Asynchronous Message Orderings Beyond Causality

Adam Shimi, Aurélie Hurault, Philippe Quéinnec

Asynchronous Message Orderings Beyond Causality

Adam Shimi[1], Aurélie Hurault[2], and Philippe Quéinnec[3]

1 **IRIT - Université de Toulouse, 2 rue Camichel, F-31000 Toulouse, France**
2 **IRIT - Université de Toulouse, 2 rue Camichel, F-31000 Toulouse, France**
3 **IRIT - Université de Toulouse, 2 rue Camichel, F-31000 Toulouse, France**

Abstract

In the asynchronous setting, distributed behavior is traditionally studied through computations, the Happened-Before posets of events generated by the system. An equivalent perspective considers the linear extensions of the generated computations: each linear extension defines a sequence of events, called an execution. Both perspective were leveraged in the study of asynchronous point-to-point message orderings over computations; yet neither allows us to interpret message orderings defined over executions. Can we nevertheless make sense of such an ordering, maybe even use it to understand asynchronicity better?

We provide a general answer by defining a topology on the set of executions which captures the fundamental assumptions of asynchronicity. This topology links each message ordering over executions with two sets of computations: its closure, the computations for which at least one linear extension satisfies the predicate; and its interior, the computations for which all linear extensions satisfy it. These sets of computations represent respectively the uncertainty brought by asynchronicity – the computations where the predicate is satisfiable – and the certainty available despite asynchronicity – the computations where the predicate must hold. The paper demonstrates the use of this topological approach by examining closures and interiors of interesting orderings over executions.

1998 ACM Subject Classification C.2.4 Distributed Systems

Keywords and phrases Asynchronous computations, Point-to-point message orderings, Causality, Topology, Interior, Closure

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2017.29

1 Introduction

1.1 Motivation

What can we know about the ordering of events in an asynchronous world? Only the causal order, answered Lamport's seminal work [20]. This insight grounds the description of asynchronous behavior through computations, the posets generated by events with their causal order – or equivalently through the linear extensions of these computations, sequences of events called executions.

It follows that message orderings defined over computations have been well studied, because they capture the type of constraints allowed by causality. One prominent example is the causal message ordering: it enforces that two messages whose send events are causally ordered and which share the same destination peer, must be received in the order they were

[1] This work was supported by project PARDI ANR-16-CE25-0006.

sent. Charron-Bost et al. [8] offer multiple equivalent characterizations of this ordering, as well as its place in a hierarchy with other orderings. Murty and Garg [22] prove that causal ordering is the most constrained ordering implementable using only messages tags – that is, without altering the causal order through control messages. Numerous works, among others Fidge [14], Mattern [21], Schwarz and Mattern [25], Raynal et al.[23] and Kshemkalyani and Singhal [19], explore the means of implementation of causal ordering.

But predicates over computations have drawbacks: they are defined on posets, objects less intuitive than sequences; and they do not capture every interesting constraint. For example, Fifo_{n-n} ordering formalized in Chevrou et al. [11] requires that all receptions follow the same order as their respective sends, even if neither pair of events is causally ordered. Fifo_{n-n}'s definition is simple, and it reduces asynchronous communication to a single FIFO queue. But since it demands a stronger order than the causal one, it conflicts with the very foundations of asynchronous distributed systems. As we will see, there is a general way to consider any predicate over executions in an asynchronous setting, which gives us both a powerful tool and a deeper understanding of asynchronicity.

1.2 Computations and Executions

▶ **Definition 1** (Events)**.** Let MES the set of messages and PEERS the set of communicating peers. Then EVENTS $\triangleq \{send, receive\} \times MES \times PEERS$, where the components correspond respectively to the type of events, the message and the peer where the event happens.

We write $peer(e)$ for the projection of an event into its peer component. Since in the following we will only consider sets of events where multiple sends and/or receptions of the same message are forbidden, events are considered uniquely characterized by their type and their message. We thus abuse notation by writing $s(m)$ and $r(m)$ instead of $\langle send, m, peer\rangle$ and $\langle receive, m, peer\rangle$ respectively. Note that we do not take into account internal events, since we are only interested in the order of communication ones.

▶ **Definition 2** (Computation)**.** Let X be a set of events containing at most one reception by message and one send event by message. Then the partially ordered set (or poset) $x = (X, \prec_c^x)$ is a **computation** iff

- $\prec_c^x$ is a causal order on X, that is the smallest partial order such that

$$\left(\begin{array}{lcll} peer(e_1) = peer(e_2) & \implies & e_1 \prec_c^x e_2 \vee e_2 \prec_c^x e_1 & \text{(Peer order)} \\ \exists m : e_1 = s(m) \wedge e_2 = r(m) & \implies & e_1 \prec_c^x e_2 & \text{(Message transfer)} \\ \exists e \in X : e_1 \prec_c^x e \wedge e \prec_c^x e_2 & \implies & e_1 \prec_c^x e_2 & \text{(Transitivity)} \end{array}\right).$$

- Any received message has been sent: $\forall m \in MES : r(m) \in x \implies s(m) \in x$.
- No message is sent to oneself: $\forall m \in MES : peer(r(m)) \neq peer(s(m))$.

The peer order $\prec_p^x$ is defined as the projection of $\prec_c^x$ on pairs of events happening on the same peer. When two events e_1 and e_2 are not causally ordered, we write $e_1 \parallel_c^x e_2$. We note $Comp$ the set of all computations over subsets of $EVENTS$. Finally, for b a predicate over computations, $Comp(b)$ is the set of computations satisfying b.

▶ **Definition 3** (Execution)**.** Let Σ be a set of events containing at most one reception by message and one send event by message. Then the totally ordered set $\sigma = (\Sigma, \prec^\sigma)$ is an **execution** iff $\exists x = (X, \prec_c^x) \in Comp$ such that $\Sigma = X$ and $\prec^\sigma$ is a linear extension of $\prec_c^x$. By the minimality of the causal order, such x is unique.

We write $comp(\sigma)$ for x, $\prec_c^\sigma$ for $\prec_c^x$ and $\prec_p^\sigma$ for $\prec_p^x$. We note $Exec$ the set of all executions over subset of $EVENTS$. Finally, for b a predicate over executions, $Exec(b)$ is the set of executions satisfying b.

Both are represented with so-called space-time diagrams, where each peer is represented by a vertical line, events at a peer are ordered from left to right and messages are drawn by connecting a send event with the corresponding reception. For example, Figure 3 defines the execution $s(m_1)r(m_1)s(m_2)r(m_2)$ as well as the computation over the same events and with partial order $\{(s(m_1), r(m_1)), (s(m_2), r(m_2)), (s(m_1), s(m_2)), ((s(m_1), r(m_2)))\}$.

1.3 Message Orderings

We conclude these preliminary definitions by introducing predicates over computations and executions. Since we consider only communication events, we will call these predicates message orderings. Table 1 states the message orderings we will study in the following. They are given as in Chevrou et al. [11] (except for RSC, where our definition is equivalent and easier to manipulate).

- Fifo_{1-1} ordering is the classical Fifo ordering between every pair of peers.
- Causal ordering is the ordering where messages to the same peer and with causally ordered send events are received according to their send order.
- Fifo_{n-1} ordering is the "mailbox" ordering that is used notably by [2], where messages to a peer are put into its mailbox, and the peer retrieves them according to their send order.
- Fifo_{1-n} ordering is the dual of Fifo_{n-1}. It can be thought of as a "sending box" by peer, in which each peer put messages it sends, and from which receivers fetch messages according to their send order.
- Fifo_{n-n} ordering is the ordering where all messages are received according to their send order, even if when neither their send events nor their receptions are causally ordered.
- RSC ordering (Realizable with Synchronous Communication) requires that every reception is immediately preceded by its corresponding send event.

The sets of executions defined by these models form a hierarchy, as stated and proved in [11].

▶ **Lemma 4** (Ordering hierarchy over executions).

1. $Exec(\text{RSC}) \subsetneq Exec(\text{Fifo}_{n-n}) \subsetneq Exec(\text{Fifo}_{1-n}) \subsetneq Exec(\text{Causal}) \subsetneq Exec(\text{Fifo}_{1-1})$
2. $Exec(\text{RSC}) \subsetneq Exec(\text{Fifo}_{n-n}) \subsetneq Exec(\text{Fifo}_{n-1}) \subsetneq Exec(\text{Causal}) \subsetneq Exec(\text{Fifo}_{1-1})$
3. $Exec(\text{Fifo}_{n-1}) \not\subset Exec(\text{Fifo}_{1-n})$ *and* $Exec(\text{Fifo}_{1-n}) \not\subset Exec(\text{Fifo}_{n-1})$

Turning to computations, only Fifo_{1-1} and Causal are well-defined over them, since they are defined only in terms of causal and peer order. Those two orderings obey the same hierarchy as their execution-based counterparts, as shown in Charron-Bost et al. [8].

▶ **Lemma 5.** $Comp(\text{Causal}) \subsetneq Comp(\text{Fifo}_{1-1})$

1.4 Overview of the Results

Our results are threefold.

- First, we leverage elementary topology to link any message ordering over executions with two sets of computations: its closure, corresponding to existential quantification over linear extensions of the ordering; and its interior, corresponding to universal quantification. Both sets provide a mean to study the message ordering in an asynchronous context, where computations are the least distinguishable unit.

Table 1 Ordering predicates over executions.

Name	Expression
Fifo_{1-1}	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1), r(m_2) \in \sigma \\ \wedge & peer(r(m_1)) = peer(r(m_2)) \\ \wedge & s(m_1) \prec_p^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec_p^\sigma r(m_2)$
Causal	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1), r(m_2) \in \sigma \\ \wedge & peer(r(m_1)) = peer(r(m_2)) \\ \wedge & s(m_1) \prec_c^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec_p^\sigma r(m_2)$
Fifo_{n-1}	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1), r(m_2) \in \sigma \\ \wedge & peer(r(m_1)) = peer(r(m_2)) \\ \wedge & s(m_1) \prec^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec_p^\sigma r(m_2)$
Fifo_{1-n}	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1), r(m_2) \in \sigma \\ \wedge & s(m_1) \prec_p^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec^\sigma r(m_2)$
Fifo_{n-n}	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1), r(m_2) \in \sigma \\ \wedge & s(m_1) \prec^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec^\sigma r(m_2)$
RSC	$\forall m_1, m_2 \in MES : \left(\begin{array}{ll} & r(m_1) \in \sigma \\ \wedge & s(m_1) \prec^\sigma s(m_2) \end{array} \right) \implies r(m_1) \prec^\sigma s(m_2)$

- Second, we characterize the closures of our message orderings (see Figure 1) by forbidden patterns in the causal order. This in turn yields a precise understanding of whether they can be distinguished at the computation level, as well as their comparative discriminating power.
- Third, we turn to interiors of the aforementioned orderings, and characterize them through the concept of chain (see Figure 2). Such characterizations expand our understanding of the orderings implementation: they provide system-level guidelines that allows us to circumvent the result of [22] for ensuring a message ordering.

The rest of the paper is organized as follows. Section 2 develops the topological approach to message orderings over executions. Then the closures and interiors of the message orderings mentioned above are studied respectively in Sections 3 and 4. Finally, Section 5 surveys related works while conclusions and perspectives are drawn in Section 6.

2 A Topological Bridge

Recall that a message ordering over executions characterizes a set of executions. On the other hand, we know from [20] that the only meaningful sets of executions in an asynchronous world correspond to the sets of all the linear extensions of a given computation, or the union of such sets. The subtlety here stems from the "all": a truncated set of linear extensions asks for an order with more discriminating power than the causal one. Making sense of any set of executions in terms of computations – and by extension in terms of causal order – will allow us to interpret any message ordering in the asynchronous setting.

First, we need a formal definition of what we call meaningful sets of executions, that is the sets defined by all the linear extensions of a given computation. They can be characterized as the equivalence classes from the causal equivalence relation.

▶ **Definition 6** (Causal Equivalence). Let $\sigma, \sigma' \in Exec$. Then $\sigma \equiv_c \sigma' \triangleq comp(\sigma) = comp(\sigma')$. We say σ and σ' are causally equivalent and write $[\sigma]_{\equiv_c}$ for the equivalence class of σ.

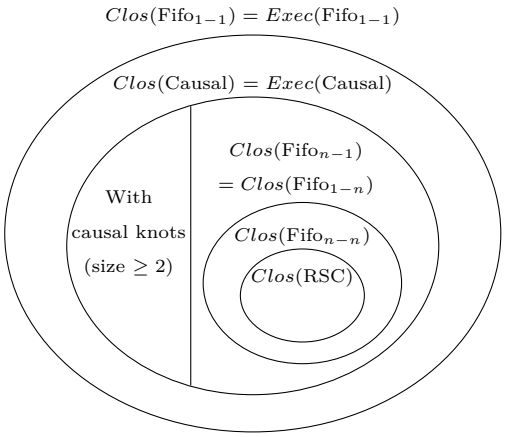

Figure 1 The closures of our message orderings.

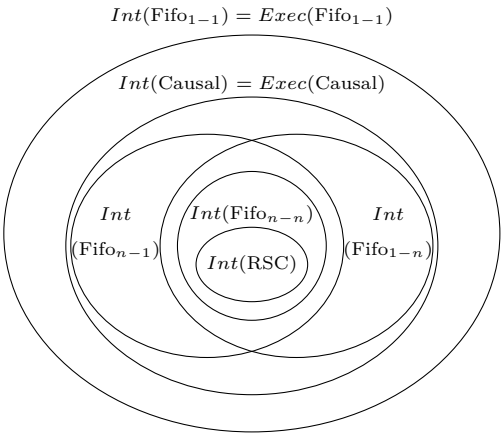

Figure 2 The interiors of our message orderings.

The quotient set of $Exec$ by causal equivalence is thus isomorphic with $Comp$. The causal equivalence classes are only the building block: any union of such blocks is also a meaningful set for our endeavors. These properties correspond to the topological concept of an open set, and motivate our introduction of the following simple topology over $Exec$.

▶ **Definition 7** (Open Set). Let $S \subseteq Exec$. Then S is an open set in the **Computation Topology** iff $\exists X \in \mathcal{P}(Exec)$ such that $S = \bigcup_{x \in X} [x]_{\equiv_c}$, where $\mathcal{P}(Exec)$ is the powerset of $Exec$.

A property of our topology that will prove useful is that any open set is also a closed one. Indeed, by definition of an equivalence relation, any union of equivalence classes is the complement of a union of equivalence classes; thus every open set is also closed. This might strike the reader used to more classical topologies as odd. Yet it allows us to state in terms of open sets both the closure and interior of a set, the two basic operations in elementary topology.

These extensions can be interpreted respectively as existential and universal quantification over equivalence classes: the set of equivalence classes (or equivalently, of computations) containing at least one execution from the original set; and the set of equivalence classes containing only executions from the original set. Intuitively, this translates respectively into the set of computations where a message ordering might hold, and the set of computations where it necessarily holds.

▶ **Definition 8** (Interior and Closure). Let $S \subseteq Exec$.

$Clos(S)$, the closure of S, is the smallest closed set such that $S \subseteq Clos(S)$.

$Int(S)$, the interior of S, is the greatest open set such that $Int(S) \subseteq S$.

We write $Clos(b)$ and $Int(b)$ for the closure and interior of the set of executions defined by the predicate b.

▶ **Lemma 9** (Collapse Lemma). *Let $S \subseteq Exec$. Then $S = Clos(S) \iff S = Int(S)$.*

Proof. ($\Rightarrow$) Since $S = Clos(S)$, S is a closed set. Given our topology, it is thus also an open set. It is therefore the greatest open set contained by itself.

($\Leftarrow$) Since $S = Int(S)$, S is an open set. Given our topology, it is thus also a closed set. It is therefore the smallest closed set containing itself. ◀

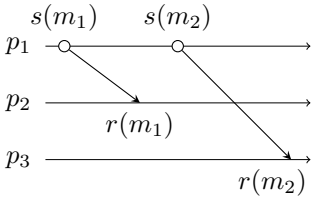

Figure 3 Fifo_{1-n} does not collapse.

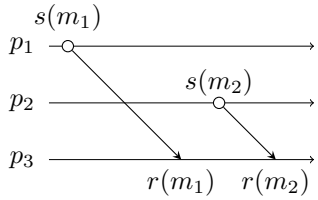

Figure 4 Fifo_{n-1} does not collapse.

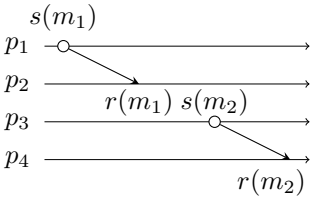

Figure 5 Fifo_{n-n} and RSC do not collapse.

We now show that the collapse described by Lemma 9 happens for Fifo_{1-1} and Causal but not the other orderings. Our topology thus captures the distinction between message orderings on executions which translate straightforwardly over computations, and message orderings on executions "asking" for more than causal order.

▶ **Theorem 10** (Fifo_{1-1} and Causal).
1. $Exec(\text{Fifo}_{1-1}) = Int(\text{Fifo}_{1-1}) = Clos(\text{Fifo}_{1-1})$
2. $Exec(\text{Causal}) = Int(\text{Causal}) = Clos(\text{Causal})$

Proof.
1. Fifo_{1-1} is defined only in terms of peer order, which is a projection of the causal order. It is therefore invariant by causal equivalence. We conclude that $Exec(\text{Fifo}_{1-1}) = Clos(\text{Fifo}_{1-1})$, which by Lemma 9 entails $Exec(\text{Fifo}_{1-1}) = Int(\text{Fifo}_{1-1}) = Clos(\text{Fifo}_{1-1})$.
2. Causal is invariant by causal equivalence for the same reason as Fifo_{1-1}. We conclude that $Exec(\text{Causal}) = Clos(\text{Causal})$, which by Lemma 9 gives us $Exec(\text{Causal}) = Int(\text{Causal}) = Clos(\text{Causal})$. ◀

▶ **Theorem 11** (Fifo_{n-1}, Fifo_{1-n}, Fifo_{n-n} and RSC).
1. $Int(\text{Fifo}_{1-n}) \subsetneq Exec(\text{Fifo}_{1-n}) \subsetneq Clos(\text{Fifo}_{1-n})$
2. $Int(\text{Fifo}_{n-1}) \subsetneq Exec(\text{Fifo}_{n-1}) \subsetneq Clos(\text{Fifo}_{n-1})$
3. $Int(\text{Fifo}_{n-n}) \subsetneq Exec(\text{Fifo}_{n-n}) \subsetneq Clos(\text{Fifo}_{n-n})$
4. $Int(\text{RSC}) \subsetneq Exec(\text{RSC}) \subsetneq Clos(\text{RSC})$

Proof. We only consider the cases where $EVENTS$ is non trivial, which here means that it contains at least two send events and two receptions.

The inclusions then follow directly from the definition of Int and $Clos$. As for strictness, counter-examples separating either the interior or the closure with the initial sets are enough: Lemma 9 then yields the strictness of inclusion for both.
1. Figure 3 is in $Exec(\text{Fifo}_{1-n})$ but not in its interior: the execution $s(m_1)s(m_2)r(m_2)r(m_1)$ is causally equivalent to it while breaking the Fifo_{1-n} predicate. Thus $Exec(\text{Fifo}_{1-n}) \neq Int(\text{Fifo}_{1-n})$.
2. Figure 4 is in $Exec(\text{Fifo}_{n-1})$ but not in its interior: the execution $s(m_2)s(m_1)r(m_1)r(m_2)$ is causally equivalent to it while breaking the Fifo_{n-1} predicate. Thus $Exec(\text{Fifo}_{n-1}) \neq Int(\text{Fifo}_{n-1})$.
3. Figure 5 is in $Exec(\text{Fifo}_{1-n})$ but not in its interior: the execution $s(m_1)s(m_2)r(m_2)r(m_1)$ is causally equivalent to it, while breaking the Fifo_{n-n} predicate. Thus $Exec(\text{Fifo}_{n-n}) \neq Int(\text{Fifo}_{n-n})$.
4. Figure 5 is also in $Exec(\text{RSC})$ and not in its interior: the execution exhibited in the previous case breaks RSC too. Thus $Exec(\text{RSC}) \neq Int((\text{RSC}))$. ◀

3 Closures

3.1 Characterization of the Closure of RSC

When we apply the closure operation on our RSC over executions, we obtain the set of computations with at least one RSC linear extension. This in turn corresponds to the RSC over computations from Charron-Bost et al. [8]. We thus reuse the characterization of $Clos$(RSC) by Charron-Bost et al., through forbidden patterns in the causal order named crowns.

▶ **Definition 12** (Crown). Let σ an execution. Then a set of messages $m_1, .., m_n$ forms a **crown** of size $n \iff \forall i \in [1, n] : s(m_i) \prec_c^\sigma r(m_{i+1})$, where $m_{n+1} = m_1$.

For instance, Figure 6 is a crown of size 2.

▶ **Theorem 13** (Crown Characterization). *Let $\sigma \in Exec$. Then $\sigma \in Clos$(RSC) $\iff$ σ contains no crown.*

Proof. See [8]. ◀

3.2 Characterization of $Clos$(Fifo$_{n-1}$) and $Clos$(Fifo$_{1-n}$)

A similar negative characterization of $Clos$(Fifo$_{n-1}$) and $Clos$(Fifo$_{1-n}$) is given by special crowns we call causal knots.

▶ **Definition 14** (Causal knot). Let σ an execution. Then a set of messages $m_1, .., m_{2s}$ forms a **causal knot** of size $s \iff \forall i \in [1, 2s] : \left(\begin{array}{l} i \equiv 1 \bmod 2 : s(m_i) \prec_c^\sigma s(m_{i+1}) \\ i \equiv 0 \bmod 2 : r(m_i) \prec_c^\sigma r(m_{i+1}) \end{array} \right)$,
where $m_{2s+1} = m_1$.

Figure 7 is an example of a causal knot of size 2. We show in the rest of this section that causal knots characterize $Clos$(Fifo$_{n-1}$) and $Clos$(Fifo$_{1-n}$). Let us start with $Clos$(Fifo$_{n-1}$): we first define a relation for each causal equivalence class characterizing the additional constraints on executions of this class to satisfy Fifo$_{n-1}$.

▶ **Definition 15** (N-1 order). Let $\sigma \in Exec$ and Σ the underlying set of events. Then $\lhd_{n-1}^\sigma \triangleq \{(s(m_1), s(m_2)) \in \Sigma \times \Sigma \mid r(m_1) \prec_p^\sigma r(m_2)\}$. And $\prec_{n-1}^\sigma \triangleq \prec_c^\sigma \cup \lhd_{n-1}^\sigma$.

Then the next Lemma reduces the membership of an execution σ in $Clos$(Fifo$_{n-1}$) to whether or not $\prec_{n-1}^\sigma$ contains a cycle.

▶ **Lemma 16.** *Let $\sigma \in Exec$. Then $\sigma \in Clos(Fifo_{n-1}) \iff \prec_{n-1}^\sigma$ is antisymmetric.*

Proof. ($\Rightarrow$) We show the contrapositive: If $\prec_{n-1}^\sigma$ contains a cycle, then $\sigma \notin Clos$(Fifo$_{n-1}$).

Let $\sigma \in Exec$ such that $\prec_{n-1}^\sigma$ contains a cycle. Then its transitive closure is not a partial order: it has no linear extensions. From that fact, we now prove that no execution in $[\sigma]_{\equiv_c}$ satisfies Fifo$_{n-1}$, and thus that $\sigma \notin Clos$(Fifo$_{n-1}$).

Let $\sigma' \equiv_c \sigma$. By our hypothesis that $\prec_{n-1}^\sigma$ contains a cycle, we have $\prec_{n-1}^\sigma \not\subset \prec^{\sigma'}$. But since $\sigma' \equiv_c \sigma$, we have $\prec_c^\sigma \subset \prec^{\sigma'}$. We thus conclude that $\lhd_{n-1}^\sigma \not\subset \prec^{\sigma'}$. Thus $\exists m_1, m_2$ such that $r(m_1) \prec_p^\sigma r(m_2) \wedge s(m_1) \not\prec^{\sigma'} s(m_2)$. By totality of $\prec^{\sigma'}$, we have $r(m_1) \prec_p^\sigma r(m_2) \wedge s(m_2) \prec^{\sigma'} s(m_1)$, and thus σ' violates Fifo$_{n-1}$.

($\Leftarrow$) Let $\sigma \in Exec$ such that $\prec_{n-1}^\sigma$ is antisymmetric. Then the transitive closure of $\prec_{n-1}^\sigma$ is reflexive (because $\prec_c^\sigma$ is reflexive), transitive and antisymmetric. It is therefore a partial order, who has a linear extension agreeing both with $\prec_c^\sigma$ and $\lhd_{n-1}^\sigma$. This linear extension defines an execution $\sigma' \equiv_c \sigma$ (because $\prec_c^\sigma \subset \prec^{\sigma'}$) such that σ' satisfies Fifo$_{n-1}$ (because $\lhd_{n-1}^\sigma \subset \prec^{\sigma'}$). We conclude that $\sigma \in Clos$(Fifo$_{n-1}$). ◀

▶ **Theorem 17** (Causal Knot Criterion for Fifo_{n-1}). *Let $\sigma \in Exec$. Then $\sigma \in Clos(\text{Fifo}_{n-1})$ $\iff$ σ contains no causal knot.*

Proof. By Lemma 16 and the transitivity of equivalence, we only need to prove that $\prec^\sigma_{n-1}$ is antisymmetric $\iff$ σ contains no causal knot.

($\Rightarrow$) We prove the contrapositive: if σ contains a causal knot, then $\prec^\sigma_{n-1}$ is not antisymmetric. Let $\sigma \in Exec$ with a causal knot of size k. By definition of causal knots, there are $2k$ messages $m_1, .., m_{2k}$ such that $\iff \forall i \in [1, 2k] : \left(\begin{array}{l} i \equiv 1 \bmod 2 : s(m_i) \prec^\sigma_c s(m_{i+1}) \\ i \equiv 0 \bmod 2 : r(m_i) \prec^\sigma_c r(m_{i+1}) \end{array} \right)$, where $m_{2k+1} = m_1$. We define another sequence of messages $m'_1, .., m'_{2k}$ from the messages of the causal knot: $\forall i \in [1, 2k]$:

$$\left(\begin{array}{l} i \equiv 1 \bmod 2 : m'_i = m_i \\ i \equiv 0 \bmod 2 : if\ r(m_i) \prec^\sigma_p r(m_{i+1}) \\ \qquad\qquad\ then\ m'_i = m_i \\ \qquad\qquad\ else\ m'_i = m, where\ m \in MES : r(m_i) \prec^\sigma_c s(m) \prec^\sigma_c r(m) \prec^\sigma_p r(m_{i+1}) \end{array} \right),$$

where $m'_{2k+1} = m'_1$. The case $i \equiv 0 \bmod 2$ is well-defined since the causal order corresponds to either the peer order or a chain of messages. We then have $\forall i \in [1, 2k]$: $\left(\begin{array}{l} i \equiv 1 \bmod 2 : s(m'_i) \prec^\sigma_c s(m'_{i+1}) \\ i \equiv 0 \bmod 2 : s(m'_i) \lhd^\sigma_{n-1} s(m'_{i+1}) \end{array} \right)$, where $m'_{2k+1} = m'_1$. Thus $\prec^\sigma_{n-1}$ contains a cycle and is not antisymmetric.

($\Leftarrow$) We prove the contrapositive: if $\prec^\sigma_{n-1}$ contains a cycle, then σ contains a causal knot.

Let $\sigma \in Exec$ such that $\prec^\sigma_{n-1}$ contains a cycle. Since both $\prec^\sigma_c$ and $\lhd^\sigma_{n-1}$ are partial orders, they are both antisymmetric: a minimal cycle thus consists of an alternation of those two. Considering such a minimal cycle of size s, we can deduce by transitivity of $\prec^\sigma_c$ and $\lhd^\sigma_{n-1}$ that it is composed of s events $e_1, .., e_s$ such that $\forall i \in [1, s] : \left(\begin{array}{l} i \equiv 1 \bmod 2 : e_i \prec^\sigma_c e_{i+1} \\ i \equiv 0 \bmod 2 : e_i \lhd^\sigma_{n-1} e_{i+1} \end{array} \right)$, where $e_{s+1} = e_1$. By minimality of the cycle and $e_1 \prec^\sigma_c e_2$, we have $e_s \lhd^\sigma_{n-1} e_1$ and thus $s = 0 \bmod 2$. Let $k = s/2$.

Recall that $\lhd^\sigma_{n-1}$ only orders send events: $e_1, .., e_s$ are thus send events. Their messages are distinct, by minimality of the cycle. Then the previous characterization can be rewritten as $\exists m_1, .., m_{2k} \in MES$ such that $\forall i \in [1, 2k] : \left(\begin{array}{l} i \equiv 1 \bmod 2 : s(m_i) \prec^\sigma_c s(m_{i+1}) \\ i \equiv 0 \bmod 2 : s(m_i) \lhd^\sigma_{n-1} s(m_{i+1}) \end{array} \right)$, where $m_{s+1} = m_1$. By definition of $\lhd^\sigma_{n-1}$, this is a causal knot of size k. ◀

$Clos(\text{Fifo}_{1-n})$ is also negatively characterized by causal knots. Thus $Clos(\text{Fifo}_{n-1}) = Clos(\text{Fifo}_{1-n})$.

▶ **Theorem 18** (Causal Knot Criterion for Fifo_{1-n}). *Let $\sigma \in Exec$. Then $\sigma \in Clos(\text{Fifo}_{1-n})$ $\iff$ σ contains no causal knot.*

Proof. The proof follows the exact same lines as the Causal knot criterion for Fifo_{n-1}, except that we substitute $\lhd^\sigma_{1-n} = \{(r(m_1), r(m_2)) \in \Sigma \times \Sigma \mid s(m_1) \prec^\sigma_p s(m_2)\}$ for $\lhd^\sigma_{n-1}$. It was moved to Appendix A due to space constraints. ◀

This surprising result shows that at the level of computations, those two message orderings cannot be separated.

3.3 Inclusion of closures

We lack a characterization of $Clos(\text{Fifo}_{n-n})$ in terms of forbidden patterns akin to crowns or causal knots. We can nonetheless separate it from $Clos(\text{Fifo}_{n-1}) = Clos(\text{Fifo}_{1-n})$, and from $Clos(\text{RSC})$.

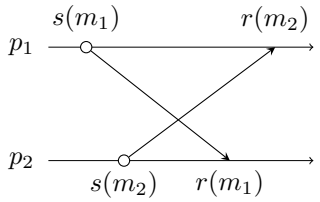

Figure 6 A crown of size 2.

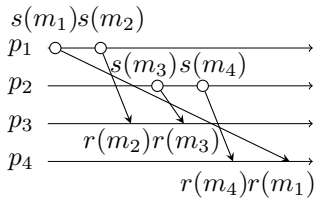

Figure 7 A causal knot of size 2.

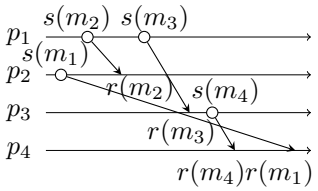

Figure 8 Not in $Clos(\text{Fifo}_{n-n})$; in $Clos(\text{Fifo}_{n-1}) = Clos(\text{Fifo}_{1-n})$.

▶ **Theorem 19** (Inclusion of closures). $Clos(\text{RSC}) \subsetneq Clos(\text{Fifo}_{n-n}) \subsetneq Clos(\text{Fifo}_{1-n}) = Clos(\text{Fifo}_{n-1})$

Proof. The inclusions follow straight from the hierarchy over executions and the definition of an open set: for $A, B \in Exec$, $A \subsetneq B$ implies that $Clos(A) \subseteq Clos(B)$. The first strictness follows from the fact that Figure 6 defines an execution in $Clos(\text{Fifo}_{n-n})$ with a crown, thus not in $Clos(\text{RSC})$. The second strictness follows from the fact that Figure 8 defines an execution σ in $Exec(\text{Fifo}_{1-n})$ (and thus in $Clos(\text{Fifo}_{1-n})$), yet this execution is not in $Clos(\text{Fifo}_{n-n})$.

Assume the contrary: $\exists \sigma' \equiv_c \sigma$ such that $\text{Fifo}_{n-n}(\sigma')$. Then $s(m_4) \prec^{\sigma'} s(m_1) \wedge r(m_2) \prec^{\sigma'} r(m_3)$, since $r(m_4) \prec_p^{\sigma} s(m_1) \wedge s(m_2) \prec_p^{\sigma} s(m_3)$. By $s(m_1) \prec_p^{\sigma} r(m_2) \wedge r(m_3) \prec_p^{\sigma} s(m_4)$, this yields $r(m_2) \prec^{\sigma'} r(m_3) \prec_p^{\sigma} s(m_4) \prec^{\sigma'} s(m_1) \prec_p r(m_2)$. **Contradiction**. ◀

3.4 Interpretation of closures

Closures capture a fundamental part of asynchronicity: the loss of knowledge it entails. Expanding the exact set of executions characterized by a message ordering to its closure introduces the additional uncertainty that comes with the non-determinism of asynchronicity.

First, closures yield a general method to check the distinguishability of a distributed system with message ordering A from one with message ordering B. For example, we deduce from the causal knot criterion that Fifo_{n-1} and Fifo_{1-n} are indistinguishable in the asynchronous world. There is no computation one can exhibit to separate them, showing with certainty which one is implemented.

Second, let us consider the complement of a closure. It represents certainty amidst asynchronicity: assurance of the non-satisfiability of the predicate. This yields a technique for ensuring that forbidden executions do not arise in a distributed system. Take the set of computations with a causal knot of size ≥ 2 as an example. These are "just causal": despite the inherent uncertainty stemming from asynchronicity, we know for sure that a system generating any of these computations cannot have a Fifo_{n-1} or Fifo_{1-n} (let alone Fifo_{n-n} or RSC) execution. Since algorithms and implementations alike usually work better with more constrained message orderings, the less constrained a computation, the more susceptible it is to cause an error or a failure. Just causal computations are thus good candidates for testing and debugging.

4 Interiors

Interiors were defined above as universal quantification over linear extensions: they thus characterize computations where, despite the inherent uncertainty of asynchronous communication, the ordering in question necessarily holds. Through chains, a simple concept from order theory, we characterize and interpret our orderings' interiors.

▶ **Definition 20** (Chain). Let $(A, \prec^A)$ a poset. Then $S \subseteq A$ is a **chain** if its elements are totally ordered for $\prec^A$: $\forall a, b \in S, a \prec^A b \vee b \prec^A a$.

We write that an execution is a chain when its set of events is a chain for its causal order.

4.1 The Interior of RSC

The interior of RSC is characterized by executions which are necessarily a chain with regard to causality. This means that all events of an execution in Int(RSC) are fully ordered by causality. Before stating the theorem, we introduce a result from dimension theory, and its corollary for inverting events in an execution while staying in the causal equivalence class.

▶ **Lemma 21** (Interpolation). *Let $(A, \prec^A)$ a poset, $S \subset A$ and $\prec_l$ a linear extension of $\prec^A_{|S}$ (where $\prec^A_{|S}$ is the projection of $\prec^A$ on $S \times S$). Then $\exists \prec_{l'}$ a linear extension of $\prec^A$ such that $\prec_{l'|S} = \prec_l$.*

Proof. It is a classical result from dimension theory. See Trotter [28] Chapter 1. ◀

▶ **Corollary 22** (Interpolation of executions). *Let σ an execution, Σ the underlying set of events, $S \subset \Sigma$ and $\prec_l$ a linear extension of $\prec^\sigma_{c|S}$. Then $\exists \sigma' \equiv_c \sigma$ such that $\prec^{\sigma'}_{|S} = \prec_l$.*

Proof. It follows immediately from Lemma 21 and the fact that a linear extension of a causal order defines an execution. ◀

▶ **Theorem 23** (Characterization of Int(RSC)). *Let σ an execution with at least one reception. Then $\sigma \in Int(\text{RSC}) \iff \sigma \in Exec(Causal) \wedge \sigma$ is a chain.*

Proof. ($\Rightarrow$) Let $\sigma \in Int$(RSC) with at least one reception. By the hierarchy of message orderings and the definition of interiors, $\sigma \in Exec$(Causal); we show that σ is a chain.

Let $s(m_1), s(m_2) \in \sigma$ such that $s(m_1) \prec^\sigma s(m_2)$. By considering all the possible cases concerning their receptions, we show that all events in σ form a chain.

1. $r(m_1) \in \sigma$. We prove by contradiction that $r(m_1) \prec^\sigma_c s(m_2)$ and thus that $s(m_1) \prec^\sigma_c r(m_1) \prec^\sigma_c s(m_2)$.
   Assume the contrary: $r(m_1) \not\prec^\sigma_c s(m_2)$. Then $\prec_l$ defined by $s(m_1) \prec_l s(m_2) \prec_l r(m_1)$ is a linear extension of $\prec^\sigma_{c|\{s(m_1), s(m_2), r(m_1)\}}$. Thus Corollary 22 ensures the existence of $\sigma' \equiv_c \sigma$ such that $s(m_1) \prec^{\sigma'} s(m_2) \prec^{\sigma'} r(m_1)$. Since σ' violates RSC, we conclude that $\sigma \notin Int$(RSC). **Contradiction**.
   We conclude that $s(m_1) \prec^\sigma_c r(m_1) \prec^\sigma_c s(m_2)$.
2. $r(m_2) \in \sigma$. Then the same reasoning by contradiction than above, here assuming $s(m_1) \not\prec^\sigma_c s(m_2)$, yields $s(m_1) \prec^\sigma_c s(m_2)$ and thus $s(m_1) \prec^\sigma_c s(m_2) \prec^\sigma_c r(m_2)$.
3. $r(m_1) \notin \sigma \wedge r(m_2) \notin \sigma$. Yet by hypothesis, σ contains at least one reception. We split by cases depending on where this reception is placed according to $s(m_1)$ and $s(m_2)$.
   - $\exists m \in MES : s(m_1) \prec^\sigma r(m) \prec^\sigma s(m_2)$. Since $\sigma \in$ RSC, we have $s(m_1) \prec^\sigma s(m) \prec^\sigma r(m) \prec^\sigma s(m_2)$. By case 1 above we have $s(m_1) \prec^\sigma_c r(m)$, and by case 2 above we have $r(m) \prec^\sigma_c s(m_2)$. Transitivity then yields $s(m_1) \prec^\sigma_c s(m_2)$.
   - $\{m \in MES \mid s(m_1) \prec^\sigma r(m) \prec^\sigma s(m_2)\} = \emptyset \wedge \exists m \in MES : r(m) \prec^\sigma s(m_1) \prec^\sigma s(m_2)$. We take m such that it is the last received message before both send events. Then by the case 1 above, $r(m) \prec^\sigma_c s(m_1) \wedge r(m) \prec^\sigma_c s(m_2)$. By definition of m, this gives us $r(m) \prec^\sigma_p s(m_1) \wedge r(m) \prec^\sigma_p s(m_2)$.
     We conclude $s(m_1) \prec^\sigma_p s(m_2)$ and thus $s(m_1) \prec^\sigma_c s(m_2)$.

- $\{m \in MES \mid s(m_1) \prec^\sigma r(m) \prec^\sigma s(m_2)\} = \emptyset \wedge \exists m \in MES : s(m_1) \prec^\sigma s(m_2) \prec^\sigma r(m)$. We take m such that it is the first received message after both send events. The same reasoning as in the previous case, using case 2 instead of case 1, yields $s(m_1) \prec_p^\sigma s(m_2)$ and thus $s(m_1) \prec_c^\sigma s(m_2)$.

($\Leftarrow$) We prove the contrapositive: $\sigma \notin Int(\text{RSC}) \implies \sigma \notin Exec(\text{Causal}) \vee \sigma$ not a chain.

Let σ an execution with at least one reception such that $\sigma \notin Int(\text{RSC})$. Thus $\exists \sigma' \equiv_c \sigma$ such that $\exists m_1, m_2 \in MES : s(m_1) \prec^{\sigma'} s(m_2) \prec^{\sigma'} r(m_1)$. We then have two possible cases.

- $s(m_1) \prec_c^\sigma s(m_2) \prec_c^\sigma r(m_1)$. Then $\exists m \in MES$ such that either $s(m_1) \prec_c^\sigma s(m_2) \prec_c^\sigma s(m) \prec_c^\sigma r(m) \prec_p^\sigma r(m_1)$ or $s(m_1) \prec_c^\sigma s(m) \prec_c^\sigma r(m) \prec_p^\sigma s(m_2) \prec_p^\sigma r(m_1)$, since the causal order corresponds to either the peer order or a chain of messages and since $peer(s(m_1)) \neq peer(r(m_1))$ by definition of a computation. From $s(m_1) \prec_c^\sigma s(m) \prec_c^\sigma r(m) \prec_p^\sigma r(m_1)$, we conclude $\sigma \notin Exec(\text{Causal})$.
- $s(m_1) \not\prec_c^\sigma s(m_2) \vee s(m_2) \not\prec_c^\sigma r(m_1)$. Since $s(m_1) \prec^{\sigma'} s(m_2) \prec^{\sigma'} r(m_1)$, we have $s(m_2) \not\prec_c^\sigma s(m_1) \wedge r(m_1) \not\prec_c^\sigma s(m_2)$. Thus σ is not a chain. ◀

All executions in $Int(\text{RSC})$ are therefore totally determined by their causal order (except the degenerate ones without receptions). This is an extraordinarily strong condition, since it means the causal equivalence class of the execution is the singleton of the execution itself!

4.2 The Interior of Fifo$_{n-n}$

By relaxing the previous characterization, we derive a characterization for $Int(\text{Fifo}_{n-n})$ based on two chains: one for the send events of received messages, and one for receptions. This results in two differences with $Int(\text{RSC})$. First, send events of never received messages are not constrained. Second, consecutive send events are possible even when the corresponding messages are eventually received. In this case, the send events must happen on the same peer; the corresponding receptions must also happen on the same peer.

▶ **Theorem 24** (Characterization of $Int(\text{Fifo}_{n-n})$). *Let $\sigma \in Exec$, $r_\sigma = \{r(m) \in \sigma\}$, and $s_\sigma = \{s(m) \in \sigma \mid r(m) \in \sigma\}$. Then $\sigma \in Int(\text{Fifo}_{n-n}) \iff \sigma \in Exec(Causal) \wedge r_\sigma$ is a chain $\wedge$ s_σ is a chain.*

Proof. This proof and the next two follow the same scheme as the proof of theorem 23 (see Appendix B). ◀

4.3 The Interiors of Fifo$_{n-1}$ and Fifo$_{1-n}$

The characterization of $Int(\text{Fifo}_{n-n})$ is itself weakened in two distinct ways to yield those of $Int(\text{Fifo}_{n-1})$ and $Int(\text{Fifo}_{1-n})$: in $Int(\text{Fifo}_{n-1})$, only send events to a same peer are required to form chains, while in $Int(\text{Fifo}_{1-n})$ it is the receptions from the same peer that need to form chains.

▶ **Theorem 25** (Characterization of $Int(\text{Fifo}_{n-1})$). *Let $\sigma \in Exec$, and $s_\sigma(p) = \{s(m) \in \sigma \mid r(m) \in \sigma \wedge peer(r(m)) = p\}$. Then $\sigma \in Int(\text{Fifo}_{n-1}) \iff \sigma \in Exec(Causal) \wedge \forall p \in PEERS, s_\sigma(p)$ is a chain.*

▶ **Theorem 26** (Characterization of $Int(\text{Fifo}_{1-n})$). *Let $\sigma \in Exec$, and $r_\sigma(p) = \{r(m) \in \sigma \mid peer(s(m)) = p\}$. Then $\sigma \in Int(\text{Fifo}_{1-n}) \iff \sigma \in Exec(Causal) \wedge \forall p \in PEERS, r_\sigma(p)$ is a chain.*

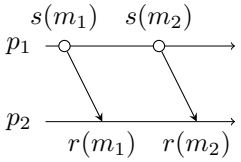

Figure 9 In $Int(\text{RSC})$; not in $Int(\text{Fifo}_{n-n})$

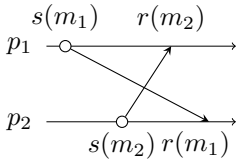

Figure 10 In $Int(\text{Fifo}_{n-n})$; not in $Int(\text{Fifo}_{1-n})$ or $Int(\text{Fifo}_{n-1})$

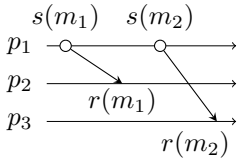

Figure 11 In $Int(\text{Fifo}_{n-1})$; not in $Int(\text{Fifo}_{1-n})$

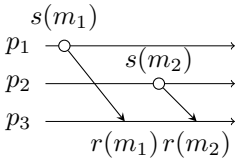

Figure 12 In $Int(\text{Fifo}_{1-n})$; not in $Int(\text{Fifo}_{n-1})$

4.4 Inclusion of FIFO Interiors

We close this study of interiors by showing the various inclusions between them. These turn out to form the same hierarchy as the initial orderings.

▶ **Theorem 27.**

1. $Int(\text{RSC}) \subsetneq Int(\text{Fifo}_{n-n}) \subsetneq Int(\text{Fifo}_{1-n})$
2. $Int(\text{RSC}) \subsetneq Int(\text{Fifo}_{n-n}) \subsetneq Int(\text{Fifo}_{n-1})$
3. $Int(\text{Fifo}_{n-1}) \not\subset Int(\text{Fifo}_{1-n})$ *and* $Int(\text{Fifo}_{1-n}) \not\subset Int(\text{Fifo}_{n-1})$

Proof. The inclusions follow straight from the hierarchy over executions and the definition of an open set: for $A, B \in Exec$, $A \subsetneq B$ implies that $Int(A) \subseteq Int(B)$. For the strictness, it is enough to give examples separating the characterizations.

1. Figure 9 separates $Int(\text{RSC})$ with $Int(\text{Fifo}_{n-n})$ by giving an execution σ where s_σ and r_σ are both chains, but σ itself is not one. Similarly, Figure 10 separates $Int(\text{Fifo}_{n-n})$ with $Int(\text{Fifo}_{1-n})$ giving an execution σ where $r_\sigma(p_1)$ and $r_\sigma(p_2)$ are chains but not σ.
2. Here we only need to prove the separation of $Int(\text{Fifo}_{n-n})$ with $Int(\text{Fifo}_{n-1})$: Figure 10 represents an adequate execution σ where $s_\sigma(p_1)$ and $s_\sigma(p_2)$ are chains but not σ.
3. Incomparability is shown by the executions from Figure 11 (where $r_\sigma(p_1)$ is not a chain but $s_\sigma(p_1), s_\sigma(p_2)$ and $s_\sigma(p_3)$ are) and Figure 12 (where $s_\sigma(p_3)$ is not a chain, but $r_\sigma(p_1), r_\sigma(p_2)$ and $r_\sigma(p_3)$ are). ◀

4.5 Interpretation of interiors

Whereas closures expand the initial set of executions to account for additional uncertainty, interiors trim it down to introduce certainty. In interiors, the message ordering is enforced by the causal order itself, ensuring it despite the non-determinism of asynchronicity. Such certainty can be leveraged as a mean of implementation, by ensuring that a given system only generates computations in the interior of a message ordering.

Let us take $Int(\text{Fifo}_{n-n})$ as an example: due to its characterization (Theorem 24), its computations must ensure both causal ordering and that all messages simultaneously in transit are between the same pair of peers. This description fits the definition of a token network, where the peer holding the token is the only one allowed to send (to a single destination peer). It then sends the token along in its last message, and on again. Similar

characterizations can be worked out for the other interiors: Int(RSC) defines a token model where the token is sent with each message, and $Int(\text{Fifo}_{n-1})$ and $Int(\text{Fifo}_{1-n})$ respectively defines token models with a send token for each peer in one case and a receive token for each peer in the other. Thus a distributed system using any of these token models implements the corresponding interior.

One apparent contradiction with the literature is that our interiors cannot be implemented in the sense of Murty and Garg [22]. Indeed, the latter paper shows that any message ordering (over computations) implementable by an inhibitory protocol must contain a specific subset of computations, which is equivalent to our $Clos$(RSC): our interiors don't satisfy this condition. By inhibitory protocol, [22] means a protocol only able to delay both send events requested by the user and deliveries of received messages. Even if our token-based conditions do not help building an inhibitory protocol, they do implement the interiors in a different way: such systems only generate computations in the corresponding interiors, and thus executions in the corresponding orderings over executions.

5 Related Work

Message orderings over computations. The Fifo_{1-1} ordering (traditionally called FIFO) dates back to the first distributed algorithms, such as Chandy-Lamport Snapshot [5]. The latter paper exemplifies the treatment of this ordering: it is brushed aside in one sentence, not even given a proper name.

Causal ordering has a deeper history. Following the connection drawn by Lamport between Happened-Before and potential causality [20], the former was used as a basis for causal broadcast ordering by Birman et al. [4]. But the broadcast part blurred the exact definition and eased the implementation: a point-to-point formalization by Schiper and al. [24] followed. It made use notably of multiple vector clocks (independently invented by Fidge [14] and Mattern [21]) to track potential causality beyond the logical clocks introduced by Lamport [20]. Following this line of research, Schwarz and Mattern [25] motivated the general problem of detecting the needed causality, while Kshemkalyani and Singhal [19] proved necessary and sufficient conditions for implementing Causal ordering, as well as an optimal implementation based on these.

$Clos$(RSC) (usually simply called RSC) is a late invention compared to the two previous orderings: it was introduced independently by Soneoka and Ibaraki [26] and Charron-Bost et al. [8] after almost all the papers mentioned above. By being non-blocking yet allowing each message to be alone in transit, $Clos$(RSC) represents the best approximation of synchronous (or rendez-vous) communication in an asynchronous setting.

Causal ordering and $Clos$(RSC) were used for characterizing implementability of message orderings by inhibitory protocols: Murty and Garg [22] showed that an ordering must contain Causal to be implementable without control messages, and the equivalent of $Clos$(RSC) to be implementable at all. Charron-Bost et al. [8] also used these three orderings as a basis for their hierarchy.

Beyond Happened-Before. Following Birman et al. [4] definition and implementation of causal broadcast ordering, Cheriton and Skeen [10] shed light into the limitations of Happened-Before as causality. Namely, that local events are always totally ordered by the Happened-Before relation while they might be causally independent. This in turn causes additional latency: if two send events are causally independent but ordered by Happened-Before, one reception might unnecessarily wait for the other.

As a follow-up, Tarafdar and Garg [27] developed the idea of potential causality for approximating true causality in the same way Happened-Before order approximates real time.

The difference stems from the local ordering of events: whereas the Happened-Before relation totally orders them, potential causality orders local events using an applicative and thus domain-dependent criterion. It can therefore dodge false causality. Tarafdar and Garg [27] also provided an example of potential causality applied to predicate detection, a problem where false causality causes both false positives and false negatives.

Finally, Ben-Zvi and Moses [3] extended the Happened-Before relation to the synchronous case. Their Syncausality captures the ability, when communication is bounded, to detect that no message was sent at a given time by a given peer. And their Bound Guarantees captures that if one knows a message was sent and the bound of the channel, then one knows after which point the message was necessarily received.

Knowledge About Uncertainty. Both closures and interiors provide additional knowledge about the computation: whether it might follow the ordering and whether it must, respectively.

Cooper and Marzullo [12] also treat knowledge through existential and universal quantification, but this time on path of consistent cuts. They define the *Possibly* and *Definitely* operators for predicates over observations (consistent cuts): *Possibly* ϕ means that there is a path in the lattice of observations passing through an observation satisfying ϕ; *Definitely* ϕ means that all paths pass through an observation satisfying ϕ. In a subsequent paper, Charron-Bost et al. [7] showed that these two operators were temporal analogous to the local knowledge operator *Knows*. They leveraged this analogy to prove a temporal counterpart to Chandy and Misra's Knowledge Change Theorem [6].

The latter paper by Chandy and Misra [6] was part of the effort to formalize knowledge in distributed systems. As almost all concurrent and subsequent efforts, it was based on the notion of indistinguishability: a process knows a predicate valuation if and only if this valuation is constant over all possible "worlds" the process cannot distinguish. Halpern and Moses [15] anchored this possible worlds semantics with axiomatisations of different knowledge characterizations, using Kripke structures as models. The interested reader is directed to Fagin et al. [13] for a thorough treatment of the subject.

Topology in distributed systems. Lastly, our approach is based on topology. The characterization of Liveness and Safety by Alpern and Schneider [1] can be considered the first application of topology to distributed computing: although their topology on the execution space is not specific to distributed systems, their characterizations play a significant role in proving correctness and efficiency of distributed algorithms. A version taking into account failures was subsequently proposed by Charron-Bost et al. [9].

Another application of topology concerns wait-freeness. A wait-free algorithm, following Herlihy's definition [16], is one where each process must terminate within a finite number of steps. Waiting for another process is thus not possible, which makes wait-free algorithms a powerful tool for fault-tolerance. Herlihy and Shavit [18] applied algebraic topology to the problem of solving tasks in a wait-free way. They showed that both the possible input and output of a task could be represented as simplicial complexes, mathematical structures from algebraic topology. Then the possibility or impossibility of wait-freely solving a task can be rephrased as the existence or nonexistence of a specific map from the input complex to the output one. We direct the interested reader to Herlihy et al. [17] for an ample treatment of this applications of algebraic topology to distributed computability.

6 Conclusion and Perspectives

Through our topology, we offered an analysis of asynchronous message orderings over executions. Their closures precisely characterize the additional uncertainty brought by asynchronicity, allowing us to show that orderings are indistinguishable at the level of computations. Interiors, on the other hand, characterize the necessary reduction to ensure the message ordering despite the non-determinism of asynchronicity. This in turn can be used as an operational constraint for implementing the ordering.

Far from putting an end to this line of research, we feel this work opens up interesting directions for further inquiry.

- First, finding a characterization for $Clos(\text{Fifo}_{n-n})$ will precise the distinguishability condition between message orderings, as well as give another tool for distributed systems programmers to decide the ordering they need.
- Multicast and broadcast message orderings are also predicates over computations and executions. Thus the topological approach introduced in this paper can be applied to them too.
- The interpretation of interiors brought up the question of implementation. Is there a more general definition than the existence of an inhibitory protocol? If so, then it will need to account for our implementations through system-level constraints such as tokens.
- Message orderings over executions which do not collapse ask for a stronger order than the causal one, thus more knowledge about the ordering of events. Studying this additional knowledge will precise the respective power of message orderings.

References

1 Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985. doi:10.1016/0020-0190(85)90056-0.

2 Samik Basu, Tevfik Bultan, and Meriem Ouederni. Synchronizability for verification of asynchronously communicating systems. In Viktor Kuncak and Andrey Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation - 13th International Conference, VMCAI 2012, Philadelphia, PA, USA, January 22-24, 2012. Proceedings*, volume 7148 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 2012. doi:10.1007/978-3-642-27940-9_5.

3 Ido Ben-Zvi and Yoram Moses. Beyond lamport's *Happened-before*: On time bounds and the ordering of events in distributed systems. *J. ACM*, 61(2):13:1–13:26, 2014. doi:10.1145/2542181.

4 Kenneth P. Birman and Thomas A. Joseph. Reliable communication in the presence of failures. *ACM Trans. Comput. Syst.*, 5(1):47–76, 1987. doi:10.1145/7351.7478.

5 K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, 1985. doi:10.1145/214451.214456.

6 K. Mani Chandy and Jayadev Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986. doi:10.1007/BF01843569.

7 Bernadette Charron-Bost, Carole Delporte-Gallet, and Hugues Fauconnier. Local and temporal predicates in distributed systems. *ACM Trans. Program. Lang. Syst.*, 17(1):157–179, 1995. doi:10.1145/200994.201005.

8 Bernadette Charron-Bost, Friedemann Mattern, and Gerard Tel. Synchronous, asynchronous, and causally ordered communication. *Distributed Computing*, 9(4):173–191, 1996. doi:10.1007/s004460050018.

9 Bernadette Charron-Bost, Sam Toueg, and Anindya Basu. Revisiting safety and liveness in the context of failures. In Catuscia Palamidessi, editor, *CONCUR 2000 - Concurrency*

Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings, volume 1877 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2000. doi:10.1007/3-540-44618-4_39.

10 David R. Cheriton and Dale Skeen. Understanding the limitations of causally and totally ordered communication. *SIGOPS Oper. Syst. Rev.*, 27(5):44–57, 1993. doi:10.1145/173668.168623.

11 Florent Chevrou, Aurélie Hurault, and Philippe Quéinnec. On the diversity of asynchronous communication. *Formal Asp. Comput.*, 28(5):847–879, 2016. doi:10.1007/s00165-016-0379-x.

12 Robert Cooper and Keith Marzullo. Consistent detection of global predicates. *SIGPLAN Not.*, 26(12):167–174, dec 1991. doi:10.1145/127695.122774.

13 Ronald Fagin, Joseph Y. Halpern, Moshe Y. Vardi, and Yoram Moses. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 1995.

14 Colin J. Fidge. Timestamps in message-passing systems that preserve the partial ordering. *Proceedings of the 11th Australian Computer Science Conference*, 10(1):56–66, 1988.

15 Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *J. ACM*, 37(3):549–587, 1990. doi:10.1145/79147.79161.

16 Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, 1991. doi:10.1145/114005.102808.

17 Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann Publishers Inc., 1st edition, 2013.

18 Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999. doi:10.1145/331524.331529.

19 Ajay D. Kshemkalyani and Mukesh Singhal. Necessary and sufficient conditions on information for causal message ordering and their optimal implementation. *Distributed Computing*, 11(2):91–111, 1998. doi:10.1007/s004460050044.

20 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978. doi:10.1145/359545.359563.

21 Friedemann Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1989.

22 Venkatesh V. Murty and Vijay K. Garg. Characterization of message ordering specifications and protocols. In *Proceedings of the 17th International Conference on Distributed Computing Systems, Baltimore, MD, USA, May 27-30, 1997*, pages 492–499. IEEE Computer Society, 1997. doi:10.1109/ICDCS.1997.603392.

23 Michel Raynal, André Schiper, and Sam Toueg. The causal ordering abstraction and a simple way to implement it. *Inf. Process. Lett.*, 39(6):343–350, 1991. doi:10.1016/0020-0190(91)90008-6.

24 André Schiper, Jorge Eggli, and Alain Sandoz. A new algorithm to implement causal ordering. In Jean-Claude Bermond and Michel Raynal, editors, *Distributed Algorithms, 3rd International Workshop, Nice, France, September 26-28, 1989, Proceedings*, volume 392 of *Lecture Notes in Computer Science*, pages 219–232. Springer, 1989. doi:10.1007/3-540-51687-5_45.

25 Reinhard Schwarz and Friedemann Mattern. Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing*, 7(3):149–174, 1994. doi:10.1007/BF02277859.

26 Terunao Soneoka and Toshihide Ibaraki. Logically instantaneous message passing in asynchronous distributed systems. *IEEE Trans. Computers*, 43(5):513–527, 1994. doi:10.1109/12.280800.

27 Ashis Tarafdar and Vijay K. Garg. Addressing false causality while detecting predicates in distributed programs. In *Proceedings of the 18th International Conference on Distributed*

Computing Systems, Amsterdam, The Netherlands, May 26-29, 1998, pages 94–101. IEEE Computer Society, 1998. doi:10.1109/ICDCS.1998.679491.

28 William T. Trotter. *Combinatorics and Partially Ordered Sets*. The Johns Hopkins University Press, 1992.

A Proof of the causal knot criterion for Fifo_{1-n}

We first introduce $\prec^{\sigma}_{1-n}$, which corresponds to $\prec^{\sigma}_{c}$ augmented with the constraints of Fifo_{1-n}.

▶ **Definition 28** (1-N order). Let $\sigma \in Exec$ and Σ the underlying set of events. Then $\lhd^{\sigma}_{1-n} \triangleq \{(r(m_1), r(m_2)) \in \Sigma \times \Sigma \mid s(m_1) \prec^{\sigma}_{p} s(m_2)\}$. And $\prec^{\sigma}_{1-n} \triangleq \prec^{\sigma}_{c} \cup \lhd^{\sigma}_{1-n}$.

Then the next Lemma reduces the membership of an execution σ in $Clos(\text{Fifo}_{1-n})$ to whether or not $\prec^{\sigma}_{1-n}$ contains a cycle.

▶ **Lemma 29.** *Let $\sigma \in Exec$. Then $\sigma \in Clos(Fifo_{1-n}) \iff \prec^{\sigma}_{1-n}$ is antisymmetric.*

Proof. ($\Rightarrow$) We show the contrapositive: If $\prec^{\sigma}_{1-n}$ contains a cycle, then $\sigma \notin Clos(\text{Fifo}_{1-n})$.

Let $\sigma \in Exec$ such that $\prec^{\sigma}_{1-n}$ contains a cycle. Then its transitive closure is not a partial order: it has no linear extensions. From that fact, we now prove that no execution in $[\sigma]_{\equiv_c}$ satisfies Fifo_{1-n}, and thus that $\sigma \notin Clos(\text{Fifo}_{1-n})$.

Let $\sigma' \equiv_c \sigma$. By our hypothesis that $\prec^{\sigma}_{1-n}$ contains a cycle, we have $\prec^{\sigma}_{1-n} \not\subset \prec^{\sigma'}$. But since $\sigma' \equiv_c \sigma$, we have $\prec^{\sigma}_{c} \subset \prec^{\sigma'}$. We thus conclude that $\lhd^{\sigma}_{1-n} \not\subset \prec^{\sigma'}$. Thus $\exists m_1, m_2$ such that $s(m_1) \prec^{\sigma}_{p} s(m_2) \wedge r(m_1) \not\prec^{\sigma'} r(m_2)$. By totality of $\prec^{\sigma'}$, we have $s(m_1) \prec^{\sigma}_{p} s(m_2) \wedge r(m_2) \prec^{\sigma'} r(m_1)$, and thus σ' violates Fifo_{1-n}.

($\Leftarrow$) Let $\sigma \in Exec$ such that $\prec^{\sigma}_{1-n}$ is antisymmetric. Then the transitive closure of $\prec^{\sigma}_{1-n}$ is reflexive (because $\prec^{\sigma}_{c}$ is reflexive), transitive and antisymmetric. It is therefore a partial order, who has a linear extension agreeing both with $\prec^{\sigma}_{c}$ and $\lhd^{\sigma}_{1-n}$. This linear extension defines an execution $\sigma' \equiv_c \sigma$ (because $\prec^{\sigma}_{c} \subset \prec^{\sigma'}$) such that σ' satisfies Fifo_{1-n} (because $\lhd^{\sigma}_{1-n} \subset \prec^{\sigma'}$).

We conclude that $\sigma \in Clos(\text{Fifo}_{1-n})$. ◀

We can now prove the causal knot criterion for Fifo_{1-n}, which characterizes $Clos(\text{Fifo}_{1-n})$ by the absence of causal knots.

▶ **Theorem 30** (Causal Knot Criterion for Fifo_{1-n}). *Let $\sigma \in Exec$. Then $\sigma \in Clos(\text{Fifo}_{1-n})$ $\iff$ σ contains no causal knot.*

Proof. By Lemma 29 and the transitivity of equivalence, we only need to prove that $\prec^{\sigma}_{1-n}$ is antisymmetric $\iff$ σ contains no causal knot.

($\Rightarrow$) We prove the contrapositive: if σ contains a causal knot, then $\prec^{\sigma}_{1-n}$ is not antisymmetric.

Let $\sigma \in Exec$ with a causal knot of size k. By definition of causal knots, there are $2k$ messages $m_1, .., m_{2k}$ such that $\iff \forall i \in [1, 2k] : \begin{pmatrix} i \equiv 1 \bmod 2 : s(m_i) \prec^{\sigma}_{c} s(m_{i+1}) \\ i \equiv 0 \bmod 2 : r(m_i) \prec^{\sigma}_{c} r(m_{i+1}) \end{pmatrix}$, where $m_{2k+1} = m_1$. We now define another sequence of messages $m'_1, .., m'_{2k}$ from the messages of the causal knot: $\forall i \in [1, 2k]$:

$$\begin{pmatrix} i \equiv 1 \bmod 2 : if\ s(m_i) \prec^{\sigma}_{p} s(m_{i+1}) \\ \qquad\qquad\quad then\ m'_i = m_i \\ \qquad\qquad\quad else\ m'_i = m, where\ m \in MES : s(m_i) \prec^{\sigma}_{p} s(m) \prec^{\sigma}_{c} r(m) \prec^{\sigma}_{c} s(m_{i+1}) \\ i \equiv 0 \bmod 2 : m'_i = m_i \end{pmatrix},$$

where $m'_{2k+1} = m'_1$. The case $i \equiv 0 \bmod 2$ is well-defined since the causal order corresponds to either the peer order or a chain of messages. We then have $\forall i \in [1, 2k]$: $\begin{pmatrix} i \equiv 1 \bmod 2 : r(m'_i) \lhd^\sigma_{1-n} r(m'_{i+1}) \\ i \equiv 0 \bmod 2 : r(m'_i) \prec^\sigma_c r(m'_{i+1}) \end{pmatrix}$, where $m'_{2k+1} = m'_1$. Thus $\prec^\sigma_{1-n}$ contains a cycle and is not antisymmetric.

($\Leftarrow$) We prove the contrapositive: if $\prec^\sigma_{1-n}$ contains a cycle, then σ contains a causal knot.

Let $\sigma \in Exec$ such that $\prec^\sigma_{1-n}$ contains a cycle. Since both $\prec^\sigma_c$ and $\lhd^\sigma_{1-n}$ are partial orders, they are both antisymmetric: any minimal cycle consists of an alternation of those two. Considering such a minimal cycle of size s, we can deduce by transitivity of $\prec^\sigma_c$ and $\lhd^\sigma_{1-n}$ that it is composed of s events $e_1, .., e_s$ such that $\forall i \in [1, s]$: $\begin{pmatrix} i \equiv 1 \bmod 2 : e_i \lhd^\sigma_{1-n} e_{i+1} \\ i \equiv 0 \bmod 2 : e_i \prec^\sigma_c e_{i+1} \end{pmatrix}$, where $e_{s+1} = e_1$. By minimality of the cycle and $e_1 \lhd^\sigma_{1-n} e_2$, we have $e_s \prec^\sigma_c e_1$ and thus $s \equiv 0 \bmod 2$. Let $k = s/2$.

Recall that $\lhd^\sigma_{1-n}$ only orders receptions: $e_1, .., e_s$ are thus receptions. Their messages are distinct, by minimality of the cycle. Then the previous characterization of the cycle can be stated as $\exists m_1, .., m_{2k} \in MES$ such that $\forall i \in [1, 2k]$: $\begin{pmatrix} i \equiv 1 \bmod 2 : r(m_i) \lhd^\sigma_{1-n} r(m_{i+1}) \\ i \equiv 0 \bmod 2 : r(m_i) \prec^\sigma_c r(m_{i+1}) \end{pmatrix}$, where $m'_{2k+1} = m'_1$. By definition of $\lhd^\sigma_{1-n}$, this is a causal knot of size k. ◀

B Proof of the characterizations of $Int(\text{Fifo}_{n-n})$, $Int(\text{Fifo}_{1-n})$ and $Int(\text{Fifo}_{n-1})$

B.1 Proof of preliminary lemma

The reduction from Corollary 22 allows us to prove that when two send events (respectively two receptions) are not causally ordered in an execution, there is a causally equivalent execution where these two events are inverted, whereas the corresponding communication events (the receptions for the send events and conversely) keep the same order. These inversions correspond to potential breaking points by causal equivalence for message orderings, a formalization of the intuition behind the counter-examples used in the proof of Theorem 11. They will therefore constitute our main tool for showing non-membership in a given interior.

▶ **Lemma 31** (Inversion of communication events). *Let $\sigma \in Exec$, and $m_1, m_2 \in MES$ such that $r(m_1), r(m_2) \in \sigma$. Then:*

1. $s(m_1) \prec^\sigma s(m_2) \wedge s(m_1) \not\prec^\sigma_c s(m_2)$
$\implies \exists \sigma' \equiv_c \sigma : s(m_2) \prec^{\sigma'} s(m_1) \wedge \prec^{\sigma'}_{|\{r(m_1), r(m_2)\}} = \prec^\sigma_{|\{r(m_1), r(m_2)\}}$
2. $r(m_1) \prec^\sigma r(m_2) \wedge r(m_1) \not\prec^\sigma_c r(m_2)$
$\implies \exists \sigma' \equiv_c \sigma : r(m_2) \prec^{\sigma'} r(m_1) \wedge \prec^{\sigma'}_{|\{s(m_1), s(m_2)\}} = \prec^\sigma_{|\{s(m_1), s(m_2)\}}$

Proof. Let $\sigma \in Exec$, and let $m_1, m_2 \in MES$ such that $r(m_1), r(m_2) \in \sigma$. Both cases boil down to exhibiting a $\sigma' \equiv_c \sigma$ satisfying the condition. If we had a linear extension of $\prec^\sigma_{c|\{s(m_1), s(m_2), r(m_1), r(m_2)\}}$ satisfying the condition, Lemma 22 would give us such valid σ'. We therefore prove the existence of a suitable linear extension of $\prec^\sigma_{c|\{s(m_1), s(m_2), r(m_1), r(m_2)\}}$ in each case.

1. Let $s(m_1) \prec^\sigma s(m_2)$ and $s(m_1) \not\prec^\sigma_c s(m_2)$. Let $\prec_l$ a total order on $\{s(m_1), s(m_2), r(m_1), r(m_2)\}$ defined by $s(m_2) \prec_l s(m_1) \prec_l r(m_1)$, $s(m_2) \prec_l s(m_1) \prec_l r(m_2)$ and $\prec_{l|\{r(m_1), r(m_2)\}} = \prec^\sigma_{|\{r(m_1), r(m_2)\}}$.
Since $s(m_1) \not\prec^\sigma_c s(m_2)$ by hypothesis and $\prec^\sigma_c$ is transitive, we get $r(m_1) \not\prec^\sigma_c s(m_2)$. This in turn implies $r(m_1) \not\prec^\sigma_c s(m_1) \wedge r(m_1) \not\prec^\sigma_c s(m_2)$.

On the other hand, transitivity of $\prec^\sigma$ yields $s(m_1) \prec^\sigma r(m_2)$. Thus $r(m_2) \not\prec^\sigma_c s(m_1)$, and we deduce $r(m_2) \not\prec^\sigma_c s(m_1) \wedge r(m_2) \not\prec^\sigma_c s(m_2)$.
$\prec_l$ is therefore a linear extension of $\prec^\sigma_{c|\{s(m_1),s(m_2),r(m_1),r(m_2)\}}$.

2. The proof is similar to the previous case, with $\prec_{l'}$ a total order on $\{s(m_1), s(m_2), r(m_1), r(m_2)\}$ such that $s(m_1) \prec_{l'} r(m_2) \prec_{l'} r(m_1)$, $s(m_1) \prec_{l'} r(m_2) \prec_{l'} r(m_1)$ and $\prec_{l'|\{s(m_1),s(m_2)\}} = \prec^\sigma_{|\{s(m_1),s(m_2)\}}$. ◀

B.2 Proof of the characterization of $Int(\text{Fifo}_{n-n})$

We now prove the characterization of $Int(\text{Fifo}_{n-n})$ from Theorem 24

Proof. ($\Rightarrow$) **Assume** the opposite: let $\sigma \in Int(\text{Fifo}_{n-n})$ and r_σ or s_σ is not a chain ($\sigma \in Exec(\text{Causal})$ necessarily).

- If r_σ is not a chain, $\exists m_1, m_2 \in MES$ such that $r(m_1) \prec^\sigma r(m_2)$ and $r(m_1) \not\prec^\sigma_c r(m_2)$. Since $\sigma \in Int(\text{Fifo}_{n-n})$, we get $s(m_1) \prec^\sigma s(m_2)$. The hypothesis $r(m_1) \not\prec^\sigma_c r(m_2)$ and Lemma 31 then entail the existence of $\sigma' \equiv_c \sigma$ with $s(m_1) \prec^{\sigma'} s(m_2)$ and $r(m_2) \prec^{\sigma'} r(m_1)$. But $\sigma' \notin Exec(\text{Fifo}_{n-n})$. Hence $\sigma \notin Int(\text{Fifo}_{n-n})$. **Contradiction**.
- If s_σ is not a chain, the same application of Lemma 31 to the case $s(m_1) \prec^\sigma s(m_2)$, $s(m_1) \not\prec^\sigma_c s(m_2)$ and $r(m_1) \prec^\sigma r(m_2)$ gives us a $\sigma'' \equiv_c \sigma$ with $\sigma'' \notin Exec(\text{Fifo}_{n-n})$. Hence $\sigma \notin Int(\text{Fifo}_{n-n})$. **Contradiction**.

($\Leftarrow$) We prove the contrapositive, namely: $\sigma \notin Int(\text{Fifo}_{n-n}) \implies \sigma \notin Exec(\text{Causal}) \vee s_\sigma$ not a chain $\vee r_\sigma$ not a chain.

Let $\sigma \in Exec$ such that $\sigma \notin Int(\text{Fifo}_{n-n})$. Thus $\exists \sigma' \equiv_c \sigma$ such that $\exists m_1, m_2 \in MES$: $s(m_1) \prec^{\sigma'} s(m_2) \wedge r(m_2) \prec^{\sigma'} r(m_1)$. We then have 2 cases.

- $s(m_1) \prec^\sigma_c s(m_2) \wedge r(m_2) \prec^\sigma_c r(m_1)$. Then by definition of the causal order, either $r(m_2) \prec^\sigma_p r(m_1)$ or $\exists m \in MES$ such that $s(m_1) \prec^\sigma_c s(m_2) \prec^\sigma_c r(m_2) \prec^\sigma_c s(m) \prec^\sigma_c r(m) \prec^\sigma_p r(m_1)$. Either way, we conclude $\sigma \notin Exec(\text{Causal})$.
- $s(m_1) \not\prec^\sigma_c s(m_2) \vee r(m_2) \not\prec^\sigma_c r(m_1)$. Then s_σ or r_σ is not a chain. ◀

B.3 Proof of the characterization of $Int(\text{Fifo}_{n-1})$ and $Int(\text{Fifo}_{1-n})$

We now prove both characterizations of $Int(\text{Fifo}_{n-1})$ and $Int(\text{Fifo}_{1-n})$ from Theorem 25 and Theorem 26 respectively.

Proof. ($\Rightarrow$) **Assume** the opposite: let $\sigma \in Int(\text{Fifo}_{n-1})$ and $p \in PEERS$ such that $s_\sigma(p)$ is not a chain ($\sigma \in Exec(\text{Causal})$ necessarily).

Thus $\exists m_1, m_2 \in MES$ such that $s(m_1), s(m_2) \in s_\sigma(p)$, $s(m_1) \prec^\sigma s(m_2)$ and $s(m_1) \not\prec^\sigma_c s(m_2)$. Since $\sigma \in Exec(\text{Fifo}_{n-1})$ and $peer(r(m_1)) = peer(r(m_2))$, we have $r(m_1) \prec^\sigma_p r(m_2)$. Then Lemma 31 implies $\exists \sigma' \equiv_c \sigma$ such that $s(m_2) \prec^{\sigma'} s(m_1), r(m_1) \prec^{\sigma'} r(m_2)$ and $peer(r(m_1)) = peer(r(m_2))$. But $\sigma' \notin Exec(\text{Fifo}_{n-1})$, and thus $\sigma \notin Int(\text{Fifo}_{n-1})$. **Contradiction**.

($\Leftarrow$) We prove the contrapositive, namely: $\sigma \notin Int(\text{Fifo}_{n-1}) \implies \sigma \notin Exec(\text{Causal}) \vee \exists p \in PEERS : s_\sigma(p)$ not a chain.

Let $\sigma \in Exec$ such that $\sigma \notin Int(\text{Fifo}_{n-1})$. Thus $\exists \sigma' \equiv_c \sigma$ such that $\exists m_1, m_2 \in MES$: $s(m_1) \prec^{\sigma'} s(m_2) \wedge r(m_2) \prec^\sigma_p r(m_1)$. We then have 2 cases.

- $s(m_1) \prec^\sigma_c s(m_2)$. From $s(m_1) \prec^\sigma_c s(m_2) \prec^\sigma_c r(m_2) \prec^\sigma_p r(m_1)$, we conclude $\sigma \notin Exec(\text{Causal})$.
- $s(m_1) \not\prec^\sigma_c s(m_2)$. Then $s_\sigma(peer(r(m_1)))$ is not a chain. ◀

Proof. ($\Rightarrow$) **Assume** the opposite: let $\sigma \in Int(\text{Fifo}_{1-n})$ and $p \in PEERS$ such that $r_\sigma(p)$ is not a chain ($\sigma \in Exec(\text{Causal})$ necessarily).

Thus $\exists m_1, m_2 \in MES$ such that $r(m_1), r(m_2) \in r_\sigma(p)$, $r(m_1) \prec^\sigma r(m_2)$ and $r(m_1) \not\prec_c^\sigma r(m_2)$. Since $\sigma \in Exec(\text{Fifo}_{1-n})$ and $peer(s(m_1)) = peer(s(m_2))$, we have $s(m_1) \prec_p^\sigma s(m_2)$. Then Lemma 31 implies $\exists \sigma' \equiv_c \sigma$ such that $r(m_2) \prec^{\sigma'} r(m_1), s(m_1) \prec^{\sigma'} s(m_2)$ and $peer(s(m_1)) = peer(s(m_2))$. But $\sigma' \notin Exec(\text{Fifo}_{1-n})$, and thus $\sigma \notin Int(\text{Fifo}_{1-n})$. **Contradiction**.

($\Leftarrow$) We prove the contrapositive, namely: $\sigma \notin Int(\text{Fifo}_{1-n}) \implies \sigma \notin Exec(\text{Causal}) \vee \exists p \in PEERS : r_\sigma(p)$ not a chain.

Let $\sigma \in Exec$ such that $\sigma \notin Int(\text{Fifo}_{1-n})$. Thus $\exists \sigma' \equiv_c \sigma$ such that $\exists m_1, m_2 \in MES : s(m_1) \prec_p^\sigma s(m_2) \wedge r(m_2) \prec^{\sigma'} r(m_1)$. We then have 2 cases.

- $r(m_2) \prec_c^\sigma r(m_1)$. Then either $r(m_2) \prec_p^\sigma r(m_1)$ or $\exists m \in MES : r(m_2) \prec_c^\sigma s(m) \prec_c^\sigma r(m) \prec_p^\sigma r(m_1)$, by definition of the causal order. Both cases yield $\sigma \notin Exec(\text{Causal})$.
- $r(m_2) \not\prec_c^\sigma r(m_1)$. Since $r(m_2) \prec^{\sigma'} r(m_1)$ gives us $r(m_1) \not\prec_c^\sigma r(m_2)$, we conclude that $r_\sigma(peer(s(m_1)))$ is not a chain. ◀