J. Parallel Distrib. Comput. 72 (2012) 791–795

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

# From the *Happened-Before* Relation to the Causal Ordered Set Abstraction

S.E. Pomares Hernandez [a,*], J.R. Perez Cruz [a], M. Raynal [b]

[a] Computer Science Department, National Institute of Astrophysics, Optics and Electronics (INAOE), Luis Enrique Erro 1, C.P. 72840, Tonantzintla, Puebla, Mexico

[b] Institut Universitaire de France & IRISA-ISTIC Université de Rennes, 35 042 Rennes-cedex, France

ARTICLE INFO

*Article history:*
Received 13 May 2011
Received in revised form
15 February 2012
Accepted 22 February 2012
Available online 5 March 2012

*Keywords:*
Happened-Before Relation
Event ordering
Distributed systems
Ordered sets

ABSTRACT

Several works in distributed systems have been designed based on the *Happened-Before Relation* (HBR). Most of these works intend to be efficient in their implementation by identifying and ensuring dependency constraints among single events. Even when the minimal causal dependencies among events have been clearly identified, the evolution of systems, which may involve a high number of processes and a high volume of transmitted data, calls for the need to design even more efficient approaches. This paper proposes the *Causal Ordered Set Abstraction* (CAOS) where the causally related events are arranged in sets that are strictly causally ordered. As for single events, CAOS establishes that any pair of resultant sets can be, and can only be, causally or concurrently related. We claim that our ordered set abstraction can be used to design more efficient algorithms based on the HBR principle. This assertion is based on two main properties. First, CAOS attains a consistent compact representation of a distributed computation. Second, as a consequence of the causal ordering of the events in the resultant sets, it is sufficient to verify only a pair of single events, one per each set, in order to determine whether these sets are causally or concurrently related, regardless of the cardinality of the sets.



## 1. Introduction

*The Happened-Before Relation* (HBR) introduced by Lamport [6] has been one of the most important contributions in the field of distributed systems; this work received the ACM PODC Influential Paper Award in 2000. The HBR establishes, without using global references, the conditions to determine for any pair of single events $a$, $b$ in a system if the event $a$ occurs before the event $b$ (denoted by $a \rightarrow b$). Based on the HBR, it has been established that if two events are not causally related, then they are concurrently related (denoted by $a \| b$). Several works of different domains in distributed systems have been designed based on the HBR principle. Most of these works intend to be efficient in their implementation by identifying and ensuring dependency constraints among single events. At single events, the necessary and sufficient dependency constraints that must be satisfied are determined by the transitive reduction of the HBR [1,11] (See Definition 2). Even when the minimal causal dependencies among events have been clearly identified, the evolution in the systems, which considers a high number of processes and a high volume of transmitted data, calls for the need to design even more efficient approaches. In this pursuit, some previous works have proposed to work instead with causal dependencies considering an ordered event set level. The most important works are: Shimamura et al. [12] and Pomares et al. [10] for multimedia synchronization, Chandra and Kshemkalyani [2] for predicate detection, and Hélary et al. [5,4] and Netzer and Xu [9] for checkpointing. All these works use and/or extend the theory presented by Lamport in [7] that defines two abstract precedence relations for sets called *precedes*, denoted in this paper by "$\rightarrow$", and *can affect* relation denoted by "$\dashrightarrow$". The precedes relation establishes that two sets are causally related $A \rightarrow B$ if every event in $A$ precedes every event in $B$. On the other hand, the second relation establishes that $A \dashrightarrow B$ if some event in $A$ precedes some event in $B$. The referred works partially take into account the HBR principle and the theory of sets of Lamport since they only consider for the precedes relation that a set is composed by local events of a process $p_i$ and totally ordered $\rightarrow_i$. The present paper proposes the Causal Ordered Set Abstraction (CAOS) that defines the rules and conditions of association to construct ordered sets of events according to all three conditions of the HBR, thus allowing the events that compose a set to originate from different sources (processes). Furthermore, CAOS is characterized by strictly fulfilling the causal ordering at the set level. This is important because, as for single events, we establish that any pair of resultant sets in CAOS can be, and can only be, causally or concurrently related. In our proposal, two sets are concurrently related, denoted by $A \| B$, if all the events of $A$ are concurrent to all the events of $B$.

We claim that the Causal Ordered Set Abstraction of events introduced in this paper can be used to design more efficient

* Corresponding author.
*E-mail addresses:* spomares@inaoep.mx (S.E. Pomares Hernandez), jrpc@inaoep.mx (J.R. Perez Cruz), michel.raynal@irisa.fr (M. Raynal).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.02.015

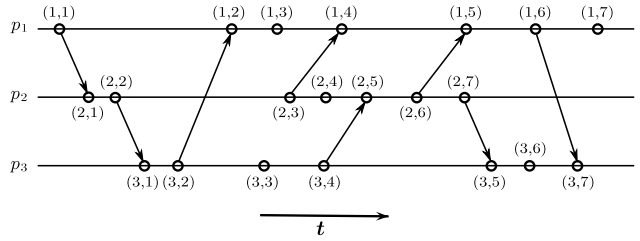


**Fig. 1.** A distributed computation scenario.

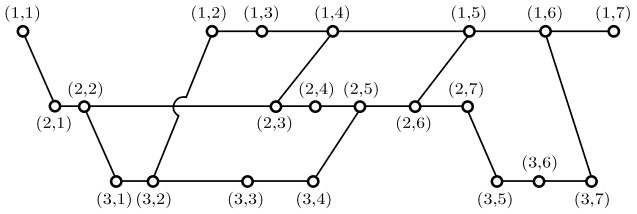


**Fig. 2.** Graph of the IDR of the scenario in Fig. 1.

algorithms based on the HBR principle. This assertion is based on two main properties. The first property is that CAOS attains a consistent compact representation of a distributed computation. The second property is that, as a consequence of the ordering of the events in the resulting sets, it is sufficient to check a single pair of events (one from each set) in order to determine whether these sets are causally or concurrently related regardless of the cardinality of the sets.

This paper proceeds as follows. Section 2 presents the system model and associated definitions. Section 3 introduces the CAOS abstraction. Finally, Section 4 concludes with a few remarks.

## 2. Preliminaries

### 2.1. System model

*Processes.* The system under consideration (see Fig. 1) is composed of a set of processes $P = \{p_1, p_2, \ldots, p_n\}$. The processes present an asynchronous execution and communicate only by message passing.

*Messages.* We consider a finite set of messages $M$, where each message $m \in M$ is sent considering an asynchronous reliable network that is characterized by no transmission time boundaries, no order delivery, and no loss of messages. The set of destinations of a message $m$ is identified by $Dest(m)$.

*Events.* We consider two types of events: internal and external events. An *internal* event is a unique action that occurs at a process $p$ in a local manner and which changes only the local process state. We denote the finite set of internal events as $I$. On the other hand, while an external event is also a unique action that occurs at a process, it is seen by other processes, thus, affecting the global state of the system. The external events considered in this paper are the *send* and *delivery* events. Let $m$ be a message. We denote by $send(m)$ the emission event and by $delivery(p, m)$ the delivery event of $m$ to participant $p \in P$. The set of events associated to $M$ is the set $E(M) = \{send(m) : m \in M\} \cup \{delivery(p, m) : m \in M \wedge p \in P\}$. The whole set of events in the system is the finite set $E = I \cup E(M)$. Each event $e \in E$ is identified by a tuple $id(e) = (p, x)$, where $p \in P$ is the producer of $e$, and $x$ is the local logical clock for events of $p$, when $e$ is carried out. When we need to refer to a specific event we use the notation $e_{p,x}$ or simply $(p, x)$.

### 2.2. Background and definitions

*The Happened-Before Relation for single events (HBR).* The HBR was defined by Lamport [6]. This relation establishes causal precedence dependencies over a set of events. The HBR is a strict partial order (i.e. transitive, irreflexive and antisymmetric) defined as follows:

**Definition 1.** The causal relation "$\rightarrow$" is the smallest relation on a set of events $E$ satisfying the following conditions:

1. If $a$ and $b$ are events belonging to the same process, and $a$ was originated before $b$, then $a \rightarrow b$.
2. If $a$ is the sending of a message by one process, and $b$ is the receipt of the same message in another process, then $a \rightarrow b$.
3. If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.

By using Definition 1, Lamport defines that a pair of events is concurrently related "$a \| b$" if it satisfies the following condition:

$$a \| b \quad \text{if } \neg(a \rightarrow b \vee b \rightarrow a).$$

The poset $\widehat{E} = (E, \rightarrow)$ constitutes the formal model adopted in this paper for a distributed computation.

*The Immediate Dependency Relation (IDR).* The IDR is the *transitive reduction* of the HBR [1,11]. We denote it by "$\downarrow$", and it is defined as follows:

**Definition 2.** Two events $a, b \in E$ have an immediate dependency relation "$a \downarrow b$" if the following restriction is satisfied.

$$a \downarrow b \quad \text{if } a \rightarrow b \text{ and } \forall c \in E, \neg(a \rightarrow c \rightarrow b).$$

Thus, an event $a$ causal-immediately precedes an event $b$, if and only if no other event $c$ belonging to $E$ exists, such that $c$ belongs to the causal future of $a$ and to the causal past of $b$.

The graph for the IDR of the scenario in Fig. 1 is shown in Fig. 2. The partial order of events is established from left to right.

Based on the IDR, we now present the following property.

**Property 1.** *For all pair of events $a, b \in E$, $a \neq b$*

*if* $\exists c \in E$ *such that* $(a \downarrow c$ *and* $b \downarrow c)$ *or*
$(c \downarrow a$ *and* $c \downarrow b)$ *then* $a \| b$.

This means that for every pair of events $a, b \in E$ with common IDR dependencies, these events are concurrently related.

*The Happened-Before Relation for sets of events.* Lamport in [7] introduced two abstract precedence relations for sets called *precedes*, and *can affect* relation. In the present work we are only interested in the precedes relation. We refer to it indistinctly as HBR or causal relation for sets. The HBR for sets is also a strict partial ordering. We formally define it as follows:

**Definition 3.** The causal relation "$\rightarrow$" is established at the set level by satisfying the following conditions:

1. $A \rightarrow B$ if $a \rightarrow b, \forall (a, b) \in A \times B$,
2. $A \rightarrow B$ if $\exists C \mid (A \rightarrow C \wedge C \rightarrow B)$.

However, according to the specification of ordered sets presented by Shimamura et al. [12] and Pomares et al. [10], which assume a local total ordering among the events that compose a set, the causal relation for sets can be accomplished only in terms of the endpoints as follows:

**Property 2.** *The relation "$\rightarrow$" is accomplished at the ordered set level if the following conditions are satisfied:*

1. $A \rightarrow B$ *if* $a^+ \rightarrow b^-$,
2. $A \rightarrow B$ *if* $\exists C \mid (a^+ \rightarrow c^- \wedge c^+ \rightarrow b^-)$

**Table 1**
Causal set abstraction.

| | | | |
|---|---|---|---|
| I. A new set $W(e)$ is created in $S$ when: | | | |
| C1. $\exists e \in E, \neg(\exists Z \in S : e \in Z)$ | | | 1 |
| $W(e)$ | $\leftarrow$ | R1. $\{e : \emptyset \downarrow e\}$ or | 2 |
| | | R2. $\{e : \exists(e_a, e_b) \in E$ such that $e_a \downarrow e$ and $e_a \downarrow e_b\}$ or | 3 |
| | | R3. $\{e : \exists(e_a, e_b), \in E, e_a \neq e_b$ such that $e_a \downarrow e$ and $e_b \downarrow e\}$. | 4 |
| II. The rest of the events $e' \in E$ are assigned to a set $W(e) \in S$ as follows: | | | |
| C2. $\exists W(e) \in S, \exists e' \in E, \neg(\exists Z \in S : e' \in Z)$ | | | 5 |
| $W(e)$ | $\leftarrow$ | R4. $W(e) \cup \{e' : \exists e_a \in W(e), \exists e_b \in E$ such that $e_a \downarrow e'$ and $\neg(e_a \downarrow e_b)\}$. | 6 |

where $a^+$ and $b^-$ are the right and the left endpoints of $A$ and $B$, respectively; $c^-$ and $c^+$ are the endpoints of $C$.

Finally, we present the concurrent relation for ordered sets, which is defined as follows:

**Definition 4.** Two ordered sets of events, $A$ and $B$, are said to be concurrently related "$A \| B$" if the following condition is satisfied:

1. $A \| B$ if $a \| b, \forall a \in A, \forall b \in B$.

We note that for our purpose, we assume that $a \| b$ presents the same behavior as $b \| a$.

## 3. The Causal Ordered Set Abstraction (CAOS)

Assuming the poset $\widehat{E} = (E, \rightarrow)$ as the model adopted for a distributed computation, the objective of CAOS is to establish over this poset the rules and conditions of association of events and the conditions of ordering between the resulting sets. For this, we begin by defining the composition of an ordered set considered in this work.

*Ordered sets of events.* We consider a finite collection $S$ of sets of events, where each set $W \in S$ is a set of events $W \subseteq E$. The elements of a set are ordered according to the IDR (See Definition 2). Such elements compose a causal path (linearization) from an event $e_1$ to an event $e_k$ such that $W = \{e_1 \downarrow e_2 \downarrow \cdots \downarrow e_k\}$. We denote by $w^-$ and $w^+$ the endpoint events of $W$ ($w^- = e_1$ and $w^+ = e_k$). When $|W| = 1$, we have that $w^- = w^+$ and we denote it simply by $w$. Each set $W \in S$ is identified by the tuple $id(W) = (w^-, w^+)$. When we need to refer to a specific set, we use $W(w^-, w^+)$ or the short notation $W(w^-)$ according to the context.

### 3.1. CAOS specification

The definition of CAOS is made up of three parts. The first part specifies the conditions and rules for the identification/creation of new sets to be included in the collection $S$. The second part specifies the conditions and rules to associate events to an existing set in $S$. The third part specifies the arrangement of sets by establishing the conditions of order between sets. All the rules of parts one and two (lines 2–4 and 6, Table 1) are based on the IDR (See Definition 2) and particularly the rules R2–R4 make use of Property 1. According to the conditions C1 and C2 (Lines 1 and 5, Table 1), an event $e \in E$ can be only associated to one set $W \in S$; this means that for every pair of resultant sets $X, Y \in S$ it implies that $X \bigcap Y = \emptyset$. The events are verified from left to right according to the partial order established by the IDR (See Fig. 2). For each event, the conditions and rules of Table 1 are verified from top to bottom.

*Part I—Creation of sets.* The rules R1–R3 establish the creation of sets (See lines 2–4, Table 1). An event that satisfies one of these rules creates a new set, and it is by default associated to such set as its left endpoint. Rule R1 creates a new set $W(e)$ when an event $e$ does not have causal history. Rule R2 creates a new set $W(e)$ when it is detected that $e$ is concurrent with respect to another event $e_b$. Only as a reminder of Property 1, $e_a \downarrow e$ and $e_a \downarrow e_b$ implies $e \| e_b$. R2 ensures that when the pattern $e_a \downarrow (e \| e_b)$ occurs, the event $e$ will be associated to a different set $W$ than the sets for $e_a$ and $e_b$. Finally, rule R3 creates a new set $W(e)$ when it is detected that two concurrent events $e_a \| e_b$ converge to a same event $e$. The concurrency of events is also determined by Property 1. R3 ensures that when the pattern $(e_a \| e_b) \downarrow e$ occurs, the event $e$ will be associated to a different set $W$ than the sets for $e_a$ and $e_b$.

*Part II—Association of events.* It is important to note that if an event does not accomplish any of the rules of Part I, then it will be associated to an existing set $W$ according to R4 (See Line 6, Table 1). R4 associates events to sets by respecting the specification of the ordered set of events previously presented where the elements of a set compose a linearization based on the IDR. Each new event associated to a set $W$ becomes the right endpoint of the linearization represented by $W$.

*Part III—Arrangement of sets.* The resulting sets $W \in S$ are arranged according to the IDR. We say that a pair of sets $X, Y \in S$ are IDR related "$X \downarrow Y$" if the following condition is satisfied:

$X \downarrow Y$ if $x^+ \downarrow y^-$.

After the specification of the creation of sets, association of events, and arrangement of sets, for every pair of sets $X, Y \in S$, one of the following conditions is satisfied:

- $X \rightarrow Y$ if $x^+ \rightarrow y^-$, or
- $Y \rightarrow X$ if $y^+ \rightarrow x^-$, or
- $X \| Y$ if $\neg(x^+ \rightarrow y^- \vee y^- \rightarrow x^+)$.

This means that for any pair $X, Y \in S$, according to Definitions 3 and 4, the pair can be only causally or concurrently related. Definitions 3 and 4 say that all the events of a set $X$, are causally or concurrently related with respect to all the events of a set $Y$, respectively. The importance of this result is that it is possible to determine how two sets $X$ and $Y$ in $S$ are related by only verifying their right and their left endpoints, regardless of the cardinality of the sets. The proof is given in Section 3.4.

### 3.2. Scenario example

After applying the CAOS abstraction to the scenario in Fig. 1 which consider the verification of each event from left to right according to the IDR, and by following the rules from R1 to R4, twelve sets have been obtained (See Table 2). It is important to note that an event can satisfy more than one rule of Part I. For example, the event (1, 4) satisfies R2 and R3. The ambiguity of rule for this case is avoided by following the order of verification from R1 to R4; the events such as (1, 4) will always be taken by R2. The important aspect ensured by CAOS is that for each event in $E$, the events will be associated only once, and this is achieved by C1 and C2.

The arrangement of the sets is presented in Fig. 3. The partial order at the set level is established from left to right. In such arrangement, an arc is established between a pair of sets $X, Y \in S$ if they are IDR related $X \downarrow Y$. For example, $W(e_{3,1}, e_{3,2})$ is IDR related with $W(e_{1,2}, e_{1,3})$ and $W(e_{3,3}, e_{3,4})$, since we have $e_{3,2} \downarrow e_{1,2}$ and $e_{3,2} \downarrow e_{3,3}$.

**Table 2**
Resultant sets of scenario in Fig. 1.

| | |
|---|---|
| $W(e_{1,1}) = \{(1, 1), (2, 1), (2, 2)\}$, | $W(e_{3,3}) = \{(3, 3), (3, 4)\}$, |
| $W(e_{3,1}) = \{(3, 1), (3, 2)\}$, | $W(e_{2,5}) = \{(2, 5), (2, 6)\}$, |
| $W(e_{1,2}) = \{(1, 2), (1, 3)\}$, | $W(e_{1,5}) = \{(1, 5), (1, 6)\}$, |
| $W(e_{2,3}) = \{(2, 3)\}$, | $W(e_{2,7}) = \{(2, 7), (3, 5), (3, 6)\}$, |
| $W(e_{1,4}) = \{(1, 4)\}$, | $W(e_{1,7}) = \{(1, 7)\}$, |
| $W(e_{2,4}) = \{(2, 4)\}$, | $W(e_{3,7}) = \{(3, 7)\}$. |

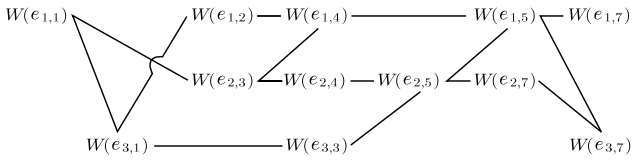


**Fig. 3.** Graph of the causal set abstraction of the scenario in Fig. 1.

### 3.3. Discussion about CAOS

Where can CAOS be useful? Let us first consider the following properties provided by CAOS. The CAOS sets define a poset $\widehat{S} = (S, \rightarrow)$ that captures the strict causal partial order on sets of events. This means that $\widehat{S}$ is irreflexive, antisymmetric and transitive. Based on these partial order properties, the principles developed for the HBR relation naturally extend and can be easily adapted to work at the causal set level. For example, IDR, which is the minimal expression of the HBR for single events (see Definition 2), can be directly modified to express causal immediate dependency between sets. This is important because the extension of IDR to sets represents exactly the causality at the set level.

By taking these properties into consideration, the theory of *vector of clocks* [8] can be adapted to ensure the causal ordering on sets. An interesting approach to explore is to update the vector clocks according to the number of messages sent into the linearization of each causal set. The premise behind is that it will be less costly to ensure the causal ordering of single events at the set level than at the single event level since we do not need to verify all single events in the causal past.

Another domain where CAOS can be useful is in *model checking*. One main problem of model checking in asynchronous systems is the state explosion. The most successful techniques for dealing with this problem are based on *partial order reduction* [3]. The compact representation of a distributed computation achieved by CAOS can be exploited for the partial order reduction to reduce the state space to be explored. Specifically, the resultant graph can be adapted for testing state *reachability* and the identification of *independent transitions/actions*. We note that concurrent sets in CAOS represent independent sets of actions.

### 3.4. Proof

In this section we prove that for every pair of sets $X = \{x^- \downarrow \cdots \downarrow x^+\}, Y = \{y^- \downarrow \cdots \downarrow y^+\} \in S$, these sets are causally or concurrently related according to Definitions 3 and 4, respectively. The proof of the causal and concurrent relation between sets is done by only comparing the $x^+$ and $y^-$ endpoints. The causality of sets is proven by Theorem 1 and the concurrency of sets is proven by Theorem 2.

**Theorem 1.** *If $x^+ \in X, y^- \in Y, X, Y \in S$ such that $x^+ \rightarrow y^-$, then $a \rightarrow d, \forall (a, d) \in X \times Y$.*

**Proof.** We prove Theorem 1 by direct proof. We have that the events of the sets in $X, Y \in S$ compose a separate causal path between $x^-, x^+$ and $y^-, y^+$, respectively. Therefore, $\forall a \in X, a \neq x^+$ we have $a \rightarrow x^+$, since $x^+ \rightarrow y^-$ by transitivity $\forall a \in X$ implies $a \rightarrow y^-$. Finally, $\forall d \in Y, d \neq y^-$ implies $y^- \rightarrow d$, since the HBR is antisymmetric and transitive, we conclude that $\forall a \in X, \forall d \in Y$ implies $a \rightarrow d$. □

**Theorem 2.** *If $x^+ \in X, y^- \in Y, X, Y \in S$ such that $x^+ \parallel y^-$, then $a \parallel d, \forall a \in X, \forall d \in Y$.*

**Proof.** We prove Theorem 2 by contradiction. We begin by denying the conclusion statement as follows:

$$\exists a \in X, \quad \exists d \in Y \quad \text{such that } \neg(a \parallel d).$$

Replacing the concurrent relation by its definition the following sentence is obtained.

$$\neg(\neg(a \rightarrow d \vee d \rightarrow a)) \equiv a \rightarrow d \vee d \rightarrow a.$$

So we need to find

$$\exists a \in X, \quad \exists d \in Y \quad \text{such that } (a \rightarrow d \vee d \rightarrow a).$$

The proof is divided into two general cases. The first case considers $a = x^+$ and $d = y^-$. The second case considers two particular cases: when $a \neq x^+$ and when $d \neq y^-$.

*Case* 1. Let $a = x^+$ and $d = y^-$. We replace them in the conclusion statement and we obtain:

$$x^+ \in X, \quad y^- \in Y \quad \text{such that } x^+ \rightarrow y^- \vee y^- \rightarrow x^+.$$

This first conclusion statement is false because according to the HBR of Lamport, a pair of events $a, b \in (E, \rightarrow)$ can be causally or concurrently related but not both simultaneously. The conditional statement establishes that $x^+ \| y^-$, while the conclusion statement affirms that $x^+ \rightarrow y^- \vee y^- \rightarrow x^+$. Thus, this statement is false.

*Case* 2. When $a \neq x^+$, we search an $a \in X$ such that $a \rightarrow d$ and $a \rightarrow x^+$. And when $d \neq y^-$, we search an event $d \in Y$ such that $a \rightarrow d$ and $y^- \rightarrow d$.

*Case* 2.1. When $a \neq x^+$, we have again two cases. The first case is when $a \downarrow x^+$ and $a \downarrow d$. The second case considers that for $b, h \in X$ and an event $g \in Y$, the causal paths $(b \downarrow h \downarrow \cdots \downarrow x^+)$ and $(b \downarrow g \downarrow \cdots \downarrow d)$ exist, if $b \neq a$ we assume that $a \rightarrow b$.

(2.1.a) When $a \downarrow x^+$ and $a \downarrow d$, by applying R2 (See Line 3, Table 1), the events $a$ and $x^+$ are associated to different sets in $S$. This means that $a \in X, x^+ \in Z$: for some $Z \in S, X \neq Z$, and this statement is false.

(2.1.b) When $(b \downarrow h \downarrow \cdots \downarrow x^+)$ and $(b \downarrow g \downarrow \cdots \downarrow d)$, by applying R2 we have again $a \in X, x^+ \in Z$: for some $Z \in S, X \neq Z$ and this statement is once again false.

*Case* 2.2. When $d \neq y^-$ two other cases exist. The first case is when $y^- \downarrow d$ and $a \downarrow d$. The second case considers that for an event $h \in X$ and $c, g \in Y$ the causal paths $(a \downarrow \cdots \downarrow h \downarrow c)$ and $(y^- \downarrow \cdots \downarrow g \downarrow c)$ exist, if $c \neq d$ we assume that $c \rightarrow d$.

(2.2.a) When $a \downarrow d$ and $y^- \downarrow d$ by applying R3 (See Line 4, Table 1), the events $d$ and $y^-$ are associated to different sets in $S$. This means that $y^- \in Y, d \in Z$: for some $Z \in S, Y \neq Z$, and this statement is false.

(2.2.b) When $(a \downarrow \cdots \downarrow h \downarrow c)$ and $(y^- \downarrow \cdots \downarrow g \downarrow c)$ by applying R3 we have again $y^- \in Y, d \in Z$: for some $Z \in S, Y \neq Z$ and this statement is once again false. □

## 4. Conclusions

The Causal Ordered Set Abstraction (CAOS) has been presented. Assuming the poset $\widehat{E} = (E, \rightarrow)$ as the model of a distributed computation, CAOS establishes over this poset the rules for the association of events to compose sets while also establishing the conditions of ordering among the resultant sets. CAOS attains a compact and consistent representation of a causal distributed computation. The resultant sets in CAOS are only causally or concurrently related. We proved that such relation between sets can be determined by comparing exclusively their endpoints, regardless of the cardinality of the sets involved.

## References


[1] E. Anceaume, J.M. Helary, M. Raynal, A note on the determination of the immediate predecessors in a distributed computation, International Journal of Foundations of Computer Science 13 (6) (2002) 865–872.
[2] P. Chandra, A.D. Kshemkalyani, Data-stream-based global event monitoring using pairwise interactions, Journal of Parallel and Distributed Computing 68 (6) (2008) 729–751.
[3] E.M Clarke, O. Grumberg, M. Minea, D. Peled, State space reduction using partial order techniques, International Journal on Software Tools for Technology Transfer (STTT) 2 (3) (1999) 279–287.
[4] J.M. Helary, A. Mostefaoui, M. Raynal, Interval consistency of asynchronous distributed computations, Journal of Computer and System Science 64 (2002) 329–349.
[5] J.M. Helary, R.H.B. Netzer, M. Raynal, Consistency issues in distributed checkpoints, IEEE Transactions on Software Engineering 25 (2) (1999) 274–281.
[6] L. Lamport, Time, clocks and the ordering of events in distributed systems, Communications ACM 21 (7) (1978) 558–565.
[7] L. Lamport, On interprocess communications: I. basic formalism, Distributed Computing 1 (2) (1986) 77–85.
[8] F. Mattern, Virtual time and global states of distributed systems, in: Proceedings of International Workshop on Parallel and Distributed Algorithms, 1989, pp. 215–226.
[9] R.H.B. Netzer, J. Xu, Necessary and sufficient conditions for consistent global snapshots, IEEE Transactions on Parallel Distribributed Systems 6 (2) (1995) 165–169.
[10] S.E. Pomares Hernandez, J. Estudillo Ramirez, L.A. Morales Rosales, G. Rodriguez Gomez, An intermedia synchronization mechanism for multimedia distributed system, International Journal of Internet Protocol Technology 4 (3) (2009) 207–218.
[11] S.E. Pomares Hernandez, J. Fanchon, K. Drira, The immediate dependency relation: an optimal way to ensure causal group communication, in: Annual Review of Scalable Computing, in: Ser. Scal. Compt., vol. 6, World Scientific, 2004, pp. 61–79.
[12] K. Shimamura, K. Tanaka, M. Takizawa, Group communication protocol for multimedia applications, in: Proceedings of the IEEE ICCNMC'01, 2001, pp. 303–308.


**S.E. Pomares Hernandez** is a Researcher in the Computer Science Department at the National Institute of Astrophysics, Optics and Electronics (INAOE), in Puebla, Mexico. He completed his PhD Degree at the Laboratory for Analysis and Architecture of Systems of CNRS, France in 2002. Since 1998, he has been researching in the field of distributed systems and partial order algorithms.

**J.R. Perez Cruz** is currently a PhD student in the Department of Computer Science at the INAOE. He obtained the M.Sc. degree in Computer Science from the same institute in 2009. His research interests include distributed systems, partial order algorithms, sensor networks and secure group communications. His postgraduate studies are supported by the National Council of Science and Technology of Mexico (CONACYT).

**M. Raynal** is a full time professor at IRISA, University of Rennes, France. His main research interest lies in distributed computing. He has published more than 125 papers in journals and more than 250 in conferences. His h-index is 45. He has recently written two books, both published by Morgan & Claypool Publishers: "*Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Distributed Systems*" 251 pages, 2010 (ISBN 978-1-60845-293-4) and "*Fault-Tolerant Agreement in Synchronous Message-Passing Systems*" 165 pages, 2010 (ISBN 978-1-60845-525-6).