# Realisability of branching pomsets (technical report)

# Realisability of branching pomsets (technical report)

Luc Edixhoven[0000−0002−6011−9535] (✉) and Sung-Shik Jongmans[0000−0002−4394−8745]

[1] Open University, Heerlen, Netherlands
[2] Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands
{led,ssj}@ou.nl

**Abstract** A communication protocol is realisable if it can be faithfully implemented in a distributed fashion by communicating agents. Pomsets offer a way to compactly represent concurrency in communication protocols and have been recently used for the purpose of realisability analysis. In this paper we focus on the recently introduced branching pomsets, which also compactly represent choices. We define well-formedness conditions on branching pomsets, inspired by multiparty session types, and we prove that the well-formedness of a branching pomset is a sufficient condition for the realisability of the represented communication protocol.

**Keywords:** Realisability · Pomsets · Choreographies

## 1 Introduction

Designing and implementing distributed systems is difficult. They are becoming ever more important, and yet the complexity resulting from combinations of inter-participant concurrency and dependencies makes the process error-prone and debugging non-trivial. As a consequence, much research has been dedicated to analysing communication patterns, or protocols, in distributed systems. Examples of such research goals are to show the presence or absence of certain safety properties in a given system, to automate such analysis or to guarantee the presence of desirable properties by construction. We are interested in particular in the *realisability* property, i.e., whether a global specification of a protocol can be faithfully implemented in a distributed fashion in the first place. This problem has been well-studied in the last two decades in a variety of settings [15,2,24,3,31].

A recent development has been the use of *partially ordered multisets*, or *pomsets*, for the purpose of realisability analysis [17]. Guanciale and Tuosto use pomsets as a syntax-oblivious specification model for communication protocols and define conditions that ensure realisability directly over pomsets. Pomsets offer a way to compactly represent concurrent behaviour. By using a partial order to explicitly capture causal dependencies between pairs of actions, they avoid the exponential blowup from finite state machines. However, a single pomset does not offer any means to represent choices. Instead, a choice is represented as a set of pomsets, one for each possible branch. Multiple choices result in one pomset for each possible combination of branches, which can yield an exponentially large set of pomsets. Recent work by the authors and Proença and Cledou introduces *branching pomsets* [12]. These extend pomsets with a branching structure to compactly represent both concurrency and choices, avoiding both exponential blowups. In this paper we present a first step in the analysis of the realisability of these branching pomsets.

We consider distributed systems using asynchronous message passing and ordered buffers (FIFO queues) between (ordered pairs of) participants. In the aforementioned paper ([17]) Guanciale and Tuosto consider systems with unordered (non-FIFO) buffers. Using the work by Guanciale and Tuosto as a basis for our analysis would require first adapting their work to a FIFO setting and then adapting it further to branching pomsets. Instead, we choose to draw inspiration from multiparty session types (MPST) [19], which already use FIFO buffers, and thus use MPST as a basis for our analysis. Through its syntax and projection operator, MPST defines a number of well-formedness conditions on global types which ensure realisability. We define similar well-formedness conditions on branching pomsets and prove that they ensure realisability as well. These conditions are sufficient but not necessary, i.e., a protocol may be realisable without being well-formed. We discuss some possible relaxations of the conditions at the end of the paper.

*Outline* We recall the concept and definition of branching pomsets in Section 2. In Section 3 we define our notion of realisability, we define our well-formedness conditions in Section 4 and we prove in Section 5 that, if a branching pomset is well-formed, then the corresponding communication protocol is realisable. In Section 6 we briefly discuss two examples. Finally, we discuss related work in Section 7 and we end the paper with our conclusions and a discussion in Section 8.

We omit a number of technical lemmas and the majority of proofs in favour of more informal proof sketches or highlights. All omitted content can be found in the appendix.

## 2 Preliminaries on branching pomsets

In this section we recall the concept and definitions of branching pomsets. This section is heavily based on the original introduction of branching pomsets [12]; a more thorough explanation can be found in that paper.

*Notation* Let $\mathcal{A} = \{\mathsf{a}, \mathsf{b}, \ldots\}$ be the set of participants (or agents). Let $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \ldots\}$ be the set of message types. Let $\mathcal{L} = \bigcup_{\mathsf{a},\mathsf{b} \in \mathcal{A}, \mathsf{x} \in \mathcal{X}} \{\mathsf{ab!x}, \mathsf{ab?x}\}$ be the set of labels (actions), ranged over by $\ell$, where $\mathsf{ab!x}$ is a send action from $\mathsf{a}$ to $\mathsf{b}$ of a message of type $\mathsf{x}$, and $\mathsf{ab?x}$ is the corresponding receive action by $\mathsf{b}$. The *subject* of an action $\ell$, written $\mathit{subj}(\ell)$, is its active agent: $\mathit{subj}(\mathsf{ab!x}) = \mathsf{a}$ and $\mathit{subj}(\mathsf{ab?x}) = \mathsf{b}$.

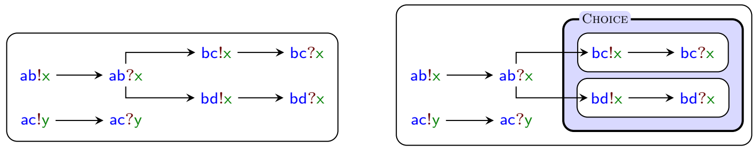


Figure 1: A pomset (left) and a branching pomset (right) depicting simple communication patterns.

A partially ordered multiset [27], or pomset for short, consists of a set of nodes (events) $E$, a labelling function $\lambda$ mapping events to a set of labels (e.g., send and receive actions), and a partial order $\leq$ defining causal dependencies between pairs of events (i.e., an event, or rather its corresponding action, can only fire if all events preceding it in the partial order have already fired). Its behaviour is the set of all sequences of the labels of its events that abide by $\leq$.

An example pomset is shown graphically in Figure 1 (left), depicting a simple communication pattern between four agents Alice (a), Bob (b), Carol (c) and Dave (d). Alice first sends a message of type x to Bob, who then sends a message of type x to both Carol and Dave. In parallel, Alice sends a message of type y to Carol. The graphical pomset representation shows the labels of the events and the arrows visualising the partial order: an event precedes any other event to which it has an outgoing arrow, either directly or transitively. Formally, $E = \{e_1, \ldots, e_8\}$, $\lambda$ is such that $e_1, \ldots, e_8$ map to respectively ab!x, ab?x, bc!x, bc?x, bd!x, bd?x, ac!y, ac?y, and $\leq \; = \{(e_i, e_j) \mid i \leq j \wedge (i, j \in \{1,2,3,4\} \vee i, j \in \{1,2,5,6\} \vee i, j \in \{7,8\})\}$.

Branching pomsets extend pomsets with a branching structure, which is a tree structure containing events and (binary) choices. Formally, the branching structure is defined below as a tree with root node $\mathcal{B}$, whose children $\mathcal{C}$ are either a single event $e$ or a choice node with children $\mathcal{B}_1, \mathcal{B}_2$. All leaves are events.

$$\mathcal{B} ::= \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$$
$$\mathcal{C} ::= e \mid \{\mathcal{B}_1, \mathcal{B}_2\}$$

We write $\mathcal{B}_1 \preceq \mathcal{B}_2$ if $\mathcal{B}_1$ is a subtree of $\mathcal{B}_2$, and $\mathcal{B}_1 \prec \mathcal{B}_2$ if $\mathcal{B}_1$ is a strict subtree of $\mathcal{B}_2$, i.e., if $\mathcal{B}_1 \preceq \mathcal{B}_2$ and $\mathcal{B}_1 \neq \mathcal{B}_2$. We use the same notation for $\mathcal{C}$s, $e$s (a special case of $\mathcal{C}$s) and combinations of all the aforementioned.

We formally define branching pomsets in Definition 1.

**Definition 1 (Branching pomset [12]).** *A branching pomset is a four-tuple $R = \langle E, \leq, \lambda, \mathcal{B} \rangle$, where:*

- *$E$ is a set of events;*
- *$\leq \subseteq E \times E$ is a causality relation on events such that $\leq^\star$ (the transitive closure of $\leq$) is a partial order on events;*
- *$\lambda : E \mapsto \mathcal{L}$ is a labelling function assigning an action to every event; and*
- *$\mathcal{B}$ is a branching structure such that the set of leaves of $\mathcal{B}$ is $E$ and no event in $E$ occurs in $\mathcal{B}$ more than once.*

We use $R.E$, $R.\leq$, $R.\lambda$ and $R.\mathcal{B}$ to refer to the components of $R$. We generally omit the prefix if doing so causes no confusion. We also write $e_1 < e_2$ if $e_1 \leq e_2$ and $e_1 \neq e_2$. We say that two events $e_1$ and $e_2$ are *causally ordered* if either $e_1 \leq e_2$ or $e_2 \leq e_1$. We say that two events $e_1$ and $e_2$ are *mutually exclusive* if there exists some $\mathcal{C} = \{\mathcal{B}_1, \mathcal{B}_2\} \prec R.\mathcal{B}$ such that $e_1 \prec \mathcal{B}_1$ and $e_2 \prec \mathcal{B}_2$.

An example branching pomset is shown graphically in Figure 1 (right). It depicts the same communication pattern as that in the pomset on the left, except that now Bob sends a message of type x to *either* Carol or Dave instead of to both. This is visualised as a choice box containing two branches. Formally, $E$, $\lambda$ and $\leq$ are the same as before. New is $\mathcal{B} = \{e_1, e_2, e_7, e_8, \mathcal{C}\}$, where $\mathcal{C} = \{\{e_3, e_4\}, \{e_5, e_6\}\}$ is Bob's choice between the events corresponding to Carol and Dave. The choice can be resolved by picking one of the branches, e.g., Carol's ($\{e_3, e_4\}$), and merging it with $\mathcal{C}$'s parent, $\mathcal{B}$, resulting in $\mathcal{B}' = \{e_1, e_2, e_3, e_4, e_7, e_8\}$.

Informally, to fire an event whose ancestor in the branching structure is a choice, first the choice must be resolved: it is replaced by one of its children (branches). The other child is discarded and the branching pomset is updated accordingly: the events contained in the discarded subtree are removed, as well as the related entries in the causality relation and the labelling function.

Formally, the semantics of branching pomsets are defined using a *refinement* relation on the branching structure. A structure $\mathcal{B}$ can refine to $\mathcal{B}'$, written $\mathcal{B} \sqsupseteq \mathcal{B}'$, by resolving a number of choices as above. We write $\mathcal{B} \sqsupset \mathcal{B}'$ to specify that $\mathcal{B} \neq \mathcal{B}'$. The refinement rules are formalised in Figure 2a. The first two rules state that refinement is reflexive and transitive. The third rule, Choice, resolves choices. It states that we can replace a choice with one of its branches. Finally, the fourth rule overloads the refinement notation to also apply to branching pomsets themselves: if $R.\mathcal{B}$ can refine to some $\mathcal{B}'$, then $R$ itself can refine to a derived branching pomset with branching structure $\mathcal{B}'$, whose events are restricted to those occurring in $\mathcal{B}'$ and likewise for $\leq$ and $\lambda$ — as defined in Figure 2c. We note that we omit one of the rules in [12], since our later well-formedness conditions lead to it never being used.

The reduction and termination rules are defined in Figure 2b. The first rule states that a branching pomset can terminate if its branching structure can refine to the empty set. The second rule states the conditions for *enabling* an event $e$, written $R \xrightarrow{\checkmark_e} R'$: $R$ can enable $e$ by refining to $R'$ if $e$ is both *minimal* and *active* in $R'$ and if there is no other refinement between $R$ and $R'$ for which this is the case. An event $e$ is minimal if there exists no other event $e'$ such that $e' < e$. It is active if it is not inside a choice, i.e., if $e \in R.\mathcal{B}$. In other words, $R$ may only refine as far as strictly necessary to enable $e$. Finally, the last two rules state that, if $R \xrightarrow{\checkmark_e} R'$, then $R$ can fire $e$ by reducing to $R' - e$, which is the branching pomset obtained by removing $e$ from $R'$ — as defined in Figure 2c. This reduction is defined both on $e$'s label and on the event itself, the latter for internal use in proofs since $\lambda(e)$ is not necessarily unique but $e$ always is.

For example, let $R$ be the branching pomset (right) in Figure 1. Its initial active events are those labelled with ab!x, ab!y, ab?x and ab?y, of which the first

$$\frac{}{\mathcal{B} \sqsupseteq \mathcal{B}}[\textsc{Refl}] \quad \frac{\mathcal{B} \sqsupseteq \mathcal{B}' \sqsupseteq \mathcal{B}''}{\mathcal{B} \sqsupseteq \mathcal{B}''}[\textsc{Trans}] \quad \frac{i \in \{1,2\}}{\{\{\mathcal{B}_1, \mathcal{B}_2\}\} \cup \mathcal{B} \sqsupseteq \mathcal{B}_i \cup \mathcal{B}}[\textsc{Choice}] \quad \frac{R.\mathcal{B} \sqsupseteq \mathcal{B}'}{R \sqsupseteq R[\mathcal{B}']}$$

(a) Refinement rules, where we assume for Choice that $\{\mathcal{B}_1, \mathcal{B}_2\} \notin \mathcal{B}$.

$$\frac{R.\mathcal{B} \sqsupseteq \emptyset}{R\downarrow} \quad \frac{R \sqsupseteq R' \quad e \in \text{a-min}(R') \quad \forall R'' : R \sqsupseteq R'' \sqsupset R' \Rightarrow e \notin \text{a-min}(R'')}{R \xrightarrow{\checkmark_e} R'} \quad \frac{R \xrightarrow{\checkmark_e} R'}{R \xrightarrow{e} R' - e} \quad \frac{R \xrightarrow{e} R'}{R \xrightarrow{\lambda(e)} R'}$$

(b) Reduction and termination rules.

$$\begin{aligned}
\langle E, \leq, \lambda, \mathcal{B}\rangle[\mathcal{B}'] &= \langle E|_{\mathcal{B}'}, \leq|_{\mathcal{B}'}, \lambda|_{\mathcal{B}'}, \mathcal{B}'\rangle \\
X|_{\mathcal{B}} &= \text{restricts } X \text{ only to the events in } \mathcal{B} \\
\text{a-min}(R) &= \{e \in R.E \mid \nexists e' \in R.E : e' < e\} \land e \in R.\mathcal{B} \\
\hat{e} - e &= \hat{e} \\
\{\mathcal{C}_1, \ldots, \mathcal{C}_n\} - e &= \begin{cases} \{\mathcal{C}_1, \ldots, \mathcal{C}_{i-1}, \mathcal{C}_{i+1}, \ldots, \mathcal{C}_n\} & \text{if } C_i = e \\ \{\mathcal{C}_1 - e, \ldots, \mathcal{C}_n - e\} & \text{otherwise} \end{cases} \\
\{\mathcal{B}_1, \mathcal{B}_2\} - e &= \{\mathcal{B}_1 - e, \mathcal{B}_2 - e\} \\
R - e &= R[R.\mathcal{B} - e]
\end{aligned}$$

(c) Operations on branching pomsets.

Figure 2: Simplified semantics of branching pomsets [12].

two are also minimal. After firing either one, the corresponding receive action becomes minimal. In these cases $R \xrightarrow{\checkmark_e} R$ for the relevant event $e$, i.e., it suffices to refine $R$ to itself. After firing ab?x the two events labelled with bc!x and bd!x become minimal but not yet active. Only now are we allowed to resolve the choice by applying Choice to pick one of the branches. After this either the events labelled with bc!x and bc?x or those with bd!x and bd?x will become active, and the first event of the chosen pair can be fired.

## 3 Realisability

In this section we define our notion of realisability and illustrate it with examples.

We model distributed implementations as compositions of a collection of local branching pomsets $\vec{R}$ and ordered buffers (FIFO queues) $\vec{b}$ containing the messages in transit (sent but not yet received) between directed pairs of agents (or channels), similar to communicating finite-state machines [5]. The local pomsets only contain actions for a single agent; there should be one local branching pomset for each agent and one buffer for each channel.

Composition is formally defined below. We use three auxiliary functions: $add(\mathsf{ab}!\mathsf{x}, \vec{b})$ returns $\vec{b}$ with $\mathsf{x}$ added to $b_{\mathsf{ab}}$, $remove(\mathsf{ab}!\mathsf{x}, \vec{b})$ returns $\vec{b}$ with $\mathsf{x}$ removed from $b_{\mathsf{ab}}$ and $has(\mathsf{ab}!\mathsf{x}, \vec{b})$ returns whether $\mathsf{x}$ is pending in $b_{\mathsf{ab}}$. Since we consider ordered buffers, *add* appends message types to the end of the corresponding queue, *remove* removes message types from the front, and *has* only checks whether the first message matches.

We note that our termination condition does not require the buffers to be empty. In practice asynchronous communication channels will typically have some latency, and requiring empty buffers would require processes (the local branching pomsets) to be aware of messages in transit. Instead, in our model the presence or absence of orphan messages (messages unreceived on termination) is a separate property from realisability, to be verified in isolation. It does, however, follow from our well-formedness conditions in Section 4 that a well-formed and realisable protocol is also free of orphan messages.

**Definition 2 (Composition).** *Let $\vec{R}$ be an agent-indexed vector of local branching pomsets. Let $\vec{b}$ be a channel-indexed vector of ordered buffers. Their composition is the tuple $\langle \vec{R}, \vec{b} \rangle$, whose semantics is defined as the labeled transition system defined by the rules below.*

$$\textit{(Send)}\ \frac{R_{\mathsf{a}} \xrightarrow{\mathsf{ab}!\mathsf{x}} R'_{\mathsf{a}} \qquad \vec{b}' = add(\mathsf{ab}!\mathsf{x}, \vec{b})}{\langle \vec{R}, \vec{b} \rangle \xrightarrow{\mathsf{ab}!\mathsf{x}} \langle \vec{R}[R'_{\mathsf{a}}/R_{\mathsf{a}}], \vec{b}' \rangle} \qquad \textit{(Receive)}\ \frac{R_{\mathsf{b}} \xrightarrow{\mathsf{ab}?\mathsf{x}} R'_{\mathsf{b}} \qquad has(\mathsf{ab}!\mathsf{x}, \vec{b}) \quad \vec{b}' = remove(\mathsf{ab}!\mathsf{x}, \vec{b})}{\langle \vec{R}, \vec{b} \rangle \xrightarrow{\mathsf{ab}?\mathsf{x}} \langle \vec{R}[R'_{\mathsf{b}}/R_{\mathsf{b}}], \vec{b}' \rangle}$$

$$\textit{(Terminate)}\ \frac{\forall \mathsf{a} : R_{\mathsf{a}}\downarrow}{\langle \vec{R}, \vec{b} \rangle\downarrow}$$

A protocol is *realisable* if there exists a faithful distributed implementation of it, i.e., one defining the same behaviour. We formally define realisability below. We note that it is typically defined in terms of language (trace) equivalence [17]. However, as the exact branching of choices plays an important part in branching pomsets, we use a more strict notion of equivalence and require the global branching pomset and the composition to be bisimilar [28]. As an example, consider the branching pomsets in Figure 3. After firing $\mathsf{ab}!\mathsf{int}$ $\mathsf{ab}?\mathsf{int}$, the branching pomset on the left can still fire either $\mathsf{bc}!\mathsf{yes}$ or $\mathsf{bc}!\mathsf{no}$, while the branching pomset on the right has already committed to one of the two upon firing $\mathsf{ab}!\mathsf{int}$. We wish to distinguish these two branching pomsets, which cannot be done using language equivalence since they yield the same set of two traces. It is then most natural to compare two branching pomsets using branching equivalence, i.e., bisimilarity. As we wish to be able to make the same distinction in our realisability analysis, we also define realisability in terms of bisimilarity. We note that our well-formedness conditions enforce a deterministic setting, in which bisimilarity agrees with language equivalence. We then choose to prove bisimilarity rather than language equivalence because the proofs are typically more straightforward.

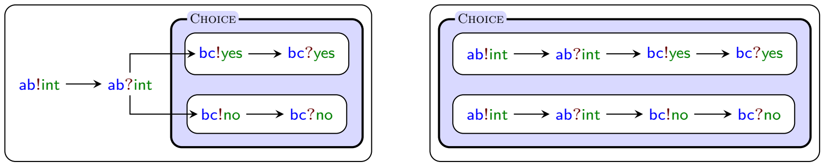


Figure 3: Two similar yet not *bi*similar branching pomsets.

Formally, if two branching pomsets $R_1$ and $R_2$ are bisimilar, written $R_1 \sim R_2$, then, for every reduction $R_1 \xrightarrow{\ell} R_1'$ there should exist a reduction $R_2 \xrightarrow{\ell} R_2'$ such that $R_1'$ and $R_2'$ are again bisimilar, and vice-versa. Additionally, we require that two bisimilar branching pomsets $R_1$ and $R_2$ can terminate iff the other can do so as well.

**Definition 3 (Realisability).** *Let $R$ be a branching pomset. The protocol it represents is realisable iff there exists a composition $\langle \vec{R}, \vec{b} \rangle$ such that $b_{\mathsf{ab}}$ is empty for all* $\mathsf{ab}$ *and $R \sim \langle \vec{R}, \vec{b} \rangle$.*

As an example, consider the branching pomsets in Figure 4:

- $R_1$ is unrealisable. Alice and Bob both have to send a yes or a no to the other but the two messages must be the same. It is impossible without further synchronisation or communication to prevent a scenario in which one will send a different message than the other.
- $R_2$ is realisable. Alice first sends an int and then a bool to Bob. After receiving the int, Bob returns either a yes or a no.
- $R_3$ is unrealisable. Alice sends an int and a bool to Bob, but while they agree that Alice first sends the int and then the bool, the order in which Bob receives the message is unspecified. As we assume ordered buffers, Bob will first receive the int, but the global branching pomset allows an execution in which Bob first receives the bool.
- $R_4$ is realisable. Alice sends a yes or a no to Bob, followed by an int.

We note that it is easy to go from a global branching pomset $R$ to a local branching pomset for some agent a by *projecting* it on a, written $R{\downharpoonleft_{\mathsf{a}}}$. We will use projections in our well-formedness conditions and realisability proof, and we formally define them below. The projection $R{\downharpoonleft_{\mathsf{a}}}$ restricts $R$ to the events whose subject is a, and restricts $\leq$ and $\lambda$ accordingly. The branching structure is pruned by removing all discarded events (leaves), but no inner nodes of the tree are removed, even if they are left without any children. This is done to safeguard the symmetry with the branching structure of $R$ to ease our proofs.

**Definition 4 (Projection).** $\langle E, \leq, \lambda, \mathcal{B} \rangle{\downharpoonleft_{\mathsf{a}}} = \langle E_{\mathsf{a}}, \leq_{\mathsf{a}}, \lambda_{\mathsf{a}}, \mathcal{B}_{\mathsf{a}} \rangle$ *where:*

- $E_{\mathsf{a}} = \{e \in E \mid \mathit{subj}(e) = \mathsf{a}\}$
- $\leq_{\mathsf{a}} = \leq \cap \ (E_{\mathsf{a}} \times E_{\mathsf{a}})$
- $\lambda_{\mathsf{a}} = \lambda \cap (E_{\mathsf{a}} \times \mathcal{L})$

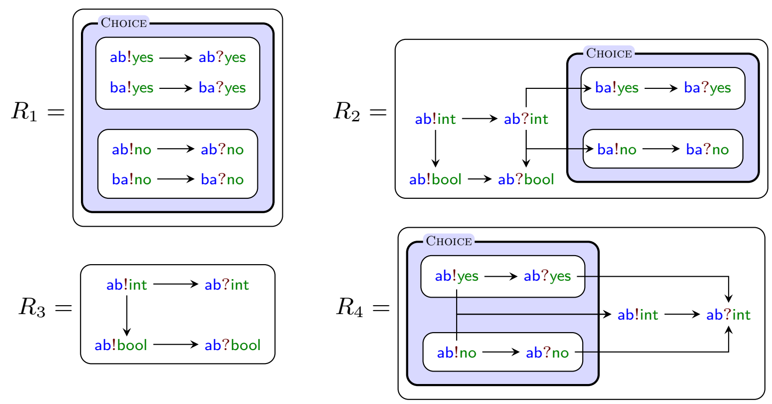


Figure 4: A collection of realisable and unrealisable branching pomsets.

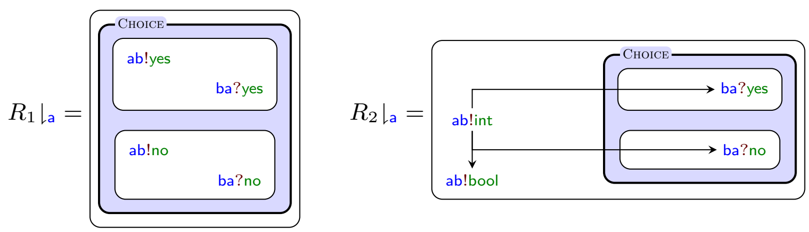


Figure 5: The projection of the branching pomsets $R_1$ and $R_2$ in Figure 4 on a.

– $\mathcal{B}_{\mathsf{a}} = \mathcal{B}{\downharpoonleft}_{\mathsf{a}}$ *as defined below.*

$$
\begin{aligned}
e{\downharpoonleft}_{\mathsf{a}} &= e \textit{ if } e \in E_{\mathsf{a}} \\
\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}{\downharpoonleft}_{\mathsf{a}} &= \{\mathcal{C}_i{\downharpoonleft}_{\mathsf{a}} \mid 1 \le i \le n \wedge \mathcal{C}_i{\downharpoonleft}_{\mathsf{a}} \textit{ is defined}\} \\
\{\mathcal{B}_1, \mathcal{B}_2\}{\downharpoonleft}_{\mathsf{a}} &= \{\mathcal{B}_1{\downharpoonleft}_{\mathsf{a}}, \mathcal{B}_2{\downharpoonleft}_{\mathsf{a}}\}
\end{aligned}
$$

As an example, Figure 5 shows the projection of two of the branching pomsets in Figure 4 on agent a. The events with subject b are removed, as are dependencies involving them. $R_1{\downharpoonleft}_{\mathsf{a}}$ is left with no dependencies at all. We note that, as the graphical representation of a branching pomset shows the transitive reduction of the causality relation and not the full relation, it is unclear from just Figure 4 whether $R_2{\downharpoonleft}_{\mathsf{a}}$ should contain dependencies between ab!int and the events ba?yes and ba?no. This is unambiguous in the formal textual definition, which we omitted but which also relates these events.

Finally, we prove that $R$ and its projections can mirror each others refinements. Both proofs are straightforward by induction on the structure of the premise's derivation tree.

**Lemma 1.** *If* $R \sqsupseteq R'$ *then* $R\lfloor_{\mathsf{a}} \sqsupseteq R'\lfloor_{\mathsf{a}}$.

**Lemma 2.** *If* $R\lfloor_{\mathsf{a}} \sqsupseteq R'_{\mathsf{a}}$ *then* $R \sqsupseteq R'$ *for some* $R'$ *such that* $R'_{\mathsf{a}} = R'\lfloor_{\mathsf{a}}$.

## 4 Well-formedness

In this section we define our well-formedness conditions on branching pomsets.

We define four well-formedness conditions (Definition 10): to be well-formed, a branching pomset must be well-branched, well-channeled, tree-like and choreographic. Well-branchedness, well-channeledness and tree-likeness are inspired by MPST [19] and ensure some safety properties. Choreographicness ensures that the branching pomset represents some sort of meaningful protocol.

– **Well-branched** (Definition 6): every choice is made only on the label of a send-receive pair, i.e., the first events in every branch must be a send and receive between some agents a and b, with the message type being different in every branch. Additionally, the projection on every agent uninvolved in the choice must be the same in every branch. Then a and b are both aware of the chosen branch and all other agents are unaffected by the choice.
  Although the branching pomset model only contains binary choices, an $n$-ary choice $\mathcal{C}$ can be encoded as a nested binary one, where the $n$ children of $\mathcal{C}$ become the leaves of a binary tree. We call such a leaf $\mathcal{B}$ an *option* of $\mathcal{C}$, written $\mathcal{B} \lhd \mathcal{C}$, which is formally defined below. This allows us to properly interpret $\mathcal{C}$ as an $n$-ary choice again and reason about it accordingly.
– **Well-channeled** (Definition 7): pairs of sends and pairs of receives on the same channel that can occur in the same execution should be ordered, and the pairs of sends should have the same order as the pairs of their corresponding receives. A branching pomset which is not well-channeled could, for example, yield a trace ab!x ab!y ab?y ab?x, which cannot be reproduced by a composition using ordered buffers.
– **Tree-like** (Definition 8): events inside of choices can only affect future events in the same branch. Graphically speaking, arrows can only enter boxes, not leave them. As a consequence, the causality relation $\leq$ follows the branching structure $\mathcal{B}$ and has a tree-like shape — hence the name.
– **Choreographic** (Definition 9): the branching pomset represents a choreography of some sort, i.e., a communication protocol in which the send and receive events are properly paired and all dependencies can be logically derived. Specifically, all dependencies are between send-receive pairs or between same-subject events, or they can be transitively derived from those. Additionally, there is some correspondence between the send and receive events: every send can be matched to exactly one corresponding receive, and every non-top-level receive has some corresponding send at the same level of the branching structure $\mathcal{B}$. This definition is similar to the definition of well-formedness by Guanciale and Tuosto [17].

**Definition 5 (Option).** *Let $\mathcal{C} \prec R.\mathcal{B}$. $\mathcal{B}$ is an* option *of $\mathcal{C}$, written $\mathcal{B} \lhd \mathcal{C}$, if $\mathcal{B} \in \{\mathcal{B}^\dagger \mid \mathcal{B}^\dagger \lhd^\dagger \mathcal{C} \wedge \nexists \mathcal{B}^\ddagger : (\mathcal{B}^\ddagger \lhd^\dagger \mathcal{C} \wedge \mathcal{B}^\ddagger \prec \mathcal{B}^\dagger)\}$, where $\lhd^\dagger$ is defined as follows:*

$$\frac{\mathcal{B} \in \mathcal{C}}{\mathcal{B} \lhd^\dagger \mathcal{C}} \qquad \frac{\mathcal{B} \in \mathcal{C}' \quad \{\mathcal{C}'\} \lhd^\dagger \mathcal{C}}{\mathcal{B} \lhd^\dagger \mathcal{C}}$$

**Definition 6 (Well-branched).** *A branching pomset $R$ is* well-branched *iff, for every $\mathcal{C} \prec R.\mathcal{B}$ there exist participants $\mathsf{a}, \mathsf{b}$ such that for every $\mathcal{B}_i \neq \mathcal{B}_j \lhd \mathcal{C}$ there exist events $e_{i1}, e_{i2} \in \mathcal{B}_i$ and $e_{j1}, e_{j2} \in \mathcal{B}_j$ such that:*

- *$\lambda(e_{i1}) = \mathsf{ab!x}$, $\lambda(e_{i2}) = \mathsf{ab?x}$, $\lambda(e_{j1}) = \mathsf{ab!y}$ and $\lambda(e_{j2}) = \mathsf{ab?y}$ for some $\mathsf{x} \neq \mathsf{y}$;*
- *$e_{i1} \leq e_i$ for all $e_i \preceq \mathcal{B}_i$ and $e_{j1} \leq e_j$ for all $e_j \preceq \mathcal{B}_j$;*
- *$e_{i2} \leq e_i$ for all $e_i \preceq \mathcal{B}_i$ for which $subj(e_i) = \mathsf{b}$ and $e_{j2} \leq e_j$ for all $e_j \preceq \mathcal{B}_j$ for which $subj(e_j) = \mathsf{b}$; and*
- *$R[\mathcal{B}_i]\lfloor_{\mathsf{c}} = R[\mathcal{B}_j]\lfloor_{\mathsf{c}}$ for all $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$[3].*

**Definition 7 (Well-channeled).** *A branching pomset $R$ is well-channeled iff, for all events $e_1, e_2, e_3, e_4 \in R.E$:*

- *If $e_1$ and $e_2$ are either both sends or both receive events, and if they share the same channel, then they are either causally ordered or mutually exclusive.*
- *If $e_1, e_3$ and $e_2, e_4$ are two pairs of matching send-receive events sharing the same channel, and if there exists no $e_5 \in R.E$ such that $e_1 < e_5 < e_3$ or $e_2 < e_5 < e_4$, then $e_1 \leq e_2 \implies e_3 \leq e_4$.*

**Definition 8 (Tree-like).** *A branching pomset $R$ is* tree-like *iff:*
$\forall \mathcal{C} = \{\mathcal{B}_1, \mathcal{B}_2\} \prec R.\mathcal{B} : (e_1 \leq e_2 \wedge e_1 \preceq \mathcal{B}_i) \implies e_2 \preceq \mathcal{B}_i$, *where $i \in \{1, 2\}$.*

**Definition 9 (Choreographic).** *A branching pomset $R$ is* choreographic *iff, for every $e \in R.E$:*

- *If there exists $e' \in R.E$ such that $e' < e$ then there exists some event $e''$ (not necessarily distinct from $e'$) such that $e' \leq e'' < e$ and either $subj(\lambda(e'')) = subj(\lambda(e))$ or [$\lambda(e'') = \mathsf{ab!x}$ and $\lambda(e) = \mathsf{ab?x}$ for some $\mathsf{a}, \mathsf{b}, \mathsf{x}$].*
- *If $\lambda(e) = \mathsf{ab?x}$ and $e \in \mathcal{B}$ for some $\mathcal{B} \prec R.\mathcal{B}$ then there exists some $e'$ such that $e' \in \mathcal{B}$ and $\lambda(e') = \mathsf{ab!x}$ and $e' < e$.*
- *If $\lambda(e) = \mathsf{ab!x}$ then there exists exactly one $e'$ such that $e \leq e'$ and that $\lambda(e') = \mathsf{ab?x}$ and that $(\lambda(e^\dagger) = \mathsf{ab!x} \wedge e^\dagger \leq e') \Rightarrow e^\dagger \leq e$.*

**Definition 10 (Well-formed).** *A branching pomset $R$ is* well-formed *iff it is well-branched, well-channeled, tree-like and choreographic.*

As an example, recall the branching pomsets in Figure 4:

[3] Technically $R[\mathcal{B}_i]\lfloor_{\mathsf{c}}$ and $R[\mathcal{B}_j]\lfloor_{\mathsf{c}}$ have different events and should thus be isomorphic rather than precisely equal. We choose to write it as an equality to not unnecessarily complicate the definition and proofs.

- $R_1$ is not well-formed since it is not well-branched: for example, the branches of the choice have multiple minimal events. It is indeed unrealisable.
- $R_2$ is both well-formed and realisable.
- $R_3$ is not well-formed since it is not well-channeled: the two receive events are on the same channel but are unordered. It is indeed unrealisable.
- $R_4$ is not well-formed since it is not tree-like: there are arrows from events inside branches of a choice to $\mathsf{ab!int}$ and $\mathsf{ab?int}$, even though the latter are not part of the same branch. It is, however, realisable, which illustrates that, while we prove in Section 5 that our well-formedness conditions are sufficient, they are not necessary.

Finally, we show that well-formedness is retained after a reduction.

**Lemma 3.** *Let $R$ be a branching pomset and let $R \xrightarrow{\ell} R'$. If $R$ is well-formed then so is $R'$.*

*Proof (sketch).* We use that the components of $R'$ are subsets of or (in the case of the branching structure) derived from the components of $R$. It then follows that a violation of one of the properties in $R'$ would also invariably lead to a violation of one of the properties in $R$.

## 5 Bisimulation proof

In this section we prove that, if a branching pomset $R$ is well-formed, then the corresponding protocol is realisable.

To prove that $R$'s protocol is realisable, we have to show the existence of a bisimilar composition of local branching pomsets and buffers. To do this, we define a canonical decomposition of $R$ by combining our previously defined projections with a buffer construction, and we prove that this canonical decomposition is bisimilar to $R$. The (re)construction of the buffer contents of channel $\mathsf{ab}$ based on $R$, written $\mathit{buff}_{\mathsf{ab}}(R)$, and the canonical decomposition of $R$, written $\mathit{cd}(R)$, are defined below.

The buffer construction $\mathit{buff}_{\mathsf{ab}}(R)$ gathers the receive events in $R$ that have no preceding matching send event. We infer that, since the send has already been fired and the receive has not, the message must be in transit.

**Definition 11 (Buffer construction).** *Let $R$ be a branching pomset. Let $\mathsf{a}$ and $\mathsf{b}$ be agents in $R$. Let $\varepsilon$ be the empty word.*

$$\text{Then } \mathit{buff}_{\mathsf{ab}}(R) = \begin{cases} \mathsf{x} \cdot \mathit{buff}_{\mathsf{ab}}(R') & \text{if } R' = R - e \text{ and } \lambda(e) = \mathsf{ab?x} \\ & \text{and } \forall e'\text{: if } e' < e \text{ then } \lambda(e') \neq \mathsf{ab!x} \\ & \text{and } \forall e', \mathsf{y}\text{: if } e' < e \text{ then } \lambda(e') \neq \mathsf{ab?y} \\ \varepsilon & \text{otherwise} \end{cases}$$

The corresponding message types are nondeterministically put in some order that respects the order of the gathered receive events — if $e_1 < e_2$ then $e_1$'s message type must precede that of $e_2$ in the constructed buffer. We note that all unmatched receive events are top-level if $R$ is choreographic, and that the same-channel top-level receive events are totally ordered if $R$ is well-channeled. It follows that, although it may add duplicate messages and is nondeterministic in the general case, $\mathit{buff}_{\mathsf{ab}}(R)$ does not add duplicate messages and is deterministic if $R$ is well-formed.

**Definition 12 (Canonical decomposition).** *Let $R$ be a branching pomset. Let $\vec{R}$ be such that $R_{\mathsf{a}} = R{\downharpoonleft_{\mathsf{a}}}$ for all $\mathsf{a}$. Let $\vec{b}$ be such that $b_{\mathsf{ab}} = \mathit{buff}_{\mathsf{ab}}(R)$ for all $\mathsf{ab}$. Then $\mathit{cd}(R) = \langle \vec{R}, \vec{b} \rangle$ is the* canonical decomposition *of $R$.*

To prove that a well-formed $R$ is bisimilar to $\mathit{cd}(R)$, we define the relation $\mathcal{R} = \{\langle R, \langle \vec{R}^\dagger, \vec{b} \rangle \rangle \mid \langle \vec{R}^\dagger, \vec{b} \rangle \sim \langle \vec{R}, \vec{b} \rangle = \mathit{cd}(R)\}$ and we prove that $\mathcal{R}$ is a bisimulation relation (Theorem 1). Note that the vector of buffers $\vec{b}$ is the same across the definition; we only allow leeway in the vector of local branching pomsets. The proof consists of the two parts mentioned in Section 3. Given that $\langle R, \langle \vec{R}^\dagger, \vec{b} \rangle \rangle \in \mathcal{R}$, if one can make some reduction then the other must be able to make the same reduction such that the resulting configurations are again related by $\mathcal{R}$ (Lemma 7, Lemma 8). Additionally, if one can terminate then so should the other (Lemma 9, Lemma 10).

The reason that $\mathcal{R}$ is not simply the set of all $\langle R, \mathit{cd}(R) \rangle$ is that a reduction from $\mathit{cd}(R)$ may not always result in $\mathit{cd}(R')$ for some $R'$. This is because choices are only resolved in the branching pomset of the agent causing the reduction. For example, consider branching pomset $R_4$ in Figure 4. Upon Alice sending yes the global branching pomset would resolve the choice for both agents simultaneously. However, upon Alice sending yes in the canonical decomposition the projection on Bob remains unchanged and still contains receive events for both yes and no. Since yes has been added to the buffer from Alice to Bob, we know that Bob will eventually have to pick the branch containing yes — after all, there is no no to receive. In other words: this configuration is bisimilar to the canonical decomposition of the resulting global branching pomset, in which the choice has also been resolved for Bob. If there were also some additional no being sent from Alice to Bob, e.g., if we replace the messages int in $R_4$ with no, then $R_4$ being well-channeled and the buffers being ordered would still ensure that we can safely resolve Bob's choice. This crucial insight is formally proven in Lemma 4.

**Lemma 4.** *Let $R$ be a well-formed branching pomset. Let $\langle \vec{R}, \vec{b} \rangle = \mathit{cd}(R)$. Let $\ell$ be some label and let $\mathsf{a} = \mathit{subj}(\ell)$. If $R \xrightarrow{\ell} R'$ and if $\langle \vec{R}, \vec{b} \rangle \xrightarrow{\ell} \langle \vec{R}[R'_{\mathsf{a}}/R{\downharpoonleft_{\mathsf{a}}}], \vec{b}^\dagger \rangle$ and if $R'_{\mathsf{a}} = R'{\downharpoonleft_{\mathsf{a}}}$, then $\langle \vec{R}[R'_{\mathsf{a}}/R{\downharpoonleft_{\mathsf{a}}}], \vec{b}^\dagger \rangle \sim \langle \vec{R}', \vec{b}' \rangle = \mathit{cd}(R')$.*

*Proof (sketch).* If $\ell = \mathsf{ba?x}$ for some $\mathsf{b}, \mathsf{x}$ then it follows from the well-formedness of $R$ that $R' = R - e$ and the remainder is straightforward. The same is true if $\ell = \mathsf{ab!x}$ and $e$ is top-level, i.e., $e \in R.\mathcal{B}$.

Otherwise it follows from the well-formedness of $R$ that $e$ is a minimal send event in one of the options of a top-level choice, i.e., $e \in \mathcal{B} \lhd \mathcal{C} \in R.\mathcal{B}$ for some $\mathcal{B}, \mathcal{C}$, and $R' = R[(R.\mathcal{B} \setminus \mathcal{C}) \cup (\mathcal{B} - e)]$. We proceed to show that the set of unmatched receive events in $R'$ is exactly that of $R$ with the addition of the one corresponding to $e$, and then $\vec{b}' = add(\mathsf{ab!x}, \vec{b}) = \vec{b}^\dagger$. It follows that $\langle \vec{R}[R'_\mathsf{a}/R{\downharpoonleft}_\mathsf{a}], \vec{b}^\dagger\rangle = \langle \vec{R}[R'_\mathsf{a}/R{\downharpoonleft}_\mathsf{a}], \vec{b}'\rangle$. For the projections, we proceed in two steps:

- First we observe that, since $R$ is well-branched, $\mathcal{B}'{\downharpoonleft}_\mathsf{c} = \mathcal{B}{\downharpoonleft}_\mathsf{c}$ for all $\mathcal{B}' \lhd \mathcal{C}$ and for all $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$. It follows that $R{\downharpoonleft}_\mathsf{c} \sim R'{\downharpoonleft}_\mathsf{c}$, and then $\langle \vec{R}[R'_\mathsf{a}/R{\downharpoonleft}_\mathsf{a}], \vec{b}'\rangle \sim \langle \vec{R}'[R{\downharpoonleft}_\mathsf{b}/R'{\downharpoonleft}_\mathsf{b}], \vec{b}'\rangle$. Note that the projection on $\mathsf{a}$ is $R'{\downharpoonleft}_\mathsf{a}$ and the projection on $\mathsf{b}$ is $R{\downharpoonleft}_\mathsf{b}$ on both sides, and the projection on every other $\mathsf{c}$ is bisimilar.
- Next we show that, with the new message added to the buffer, no event can ever fire in $R{\downharpoonleft}_\mathsf{b}$ in any other option of $\mathcal{C}$ than $\mathcal{B}$. It follows that we can discard all other options, and then $\langle \vec{R}'[R{\downharpoonleft}_\mathsf{b}/R'{\downharpoonleft}_\mathsf{b}], \vec{b}'\rangle \sim \langle \vec{R}', \vec{b}'\rangle = cd(R')$. $\square$

To satisfy the preconditions of Lemma 4, we additionally prove that $R$'s reductions can be mirrored by its projection on the reduction label's subject (Lemma 5) and dually that the reductions of $R$'s canonical decomposition can be mirrored by $R$ (Lemma 6). Their proofs rely on the observations that the corresponding event $e$ must be minimal both in $R$ and $R{\downharpoonleft}_\mathsf{a}$, and that the branching structures of the two are the same (modulo discarded leaves). It then follows that the same refinement enables $e$ in both $R$ and $R{\downharpoonleft}_\mathsf{a}$.

**Lemma 5.** *Let $R$ be a tree-like branching pomset. If $R \xrightarrow{\ell} R'$ and $\mathsf{a} = subj(\ell)$ then $R{\downharpoonleft}_\mathsf{a} \xrightarrow{\ell} R'{\downharpoonleft}_\mathsf{a}$.*

**Lemma 6.** *Let $R$ be a well-channeled, tree-like and choreographic branching pomset. Let $\langle \vec{R}, \vec{b}\rangle = cd(R)$. If $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}[R'_\mathsf{a}/R{\downharpoonleft}_\mathsf{a}], \vec{b}'\rangle$ then $R \xrightarrow{\ell} R'$ for some $R'$ such that $R'_\mathsf{a} = R'{\downharpoonleft}_\mathsf{a}$.*

Finally, we bring everything together and prove the four necessary steps for bisimulation in the lemmas below, culminating in Theorem 1. The proof for Lemma 7 uses Lemma 5 to show the preconditions of Lemma 4 and then applies the latter. This gives us $cd(R) \xrightarrow{\ell} cd(R)' \sim cd(R')$, and since $\langle \vec{R}^\dagger, \vec{b}^\dagger\rangle \sim cd(R)$ the result is then straightforward. The proof for Lemma 8 is analogous but uses Lemma 6. The proofs for Lemma 9 and Lemma 10 respectively use Lemma 1 and Lemma 2 to show that, if one can terminate by refining to the empty set, then so must the other.

**Lemma 7.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $R \xrightarrow{\ell} R'$ then there exist $\vec{R}^\ddagger$ and $\vec{b}^\ddagger$ such that $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle$ and $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$.*

**Lemma 8.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle$ then there exists $R'$ such that $R \xrightarrow{\ell} R'$ and $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$.*

**Lemma 9.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $R{\downarrow}$ then $\langle \vec{R}^\dagger, \vec{b}\rangle{\downarrow}$.*

**Lemma 10.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $\langle \vec{R}^\dagger, \vec{b}\rangle{\downarrow}$ then $R{\downarrow}$.*

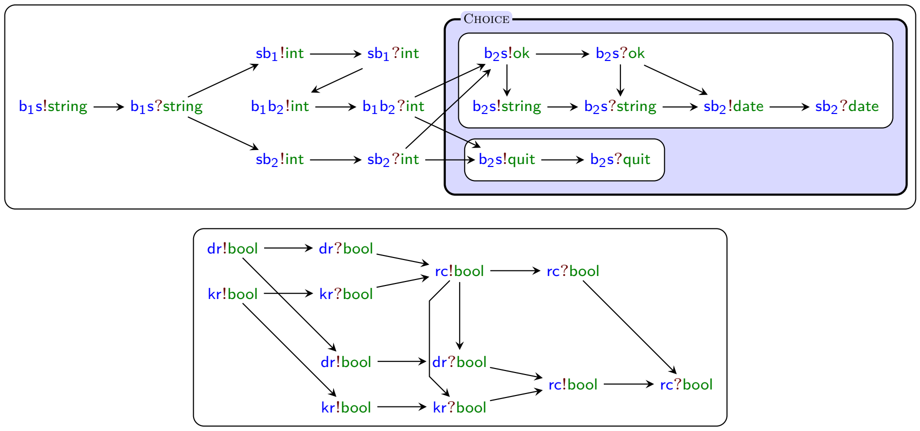


Figure 6: Branching pomsets representing the two-buyers-protocol (top) and two iterations of the simple streaming protocol (bottom) [19].

**Theorem 1.** *Let $R$ be a branching pomset. If $R$ is well-formed and $\mathit{buff}_{\mathsf{ab}}(R) = \varepsilon$ for all* ab *then the protocol represented by $R$ is realisable.*

*Proof.* It follows from Lemma 7, Lemma 8, Lemma 9 and Lemma 10 that $\mathcal{R}$ is a bisimulation relation [28]. Specifically, it then follows that $R \sim cd(R)$. Then, since $\mathit{buff}_{\mathsf{ab}}(R) = \varepsilon$ for all ab, by Definition 3 the protocol represented by $R$ is realisable. □

## 6 Examples

In this section we briefly look at two example protocols used by Honda et al. [19]. Both are depicted as branching pomsets in Figure 6.

The **two-buyers-protocol** (top) features Buyer 1 and Buyer 2 ($\mathsf{b}_1, \mathsf{b}_2$) who wish to jointly buy a book from Seller (s). Buyer 1 first sends the title of the book (string) to Seller, Seller sends its quote (int) to both Buyer 1 and Buyer 2, and Buyer 1 sends Buyer 2 the amount they can contribute (int). Buyer 2 then notifies Seller whether they accept (ok) or reject (quit) the quote. If they accept, they also send their address (string), and Seller returns a delivery date (date).

The **simple streaming protocol** (bottom) features Data Producer (d) and Key Producer (k), who continuously respectively send data and keys (both bool) to Kernel (r). Kernel performs some computation and sends the result (bool) to Consumer (c). The protocol in Figure 6 shows two iterations of this process.

Both branching pomsets are well-formed, and hence the protocols are realisable. We note that, as in the paper by Honda et al., further communication between Buyers 1 and 2 has been omitted in the two-buyers-protocol. Since this is bound to be different in the case of acceptance and rejection, the resulting branching pomset would not be well-branched and thus not well-formed —

though still realisable. We discuss relaxed well-branchedness conditions in Section 7. Also note that the ok and address (string) messages are sent sequentially; sending these in parallel would violate well-channeledness and make the protocol unrealisable with ordered buffers. The same is true for the streaming protocol: the two iterations are composed sequentially and doing so concurrently would violate well-channeledness and result in unrealisability. The size of the branching pomset for the streaming protocol scales linearly with the number of depicted iterations; showing all (infinitely many) iterations would require an infinitely large branching pomset. We briefly touch upon infinity in Section 8.

## 7 Related work

Realisability has been well-studied in the last two decades in a variety of settings. For example, Alur et al. study the realisability of message sequence charts [1]. They define the notions of weak and safe realisability of languages, the latter also ensuring deadlock-freedom, and they define closure conditions on languages which they show to precisely capture weakly and safely realisable languages. Lohmann and Wolf define the notion of distributed realisability, where a protocol is distributedly realisable if there exists a set of compositions such that every composition covers a subset of the protocol and the entire protocol is covered by their union [24]. Fu et al. [15], Basu et al. [3], Finkel and Lozes [14] and Schewe et al. [31] all study the realisability of protocols on different network configurations when considering only the sending behaviour — receive events are omitted — showing necessary and/or sufficient conditions and decidability results.

One major source of inspiration for our work has been previous work on pomsets. Pomsets were initially introduced by Pratt [27] for concurrency models and have been widely used, e.g., in the context of message sequence charts by Katoen and Lambert [23]. Recently Guanciale and Tuosto presented a realisability analysis based on sets of pomsets [17], in which they show how to capture the language closure conditions of Alur et al. [1] directly on pomsets, without having to explicitly compute their language. Typically choreography languages are limited in their expressiveness and any analysis on their realisability is then language-dependent. Both Alur et al. and Guanciale and Tuosto perform a syntax-oblivious analysis, which has the benefit of being applicable to any specification which can be encoded as a set of pomsets, regardless of the specification language. The analysis by Guanciale and Tuosto is at a higher level of abstraction than sets of traces. This allows both for a more efficient analysis and for easier identification of design errors, as these can be identified in a more abstract model.

Our approach is similarly syntax-oblivious, though the analysis itself is based on MPST (on which we will elaborate later). A major difference is that Guanciale and Tuosto use unordered buffers (e.g., non-FIFO queues) while ours are ordered. For example, the parallel composition of $\mathsf{a}{\rightarrow}\mathsf{b}{:}\mathsf{x}$ and $\mathsf{a}{\rightarrow}\mathsf{b}{:}\mathsf{y}$ is realisable in the unordered setting and not in the ordered one. The two settings agree on realisability when the two message types are the same (e.g., two concurrent copies of $a \rightarrow b{:}x$); while Guanciale and Tuosto explicitly note that they support concurrently repeated actions, however, our current well-channeledness condition does not make an exception for these. This is an obvious target for relaxation of our conditions. In their paper, Guanciale and Tuosto also separately define termination soundness, i.e., whether participants are not kept waiting indefinitely after a composition terminates. For example, the branching pomset (right) in Figure 1 is realisable but not termination-sound, as either Carol or Dave will have to wait indefinitely since they do not know that the other received Bob's message. Making this protocol termination-sound would require additional communication between Bob, Carol and/or Dave. Further inspiration for relaxation of well-formedness conditions can be found in an earlier paper by Guanciale and Tuosto [16]. In particular their definition of well-branchedness, using 'active' and 'passive' agents, could serve as a basis for a relaxed version of our own.

The other major source of inspiration for our work is multiparty session types (MPST), introduced by Honda et al. [19]. Specifically:

- Our well-branchedness condition corresponds to the branching syntax of global types in MPST and its definition of projection. Branching in MPST is done on the label of a single message, and the projection on agents uninvolved in the choice is only defined if it is the same in every branch.
- Our well-channeledness condition corresponds to the principle that same-channeled actions should be ordered. We note that our condition is more lenient: we prohibit concurrent sends or receives on the same channel, while in MPST the projection of a parallel composition on an agent is undefined if the agent occurs in both threads (even if the threads' channels are disjoint).
- Our tree-likeness condition follows from the syntax of global types in MPST, which uses a prefix operator rather than a more general sequential composition. As a consequence all global types are tree-like. The same is true for other languages that use a prefix operator and not sequential composition, such as CCS [25] and $\pi$-calculus [29].

Since its introduction, various papers have addressed the strictness of the well-branchedness condition in MPST. One line of research relaxes the condition by using a merge operation to allow all agents to have different behaviour in different branches, as long as they are timely informed of the choice [6,11,30]. Another line relaxes the condition by allowing different branches to start with different receivers, rather than enforcing the same receiver in every branch [10,7,4,9,21]. These results may also serve as inspiration for relaxations of the well-branchedness condition on branching pomsets.

While our current conditions broadly correspond with well-formedness in MPST, we believe that our approach offers three advantages. First, as discussed before, it is syntax-oblivious, meaning it is not only applicable to MPST but to any specification which can be encoded as a branching pomset. Second, we believe that branching pomsets have the potential to be more expressive than global types in MPST. As mentioned above, our well-channeledness condition is already more lenient than the one in MPST. We have described various sources of possible relaxations for our well-branchedness condition, both in the MPST and in the pomset literature. Lastly, we conjecture that our tree-likeness condition is needed to simplify our proofs, and that it is possible — though more complex — to prove realisability without it or at least with a relaxed version of it.

A proper comparison between the pomset-based approach by Guanciale and Tuosto [17] and ours, both in terms of expressiveness and efficiency, would first require further development of our conditions. In the meantime, one takeaway from their paper is the performance gain they obtain by lifting the analysis from languages (sets of traces) to a higher level of abstraction, i.e., sets of pomsets. Our hope is that a further performance gain can be obtained by lifting the analysis from sets of pomsets to an even higher level of abstraction (e.g., branching pomsets).

*Event structures* Finally, the concept of branching pomsets is reminiscent of event structures [26] and their recent usage in the context of MPST [8]. Both consist of a set of events with some causality relation and a choice mechanism. In event structures the choice mechanism consists of a conflict relation, where two conflicting events may not occur together in the same execution; in branching pomsets choices are modelled as a branching structure. A technical difference seems to be the resolving of choices. In event structures a choice (conflict) may only be resolved by firing an event in one of its branches. In contrast, in branching pomsets a choice may be resolved without doing so; instead, one branch may be discarded to fire an event outside of the choice, which is causally dependent on the discarded branch but not on the other. A thorough formal comparison between the two models, including both technical and conceptual aspects, is ongoing work.

## 8 Conclusion

We have defined well-formedness conditions on branching pomsets (Definition 10) and have proven that a well-formed branching pomset represents a realisable protocol (Theorem 1). These conditions are sufficient but not necessary, i.e., a protocol may be realisable if its branching pomset is not well-formed. Examples of this are given in Figure 1 (the branching pomset on the right is realisable but not well-branched) and Figure 4 (branching pomset $R_4$ is realisable but not tree-like). Several routes for relaxing our well-channeledness and especially our well-branchedness conditions have been discussed in Section 7, in the aim of increasing the number of branching pomsets that are well-formed while retaining well-formedness as a sufficient condition for realisability. In the remainder of this section we share some additional thoughts on well-channeledness and tree-likeness, and then briefly discuss the applicability of our work to branching pomsets of infinite size.

*Well-channeledness* The branching pomset $R_3$ in Figure 4 is not well-channeled since the events labelled with $\mathsf{ab}?\mathsf{int}$ and $\mathsf{ab}?\mathsf{bool}$ are unordered. It is unrealisable because a local system will force the $\mathsf{int}$ to be received before the $\mathsf{bool}$ while the global branching pomset allows a different order. However, in this case one may take the view that the problem is not the local system being too strict, but rather the global branching pomset being too lenient in an environment with ordered buffers: it should then in some way allow a user to specify just the two acceptable orderings without having to resort to adding a choice between the two and duplicating all events in the pomset. Therefore, instead of changing the well-channeledness condition, another avenue would be to change the reduction rules in for branching pomsets themselves (Figure 2) and specifically adapt them to ordered buffers. This could be done in such a way that reducing $R_3$ by firing ab!int then automatically adds a dependency from ab?int to ab?bool. This might allow the well-channeledness condition to be significantly relaxed or to possibly be removed altogether.

*Tree-likeness* Having the assumption of tree-likeness simplifies our proofs. It is our aim to eventually relax or even remove it and still prove realisability, but this will require a significantly more complex proof. We have noted in Section 7 that global types in MPST and expressions in, e.g., CCS and the $\pi$-calculus, are tree-like by default. Conceptually a non-tree-like branching pomset could potentially be turned into an equivalent (i.e., bisimilar) tree-like one by distributing the offending events over the branches of the involved choice. For example, consider the branching pomset $R_4$ in Figure 4. By duplicating ab!int and ab?int and adding a copy of each with the relevant dependencies to each of the two branches of the choice, we obtain a bisimilar but now tree-like (and well-formed) branching pomset. A more general scheme may be developed based on versions of the CCS expansion theorem [18,13]. However, regaining expressiveness at the cost of duplicating events effectively negates the benefits of using branching pomsets in the first place.

*Infinity* The paper introducing branching pomsets [12] supports branching pomsets of infinite size. We note that our theoretical results in this paper also hold for infinite branching pomsets. However, determining the well-formedness of an infinite branching pomset is undecidable due to its size. A solution in the case of infinity through repetition, e.g., loops in choreographies, would be to use a symbolic representation. Alternatively, a solution might be found in the extension from message sequence charts (MSCs) to MSC-graphs [20]. A similar extension could be developed for branching pomsets, where they are sequentially composed in a (possibly cyclic) graph. Finally, it may be possible to leverage the recently introduced pomset automata [22].

## References

1. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence charts. IEEE Trans. Software Eng. **29**(7), 623–633 (2003). https://doi.org/10.1109/TSE.2003.1214326, https://doi.org/10.1109/TSE.2003.1214326
2. Alur, R., Etessami, K., Yannakakis, M.: Realizability and verification of MSC graphs. Theor. Comput. Sci. **331**(1), 97–114 (2005). https://doi.org/10.1016/j.tcs.2004.09.034, https://doi.org/10.1016/j.tcs.2004.09.034

3. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: Field, J., Hicks, M. (eds.) Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012. pp. 191–202. ACM (2012). https://doi.org/10.1145/2103656.2103680, https://doi.org/10.1145/2103656.2103680
4. Bocchi, L., Melgratti, H.C., Tuosto, E.: Resolving non-determinism in choreographies. In: Shao, Z. (ed.) Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8410, pp. 493–512. Springer (2014). https://doi.org/10.1007/978-3-642-54833-8_26, https://doi.org/10.1007/978-3-642-54833-8_26
5. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983). https://doi.org/10.1145/322374.322380, https://doi.org/10.1145/322374.322380
6. Carbone, M., Yoshida, N., Honda, K.: Asynchronous session types: Exceptions and multiparty interactions. In: Bernardo, M., Padovani, L., Zavattaro, G. (eds.) Formal Methods for Web Services, 9th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2009, Bertinoro, Italy, June 1-6, 2009, Advanced Lectures. Lecture Notes in Computer Science, vol. 5569, pp. 187–212. Springer (2009). https://doi.org/10.1007/978-3-642-01918-0_5, https://doi.org/10.1007/978-3-642-01918-0_5
7. Castagna, G., Dezani-Ciancaglini, M., Padovani, L.: On global types and multiparty session. Log. Methods Comput. Sci. **8**(1) (2012). https://doi.org/10.2168/LMCS-8(1:24)2012, https://doi.org/10.2168/LMCS-8(1:24)2012
8. Castellani, I., Dezani-Ciancaglini, M., Giannini, P.: Event structure semantics for multiparty sessions. In: Boreale, M., Corradini, F., Loreti, M., Pugliese, R. (eds.) Models, Languages, and Tools for Concurrent and Distributed Programming - Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday. Lecture Notes in Computer Science, vol. 11665, pp. 340–363. Springer (2019). https://doi.org/10.1007/978-3-030-21485-2_19, https://doi.org/10.1007/978-3-030-21485-2_19
9. Castellani, I., Dezani-Ciancaglini, M., Giannini, P.: Reversible sessions with flexible choices. Acta Informatica **56**(7-8), 553–583 (2019). https://doi.org/10.1007/s00236-019-00332-y, https://doi.org/10.1007/s00236-019-00332-y
10. Deniélou, P., Yoshida, N.: Multiparty session types meet communicating automata. In: Seidl, H. (ed.) Programming Languages and Systems - 21st European Symposium on Programming, ESOP 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7211, pp. 194–213. Springer (2012). https://doi.org/10.1007/978-3-642-28869-2_10, https://doi.org/10.1007/978-3-642-28869-2_10
11. Deniélou, P., Yoshida, N., Bejleri, A., Hu, R.: Parameterised multiparty session types. Log. Methods Comput. Sci. **8**(4) (2012). https://doi.org/10.2168/LMCS-8(4:6)2012, https://doi.org/10.2168/LMCS-8(4:6)2012
12. Edixhoven, L., Jongmans, S.S., Proença, J., Cledou, G.: Branching pomsets for choreographies. In: Proceedings 15th Interaction and Concurrency Experience, ICE 2022, Lucca, Italy, 17 June 2022. EPTCS (2022). https://doi.org/10.48550/arXiv.2208.04632

13. Ferrari, G.L., Gorrieri, R., Montanari, U.: An extended expansion theorem. In: Abramsky, S., Maibaum, T.S.E. (eds.) TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Brighton, UK, April 8-12, 1991, Volume 2: Advances in Distributed Computing (ADC) and Colloquium on Combining Paradigms for Software Developmemnt (CCPSD). Lecture Notes in Computer Science, vol. 494, pp. 29–48. Springer (1991). https://doi.org/10.1007/3540539816_56, https://doi.org/10.1007/3540539816_56
14. Finkel, A., Lozes, É.: Synchronizability of communicating finite state machines is not decidable. In: Chatzigiannakis, I., Indyk, P., Kuhn, F., Muscholl, A. (eds.) 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland. LIPIcs, vol. 80, pp. 122:1–122:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ICALP.2017.122, https://doi.org/10.4230/LIPIcs.ICALP.2017.122
15. 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). https://doi.org/10.1016/j.tcs.2004.07.004, https://doi.org/10.1016/j.tcs.2004.07.004
16. Guanciale, R., Tuosto, E.: An abstract semantics of the global view of choreographies. In: Bartoletti, M., Henrio, L., Knight, S., Vieira, H.T. (eds.) Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016. EPTCS, vol. 223, pp. 67–82 (2016). https://doi.org/10.4204/EPTCS.223.5, https://doi.org/10.4204/EPTCS.223.5
17. Guanciale, R., Tuosto, E.: Realisability of pomsets. J. Log. Algebraic Methods Program. **108**, 69–89 (2019). https://doi.org/10.1016/j.jlamp.2019.06.003, https://doi.org/10.1016/j.jlamp.2019.06.003
18. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. J. ACM **32**(1), 137–161 (1985). https://doi.org/10.1145/2455.2460, https://doi.org/10.1145/2455.2460
19. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Necula, G.C., Wadler, P. (eds.) Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008. pp. 273–284. ACM (2008). https://doi.org/10.1145/1328438.1328472, https://doi.org/10.1145/1328438.1328472
20. ITU-TS: ITU-TS Recommendation Z.120: Message Sequence Chart 2011 (MSC11). Tech. rep., ITU-TS, Geneva (2011)
21. Jongmans, S., Yoshida, N.: Exploring type-level bisimilarity towards more expressive multiparty session types. In: Müller, P. (ed.) Programming Languages and Systems - 29th European Symposium on Programming, ESOP 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12075, pp. 251–279. Springer (2020). https://doi.org/10.1007/978-3-030-44914-8_10, https://doi.org/10.1007/978-3-030-44914-8_10
22. Kappé, T., Brunet, P., Luttik, B., Silva, A., Zanasi, F.: On series-parallel pomset languages: Rationality, context-freeness and automata. J. Log. Algebraic Methods Program. **103**, 130–153 (2019). https://doi.org/10.1016/j.jlamp.2018.12.001, https://doi.org/10.1016/j.jlamp.2018.12.001
23. Katoen, J., Lambert, L.: Pomsets for MSC. In: König, H., Langendörfer, P. (eds.) Formale Beschreibungstechniken für verteilte Systeme, 8. GI/ITG-Fachgespräch, Cottbus, 4. und 5. Juni 1998. pp. 197–207. Verlag Shaker (1998)

24. Lohmann, N., Wolf, K.: Realizability is controllability. In: Laneve, C., Su, J. (eds.) Web Services and Formal Methods, 6th International Workshop, WS-FM 2009, Bologna, Italy, September 4-5, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6194, pp. 110–127. Springer (2009). https://doi.org/10.1007/978-3-642-14458-5_7, https://doi.org/10.1007/978-3-642-14458-5_7
25. Milner, R.: A Calculus of Communicating Systems, Lecture Notes in Computer Science, vol. 92. Springer (1980). https://doi.org/10.1007/3-540-10235-3, https://doi.org/10.1007/3-540-10235-3
26. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part I. Theor. Comput. Sci. **13**, 85–108 (1981). https://doi.org/10.1016/0304-3975(81)90112-2, https://doi.org/10.1016/0304-3975(81)90112-2
27. Pratt, V.R.: Modeling concurrency with partial orders. Int. J. Parallel Program. **15**(1), 33–71 (1986). https://doi.org/10.1007/BF01379149, https://doi.org/10.1007/BF01379149
28. Sangiorgi, D.: Introduction to bisimulation and coinduction. Cambridge University Press (2011). https://doi.org/10.1017/CBO9780511777110
29. Sangiorgi, D., Walker, D.: The Pi-Calculus - a theory of mobile processes. Cambridge University Press (2001)
30. Scalas, A., Yoshida, N.: Less is more: multiparty session types revisited. Proc. ACM Program. Lang. **3**(POPL), 30:1–30:29 (2019). https://doi.org/10.1145/3290343, https://doi.org/10.1145/3290343
31. Schewe, K., Ameur, Y.A., Benyagoub, S.: Realisability of choreographies. In: Herzig, A., Kontinen, J. (eds.) Foundations of Information and Knowledge Systems - 11th International Symposium, FoIKS 2020, Dortmund, Germany, February 17-21, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12012, pp. 263–280. Springer (2020). https://doi.org/10.1007/978-3-030-39951-1_16, https://doi.org/10.1007/978-3-030-39951-1_16

## A Proofs from the paper

**Lemma 1.** *If $R \sqsupseteq R'$ then $R{\downharpoonright_{\mathsf{a}}} \sqsupseteq R'{\downharpoonright_{\mathsf{a}}}$.*

*Proof.* Let $\mathcal{B} = R.\mathcal{B}$, $\mathcal{B}' = R'.\mathcal{B}$, $\mathcal{B}_{\mathsf{a}} = R{\downharpoonright_{\mathsf{a}}}.\mathcal{B}$ and $\mathcal{B}'_{\mathsf{a}} = R'{\downharpoonright_{\mathsf{a}}}$. We have that $\mathcal{B} \sqsupseteq \mathcal{B}'$ and need to show that $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$. The proof is by induction on the structure of the derivation tree of $\mathcal{B} \sqsupseteq \mathcal{B}'$. Consider its root, i.e., the final step in the derivation.

- **Base:** Refl. Then $\mathcal{B} = \mathcal{B}'$ and $\mathcal{B}_{\mathsf{a}} = \mathcal{B}'_{\mathsf{a}}$. By Refl $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$.
- **Base:** Choice. Then $\mathcal{B} = \{\{\mathcal{B}_1, \mathcal{B}_2\}\} \cup \mathcal{B}^\dagger$ and $\mathcal{B}' = \mathcal{B}_i \cup \mathcal{B}^\dagger$ for some $i \in \{1,2\}$. It follows that $\mathcal{B}_{\mathsf{a}} = \{\{\mathcal{B}_1{\downharpoonright_{\mathsf{a}}}, \mathcal{B}_2{\downharpoonright_{\mathsf{a}}}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$. Then by Choice $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}_i{\downharpoonright_{\mathsf{a}}} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$.
- **Step:** Trans. Then $\mathcal{B} \sqsupseteq \mathcal{B}'' \sqsupseteq \mathcal{B}'$ for some $\mathcal{B}''$. By the induction hypothesis $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}''{\downharpoonright_{\mathsf{a}}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$. Then by Trans $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$.
- **Step:** Congr. Then $\mathcal{B} = \{\{\mathcal{B}_1, \mathcal{B}_2\}\} \cup \mathcal{B}^\dagger$ and $\mathcal{B}' = \{\{\mathcal{B}'_1, \mathcal{B}'_2\}\} \cup \mathcal{B}^\dagger$ where $\mathcal{B}_i \sqsupseteq \mathcal{B}'_i$ for some $i \in \{1,2\}$. It follows that $\mathcal{B}_{\mathsf{a}} = \{\{\mathcal{B}_1{\downharpoonright_{\mathsf{a}}}, \mathcal{B}_2{\downharpoonright_{\mathsf{a}}}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$, and likewise $\mathcal{B}'_{\mathsf{a}} = \{\{\mathcal{B}'_1{\downharpoonright_{\mathsf{a}}}, \mathcal{B}'_2{\downharpoonright_{\mathsf{a}}}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$. By the induction hypothesis $\mathcal{B}_i{\downharpoonright_{\mathsf{a}}} \sqsupseteq \mathcal{B}'_i{\downharpoonright_{\mathsf{a}}}$ for $i \in \{1,2\}$. Then by Congr $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$. □

**Lemma 2.** *If $R{\downharpoonright_{\mathsf{a}}} \sqsupseteq R'_{\mathsf{a}}$ then $R \sqsupseteq R'$ for some $R'$ such that $R'_{\mathsf{a}} = R'{\downharpoonright_{\mathsf{a}}}$.*

*Proof.* Let $\mathcal{B} = R.\mathcal{B}$, $\mathcal{B}' = R'.\mathcal{B}$, $\mathcal{B}_{\mathsf{a}} = R{\downharpoonright_{\mathsf{a}}}.\mathcal{B}$ and $\mathcal{B}'_{\mathsf{a}} = R'_{\mathsf{a}}$. We have that $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$ and need to show that $\mathcal{B} \sqsupseteq \mathcal{B}'$ and $\mathcal{B}'{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$. The proof is by induction on the structure of the derivation tree of $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$. Consider its root, i.e., the final step in the derivation.

- **Base:** Refl. Then $\mathcal{B}_{\mathsf{a}} = \mathcal{B}'_{\mathsf{a}}$ and it suffices to take $\mathcal{B}' = \mathcal{B}$.
- **Base:** Choice. Then $\mathcal{B}_{\mathsf{a}} = \{\{\mathcal{B}_1{\downharpoonright_{\mathsf{a}}}, \mathcal{B}_2{\downharpoonright_{\mathsf{a}}}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$ for some $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}^\dagger$, and, without loss of generality, $\mathcal{B}'_{\mathsf{a}} = \mathcal{B}_1{\downharpoonright_{\mathsf{a}}} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$. Then $\mathcal{B}' = \mathcal{B}_1 \cup \mathcal{B}^\dagger$. By Choice $\mathcal{B} \sqsupseteq \mathcal{B}'$ and $\mathcal{B}'{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$.
- **Step:** Trans. Then $\mathcal{B}_{\mathsf{a}} \sqsupseteq \mathcal{B}''_{\mathsf{a}} \sqsupseteq \mathcal{B}'_{\mathsf{a}}$ for some $\mathcal{B}''_{\mathsf{a}}$. By the induction hypothesis there exists some $\mathcal{B}''$ such that $\mathcal{B} \sqsupseteq \mathcal{B}'' \sqsupseteq \mathcal{B}'$ and $\mathcal{B}''{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$ and $\mathcal{B}'{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$. Then by Trans $\mathcal{B} \sqsupseteq \mathcal{B}'$.
- **Step:** Congr. Then $\mathcal{B}_{\mathsf{a}} = \{\{\mathcal{B}_1{\downharpoonright_{\mathsf{a}}}, \mathcal{B}_2{\downharpoonright_{\mathsf{a}}}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$ and $\mathcal{B}'_{\mathsf{a}} = \{\{\mathcal{B}'_{\mathsf{a}1}, \mathcal{B}'_{\mathsf{a}2}\}\} \cup \mathcal{B}^\dagger{\downharpoonright_{\mathsf{a}}}$ for some $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}^\dagger, \mathcal{B}'_{\mathsf{a}1}, \mathcal{B}'_{\mathsf{a}2}$ such that $\mathcal{B}_1{\downharpoonright_{\mathsf{a}}} \sqsupseteq \mathcal{B}'_{\mathsf{a}1}$ and $\mathcal{B}_2{\downharpoonright_{\mathsf{a}}} \sqsupseteq \mathcal{B}'_{\mathsf{a}2}$. By the induction hypothesis there exist $\mathcal{B}'_1$ and $\mathcal{B}'_2$ such that $\mathcal{B}_1 \sqsupseteq \mathcal{B}'_1$ and $\mathcal{B}'_1{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}1}$ and likewise for $\mathcal{B}'_2$. Then by Congr $\mathcal{B} \sqsupseteq \{\{\mathcal{B}'_1, \mathcal{B}'_2\}\} \cup \mathcal{B}^\dagger = \mathcal{B}'$ and $\mathcal{B}'{\downharpoonright_{\mathsf{a}}} = \mathcal{B}'_{\mathsf{a}}$. □

**Lemma 3.** *Let $R$ be a branching pomset and let $R \xrightarrow{\ell} R'$. If $R$ is well-formed then so is $R'$.*

*Proof.*

- **Well-branched:** Let $\mathcal{C}' \prec R'.\mathcal{B}$. Since $R' = R^\dagger - e$ for some $R \sqsupseteq R^\dagger$, there exists some $\mathcal{C} \prec R.\mathcal{B}$ such that, for every $\mathcal{B}' \lhd \mathcal{C}'$, there exists some $\mathcal{B} \lhd \mathcal{C}$ such that $\mathcal{B} \sqsupseteq \mathcal{B}'$. Since $R$ is well-branched the well-branchedness conditions must hold for $\mathcal{C}$, and then they also hold for $\mathcal{C}'$.

- **Well-channeled:** Let $e_1, e_2, e_3, e_4 \in R'.E$.
  - Suppose that $e_1$ and $e_2$ are either both sends or both receives sharing the same channel. Then, since $R$ is well-channeled, they are either causally ordered or mutually exclusive in $R$. It follows that they are also in $R'$.
  - Suppose that $e_1$ and $e_2$ are two send events sharing the same channel, that $e_3$ and $e_4$ are matching receives for respectively $e_1$ and $e_2$, that there exists no $e_5 \in R'.E$ such that $e_1 < e_5 < e_3$ or $e_2 < e_5 < e_4$, and that $e_1 \leq e_2$. Suppose, for the sake of contradiction, that there exists some $e_5 \in R.E$ such that $e_1 \; R.\!< e_5 \; R.\!< e_3$ (the case $e_2 \; R.\!< e_5 \; R.\!< e_4$ is analogous). Then, since $R$ is tree-like and $e_3 \in R'.E$, also $e_5 \in R'.E$ which contradicts our first assumption. It follows that there does not exist any such $e_5$ in $R.E$, and then, since $R$ is well-channeled, $e_3 \; R.\!\leq e_4$ and then also $e_3 \; R'.\!\leq e_4$.
- **Tree-like:** Let $\{\mathcal{B}'_1, \mathcal{B}'_2\} \prec R'.\mathcal{B}$ and let $e_1, e_2 \in R'.E$ such that $e_1 \; R'.\!\leq e_2$ and, without loss of generality, $e_1 \preceq \mathcal{B}'_1$. Since $R' = R^\dagger - e$ for some $R \sqsupseteq R^\dagger$, there exists some $\{\mathcal{B}_1, \mathcal{B}_2\} \prec R.\mathcal{B}$ such that $\mathcal{B}_1 \sqsupseteq \mathcal{B}'_1$ and $\mathcal{B}_2 \sqsupseteq \mathcal{B}'_2$. It follows that $e_1 \; R.\!\leq e_2$ and $e_1 \preceq \mathcal{B}_1$, and since $R$ is tree-like then $e_2 \preceq \mathcal{B}_1$. Then also $e_2 \preceq \mathcal{B}'_1$.
- **Choreographic:** Due to its size, this is covered separately by Lemma 11.

□

**Lemma 4.** *Let $R$ be a well-formed branching pomset. Let $\langle \vec{R}, \vec{b} \rangle = cd(R)$. Let $\ell$ be some label and let $\mathsf{a} = subj(\ell)$. If $R \xrightarrow{\ell} R'$ and if $\langle \vec{R}, \vec{b} \rangle \xrightarrow{\ell} \langle \vec{R}[R'_\mathsf{a}/R\lfloor_\mathsf{a}], \vec{b}^\dagger \rangle$ and if $R'_\mathsf{a} = R'\lfloor_\mathsf{a}$, then $\langle \vec{R}[R'_\mathsf{a}/R\lfloor_\mathsf{a}], \vec{b}^\dagger \rangle \sim \langle \vec{R}', \vec{b}' \rangle = cd(R')$.*

*Proof.* Let $R_\mathsf{a} = R\lfloor_\mathsf{a}$. Since $\langle \vec{R}, \vec{b} \rangle \xrightarrow{\ell} \langle \vec{R}[R'_\mathsf{a}/R_\mathsf{a}], \vec{b}^\dagger \rangle$, it follows from Definition 2 that $R_\mathsf{a} \xrightarrow{\ell} R'_\mathsf{a}$. Let $e \in R.E$ be the event such that $R \xrightarrow{e} R'$.

Since $R$ it tree-like, it follows from Lemma 13 that $e$ is minimal in $R$. It then also follows that $e$ is minimal in $R_\mathsf{a}$.

- Suppose that $\ell = \mathsf{ba?x}$ for some $\mathsf{b}, \mathsf{x}$. Then $\vec{b}^\dagger = remove(\mathsf{ba!x}, \vec{b})$.
  Since $e$ is minimal in $R$ and $R$ is choreographic, $e \in R.\mathcal{B}$ and then $R' = R - e$.
  **Buffers**:
  - Since $R$ and $R'$ are well-formed and choreographic, it follows from Lemma 12 that $\vec{b}$ and $\vec{b}'$ are uniquely defined.
  - Since $R' = R - e$, it then follows that $\vec{b}' = remove(\mathsf{ba!x}, \vec{b})$.

  **Projections**:
  - Since $R' = R - e$, it follows that $R\lfloor_\mathsf{b} = R'\lfloor_\mathsf{b}$ for all $\mathsf{b} \neq \mathsf{a}$.
  - Then $\vec{R}' = \vec{R}[R'_\mathsf{a}/R_\mathsf{a}]$.

  We conclude that $\langle \vec{R}[R'_\mathsf{a}/R_\mathsf{a}], \vec{b}^\dagger \rangle = \langle \vec{R}', \vec{b}' \rangle$ and then $\langle \vec{R}[R'_\mathsf{a}/R_\mathsf{a}], \vec{b}^\dagger \rangle \sim \langle \vec{R}', \vec{b}' \rangle$.
- Suppose that $\ell = \mathsf{ab!x}$ for some $\mathsf{b}, \mathsf{x}$. Then $\vec{b}^\dagger = add(\mathsf{ab!x}, \vec{b})$.
  If $e \in R.\mathcal{B}$ then we can proceed as above. Suppose, for the sake of contradiction, that $e \in \mathcal{B}_1 \prec \mathcal{B}_2 \lhd \mathcal{C} \in R.\mathcal{B}$ such that $\mathcal{B}_1 \not\lhd \mathcal{C}$. Since $R$ is well-branched and since $e \prec \mathcal{B}_2$, there then exists some $e' \in \mathcal{B}_2$ such that $e' < e$. However, this contradicts $e$ being minimal in $R$. We can thus assume for the remainder of this case that $e \in \mathcal{B} \lhd \mathcal{C} \in R.\mathcal{B}$ for some $\mathcal{B}, \mathcal{C}$, and $R' = R[(R.\mathcal{B} \setminus \mathcal{C}) \cup (\mathcal{B} - e)]$.

**Buffers**:

- Since $R$ is choreographic and $e$ is minimal, there exists exactly one event $e'$ with $\lambda(e') = \mathsf{ab?x}$ whose sole preceding matching send event is $e$.
- Since $R$ is tree-like and $e \preceq \mathcal{B}$ and $e \leq e'$, it follows that $e' \preceq \mathcal{B}$.
- Let $R.\hat{E}$ be the set of receive events in $R.E$ without a preceding matching send event, and $R'.\hat{E}$ the same for $R'$.
- Since $R$ is choreographic, if $\hat{e} \in R.\hat{E}$ then $\hat{e} \in R.\mathcal{B}$, and similarly for $R'$.
- Since $e' \in R'.\hat{E}$, it follows that $e' \in R'.\mathcal{B}$.
- Since $R$ is tree-like, the events in $R.\hat{E}$ are not affected by the transition on $\ell$ and it follows that $R.\hat{E} \cup \{e'\} \subseteq R'.\hat{E}$.
- Suppose, for the sake of contradiction, that there exists some $e^\dagger \in R'.\hat{E}$ such that $e^\dagger \notin R.\hat{E}$ and $e^\dagger \neq e'$.
  * Since $R'.E \subseteq R.E$, it follows that $e^\dagger \in R.E$.
  * Then $e^\dagger$ has a preceding matching send $e^\ddagger$ in $R$ but not in $R'$. Since $R' = R[(R.\mathcal{B} \setminus \mathcal{C}) \cup (\mathcal{B} - e)]$ and $R$ and $R'$ are choreographic, it follows that $e^\ddagger \prec \mathcal{B}$.
  * Since $e^\ddagger \notin R'.E$, it follows that $e^\ddagger = e$, which contradicts $R$ being choreographic.
- It follows that $R'.\hat{E} = R.\hat{E} \cup \{e'\}$.
- Since $R$ and $R'$ are well-channeled and choreographic, $R.\hat{E}\lfloor_{\mathsf{cd}}$ and $R'.\hat{E}\lfloor_{\mathsf{cd}}$ are totally ordered for all $\mathsf{cd}$ (Lemma 12).
- Since $R$ is well-channeled and tree-like, $\hat{e} \leq e'$ for all $\hat{e} \in R.\hat{E}\lfloor_{\mathsf{ab}}$.
- Then $\vec{b}' = \mathit{add}(\mathsf{ab!x}, \vec{b})$ and $\langle \vec{R}[R'_{\mathsf{a}}/R_{\mathsf{a}}], \mathit{add}(\mathsf{ab!x}, \vec{b})\rangle = \langle \vec{R}[R'_{\mathsf{a}}/R_{\mathsf{a}}], \vec{b}'\rangle$.

**Projections**:

- Since $R$ is well-branched, it follows that $\mathcal{B}_1\lfloor_{\mathsf{c}} = \mathcal{B}_2\lfloor_{\mathsf{c}}$ for all $\mathcal{B}_1, \mathcal{B}_2 \lhd \mathcal{C}$ and $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$.
- Since $R$ is tree-like, picking one of the options of $\mathcal{C}$ does not influence any event outside of $\mathcal{C}$.
- Since $R' = R[(R.\mathcal{B} \setminus \mathcal{C}) \cup (\mathcal{B} - e)]$, and since $\mathcal{B}'\lfloor_{\mathsf{c}} = \mathcal{B}\lfloor_{\mathsf{c}}$ for all $\mathcal{B}' \lhd \mathcal{C}$ and for all $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$, since $R$ is tree-like and since $\mathit{subj}(e) = \mathsf{a}$, it follows that $R\lfloor_{\mathsf{c}} \sim R'\lfloor_{\mathsf{c}}$ for all $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$.
- It then follows that $\langle \vec{R}[R'_{\mathsf{a}}/R_{\mathsf{a}}], \vec{b}'\rangle \sim \langle \vec{R}'[R\lfloor_{\mathsf{b}}/R'\lfloor_{\mathsf{b}}], \vec{b}'\rangle$. Note that the projection on $\mathsf{a}$ on both sides is $R'_{\mathsf{a}}$ and the projection on $\mathsf{b}$ on both sides is $R\lfloor_{\mathsf{b}}$, and that $R\lfloor_{\mathsf{c}} \sim R'\lfloor_{\mathsf{c}}$ for all $\mathsf{c} \neq \mathsf{a}, \mathsf{b}$. In other words: all components are pairwise bisimilar.
- Suppose, for the sake of contradiction, that $R\lfloor_{\mathsf{b}}$ would eventually be allowed to fire an event not in $R'\lfloor_{\mathsf{b}}$.
  * Since the event is in $R\lfloor_{\mathsf{b}}$ but not in $R'\lfloor_{\mathsf{b}}$, it must be in an option $\mathcal{B}_b \lhd \mathcal{C}\lfloor_{\mathsf{b}}$ such that $\mathcal{B}_b \neq \mathcal{B}\lfloor_{\mathsf{b}}$.
  * Since $R$ is well-branched, there exists an event $e_b \in \mathcal{B}_b$ such that $e_b \leq e'_b$ for all $e'_b \preceq \mathcal{B}_b$, and $\lambda(e_b) = \mathsf{ab?y}$ for some $\mathsf{y} \neq \mathsf{x}$. It follows that $e_b$ will need to be fired eventually.
  * Let $n = |\vec{b}'_{\mathsf{ab}}|$ be the number of messages on channel $\mathsf{ab}$ in $\vec{b}'$.
  * There are $n - 1$ events $\hat{e} \in R\lfloor_{\mathsf{b}}.\mathcal{B}$ with $\lambda(\hat{e}) = \mathsf{ab?z}$ for some $\mathsf{z}$. Since $R$ is well-channeled, these precede all other receive events on $\mathsf{ab}$.

- * The last message in $\vec{b}'_{\mathsf{ab}}$ is $\mathsf{x}$, while we need a first message $\mathsf{y}$ to fire $e_b$. Since the buffers are order-preserving, we thus need to fire an event $e'_b$ with $\lambda(e'_b) = \mathsf{ab?x}$ first.
- * Since $R$ is well-channeled and $e_b \neq e'_b$, either:
  - · $e_b < e'_b$, in which case we cannot fire $e'_b$ before $e_b$; or
  - · $e'_b < e_b$, in which case $e'_b \in R\lfloor_{\mathsf{b}}.\mathcal{B}$ and then it will be fired by one of the first $n-1$ messages; or
  - · $e_b$ and $e'_b$ are mutually exclusive, in which case firing $e'_b$ would permanently remove $e_b$.
- * We conclude that we cannot fire $e_b$ at any point, and therefore that we cannot fire any $\mathsf{b}$-event in another option of $\mathcal{C}$ than $\mathcal{B}\lfloor_{\mathsf{b}}$.

- Since we can then discard the redundant branches without changing the system's behaviour, we obtain: $\langle \vec{R}'[R\lfloor_{\mathsf{b}}/R'\lfloor_{\mathsf{b}}], \vec{b}'\rangle \sim \langle \vec{R}'[R\lfloor_{\mathsf{b}}/R'\lfloor_{\mathsf{b}}][R'\lfloor_{\mathsf{b}}/R\lfloor_{\mathsf{b}}], \vec{b}'\rangle = \langle \vec{R}', \vec{b}'\rangle$. □

**Lemma 5.** *Let $R$ be a tree-like branching pomset. If $R \xrightarrow{\ell} R'$ and $\mathsf{a} = subj(\ell)$ then $R\lfloor_{\mathsf{a}} \xrightarrow{\ell} R'\lfloor_{\mathsf{a}}$.*

*Proof.* By definition $R \xrightarrow{\checkmark_e} R^\dagger$ and $R' = R^\dagger - e$ for some $R^\dagger$ and $e$ such that $\lambda(e) = \ell$. Let $R_{\mathsf{a}} = R\lfloor_{\mathsf{a}}$.

Since $R$ is tree-like, by Lemma 13 $e$ is minimal in $R$. It follows that $e$ is also minimal in $R_{\mathsf{a}}$. Since $e$ is minimal in $R_{\mathsf{a}}$, it follows that $R_{\mathsf{a}} \xrightarrow{\checkmark_e} R^\dagger_{\mathsf{a}}$ and $R_{\mathsf{a}} \xrightarrow{e} R'_{\mathsf{a}}$ for some $R'_{\mathsf{a}} = R^\dagger_{\mathsf{a}} - e$.

Without loss of generality, $e \in \mathcal{B}_1 \in \mathcal{C}_1 \in \ldots \in \mathcal{B}_m \in \mathcal{C}_m \in R.\mathcal{B}$ for some $\mathcal{B}_1, \mathcal{C}_1, \ldots, \mathcal{B}_m, \mathcal{C}_m$ ($m \geq 0$). Since $e$ is minimal in $R$, the maximal refinement $R^\dagger$ consists of applying CHOICE to resolve $\mathcal{C}_1, \ldots, \mathcal{C}_m$ and bring $e$ to the top level (i.e., $e \in R.\mathcal{B}$). Formally: $R^\dagger = (R.\mathcal{B} \setminus \mathcal{C}_m) \cup (\mathcal{B}_m \setminus \mathcal{C}_{m-1}) \cup \ldots \cup (\mathcal{B}_2 \setminus \mathcal{C}_1) \cup \mathcal{B}_1$. It follows that $e \in \mathcal{B}_1\lfloor_{\mathsf{a}} \in \mathcal{C}_1\lfloor_{\mathsf{a}} \in \ldots \in \mathcal{B}_m\lfloor_{\mathsf{a}} \in \mathcal{C}_m\lfloor_{\mathsf{a}} \in R.\mathcal{B}\lfloor_{\mathsf{a}}$. Analogously, $R^\dagger_{\mathsf{a}} = (R.\mathcal{B}\lfloor_{\mathsf{a}} \setminus \mathcal{C}_m\lfloor_{\mathsf{a}}) \cup (\mathcal{B}_m\lfloor_{\mathsf{a}} \setminus \mathcal{C}_{m-1}\lfloor_{\mathsf{a}}) \cup \ldots \cup (\mathcal{B}_2\lfloor_{\mathsf{a}} \setminus \mathcal{C}_1\lfloor_{\mathsf{a}}) \cup \mathcal{B}_1\lfloor_{\mathsf{a}} = R^\dagger\lfloor_{\mathsf{a}}$.

It follows that $R'_{\mathsf{a}} = R'\lfloor_{\mathsf{a}}$. □

**Lemma 6.** *Let $R$ be a well-channeled, tree-like and choreographic branching pomset. Let $\langle \vec{R}, \vec{b}\rangle = cd(R)$. If $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}[R'_{\mathsf{a}}/R\lfloor_{\mathsf{a}}], \vec{b}'\rangle$ then $R \xrightarrow{\ell} R'$ for some $R'$ such that $R'_{\mathsf{a}} = R'\lfloor_{\mathsf{a}}$.*

*Proof.* This proof is analogous to that of Lemma 5.

By definition $R\lfloor_{\mathsf{a}} \xrightarrow{\checkmark_e} R^\dagger_{\mathsf{a}}$ and $R'_{\mathsf{a}} = R^\dagger_{\mathsf{a}} - e$ for some $R^\dagger_{\mathsf{a}}$ and $e$ such that $\lambda(e) = \ell$.

Since $R$ is tree-like, so is $R\lfloor_{\mathsf{a}}$ and then by Lemma 13 $e$ is minimal in $R\lfloor_{\mathsf{a}}$. Assume, for the sake of contradiction, that there exists some event $e' \in R.E$ such that $e' < e$, i.e., that $e$ is not minimal in $R$. Then, since $R$ is choreographic, there exists some $e'' \in R.E$ such that $e'' < e$ and either $subj(\lambda(e'')) = \mathsf{a}$, which contradicts $e$ being minimal in $R\lfloor_{\mathsf{a}}$, or $\lambda(e'') = \mathsf{ba!x}$ and $\lambda(e) = \mathsf{ba?x}$ for some $\mathsf{b}, \mathsf{x}$, in which case $\langle \vec{R}, \vec{b}\rangle \stackrel{\ell}{\not\rightarrow} \langle \vec{R}[R'_{\mathsf{a}}/R\lfloor_{\mathsf{a}}], \vec{b}'\rangle$. After all, $has(\mathsf{ba!x}, \vec{b})$ must hold but the message $\mathsf{x}$ cannot come from $e$ (Definition 11). There must then exist some $e^\dagger$ with no matching send and $\lambda(e^\dagger) = \mathsf{ba?x}$. Since $R$ is well-channeled, then either $e < e^\dagger$, in which case $e'' < e^\dagger$, or $e^\dagger < e$, in which case $e$ is not minimal in

$R\lfloor_{\mathsf{a}}$, or $e^\dagger$ and $e$ are mutually exclusive, which is impossible since $R$ is tree-like and $e^\dagger$ must then be top-level. There thus does not exist any such $e^\dagger$, and we can conclude that $e$ is also minimal in $R$.

Since $e$ is minimal in $R$, it follows that $R \xrightarrow{\checkmark_e} R^\dagger$ and $R \xrightarrow{e} R'$ for some $R' = R^\dagger - e$.

Without loss of generality, $e \in \mathcal{B}_1 \in \mathcal{C}_1 \in \ldots \in \mathcal{B}_m \in \mathcal{C}_m \in R.\mathcal{B}$ for some $\mathcal{B}_1, \mathcal{C}_1, \ldots, \mathcal{B}_m, \mathcal{C}_m$ $(m \geq 0)$. Since $e$ is minimal in $R$, the maximal refinement $R^\dagger$ consists of applying Choice to resolve $\mathcal{C}_1, \ldots, \mathcal{C}_m$ and bring $e$ to the top level (i.e., $e \in R.\mathcal{B}$). Formally: $R^\dagger = (R.\mathcal{B} \setminus \mathcal{C}_m) \cup (\mathcal{B}_m \setminus \mathcal{C}_{m-1}) \cup \ldots \cup (\mathcal{B}_2 \setminus \mathcal{C}_1) \cup \mathcal{B}_1$.

It follows that $e \in \mathcal{B}_1\lfloor_{\mathsf{a}} \in \mathcal{C}_1\lfloor_{\mathsf{a}} \in \ldots \in \mathcal{B}_m\lfloor_{\mathsf{a}} \in \mathcal{C}_m\lfloor_{\mathsf{a}} \in R.\mathcal{B}\lfloor_{\mathsf{a}}$. Analogously, $R^\dagger_a = (R.\mathcal{B}\lfloor_{\mathsf{a}} \setminus \mathcal{C}_m\lfloor_{\mathsf{a}}) \cup (\mathcal{B}_m\lfloor_{\mathsf{a}} \setminus \mathcal{C}_{m-1}\lfloor_{\mathsf{a}}) \cup \ldots \cup (\mathcal{B}_2\lfloor_{\mathsf{a}} \setminus \mathcal{C}_1\lfloor_{\mathsf{a}}) \cup \mathcal{B}_1\lfloor_{\mathsf{a}} = R^\dagger\lfloor_{\mathsf{a}}$.

It follows that $R'_{\mathsf{a}} = R'\lfloor_{\mathsf{a}}$. $\square$

**Lemma 7.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $R \xrightarrow{\ell} R'$ then there exist $\vec{R}^\ddagger$ and $\vec{b}^\ddagger$ such that $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle$ and $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$.*

*Proof.* By definition $\langle \vec{R}^\dagger, \vec{b}\rangle \sim \langle \vec{R}, \vec{b}\rangle = \mathit{cd}(R)$. Let $\mathsf{a} = \mathit{subj}(\ell)$. By Lemma 5 $R\lfloor_{\mathsf{a}} \xrightarrow{\ell} R'\lfloor_{\mathsf{a}}$.

- If $\ell = \mathsf{ab!x}$ for some $\mathsf{b}, \mathsf{x}$ then $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}[R'\lfloor_{\mathsf{a}}/R\lfloor_{\mathsf{a}}], \mathit{add}(\mathsf{ab!x}, \vec{b})\rangle$.
- If $\ell = \mathsf{ba?x}$ for some $\mathsf{b}, \mathsf{x}$ then let $e \in R.E$ be the event such that $R \xrightarrow{e} R'$. Since $R$ is tree-like, by Lemma 13 $e$ is minimal in $R$. Since $R$ is well-branched, $e \in R.\mathcal{B}$. Since $R$ is well-channeled, all other same-channeled receive events $e'$ must be either causally ordered, in which case $e < e'$ since $e$ is minimal, or mutually exclusive, which is contradictory since $e \in R.\mathcal{B}$. It then follows from Definition 11 that $\mathit{has}(\mathsf{ba!x}, \vec{b})$. Then $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}[R'\lfloor_{\mathsf{a}}/R\lfloor_{\mathsf{a}}], \mathit{remove}(\mathsf{ba!x}, \vec{b})\rangle$.

By Lemma 4 $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}, \vec{b}\rangle' \sim \mathit{cd}(R')$. Since $\langle \vec{R}^\dagger, \vec{b}\rangle \sim \langle \vec{R}, \vec{b}\rangle$ it follows that $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle \sim \mathit{cd}(R')$ for some $\vec{R}^\ddagger, \vec{b}^\ddagger$, and then $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$. $\square$

**Lemma 8.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle$ then there exists $R'$ such that $R \xrightarrow{\ell} R'$ and $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$.*

*Proof.* By definition $\langle \vec{R}^\dagger, \vec{b}\rangle \sim \langle \vec{R}, \vec{b}\rangle = \mathit{cd}(R)$. Let $\mathsf{a} = \mathit{subj}(\ell)$. Then $R\lfloor_{\mathsf{a}} \xrightarrow{\ell} R'_{\mathsf{a}}$ for some $R'_{\mathsf{a}}$. By Lemma 6 $R \xrightarrow{\ell} R'$ for some $R'$ such that $R'_{\mathsf{a}} = R'\lfloor_{\mathsf{a}}$. Then by Lemma 4 $\langle \vec{R}, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}, \vec{b}\rangle' \sim \mathit{cd}(R')$. Since $\langle \vec{R}^\dagger, \vec{b}\rangle \sim \langle \vec{R}, \vec{b}\rangle$ it follows that $\langle \vec{R}^\dagger, \vec{b}\rangle \xrightarrow{\ell} \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle \sim \mathit{cd}(R')$ for some $\vec{R}^\ddagger, \vec{b}^\ddagger$, and then $\langle R', \langle \vec{R}^\ddagger, \vec{b}^\ddagger\rangle\rangle \in \mathcal{R}$. $\square$

**Lemma 9.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b}\rangle\rangle \in \mathcal{R}$. If $R$ is well-formed and $R\downarrow$ then $\langle \vec{R}^\dagger, \vec{b}\rangle\downarrow$.*

*Proof.* Since $R\downarrow$, it follows that $R.\mathcal{B} \sqsupseteq \emptyset$ and then by Lemma 1 $R\lfloor_{\mathsf{a}}.\mathcal{B} \sqsupseteq \emptyset$ and $R\lfloor_{\mathsf{a}}\downarrow$ for every $\mathsf{a}$. It follows that $\mathit{cd}(R)\downarrow$. Since $\langle \vec{R}^\dagger, \vec{b}\rangle \sim \mathit{cd}(R)$, it then follows that $\langle \vec{R}^\dagger, \vec{b}\rangle\downarrow$. $\square$

**Lemma 10.** *Let $\langle R, \langle \vec{R}^\dagger, \vec{b} \rangle \rangle \in \mathcal{R}$. If $R$ is well-formed and $\langle \vec{R}^\dagger, \vec{b} \rangle \downarrow$ then $R\downarrow$.*

*Proof.* Since $\langle \vec{R}^\dagger, \vec{b} \rangle \downarrow$, it follows that $R\lfloor_{\mathsf{a}}\downarrow$ and $R\lfloor_{\mathsf{a}}.\mathcal{B} \sqsupseteq \emptyset$ for every $\mathsf{a}$. Suppose, for the sake of contradiction, that $R.\mathcal{B} \not\sqsupseteq \emptyset$. Then, since $R$ is well-branched, there must exist some $\mathsf{a}$ such that $\mathsf{a}$ occurs in every refinement of $R$. However, this contradicts Lemma 2, which states that there must exist some $R'$ such that $R \sqsupseteq R'$ and $R'\lfloor_{\mathsf{a}} = \emptyset$. Therefore, $R.\mathcal{B} \sqsupseteq \emptyset$ and then $R\downarrow$. ☐

## B Additional lemmas

**Lemma 11.** *Let $R$ be a tree-like and choreographic branching pomset. If $R \xrightarrow{\ell} R'$ then $R'$ is choreographic.*

*Proof.* Let $e \in R'.E$.

- Suppose there exists some event $e' \in R'.E$ such that $e' < e$.
  - If $\lambda(e) = \mathsf{ab!x}$ then since $e$ and $e'$ are also in $R$ and $R$ is choreographic there exists some $e''$ in $R$ such that $e' \leq e'' < e$ and $subj(\lambda(e'')) = \mathsf{a}$. Suppose, for the sake of contradiction, that $e''$ is not in $R'$. Then either $R \xrightarrow{e''} R'$ or $e''$ is discarded as part of the resolving of some choice. The former is not possible since either $e' = e''$, in which case $e' \notin R'.E$, or $e' < e''$, in which case $e''$ is not minimal. The latter is not possible since we assume that $R$ is tree-like: if $e''$ is discarded as part of a non-chosen branch then $e$ belongs to the same branch and should also have been discarded, yet $e$ is in $R'$. We conclude that $e''$ is also in $R'$ and then there exists some $e''$ in $R'$ such that $e' \leq e'' < e$ and $subj(\lambda(e'')) = \mathsf{a}$.
  - If $\lambda(e) = \mathsf{ab?x}$ then we proceed analogously.
- Suppose $\lambda(e) = \mathsf{ab?x}$ and $e \in \mathcal{B}$ for some $\mathcal{B} \prec R'.\mathcal{B}$. Since then also $e \in R.E$, we consider where $e$ is in $R.\mathcal{B}$. If $e \in R.\mathcal{B}$ then also $e \in R'.\mathcal{B}$, which is contradictory since $e \in \mathcal{B} \prec R'.\mathcal{B}$. Otherwise, there exists some $\mathcal{B}^\dagger \prec R.\mathcal{B}$ such that $e \in \mathcal{B}^\dagger$. Since $R$ is choreographic, there then exists some $e'$ such that $e' \in \mathcal{B}^\dagger$ and $\lambda(e') = \mathsf{ab!x}$ and $e' < e$. If $e' \not\preceq \mathcal{B}$ then $R \xrightarrow{e'} R'$ and then $e \in R'.\mathcal{B}$. If $e' \in R'.\mathcal{B}$ then, again, $e \in R'.\mathcal{B}$. Finally, otherwise there exists some $\mathcal{B}' \prec R'.\mathcal{B}$ such that $e' \in \mathcal{B}'$ and then $e \in \mathcal{B}'$. We then still have $e' < e$, which concludes the second choreographicness criterion.
- Suppose $\lambda(e) = \mathsf{ab!x}$. Since $e$ is also an event in $R$ and $R$ is choreographic, there exists some event $e'$ in $R$ such that $e \leq e'$ and $\lambda(e') = \mathsf{ab?x}$ and for all events $e^\dagger$ in $R$: if $\lambda(e^\dagger) = \mathsf{ab!x}$ and $e^\dagger \leq e'$ then $e^\dagger \leq e$; if $\lambda(e^\dagger) = \mathsf{ab?x}$ and $e^\dagger \neq e'$ and $e \leq e^\dagger$ then there exists some $e^\ddagger \neq e$ such that $\lambda(e^\ddagger) = \mathsf{ab!x}$ and $e^\ddagger \leq e^\dagger$. Suppose, for the sake of contradiction, that $e'$ is not in $R'$. Then either $R \xrightarrow{e'} R'$ or $e'$ was part of a discarded branch. The former is not possible since $e \leq e'$ and $e$ is in $R'$. The latter is ultimately also not possible since $e$ is in $R'$.

  Recall the second choreographicness criterion: $e'$ is in some branch in $R$, i.e., $e' \in \mathcal{B}$ for some $\mathcal{B} \prec R.\mathcal{B}$, from which it follows that there exists some $e''$ such that $e'' \in \mathcal{B}$ and $\lambda(e'') = \mathsf{ab!x}$ and $e'' < e'$. Since then $e'' \leq e$ and $R$

is tree-like, $e \preceq \mathcal{B}$. In fact, since $e'' \in \mathcal{B}$, it follows from $R$ being tree-like that $e \in \mathcal{B}$. Then, as for the second choreographicness criterion, as $e$ is in $R$ so must $e'$. Finally, since $R'.{\leq} \subseteq R.{\leq}$, the remaining necessary results remain for all $e^\dagger$ in $R'$. We can thus conclude that $R'$ satisfies the third choreographicness criterion.

We conclude that $R'$ is choreographic. ◻

**Lemma 12.** *Let $R$ be a branching pomset. Let $\mathsf{a}$ and $\mathsf{b}$ be agents in $R$. If $R$ is well-channeled and choreographic then $buff_{\mathsf{ab}}(R)$ is uniquely defined.*

*Proof.* Suppose, for the sake of contradiction, that there exist two distinct events $e_1, e_2 \in R.E$ that match the criteria in Definition 11. Without loss of generality, let $\lambda(e_1) = \mathsf{ab?x}$ and $\lambda(e_2) = \mathsf{ab?y}$. Since $R$ is choreographic, it follows that $e_1, e_2 \in R.\mathcal{B}$. Since $R$ is well-channeled and $e_1, e_2 \in R.\mathcal{B}$ are two receive events on the same channel, it then follows that either $e_1 \leq e_2$ or $e_2 \leq e_1$. Since there exists no $e'$ such that $e' < e_1$ or $e' < e_2$, it must be the case that $e_1 = e_2$.

**Lemma 13.** *If $R$ is tree-like and $R \xrightarrow{\checkmark_e}$ then $e$ is minimal in $R$.*

*Proof.* Suppose, for the sake of contradiction, that there exists some $e' \in R.E$ such that $e' < e$. Since $R \xrightarrow{\checkmark_e}$, there then exists some refinement of $R$ which contains $e$ and not $e'$, i.e., there exists some $\mathcal{B} \preceq R.\mathcal{B}$ such that $e' \preceq \mathcal{B}$ and $e \not\preceq \mathcal{B}$. However, then $R$ is not tree-like. Therefore, $e$ must be minimal in $R$.

**Lemma 14.** *Let $R$ be well-branched. If $R{\downarrow}$ then $R.E = \emptyset$.*

*Proof.* Since $R{\downarrow}$ it follows that $R.\mathcal{B} \sqsupseteq \emptyset$. The proof is by induction on the structure of the latter's derivation tree. Consider its root, i.e., the final step in the derivation.

- **Base:** Refl. Then $R.\mathcal{B} = \emptyset$. It follows that $R.E = \emptyset$.
- **Base:** Choice. Then $R.\mathcal{B} = \{\{\mathcal{B}_1, \mathcal{B}_2\}\} \cup \mathcal{B}^\dagger$ such that $\mathcal{B}^\dagger = \emptyset$ and, without loss of generality, $\mathcal{B}_1 = \emptyset$. Since $R$ is well-branched, $\mathcal{B}_1$ cannot be empty and this case cannot be the final step.
- **Step:** Trans. Then $R.\mathcal{B} \sqsupseteq \mathcal{B}^\dagger \sqsupseteq \emptyset$. It follows from the induction hypothesis that $\mathcal{B}^\dagger = \emptyset$ and then that $R.\mathcal{B} = \emptyset$. Then $R.E = \emptyset$.
- **Step:** Congr. This cannot be the final step as it cannot yield $\emptyset$.

**Lemma 15.** *Let $R$ be well-branched. Let $R_{\mathsf{a}} = R{\restriction_{\mathsf{a}}}$. If $R_{\mathsf{a}}{\downarrow}$ then $R_{\mathsf{a}}.E = \emptyset$.*

*Proof.* Since $R_{\mathsf{a}}{\downarrow}$ it follows that $R_{\mathsf{a}}.\mathcal{B} \sqsupseteq \emptyset$. The proof is by induction on the structure of the latter's derivation tree. Consider its root, i.e., the final step in the derivation.

- **Base:** Refl. Then $R_{\mathsf{a}}.\mathcal{B} = \emptyset$ and $R_{\mathsf{a}}.E = \emptyset$.
- **Base:** Choice. Then $R_{\mathsf{a}}.\mathcal{B} = \{\{\mathcal{B}_1, \mathcal{B}_2\}\} \cup \mathcal{B}^\dagger$ such that $\mathcal{B}^\dagger = \emptyset$ and, without loss of generality, $\mathcal{B}_1 = \emptyset$. Since $R_{\mathsf{a}}$ is well-branched, it follows that $\mathcal{B}_2 = \emptyset$ and then $R_{\mathsf{a}}.E = \emptyset$.
- **Step:** Trans. Then $R_{\mathsf{a}}.\mathcal{B} \sqsupseteq \mathcal{B}^\dagger \sqsupseteq \emptyset$. It follows from the induction hypothesis that $\mathcal{B}^\dagger = \emptyset$ and then that $R_{\mathsf{a}}.\mathcal{B} = \emptyset$. Then $R_{\mathsf{a}}.E = \emptyset$.
- **Step:** Congr. This cannot be the last step as it cannot yield $\emptyset$.