\}$ of Fig. 1 is not closed under **CC2**. In fact, the well-formed and complete word

$$\mathsf{A\,B!x};\ \mathsf{A\,B?x};\ \mathsf{D\,B!y};\ \mathsf{D\,B?y};\ \mathsf{D\,C!y};\ \mathsf{D\,C?y};\ \mathsf{A\,C!x};\ \mathsf{A\,C?x} \tag{1}$$

satisfies the conditions of **CC2**, because the projection of the word (1) on each participant equals the projection of a linearization of $r_{(1)_a}$ or of $r_{(1)_b}$ on the same participant. However, (1) is not in the language $\mathbb{L}(R_{(1)})$, because $\mathsf{A\,C?x}$ must precede $\mathsf{D\,C?y}$ in all the words obtained by the linearization of $r_{(1)_a}$, while in those obtained by a linearization of $r_{(1)_b}$, $\mathsf{D\,B?y}$ must precede $\mathsf{A\,B?x}$.

The realizability entailed by condition **CC2** is "weak" because it does not rule out possibly deadlocking systems. Therefore, an additional closure condition, dubbed **CC3**, has been identified in [16, 1]. A language $\mathbb{L}$ over the alphabet $\mathcal{L}$ has the closure condition **CC3** when

$$\mathsf{pref}(\mathbb{L}) \supseteq \{w \in \mathcal{L}^\star \mid w \text{ well-formed and } \mathcal{P}\text{-feasible for } \mathsf{pref}(\mathbb{L})\}$$

where $\mathsf{pref}(\mathbb{L})$ is the prefix closure of $\mathbb{L}$. Basically, condition **CC3** states that any (potentially partial) execution that cannot be told apart by any of the participants is a (partial) execution in $\mathbb{L}$. And now the following result characterises safe realizability.

Theorem 2 ([16, 1])**.** *A language $\mathbb{L}$ is safe realizable if, and only if, $\mathbb{L}$ contains only well-formed and complete words and satisfies **CC2** and **CC3**[4].*

Once a language $\mathbb{L}$ is known to be realizable, we get a system $S(\mathbb{L}) = (M_{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}}$ realizing $\mathbb{L}$ by defining, for all $\mathsf{A} \in \mathcal{P}$

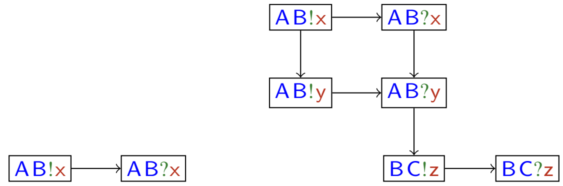
$$M_{\mathsf{A}} = (\mathsf{pref}(\mathbb{L}\downharpoonright_{\mathsf{A}}), \varepsilon, \mathbb{L}\downharpoonright_{\mathsf{A}}, \rightarrow)$$

where $w \xrightarrow{l} w.l$ if $w.l \in \mathsf{pref}(\mathbb{L}\downharpoonright_{\mathsf{A}})$. Then, in [1] the following result is shown.

Theorem 3 ([1])**.** *If $\mathbb{L}$ is a weakly realizable language then $\mathbb{L}(S(\mathbb{L})) = \mathbb{L}$. Moreover, if $\mathbb{L}$ is safely realizable then $S(\mathbb{L})$ is deadlock-free.*

We introduce a new verification condition for termination soundness. A participant $\mathsf{A} \in \mathcal{P}$ is *termination-unaware* for the language $\mathbb{L}$ over $\mathcal{L}$ if there exist $w, w' \in \mathbb{L}$ such that $w\downharpoonright_{\mathsf{A}}$ is a prefix of $w'\downharpoonright_{\mathsf{A}}$ and the first symbol in $w'\downharpoonright_{\mathsf{A}}$ after $w\downharpoonright_{\mathsf{A}}$ is in $\mathcal{L}^?$. Given a set of participants $\mathcal{P}' \subseteq \mathcal{P}$, we say that $\mathbb{L}$ is *$\mathcal{P}'$-terminating* when there is no $\mathsf{A} \in \mathcal{P}'$ termination-unaware for $\mathbb{L}$. The language of the family of pomsets $\{r_{2_a}, r_{2_b}\}$ of Fig. 2 is $\{\mathsf{A}\}$-terminating, but it is not $\{\mathsf{B}\}$-terminating. In fact, after receiving the message $\mathsf{A\,B?x}$, participant B cannot distinguish whether A terminates or will send $\mathsf{A\,B!y}$; hence B ends up in a state where the input $\mathsf{A\,B?y}$ is enabled. And likewise for C.

[4] The theorem in [1] describes a different condition, **CC2'**, which is easier to implement and is equivalent to **CC2** when in conjunction with **CC3**



(a) r_{2_a} (b) r_{2_b}

Figure 2: A set of two pomsets that is not termination sound for B or C

Theorem 4. *Given $\mathcal{P}' \subseteq \mathcal{P}$, if $\mathbb{L}$ is $\mathcal{P}'$-terminating and safely realizable then it is termination-sound for $\mathcal{P}'$.*

Proof. The proof is trivial. Let $S(\mathbb{L})$ be the system obtained from the construction of Theorem 3. $S(\mathbb{L})$ is deadlock-free and $\mathbb{L} = \mathbb{L}(S(\mathbb{L}))$. Let $\mathsf{A} \in \mathcal{P}'$, $w \in \mathbb{L}$, and s an accepting configuration reached in a run of S corresponding to w. For each $w' \in \mathbb{L}$ such that $w\!\downarrow_{\mathsf{A}}$ is prefix of $w'\!\downarrow_{\mathsf{A}}$, the first symbol in $w'\!\downarrow_{\mathsf{A}}$ after $w\!\downarrow_{\mathsf{A}}$ cannot be an input (since $\mathbb{L}$ is $\mathcal{P}'$-terminating). Therefore, by construction of $S(\mathbb{L})$, there is no input transition departing from the local state of A in s. $\square$

4 Pomset based verification conditions

We introduce a different approach to check realizability and sound termination of specifications, which does not require to explicitly compute the language of the family of pomsets. This allows us to avoid the combinatorial explosion due to interleavings. The main strategy is to provide alternative definitions of closures directly on pomsets which handle both *intra-* and *inter-participant* concurrency. Besides theoretical benefits, this yields a clear advantage for practitioners. In fact, design errors can be identified and confined in more abstract models, closer to the global specification than to traces of execution. Also, since the verification conditions are syntax-oblivious, since they require to analyze sets of pomsets, that provide a general framework to represent semantics of coordination models based on asynchronous message-passing. As discussed in Section 5, our conditions strictly entail the corresponding ones in Section 3

Definition 7 (Closure)**.** *Let ρ be a function from $\mathcal{P}$ to pomsets and $(r^{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}}$ be the tuple where $r^{\mathsf{A}} = \rho(\mathsf{A})\!\downarrow_{\mathsf{A}}$ for all $\mathsf{A} \in \mathcal{P}$. The inter-participant closure $\square((r^{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}})$ is the set of all well-formed pomsets $[\bigcup_{\mathsf{A}\in\mathcal{P}} \mathcal{E}_{r^{\mathsf{A}}},\quad \leq_I \cup \bigcup_{\mathsf{A}\in\mathcal{P}} \leq_{r^{\mathsf{A}}},\quad \bigcup_{\mathsf{A}\in\mathcal{P}} \lambda_{r^{\mathsf{A}}}]$ where $\leq_I \subseteq \{(e^{\mathsf{A}}, e^{\mathsf{B}}) \in \mathcal{E}_{r^{\mathsf{A}}} \times \mathcal{E}_{r^{\mathsf{B}}}, \mathsf{A}, \mathsf{B} \in \mathcal{P} \mid \lambda_{r^{\mathsf{A}}}(e^{\mathsf{A}}) = \mathsf{A}\,\mathsf{B}!\mathsf{m}, \lambda_{r^{\mathsf{B}}}(e^{\mathsf{B}}) = \mathsf{A}\,\mathsf{B}?\mathsf{m}\}$.*

Informally, the inter-participant closure takes one pomset for every participant and generates all “acceptable” matches between output and input events. We use Fig. 3 and Fig. 4 to illustrate the inter-participant closure. The singleton $R_{(3)}$ contains one pomset that is the composition of two independent pomsets: $r_{(3)_a}$ and $r_{(3)_b}$. Intuitively, this represents two concurrent “threads” (hereafter left and right threads) that have no interdependencies. Let r^{A} be the projection of the single pomset in $R_{(3)}$ for $\mathsf{A} \in \mathcal{P}$, then the inter-participant closure of $(r^{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}}$ consists of the two pomsets of Fig. 4, the one that uses the black and green dependencies, and the one that uses the black and red dependencies.

Definition 8. *A* pomset *r* is *less permissive* than pomset *r' (or r'* is more permissive than *r, written $r \sqsubseteq r'$) when $\mathcal{E}_r = \mathcal{E}_{r'}$, $\lambda_r = \lambda_{r'}$, and $\leq_r \supseteq \leq_{r'}$.*

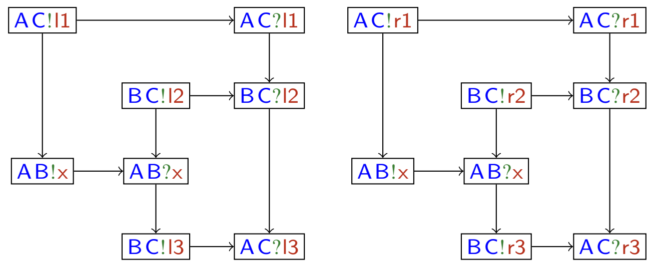


Figure 3: $R_{(3)} = \{[\mathcal{E}_{r_{(3)_a}} \cup \mathcal{E}_{r_{(3)_b}}, \leq_{r_{(3)_a}} \cup \leq_{r_{(3)_b}}, \lambda_{r_{(3)_a}} \cup \lambda_{r_{(3)_b}}]\}$

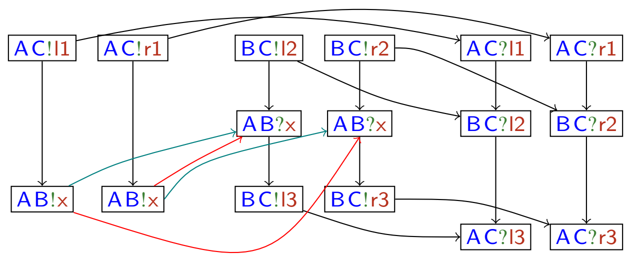


Figure 4: Inter-participant closure of pomset of Fig. 3

Lemma 1. *If $r \sqsubseteq r'$ then $\mathbb{L}(r) \subseteq \mathbb{L}(r')$.*

Definition 9 (CC2-POM). *A set of pomsets R over $\mathcal{L}$ satisfies closure condition **CC2-POM** if for all tuples $(r^{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}}$ of pomsets of R, for every pomset $r \in \square((r^{\mathsf{A}}\downharpoonright_{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}})$, there exists $r' \in R$ such that $r \sqsubseteq r'$.*

Intuitively, Definition 9 requires that if all the possible executions of a pomset cannot be distinguished by any of the participants of R, then those executions must be part of the language of R. Theorem 5 below shows that **CC2-POM** entails **CC2**; its proof is based on "counting" the number of events with a certain label l preceding an event e in the order $\leq_r$ of a pomset r: we write $\sharp_l^r(e)$ for such number (namely, $\sharp_l^r(e)$ is the cardinality of $\{e' \in \mathcal{E}_r \mid e' \leq_r e \wedge \lambda_r(e') = l\}$).

Theorem 5. *If R satisfies **CC2-POM** then $\mathbb{L}(R)$ satisfies **CC2**.*

Proof. Let w be a well-formed and complete word over $\mathcal{L}$ that satisfies hypothesis of **CC2**: for every participant $\mathsf{A} \in \mathcal{P}$ there exists $w^{\mathsf{A}} \in \mathbb{L}(R)$ for which $w\downharpoonright_{\mathsf{A}} = w^{\mathsf{A}}\downharpoonright_{\mathsf{A}}$. Then, for each $\mathsf{A} \in \mathcal{P}$, there is a pomset $r^{\mathsf{A}} \in R$ such that a linearization ℓ_{A} of r^{A} yields w^{A}. We can hence take the pomset

$$r = \left[\bigcup_{\mathsf{A}\in\mathcal{P}} \mathcal{E}_{r^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}, \quad \leq_I \cup \bigcup_{\mathsf{A}\in\mathcal{P}} \leq_{r^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}, \quad \bigcup_{\mathsf{A}\in\mathcal{P}} \lambda_{r^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}\right]$$

where

$$\leq_I = \bigcup_{\mathsf{B}\neq\mathsf{A}\in\mathcal{P}} \left\{ (e^{\mathsf{A}}, e^{\mathsf{B}}) \in \mathcal{E}_{r^{\mathsf{A}}\downharpoonright_{\mathsf{A}}} \times \mathcal{E}_{r^{\mathsf{B}}\downharpoonright_{\mathsf{B}}} \;\middle|\; \begin{array}{l} \lambda_{r^{\mathsf{A}}}(e^{\mathsf{A}}) = \mathsf{AB!m} \textit{ and } \lambda_{r^{\mathsf{B}}}(e^{\mathsf{B}}) = \mathsf{AB?m} \\ \textit{and } \sharp^{\ell_{\mathsf{A}}}_{\mathsf{AB!m}}(e^{\mathsf{A}}) = \sharp^{\ell_{\mathsf{B}}}_{\mathsf{AB?m}}(e^{\mathsf{B}}) \end{array} \right\}$$

The pomset r is in $\square((r^{\mathsf{A}}\downharpoonright_{\mathsf{A}})_{\mathsf{A}\in\mathcal{P}})$, since it is well-formed and complete and $\leq_I$ satisfies conditions of Definition 7. In fact, since w is well-formed and complete, all send and receive events have corresponding matching events. Also by construction, $w \in \mathbb{L}(r)$ and, for every A, $r\downharpoonright_{\mathsf{A}} \sqsubseteq r^{\mathsf{A}}\downharpoonright_{\mathsf{A}}$. Finally, by **CC2-POM** there exists $r' \in R$ such that $r \sqsubseteq r'$, therefore $w \in \mathbb{L}(r')$ hence $w \in \mathbb{L}(R)$. $\square$

Fig. 3 provides an example of a family of pomsets that cannot be weakly implemented. An execution of this specification can be as follows:

1. the left thread of A executes $\mathsf{A\,C!l_1}$ and $\mathsf{A\,B!x}$
2. the right thread of B executes $\mathsf{B\,C!r_2}$ and $\mathsf{A\,B?x}$, "stealing" the message x generated by the left thread of A and meant for the left thread of B
3. the right thread of B executes $\mathsf{B\,C!r_3}$.

This violates the constraint that event $\mathsf{A\,B!r_1}$ must always precede event $\mathsf{B\,C!r_3}$, which the specification imposes independently of the interleaved execution of the participants' threads. Indeed, $R_{(3)}$ does not satisfy **CC2-POM**. In fact, there are two well-formed and complete pomsets that satisfy the hypothesis of **CC2-POM**: the pomset of Fig. 4 that uses the black and green dependencies, and the one that uses the black and red dependencies. Condition **CC2-POM** is violated because there is no pomset in $R_{(3)}$ that is more permissive than the pomset using the red dependencies.

The next condition requires to introduce the concept of *prefix* of a pomset r, which is a pomset r' on a subset of the events of r that preserves the order and labelling of r; formally (following [13])

Definition 10 (Prefix pomsets)**.** *A pomset $r' = [\mathcal{E}', \leq', \lambda']$ is a* prefix *of pomset $r = [\mathcal{E}, \leq, \lambda]$ if there exists a label preserving injection $\phi : \mathcal{E}' \to \mathcal{E}$ such that $\phi(\leq') = \leq \cap (\mathcal{E} \times \phi(\mathcal{E}'))$*

We remark that an arbitrary sub-pomset satisfies the weaker condition $\phi(\leq') = \leq \cap (\phi(\mathcal{E}') \times \phi(\mathcal{E}'))$. Instead, $\phi(\leq') = \leq \cap (\mathcal{E} \times \phi(\mathcal{E}'))$ prevents events in $\mathcal{E} \setminus \phi(\mathcal{E}')$ from preceding events in $\phi(\mathcal{E}')$ and it is equivalent to say that for all $e' \in \mathcal{E}'$ if there is $e \leq \phi(e')$ then there exists $e'' \in \mathcal{E}'$ such that $\phi(e'') = e$ and $e'' \leq' e'$.

Lemma 2. *Let r be a pomset over $\mathcal{L}$ and w be a word in $\mathcal{L}^\star$, $w \in \mathsf{pref}(\mathbb{L}(r))$ iff exists a prefix r' of r such that $w \in \mathbb{L}(r')$.*

Definition 11 (**CC3-POM**)**.** *A set of pomsets R over $\mathcal{L}$ satisfies closure condition* ***CC3-POM*** *if for all tuples of pomsets $(\bar{r}^{\mathsf{A}})_{\mathsf{A} \in \mathcal{P}}$ such that for every A $\bar{r}^{\mathsf{A}}$ is a prefix of a pomset $r^{\mathsf{A}} \in R$, and for every pomset $\bar{r} \in \square((\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}})_{\mathsf{A} \in \mathcal{P}})$ there is a pomset $r' \in R$ and a prefix $\bar{r}'$ of r' such that $\bar{r} \sqsubseteq \bar{r}'$.*

Theorem 6. *If R satisfies* ***CC3-POM*** *then $\mathbb{L}(R)$ satisfies* ***CC3****.*

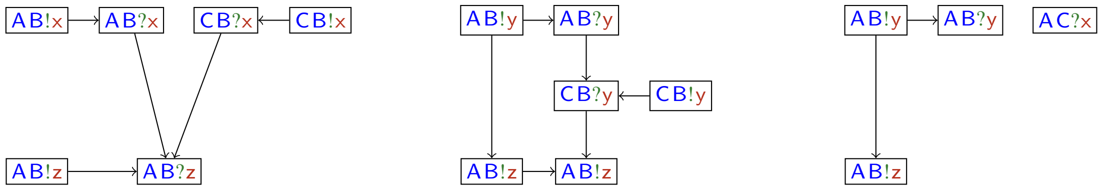
Proof. Let w be a word that satisfies hypothesis of **CC3**: for every participant $\mathsf{A} \in \mathcal{P}$, there exists a word $w^{\mathsf{A}} \in \mathsf{pref}(\mathbb{L}(R))$ such that $w\downharpoonright_{\mathsf{A}} = w^{\mathsf{A}}\downharpoonright_{\mathsf{A}}$. Therefore, there is a pomset $\bar{r}^{\mathsf{A}}$ prefix of a pomset $r^{\mathsf{A}} \in R$ such that $w^{\mathsf{A}} \in \mathbb{L}(\bar{r}^{\mathsf{A}})$ and let ℓ_{A} be one of the linearizations of $\bar{r}^{\mathsf{A}}$ that corresponds to w^{A}. Define

$$\bar{r} = \left[\bigcup_{\mathsf{A} \in \mathcal{P}} \mathcal{E}_{\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}, \qquad \leq_I \cup \bigcup_{\mathsf{A} \in \mathcal{P}} (\leq_{\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}), \quad \bigcup_{\mathsf{A} \in \mathcal{P}} \lambda_{\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}}}, \right]$$

where

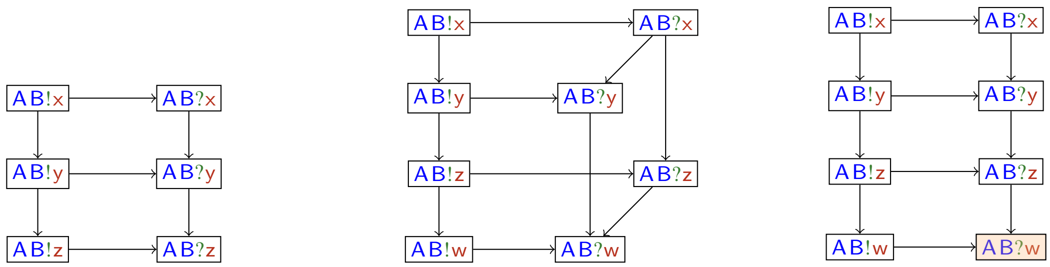
$$\leq_I = \bigcup_{\mathsf{B} \neq \mathsf{A} \in \mathcal{P}} \left\{ (e^{\mathsf{A}}, e^{\mathsf{B}}) \in \mathcal{E}_{\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}}} \times \mathcal{E}_{\bar{r}^{\mathsf{B}}\downharpoonright_{\mathsf{B}}} \;\middle|\; \begin{array}{l} \lambda_{\bar{r}^{\mathsf{A}}}(e^{\mathsf{A}}) = \mathsf{A\,B!m} \textit{ and } \lambda_{\bar{r}^{\mathsf{B}}}(e^{\mathsf{B}}) = \mathsf{A\,B?m} \\ \textit{and } \sharp^{\ell_{\mathsf{A}}}_{\mathsf{A\,B!m}}(e^{\mathsf{A}}) = \sharp^{\ell_{\mathsf{B}}}_{\mathsf{A\,B?m}}(e^{\mathsf{B}}) \end{array} \right\}$$

The pomset $\bar{r}$ is in $\square((\bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}})_{\mathsf{A} \in \mathcal{P}})$, since it is well-formed and $\leq_I$ satisfies conditions of Definition 7. In fact, since w is well-formed, all receives have matching sends. Also by construction, $w \in \mathbb{L}(\bar{r})$ and, for every A, $\bar{r}\downharpoonright_{\mathsf{A}} \sqsubseteq \bar{r}^{\mathsf{A}}\downharpoonright_{\mathsf{A}}$. Hence, by **CC3-POM** there exists $r' \in R$ and a prefix $\bar{r}'$ of r such that $\bar{r} \sqsubseteq \bar{r}'$, therefore $w \in \mathbb{L}(\bar{r}')$ and therefore $w \in \mathsf{pref}(\mathbb{L}(R))$. $\square$



(a) r_{5_a} (b) r_{5_b} (c) r_{5_c}

Figure 5: The language of $\{r_{5_a}, r_{5_b}\}$ cannot be implemented since A and C do not coordinate to commit to a common choice.



(a) r_{6_a} (b) r_{6_b} (c) $\le_{r_{6_a}} \cup \le_{r_{6_a}}$

Figure 6: The set $R_{(6)} = \{r_{6_a}, r_{6_a}\}$ is not termination sound for B

The family of pomsets $R = \{r_{5_a}, r_{5_b}\}$ of Fig. 5 exemplifies a common obstacle for safe realizability. Here, participants A and C should both send the message x or both send the message y. However, A and C do not coordinate to achieve this behaviour; this makes it impossible for them to distributively commit to a common choice. The family of pomsets R does not satisfy **CC3-POM**. In fact, pomset r_{5_c} satisfies hypothesis of **CC3-POM** (using r_{5_a} for C and r_{5_b} for both A and B), however there is no pomset in R whose prefix is more permissive that r_{5_c}.

Like for the closure conditions, we lift the sufficient condition for termination soundness to pomsets.

Definition 12 (Terminating pomsets)**.** *A participant* A $\in \mathcal{P}$ *is* termination-unaware *for a set of pomsets* R *if there are* $r, r' \in R$*, and a label-preserving injection* $\phi : \mathcal{E}_{r\downarrow_A} \to \mathcal{E}_{r'\downarrow_A}$ *such that* $\le = \phi(\le_{r\downarrow_A}) \cup \le_{r'\downarrow_A}$ *is a partial order and*

$$min_{\le}(\mathcal{E}_{r'\downarrow_A}) \subseteq \phi(min_{\le_{r\downarrow_A}}(\mathcal{E}_{r\downarrow_A})) \qquad \text{and} \qquad min_{\le}(\mathcal{E}_{r'\downarrow_A} \setminus \phi(\mathcal{E}_{r\downarrow_A})) \cap \mathcal{L}^? \neq \emptyset$$

Given a set of participants $\mathcal{P}' \subseteq \mathcal{P}$*, we say that* R *is* $\mathcal{P}'$-terminating *when there is no* A $\in \mathcal{P}'$ *termination-unaware for* R*.*

We use Fig. 6 to describe termination awareness. B is *termination-unaware* for the set of pomsets $R_{(6)}$. In fact, let $\phi : \mathcal{E}_{r_{6_a}\downarrow_B} \to \mathcal{E}_{r_{6_b}\downarrow_B}$ be the only possible label-preserving injection, then $\le = \phi(\le_{r_{6_a}\downarrow_B}) \cup \le_{r_{6_b}\downarrow_B}$ is the partial order in Fig. 6.c, and $min_{\le}(\mathcal{E}_{r_{6_a}\downarrow_B} \setminus \phi(\mathcal{E}_{r_{6_b}\downarrow_B})) = \{\text{AB?w}\}$ is not disjoint from $\mathcal{L}^?$. Intuitively, $\le$ represents the intersection of the languages of the two pomsets $r_{6_b}\downarrow_B$ and $r_{6_a}\downarrow_B$. Notice also that neither $r_{6_a}\downarrow_B \sqsubseteq r_{6_b}\downarrow_B$ or $r_{6_b}\downarrow_B \sqsubseteq r_{6_a}\downarrow_B$.

Theorem 7. *Given* $\mathcal{P}' \subseteq \mathcal{P}$*, if* R *is* $\mathcal{P}'$-terminating *then* $\mathbb{L}(R)$ *is* $\mathcal{P}'$-terminating.

Proof. Given a word $w \in \mathbb{L}(R)$, there is a pomset $r \in R$ such that $w \in \mathbb{L}(R)$. Let A $\in \mathcal{P}'$ and assume that there is $w' \in \mathbb{L}(R)$ such that $w\downarrow_A$ is a prefix of $w'\downarrow_A$. Therefore, there is a pomset $r' \in R$ such

(a) r_{7_a} (b) r_{7_b}

Figure 7: A set of pomsets language-equivalent to the pomset with red and black dependencies of Fig. 4, but explicitly interleaves the events BC!l2 and BC!r2 (cyan dependencies).

that $w' \downarrow_{\mathsf{A}} \in \mathbb{L}(r' \downarrow_{\mathsf{A}})$. Let $e_1, \ldots, e_n$ and $e'_1, \ldots, e'_{n'}$, with $n < n'$, be the linearizations of $\leq_r$ and $\leq_{r'}$ respectively for the world w and w' respectively. Let ϕ be the injection that maps e_i to e'_i for $1 \leq i \leq n$, then $\leq = \phi(\leq_{r\downarrow_{\mathsf{A}}}) \cup \leq_{r'\downarrow_{\mathsf{A}}}$ is a partial order. Therefore $min_{\leq}(\mathcal{E}_{r'\downarrow_{\mathsf{A}}} \setminus \phi(\mathcal{E}_{r\downarrow_{\mathsf{A}}})) \cap \mathcal{L}^? \neq \emptyset$ since R is $\mathcal{P}'$*-terminating*, thus the first symbol of w' after w cannot be an input. □

5 Discussion on the pomset based verification conditions

If a pomset is thought of as the specification of a possible scenario of a system, a practical advantage of using the conditions of Section 4 is that problems can be discovered at design-time. This permits to easily isolate the problematic scenarios of a specification even if they share multiple traces with non-problematic scenarios. Moreover, these conditions avoid the explicit computation of the language of the family of pomsets. This can lead to combinatorial explosion due to interleavings. For example, both pomsets $r_{(3)_a}$ and $r_{(3)_b}$ of Fig. 3 have 32 different linearizations, each one consisting of 8 events. Therefore the language of $R_{(3)}$ consists of $32 * 32 * 2^8 = 2^{18}$ words. Directly analyzing the inter-participant closure in Fig. 4 has clear advantages.

Checking conditions of Section 4 is in general exponentially expensive due to two reasons: the combinatorial explosion of the inter-participant closure and the need of finding a graph isomorphism to check relation $\sqsubseteq$ between pomsets and to prove the existence of the label preserving injection ϕ. In both cases, this complexity depends on the presence of multiple and independent instances of the same action.

Definition 13. *Let r be a pomset over $\mathcal{L}$. An action $l \in \mathcal{L}$* concurrently repeats *in r if there exist $e, e' \in \mathcal{E}_r$ such that $e \neq e'$, $\lambda_r(e) = \lambda_r(e') = l$, and neither $e \leq_r e'$ or $e' \leq_r e$.*

For every non concurrently repeated action there is only one linearization of the inter-participant dependencies. Also, for events involving non concurrently repeated actions it is straightforward to find a graph isomorphism, since there is only one possible label and order preserving injection between the pomsets. This permits to check conditions of Section 4 in polynomial time with respect to the number of events if there are no concurrent repetitions.

In practice, the presence of actions that concurrently repeat is limited. In fact, specification formalisms usually impose conditions that syntactically avoid this issue (e.g. see well-forkedness of [23] or the even more restrictive conditions of e.g., [12]) because sending the same message in two independent threads may "confuse" receivers making it hard (or impossible) to decide which receiving thread should consume the message, leading to coordination problems.

We remark that the conditions of Section 4 strictly entail the corresponding ones in Section 3. We show a counterexample for **CC2-POM** only, since the same reasoning applies for the other condition. Consider the set $R_4 = \{r4_{red}, r4_{green}\}$, where $r4_{red}$ and $r4_{green}$ respectively are the pomset with red dependencies and the pomset with green dependencies of Figure 4. Then, R_4 satisfies **CC2-POM**, since it contains all pomsets that satisfy hypothesis of the closure condition, therefore by Theorem 5 its language satisfies **CC2**. Consider the set $R_7 = \{r7_a, r7_b, r4_{green}\}$, where $r7_a$ and $r7_b$ are the two pomsets of Figure 7. Notice that $r7_a$ and $r7_b$ are equivalent to $r4_{red}$, with the exception of the dependency between B C!l2 and B C!r2. Since $r7_a$ and $r7_b$ have opposite orders between these two events, the union of their languages is equal to the language of $r4_{red}$. Therefore the language of R_7 is equal to the language of R_4, hence it also satisfies **CC2**. However, R_7 does not satisfy **CC2-POM**. In fact, the pomset $r4_{red}$ satisfies hypothesis of **CC2-POM**, but there is not pomset in R_7 that is more permissive than $r4_{red}$.

6 Related work

The surge of *message-passing* applications in industry is revamping the interest for software engineering methodologies supporting designers and developers called to realise communication-centred software. In this context, realizability of global specifications is of concern for both practical and theoretical reasons. Our approach can support choreography languages (e.g. [23]). These specifications yield at the same time (*i*) concrete support to scenario-based development, (*ii*) rigorous semantics in terms of partial order of communication events that enable the use of algorithms and tools to reason about and verify communicating applications, and (*iii*) a simple graphical syntax that supports the intuition and makes it easy to practitioners to master the specification without needing to delve into the underlying theory.

A paradigmatic class of such formalisms are *message-sequence charts* (MSCs) [22, 9, 18, 11, 10, 2]). A mechanism to statically detect realizability in MSCs is proposed in [3]. The notions of non-local choices and of termination considered in [3] are less than than our verification conditions since intra-participant concurrency is not allowed and termination awareness (Definition 6) is not enforced. In the context of choreographies, several works (e.g., [4, 7, 12]) defined constraints to guarantee the soundness of the projections of global specifications. These approaches address the problem for specific languages, thus these conditions often use information on the syntactical structure of the specification. Instead, conditions presented in Section 4 are syntax-oblivious and they make minimal assumptions on the communication model. Therefore, our results can be applied to a wide range of languages.

The closure conditions reviewed in Section 3 have been initially introduced in [1] to study realizability of MSC. The replacement in the framework of MSC with pomsets is technically straightforward and yields more general results, since it enables multi-threaded participants. In Section 3, to avoid systems where participants can get stuck due to the termination of some partners, we introduce a new flexible termination condition and demonstrate sufficient conditions that guarantee it. Then, we introduce new verification conditions for the distributed realizability of pomsets, which can tame the combinatorial explosion due to the interleaving of communication events.

A problem related to realizability is satisfiability of logical formulae. Model checkers use temporal logic, i.e. LTL, to formalize system specifications. A general problem that must be faced is that formal specifications can be wrong as their implementations. For instance, if a formula is unsatisfiable, then the specification is probably incorrect. Similarly to realizability, the problem of satisfiability of a temporal formula [20] allows to demonstrate that there exists an implementation that meets the specification.

7 Concluding remarks

We introduced and studied conditions for the realizability of pomsets, providing verification conditions for realizability of global specifications that are syntax-oblivious and avoid the combinatorial explosion due to interleaving. Also, we introduced a new flexible termination condition and demonstrated sufficient conditions that guarantee it.

There are some open questions to address. Pomset semantics of recursive processes is infinite, which precludes to directly use these results for global specifications that have loops. In [5] pomsets were used in combination with proved transition systems to give an non-interleaving semantics of CCS; basically, given a sequence of transitions $p \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} q$ between two CCS processes p and q, a pomset r can be derived from a *proved transition system* so that r represents the equivalence class of traces between p and q "compatible" with traces labelled $\alpha_1, \dots, \alpha_n$. This work can help us to generalise our results to infinite computations.

Realizability of high-level MSCs has been addressed in [16], but the verification conditions are not syntax-oblivious. The conditions of Section 4 are sufficient but not necessary conditions for realizability. This is due to the fact that the same semantics (i.e., set of traces) can be expressed using different sets of pomsets by exploring different interleavings. We do not know if a notion of normal forms for families of pomsets can be used to guarantee that our conditions are necessary. We conjecture that our semantics could be applied to other coordination paradigms such as order-preserving asynchronous message-passing (as the original semantics of CFSMs), synchronous communications, or tuple based coordination. We leave the exploration of the robustness of our framework as future work. Finally, we plan to extend ChorGram [15], a tool we are currently developing, to implement our theoretical framework and apply it to the analysis of global specifications.

References

[1] Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Inference of Message Sequence Charts. *IEEE Trans. Software Eng.*, 29(7):623–633, 2003.

[2] Rajeev Alur, Gerard J. Holzmann, and Doron Peled. An analyzer for message sequence charts. In Tiziana Margaria and Bernhard Steffen, editors, *TACAS*, pages 35–48. Springer, 1996.

[3] Hanêne Ben-Abdallah and Stefan Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 259–274. Springer, 1997.

[4] Laura Bocchi, Hernán C. Melgratti, and Emilio Tuosto. Resolving non-determinism in choreographies. In *ESOP*, pages 493–512, 2014.

[5] Gérard Boudol and Ilaria Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 411–427. Springer, 1988.

[6] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines. *Journal of the ACM*, 30(2):323–342, 1983.

[7] Marco Carbone, Kohei Honda, and Nobuko Yoshida. A calculus of global interaction based on session types. *Electronic Notes in Theoretical Computer Science*, 171(3):127 – 151, 2007.

[8] Haim Gaifman and Vaughan R Pratt. Partial order models of concurrency and the computation of functions. In *LICS*, pages 72–85, 1987.

[9] Emmanuel Gaudin and Eric Brunel. Property Verification with MSC. In *SDL 2013*. Springer, 2013.

[10] Elsa L. Gunter, Anca Muscholl, and Doron A. Peled. Compositional Message Sequence Charts. In *TACAS*, pages 496–511. Springer, 2001.

[11] David Harel and Rami Marelly. *Come, Let's Play*. Springer, 2003.

[12] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM*, 63(1):9:1–9:67, 2016. Extended version of a paper presented at POPL08.

[13] Joost-Pieter Katoen and Lennard Lambert. Pomsets for message sequence charts. *Formale Beschreibungstechniken für Verteilte Systeme*, pages 197–208, 1998.

[14] Susheel Kumar. 7 Reasons Why Organizations Struggle with Microservices Adoption. https://blogs.perficient.com/integrate/2017/06/26/7-reasons-why-organization-struggle-with-microservices-adoption/, June 2017.

[15] Julien Lange and Emilio Tuosto. ChorGram. https://bitbucket.org/emlio_tuosto/chorgram/wiki/Home.

[16] Markus Lohrey. Safe Realizability of High-Level Message Sequence Charts. In Luboš Brim, Mojmír Křetínský, Antonín Kučera, and Petr Jančar, editors, *CONCUR*, pages 177–192. Springer, 2002.

[17] Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.

[18] Anca Muscholl and Doron Peled. Deciding Properties of Message Sequence Charts. In Stefan Leue and Tarja Johanna Systä, editors, *Scenarios: Models, Transformations and Tools*, pages 43–65. Springer, 2005.

[19] Vaughan Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.

[20] Kristin Y Rozier and Moshe Y Vardi. LTL satisfiability checking. In *International SPIN Workshop on Model Checking of Software*, pages 149–167. Springer, 2007.

[21] Davide Sangiorgi and David Walker. *The π-Calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[22] Formal description techniques (FDT) – Message Sequence Chart (MSC). Recommendation ITU-T Z.120, February 2011. Available at http://www.itu.int/rec/T-REC-Z.120-201102-I/en.

[23] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies. *JLAMP*, 95:17 – 40, 2018.

[24] Web services choreography description language version 1.0. https://www.w3.org/TR/ws-cdl-10/, 2005.