A Fine-Grained Modality Classification for Global Predicates

Ajay D. Kshemkalyani, *Senior Member*, *IEEE*

Abstract—Specifying and detecting predicates in a distributed execution is an important problem. Distributed execution observation has classically used two modalities—$Possibly(\phi)$ and $Definitely(\phi)$—for predicate ϕ. Based on the temporal interactions of intervals, the author identified a complete, *orthogonal* set of relationships $\Re$ between pairs of intervals in a distributed execution. This paper shows how to map the rich, orthogonal classification of modalities of pairwise interval interactions, to the classical coarse-grained classification, $Possibly(\phi)$ and $Definitely(\phi)$, for specifying predicates defined on any number of processes. This increases the power of expressing the temporal modalities under which predicates can be specified, beyond the current $Possibly/Definitely$ classification. This paper also gives timestamp-based tests for the orthogonal modalities in the refined classification.

Index Terms—Predicates, causality, synchronization, distributed execution, distributed system, global state.

1 Introduction

Predicate specification and detection is an important problem in distributed systems. It is useful in specifying synchronization and coordination conditions in an execution in areas such as sensor networks, distributed debugging, and industrial process control. While stable predicates (once a stable predicate becomes true, it remains true) can be detected by techniques such as global snapshot algorithms, unstable predicates are difficult to deal with because snapshot algorithms cannot capture such predicates. The main difficulties in specifying and detecting unstable predicates in a distributed execution are the following: 1) The asynchronous nature of the process execution speeds and message delivery times leads to many interleavings of the events at different processes, each of which leads to a different sequence of global states corresponding to a different observation of the same execution. 2) A global observation of the execution across different processes is inherently not possible in a distributed system.

A *global predicate* is defined on variables local to more than one process. Two modalities, $Possibly(\phi)$ and $Definitely(\phi)$, for the satisfaction of a global predicate ϕ in a distributed execution were defined by Cooper and Marzullo [3] and Marzullo and Neiger [15]. For a given execution, 1) $Possibly(\phi)$ is true if some observation of the execution may pass through a state in which ϕ is true, and 2) $Definitely(\phi)$ is true if every observation of the execution will pass through a state in which ϕ is true. These two modalities have become widely accepted. The modality classification—$Possibly$ and $Definitely$—is coarse-grained. The formalism and axiom system given by Kshemkalyani [9] identified a complete *orthogonal* set $\Re$ of 40 fine-grained temporal interactions between intervals in a distributed execution. Here, any pair of intervals must be related by one and only one of the mutually exclusive interaction types identified in $\Re$. We show that this formalism provides more expressive power than the *Possibly* and *Definitely* modalities of [3] and [15] for specifying predicates. Specifically, we show how to map the rich, orthogonal classification of modalities of pairwise interval interactions to the classical coarse-grained classification, $Possibly(\phi)$ and $Definitely(\phi)$, for specifying predicates defined on any number of processes. This gives flexibility and power to monitor, synchronize, and control distributed executions. We also give timestamp-based tests to detect the modality of predicates defined on variables local to a pair of processes—these tests can then be applied to all pairs of processes to detect global predicates defined on variables local to more than two processes.

The power of our approach stems from the fact that we model intervals as in [9], as opposed to individual events as in [3], [15], in the distributed execution. The *intervals* at each process are the local time durations of interest. Specifically, in each *interval* at a process, the local variables using which the global predicate ϕ is defined, can potentially satisfy ϕ. Note that the notion of an interval corresponds to an *abstract event* or a *nonatomic event* as used by Lamport who argued that "it is useful to assume that primitive elements between which concurrency is modeled are nonatomic" [14]. A nonatomic event naturally models the execution of a code fragment by a process and is an important concept in reasoning about the interaction between multiple code fragments executed by different processes [11].

Section 2 gives the system model and reviews background material. Section 3 shows the refinement of the $Possibly(\phi)$ and $Definitely(\phi)$ classification in terms of the 40 possible interaction types $\Re$ [9]. Section 4 gives the timestamp-based tests to check for the 40 orthogonal relations. Section 5 discusses the significance and some applications of the fine-grained modality classification. Section 6 gives the conclusions.

• *The author is with the Department of Computer Science, University of Illinois at Chicago, 851 South Morgan Street, Chicago, IL 60607. E-mail: ajayk@cs.uic.edu.*

Manuscript received 23 July 2001; revised 23 Aug. 2002; accepted 13 Nov. 2002.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 114585.

TABLE 1
Some Relations for Interactions between Intervals X and Y [9]

Relation r	Expression for $r(X, Y)$
R1 (strong precedence)	$\forall x \in X \forall y \in Y, x \prec y$
R2 (partially strong precedence)	$\forall x \in X \exists y \in Y, x \prec y$
R3 (partially weak precedence)	$\exists x \in X \forall y \in Y, x \prec y$
R4 (weak precedence)	$\exists x \in X \exists y \in Y, x \prec y$
S1 (event with no coupling)	$\exists x \in X \forall y \in Y, x \not\preceq y \bigwedge y \not\preceq x$
S2 (round-trip coupling)	$\exists x1, x2 \in X \exists y \in Y, x1 \prec y \prec x2$

2 System Model and Background

2.1 System Model

In a distributed system, a set of processes N communicate with one another by asynchronous message passing over logical channels. There is no shared memory and no common clock. A poset event structure model $(E, \prec)$, where $\prec$ is an irreflexive partial ordering representing the causality relation on the event set E, is used as the model for a distributed system execution. E is partitioned into local executions at each process. E_i is the linearly ordered set of events executed by process P_i. An event executed by P_i is denoted e_i. The causality relation on E is the transitive closure of the local ordering relation on each E_i and the ordering imposed by message send events and message receive events [13]. A subset of set E_i identifies a duration or an *interval* at P_i that is demarcated by the earliest and latest events. Such intervals are *linear intervals* and denote *linear abstract events*. As stated earlier, our intervals are such that in each *interval* at a process, the local variables using which the global predicate ϕ is defined, can potentially satisfy ϕ. Further semantics of intervals of interest are discussed in Section 3. A more general definition of intervals, and an extension of the definition to a space-time setting, and to a collection of arbitrary, possibly noncontiguous events of interest from different processes, are given in [9], [11].

A *cut* C is a subset of E such that if $e_i \in C$, then $(\forall e_i')\ e_i' \prec e_i \Longrightarrow e_i' \in C$. Thus, a cut is downward-closed within partitions. A *consistent cut* C is a downward-closed subset of E in $(E, \prec)$ [17], i.e., if $e \in C$, then $(\forall e')\ e' \prec e \Longrightarrow e' \in C$. Only all downward-closed subsets of E preserve causality and denote observable execution prefixes. For event e, there are two special consistent cuts $\downarrow e$ and $e \uparrow$. $\downarrow e$ is the maximal set of events that happen before e. $e \uparrow$ is the set of all events up to and including the earliest events at each process for which e happens before the events.

Definition 1 (Past of an event e). $\downarrow e = \{e' | e' \prec e\}$

(Future of an event e). $e \uparrow = \{e' | e' \not\succeq e\} \bigcup \{e_i, i = 1, \ldots, |N| \mid e_i \succeq e \bigwedge (\forall e_i' \prec e_i,\ e_i' \not\succeq e)\}$.

The system state after the events in a cut is a global state [2]; if the cut is consistent, the corresponding system state is termed a *consistent global state* and denotes a meaningful observation of a global state.

We assume the availability of vector clocks Clk to maintain logical time [5], [17]. Vector clocks have the property that $e \prec f \Longleftrightarrow Clk(e) < Clk(f)$. A vector clock has size $n = |N|$, with the $<$ operator defined as follows: 1) $Clk \leq Clk'$ if and only if $(\forall k \in N)\ Clk[k] \leq Clk'[k]$, and 2) $Clk < Clk'$ if and only if $Clk \leq Clk'$ and $Clk \neq Clk'$. The following rules update the clock at each process:

V1. (Internal event) Before process P_i executes the event, $Clk_i[i] = Clk_i[i] + 1$.

V2. (Send event) Before process P_i executes the event, $Clk_i[i] = Clk_i[i] + 1$. Send message timestamped with Clk_i.

V3. (Receive event) When process P_j receives a message timestamped T from process P_i, $(\forall k \in N)\ Clk_j[k] = \max(Clk_j[k], T[k])$; $Clk_j[j] = Clk_j[j] + 1$; deliver the message.

The timestamp T of an event e_i is the local clock value $Clk(e_i)$ that results after the execution of the event. Unless otherwise indicated, we do not assume any restriction on the predicates defined on the execution. Given a literal/Boolean b, $\overline{b}$ denotes its complement. A variable x local to process P_i is denoted as x_i. To simplify notation, the process set N is also represented as the set of integer process indices. For example, we use "$(\forall i \in N)$" instead of "$(\forall P_i \in N)$." Such usage should be clear from the context.

2.2 Prior Work

For abstract events X and Y, Lamport defined the following two relations [14]:

1. $\longrightarrow$: $X \longrightarrow Y$ iff $\forall x \in X, \forall y \in Y, x \prec y$.
2. $--\rightarrow$: $X --\rightarrow Y$ iff $\exists x \in X, \exists y \in Y, x \prec y$.

Four additional relations besides $\longrightarrow$ and $--\rightarrow$ were defined in [9] to capture the interaction between a pair of intervals or abstract events. Henceforth, we will refer to relations $\longrightarrow$ and $--\rightarrow$ by R1 and R4, respectively. Relations R1 (strong precedence), R2 (partially strong precedence), R3 (partially weak precedence), and R4 (weak precedence) define *causality conditions*, whereas S1 (event with no coupling) and S2 (round-trip coupling) define *coupling conditions*. The relations R1-R4 and S1-S2 are expressed in Table 1.

Axiom 1 gives the density axiom for time in terms of the events/time instants in an interval. It states that any interval X is defined over an infinite dense set E, and each X that is not a single-member set contains ∞ number of elements.

Axiom 1 (Density Axiom). *Each interval X is dense, i.e.,* $\forall x1, x2 \in X, x1 \prec x2 \Longrightarrow \exists x3 \in X \mid x1 \prec x3 \prec x2$.

TABLE 2
The 40 Orthogonal Relations in $\Re$, between Intervals X and Y [9]

Interaction Type	Relation $r(X,Y)$						Relation $r(Y,X)$					
	R1	R2	R3	R4	S1	S2	R1	R2	R3	R4	S1	S2
IA $(=IQ^{-1})$	*1*	1	1	1	0	0	0	0	0	0	0	0
IB $(=IR^{-1})$	*0*	*1*	*1*	1	0	0	0	0	0	*0*	0	0
IC $(=IV^{-1})$	0	*0*	*1*	1	1	0	0	0	0	*0*	0	0
ID $(=IX^{-1})$	0	0	*1*	1	*1*	1	0	*1*	0	1	0	0
ID' $(=IU^{-1})$	0	0	*1*	1	*0*	1	0	*1*	0	1	0	*1*
IE $(=IW^{-1})$	0	*0*	*1*	1	*1*	*1*	0	*0*	0	*1*	0	0
IE' $(=IT^{-1})$	0	*0*	*1*	1	*0*	1	0	*0*	0	*1*	0	*1*
IF $(=IS^{-1})$	0	*1*	*1*	1	0	1	0	0	0	*1*	0	1
IG $(=IG^{-1})$	0	0	0	*0*	1	0	0	0	0	*0*	1	0
IH $(=IK^{-1})$	0	*0*	*0*	*1*	1	0	0	0	0	*0*	1	0
II $(=IJ^{-1})$	0	*1*	*0*	1	0	0	0	0	0	*0*	1	0
IL $(=IO^{-1})$	0	0	*0*	*1*	*1*	1	0	*1*	*0*	1	0	0
IL' $(=IP^{-1})$	0	0	*0*	*1*	*0*	1	0	*1*	*0*	1	0	*1*
IM $(=IM^{-1})$	0	*0*	*0*	*1*	*1*	*0*	0	*0*	*0*	*1*	*1*	0
IN $(=IM'^{-1})$	0	*0*	*0*	*1*	*1*	*1*	0	*0*	*0*	*1*	0	0
IN' $(=IN'^{-1})$	0	*0*	*0*	*1*	*0*	*1*	0	*0*	*0*	*1*	*0*	*1*
ID'' $(=(IUX)^{-1})$	0	0	*1*	1	*0*	1	0	*1*	0	1	0	*0*
IE'' $(=(ITW)^{-1})$	0	*0*	*1*	1	*0*	*1*	0	*0*	0	*1*	0	*0*
IL'' $(=(IOP)^{-1})$	0	0	*0*	*1*	*0*	1	0	*1*	*0*	1	0	*0*
IM'' $(=(IMN)^{-1})$	0	*0*	*0*	*1*	*0*	*0*	0	*0*	*0*	*1*	*1*	*0*
IN'' $(=(IMN')^{-1})$	0	*0*	*0*	*1*	*0*	*1*	0	*0*	*0*	*1*	0	*0*
IMN'' $(=(IMN'')^{-1})$	0	*0*	*0*	*1*	*0*	*0*	0	*0*	*0*	*1*	*0*	*0*

The upper part of the table gives the 29 relations assuming dense time. The lower part of the table gives 11 additional relations assuming nondense time. For each independent relation, the dominating values (prime implication cover) of the dependent relations in $\Re$ are in italics [10].

Assuming that time is dense, it was shown in [9] that there are 29 possible interaction types between a pair of intervals, as given in the upper part of Table 2. The significance of the second part of the table, as well as of the use of the italic type, will be explained later. Of the 29 interactions, there are 13 pairs of inverses, while three are inverses of themselves. The 29 interaction types are specified using Boolean vectors. For intervals X and Y, the six relations R1-R4 and S1-S2 form a Boolean vector of length 12, (six bits for $r(X,Y)$ and six bits for $r(Y,X)$). An example of how to interpret Table 2 follows. Consider interaction IB and its inverse IR. We have $IB(X,Y) = IR(Y,X)$. The vector components for interaction $IB(X,Y)$ are

$$\overline{R1}(X,Y), R2(X,Y), R3(X,Y), R4(X,Y), \overline{S1}(X,Y),$$
$$\overline{S2}(X,Y), \overline{R1}(Y,X), \overline{R2}(Y,X), \overline{R3}(Y,X), \overline{R4}(Y,X),$$
$$\overline{S1}(Y,X), \overline{S2}(Y,X).$$

Analogously, the vector components for $IR(X,Y)$ are

$$\overline{R1}(X,Y), \overline{R2}(X,Y), \overline{R3}(X,Y), \overline{R4}(X,Y), \overline{S1}(X,Y),$$
$$\overline{S2}(X,Y), \overline{R1}(Y,X), R2(Y,X), R3(Y,X), R4(Y,X),$$
$$\overline{S1}(Y,X), \overline{S2}(Y,X).$$

Note that with the dense time assumption, R1 to R4, and either S1 or S2 are sufficient to determine the interaction type.

Figs. 1 and 2 (reproduced from [9]) illustrate the complete set of 29 orthogonal relations for dense time. In Fig. 1, interval X is shown in a fixed position using a rectangular box, whereas interval Y, indicated using horizontal lines, is in different positions relative to X. $\min(X)$ and $\max(X)$ denote the least and greatest element of linear interval X, respectively. $\downarrow \min(X)$, $\downarrow \max(X)$, $\min(X) \uparrow$, and $\max(X) \uparrow$ denote consistent cuts. The different types of interactions are identified by the various positions of Y relative to X. Each position of Y is labeled by an interaction type, IA through IX (and their modifiers). Five positions of Y have two labels each. For each of these five positions of Y, the distinction between the two interaction types, represented by the two labels, is

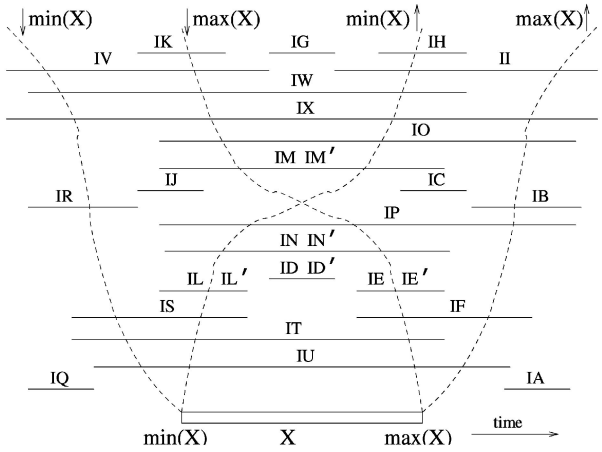

Fig. 1. Illustration of interaction types between intervals under the dense time model [9].

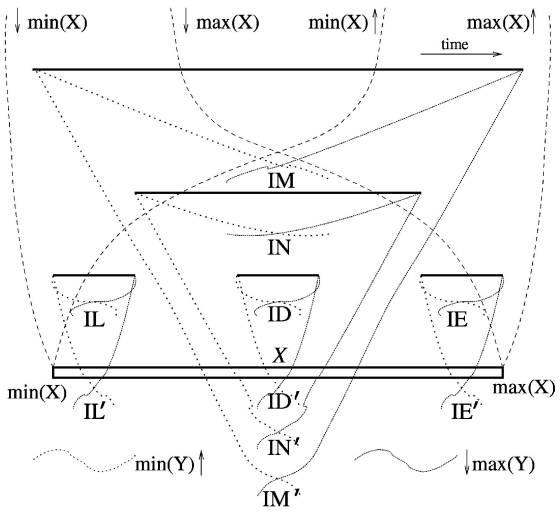

Fig. 2. Ilustration of interaction types between intervals under the dense time model (cont.) [9].

illustrated using Fig. 2. In this figure, the positions of Y are indicated using a thick horizontal line.

When Axiom 1 does not hold, the intervals may be nondense. This model is significant because clocks which measure dense linear time use a nondense linear scale in practice. Additionally, in some environments, local predicates may meaningfully be defined to hold only when a discrete event occurs. This model is also significant because actions at each node in a distributed system form a linear sequence of discrete events. This model permits 11 interaction types between a pair of intervals, defined in the lower part of Table 2, in addition to the 29 identified before. Of these, there are five pairs of inverses, while one is its own inverse. The reader is requested to refer to [9] for pictorial illustrations and further explanations of these interaction types.

3 Refining the Possibly(ϕ) / Definitely(ϕ) Classification

3.1 Background and Approach

Specifying predicates on the system state provides an important handle to specify and detect the expected behavior of the system. Unstable predicates pose two challenges for detection in a system execution, as observed in [16]. 1) The predicate may no longer be true at the time the predicate is evaluated by a monitor. 2) Even if a predicate is found to be true, it still may not have ever held during the actual run. In particular, snapshot algorithms are not useful to detect unstable predicates because even if a predicate is detected, it may never have held, and the predicate may have held even if it is not detected. Evaluating nonstable predicates over global states constructed from observations of runs has the same drawbacks. To overcome these drawbacks, Cooper and Marzullo [3], Marzullo and Neiger [15] proposed to extend predicates to apply to the entire distributed execution rather than to individual observations or global states of it. They defined:

- $Possibly(\phi)$: There exists a consistent observation of the execution such that ϕ holds in a global state of the observation.

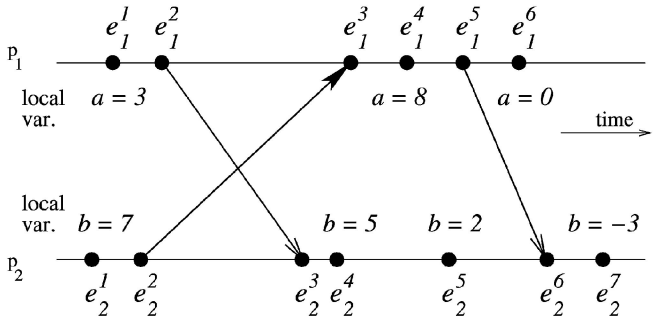

Fig. 3. An example execution.

- $Definitely(\phi)$: For every consistent observation of the execution, there exists a global state of it in which ϕ holds.

Consider the example in Fig. 3. The execution is run at processes P_1 and P_2. Event e^k_i denotes the kth event at process P_i. Variable a is local to P_1 and variable b is local to P_2. Observe that $Definitely(a + b = 10)$ holds by the following reasoning: When b is assigned 7 at event e^1_2, process P_1's execution may be in any state from the initial state up to the state preceding event e^3_1, in which $a = 3$. However, before the value of b changes from 7 to 5 at event e^4_2 and, in fact, before P_2 executes event e^3_2, P_1 must have executed event e^1_1 at which time $a = 3$. This is true for all equivalent executions. Hence, $Definitely(a + b = 10)$ holds. Although predicate $(a + b = 5)$ is never true in the shown execution, observe that $Possibly(a + b = 5)$ holds by the following reasoning: The predicate $a + b = 5$ can be true only if: 1) $a = 3 \wedge b = 2$, 2) $a = 0 \wedge b = 5$, or 3) $a = 8 \wedge b = -3$, simultaneously in some equivalent execution. State (1) is possible in physical time after the occurrence of event e^5_2 and before the occurrence of e^4_1. In the execution shown, e^5_2 occurs after e^4_1. However, in an equivalent execution, event e^4_1 may be delayed to occur after event e^5_2, in which case b changes to a value other than 2 after a becomes 8. Hence, the predicate is true in this equivalent execution. It so happens that a similar argument also holds for (2) and (3).

Clearly, $Definitely(\phi)$ implies $Possibly(\phi)$, but the implication does not hold the other way unless ϕ is stable. We observe that for any predicate ϕ, three orthogonal modal possibilities hold: 1) $Definitely(\phi)$, 2) $\neg\, Definitely(\phi) \wedge Possibly(\phi)$, and 3) $\neg\, Possibly(\phi)$. Based on the formalism in [9], we now show that these three orthogonal modal possibilities can be refined into the exhaustive set of 29 (or 40) possibilities between each pair of intervals, based on whether time is assumed to be dense (or nondense), respectively.

The definitions of the modalities on predicates use system states. However, observe that the definitions essentially demarcate intervals at each process such that in each interval, the values of local variables using which ϕ is defined, may satisfy ϕ. In general, the semantics of the intervals depends on predicate ϕ. For a conjunctive predicate $\phi = \wedge_i \phi_i$, where ϕ_i is a conjunct (predicate) local to process P_i, each interval at P_i can be locally determined to be the maximum local duration in which *local predicate* ϕ_i is true. For a nonconjunctive predicate, the interval at a

TABLE 3
Refinement Mapping

	$Definitely(\phi)$	$Possibly(\phi) \wedge \neg Definitely(\phi)$	$\neg Possibly(\phi)$
Dense time	ID and IX ID' and IU IE and IW IE′ and IT IF and IS IO and IL IP and IL′ IM IM′ and IN IN′	IB and IR IC and IV IG IH and IK II and IJ	IA and IQ
Nondense time (Interpretation 1)		all except IA and IQ	IA and IQ
Nondense time (Interpretation 2) For the 29 relations of dense time model, – use same mappings as for dense time	all 11 new interactions for nondense time		
Nondense time (Interpretation 3)	ID and IX ID′ and IU IE and IW IE′ and IT IF and IS IO and IL IP and IL′ IM′ and IN IN′ ID″ and IUX IE″ and ITW IL″ and IOP IN″ and IMN′	IB and IR IC and IV IG IH and IK II and IJ IM IM″ and IMN IMN″	IA and IQ

The upper part shows the 29 mappings when the dense time model is assumed. For the nondense time model, the mappings under three possible interpretations of $Possibly$ and $Definitely$ are shown in the lower part.

process needs to be identified carefully, possibly after reexpressing ϕ, depending on the structure of ϕ, and examining the state lattice for the execution. Whatever semantics is used to identify the intervals, the fine-grained modalities apply.

3.2 Predicate ϕ Defined over Two Processes

Let us first consider the case when predicate ϕ is defined on variables that are local to two processes. For a large class of applications, such as those using a pair of interacting agents or a client-server interaction, predicates are defined on variables local to two processes.

For the dense time model, we can formulate an alternate definition of $Possibly$ and $Definitely$, based on linear extensions of partial orders. For the poset $(E, \prec)$, its linear extension is a total order that preserves the partial order. Each linear extension represents a consistent observation of the execution.

Definition 2. *Assume the dense time model. Let intervals X and Y at two processes be such that during these intervals, the local predicates are true.*

- *$Possibly(\phi)$: (For some such intervals X and Y,) there exists a linear extension of $(E, \prec)$ such that the string $[min(X), max(X)]$ overlaps with the string $[min(Y), max(Y)]$.*
- *$Definitely(\phi)$: (For some such intervals X and Y,) for every linear extension of $(E, \prec)$, the string $[min(X), -max(X)]$ overlaps with the string $[min(Y), max(Y)]$.*

Observe that intervals are qualified before the linear extensions to express the modalities in terms of one interval at each process.

Theorem 1. *For the* dense time model, *the mappings of the 29 interaction types to the three classes 1) $Definitely(\phi)$, 2) $\neg Definitely(\phi) \wedge Possibly(\phi)$, and 3) $\neg Possibly(\phi)$, are given in the top part of Table 3.*

Proof. Definition 2 gives an alternate formulation of the $Possibly$ and $Definitely$ modalities using linear extensions of the partial order $(E, \prec)$. If the strings $[min(X), max(X)]$ and $[min(Y), max(Y)]$ do not overlap in any linear extension, then $R1(X, Y) \vee R1(Y, X)$ and even $Possibly(\phi)$ cannot be true. The overlap of the strings in every linear extension of the partial order implies $Definitely(\phi)$, and this overlapping is equivalently characterized as $R4(X, Y) \wedge R4(Y, X)$. Otherwise, the strings overlap in some linear extensions and not in some others, which implies $Possibly(\phi) \wedge \neg Definitely(\phi)$. This is expressed in Fig. 4 in terms of the dependent relations $R1$ and $R4$. Mapping these conditions of Fig. 4 to the 29 orthogonal relations, we get the mapping shown in the upper part of Table 3.□

if $R1(X,Y) \vee R1(Y,X)$ **then** $\neg Possibly(\phi)$
else if $R4(X,Y) \wedge R4(Y,X)$ **then** $Definitely(\phi)$
else $Possibly(\phi) \wedge \neg Definitely(\phi)$

Fig. 4. Pseudocode for the mapping of the refined relations under dense time.

Observe that the mapping of the 29 relations to the three classes uses only a little of the information that is necessary to distinguish the 29 relations from one another. This is because we are mapping to a coarser granularity: 1) $Definitely(\phi)$, 2) $\neg Definitely(\phi) \wedge Possibly(\phi)$, and 3) $\neg Possibly(\phi)$. To give the reader a feel of this fine-to-coarse mapping which follows from the rules in Fig. 4, we consider one interaction type in each coarse-grained class.

- For interaction type ID defined by $R3(X,Y) = 1 \wedge R4(X,Y) = 1 \wedge S1(X,Y) = 1 \wedge S2(X,Y) = 1 \wedge R2(Y,X) = 1 \wedge R4(Y,X) = 1$ and all other relations being false, we have that as $R4(X,Y) = 1 \wedge R4(Y,X) = 1$, $Definitely(\phi)$ must hold. Similarly for its inverse IX.
- For interaction type IB defined by $R2(X,Y) = 1 \wedge R3(X,Y) = 1 \wedge R4(X,Y) = 1$ and all other relations being false, $R3(X,Y) = 1$ implies Y may be concurrent with X. X and Y will necessarily overlap in time if we have $R4(Y,X)$. However, as $\overline{R4}(Y,X)$, therefore, IB gets mapped to $\neg Definitely(\phi) \wedge Possibly(\phi)$. Similarly for its inverse IR.
- For interaction type IA defined by $R1(X,Y) = 1$, the durations X and Y can never be concurrent. Interaction IQ being the inverse of interaction IA, the same holds for IQ. Hence, both get mapped to $\neg Possibly(\phi)$.

Definition 2 needs to be applied cautiously to the nondense time model because the original definitions [3], [15] were formulated implicitly assuming dense time. Three possible interpretations of $Possibly$ and $Definitely$ under nondense time are described next. The resulting rules for the fine-to-course-grained mapping are given in Fig. 5.

Interpretation 1. When events are discrete and predicates can meaningfully take on values only at the events, overlapping strings $[min(X), max(X)]$ and $[min(Y), max(Y)]$ do not imply the existence of a single instant in global time at which the global predicate is necessarily true. Hence, $Definitely(\phi)$ can never hold.

An example application where this interpretation is useful is in particle physics studies. A particle impinging on a surface denotes a discrete event, and a local predicate, although defined on other local variables at the surface, may be meaningfully true only at the time of occurrence of the discrete event.

Interpretation 2. Despite the nondense time model and events being discrete, if we apply Definition 2, then all the 11 new interaction types for nondense time map to $Definitely(\phi)$. The mapping for the other 29 interactions is the same as for dense time.

Interpretation 3. To explicitly capture the notion of the strong coupling of events in X and Y for $Definitely(\phi)$ to hold, we can require $S2(X,Y)$ or $S2(Y,X)$, which explicitly signify a "round-trip dependence chain of events" between the two intervals, to be true. This would be in addition to the requirement in Definition 2 that strings $[min(X), max(X)]$ and $[min(Y), max(Y)]$ overlap for all linear extensions. The condition for $Definitely(\phi)$ to hold can be expressed as $R4(X,Y) \wedge R4(Y,X) \wedge (S2(X,Y) \vee S2(Y,X))$.

With this interpretation, observe that interaction type IM maps to $Possibly(\phi) \wedge \neg Definitely(\phi)$, instead of to $Definitely(\phi)$ if the dense time model had been used. The mappings for the other 28 interactions (applicable to the dense time model) are the same as for the dense time model. Of the 11 new interactions for nondense time, IM″, IMN, and IMN″ map to $Possibly(\phi) \wedge \neg Definitely(\phi)$, whereas the others map to $Definitely(\phi)$.

Theorem 2. *For the nondense time model, the mappings of the 40 interaction types to the three classes 1) $Definitely(\phi)$, 2) $\neg Definitely(\phi) \wedge Possibly(\phi)$, and 3) $\neg Possibly(\phi)$, as per the three interpretations above, are given in the lower part of Table 3.*

Proof. The proof follows by combining the explanations of the three interpretations, the proof of Theorem 1, Table 2, and Definition 2. The logic for the mappings is expressed in Fig. 5 in terms of the dependent relations $R1$, $R4$, and $S2$.□

Interpretation 1
if $R1(X,Y) \vee R1(Y,X)$ **then** $\neg Possibly(\phi)$
else $Possibly(\phi) \wedge \neg Definitely(\phi)$

Interpretation 2
if $R1(X,Y) \vee R1(Y,X)$ **then** $\neg Possibly(\phi)$
else if $R4(X,Y) \wedge R4(Y,X)$ **then** $Definitely(\phi)$
else $Possibly(\phi) \wedge \neg Definitely(\phi)$

Interpretation 3
if $R1(X,Y) \vee R1(Y,X)$ **then** $\neg Possibly(\phi)$
else if $R4(X,Y) \wedge R4(Y,X) \wedge (S2(X,Y) \vee S2(Y,X))$ **then** $Definitely(\phi)$
else $Possibly(\phi) \wedge \neg Definitely(\phi)$

Fig. 5. Pseudocode for the mappings of the refined relations for nondense time. The mappings for three interpretations are given.

In the example of Fig. 3, it can be shown that $IM(X_1(a = 3), Y_2(b = 7))$ (for dense time) and $IMN''(X_1(a = 3), Y_2(b = 7))$ (for Interpretation 2 of nondense time) hold; hence, $Definitely(a = 3 \wedge b = 7)$. The following can also be shown: 1) $IB(X_1(a = 3), Y_2(b = 2))$, hence, $Possibly(a = 3 \wedge b = 2) \wedge \neg Definitely(a = 3 \wedge b = 2)$, 2) $IG(X_1(a = 8), Y_2(b = 5))$, hence, $Possibly(a = 8 \wedge b = 5) \wedge \neg Definitely(a = 8 \wedge b = 5)$, and 3) $IV(X_1(a = 8), Y_2(b = 7))$, hence, $Possibly(a = 8 \wedge b = 7) \wedge \neg Definitely(a = 8 \wedge b = 7)$.

3.3 Predicate ϕ Defined over More than Two Processes

In general, a predicate is defined on variables local to some subset V of the set of all the processes, N. When a predicate ϕ is defined over variables that are local to more than two processes, one can express the three modal possibilities 1) $Definitely(\phi)$, 2) $\neg Definitely(\phi) \wedge Possibly(\phi)$, and 3) $\neg Possibly(\phi)$, in terms of the more fine-grained pairwise orthogonal modalities in $\Re$, between $C_2^{|V|}$ pairs of processes. Note that not all $40^{C_2^{|V|}}$ combinations will be valid—the combinations have to satisfy the axioms given in [9]. When considering the nondense time model in the rest of this section, the proofs implicitly assume Interpretation 2 of $Possibly$ and $Definitely$. The proofs for the other interpretations follow directly.

Theorem 3. *For a conjunctive predicate ϕ defined on processes in $V \subseteq N$, let ϕ be expressed as $\bigwedge_{i \in V} \phi_i$, where ϕ_i is the component of ϕ local to process P_i. The following results are implicitly qualified over a set of intervals, containing one interval from each process.*

1. $Definitely(\phi)$ *holds if and only if* $\bigwedge_{(\forall i \in V)(\forall j \in V)} [Definitely(\phi_i \wedge \phi_j)]$.
2. $\neg Definitely(\phi) \wedge Possibly(\phi)$ *holds if and only if*

$$\bullet \quad (\exists i \in V)(\exists j \in V) \neg Definitely(\phi_i \wedge \phi_j) \bigwedge \left(\bigwedge_{(\forall i \in V)(\forall j \in V)} [Possibly(\phi_i \wedge \phi_j)] \right).$$

3. $\neg Possibly(\phi)$ *holds if and only if* $(\exists i \in V)(\exists j \in V)$ $\neg Possibly(\phi_i \wedge \phi_j)$.

Proof. Part 1 : $Definitely(\phi)$ implies that every observation of the execution must include a state in which ϕ holds. Thus, in *every* linear extension of the execution, the strings $[min(X), max(X)]$ and $[min(Y), max(Y)]$ for each pair of intervals X and Y overlap. We also have that in every observation of the execution, there must exist an instant in time when $\phi = \bigwedge_i \phi_i$ is true. At this instant, the consequent must hold for the predicate decomposed pairwise, i.e., the execution must pass through this instant at which $\bigwedge_{(\forall i \in V)(\forall j \in V)} (\phi_i \wedge \phi_j)$.

We now prove that $\bigwedge_{(\forall i \in V)(\forall j \in V)}$ $[Definitely(\phi_i \wedge \phi_j)]$ implies $Definitely(\phi)$. The antecedent implies that in *each* linear extension, for each pair of intervals X and Y, $min(X)$ precedes $max(Y)$. Hence, given any linear extension, its $max(\{min(X_i) \mid i \in V\})$ precedes its $min(\{max(Y_j) \mid j \in V\})$ with ϕ being true between the two. Hence, $Definitely(\phi)$ holds. We prove this alternately using time durations. $Definitely(\phi_i \wedge \phi_j)$ implies that for *every* observation of the execution, there exists a time instant when intervals I_i and I_j overlap. Let $\mathcal{I}$ be the set of $|V|$ intervals, one per process, that overlap pairwise thus. Assume that $|\mathcal{I}|$ is a power of 2. (If it is not, add $2^{(\lceil log|\mathcal{I}| \rceil)} - |\mathcal{I}|$ duplicates of any elements of $\mathcal{I}$ to $\mathcal{I}$; $\mathcal{I}$ can be viewed as a multiset in this part of the proof.) Assign the elements of $\mathcal{I}$ to an array A that represents the leaf nodes of a complete binary tree. We use induction on the distance h of nonleaf nodes from the leaf nodes in the tree to show that the invariant on line 4 of the algorithm in Fig. 6 is true.

$A[1 \ldots |\mathcal{I}|]$: **array** of intervals

(0) Assign to A the elements of $\mathcal{I}$
(1) $d = \log |\mathcal{I}|$
(2) **for** $h = 1$ **to** d **do**
(3) **for** $j = 1$ **to** 2^{d-h} **do** claim **Invariant**;
(4) // **Invariant:** The 2^h intervals in the set
// $\{A[2^h \cdot (j-1)], \ldots, A[2^h \cdot j - 1]\}$ overlap in time

Fig. 6. Recursive combining to demonstrate overlap of intervals.

For the base case iteration $h = 1$, the invariant is true for all j between 1 and 2^{d-1} because there are two intervals in the set, and we have that each pair of intervals overlaps in time. We assume that the invariant is true for iteration $h = x$, $x \geq 1$. We now show that the invariant holds for iteration $h = x + 1$. The 2^{x+1} intervals in the set $\{A[2^{x+1} \cdot (j-1)], \ldots, A[2^{x+1} \cdot j - 1]\}$ correspond to an internal node in the tree at height $x + 1$ from the leaves. The two child nodes correspond to intervals in the two sets $G_L = \{A[2^{x+1} \cdot (j-1)], \ldots, A[2^{x+1} \cdot (j-1) + 2^x - 1]\}$ and $G_R = \{A[2^{x+1} \cdot (j-1) + 2^x], \ldots, A[2^{x+1} \cdot j - 1]\}$ of size 2^x each. As the invariant holds for height x, intervals in each set overlap at some point in time. Let d_L and d_R denote the overlap durations of the intervals in G_L and G_R, respectively. If d_L and d_R do not overlap, we have the scenario in Fig. 7 where time is divided into five zones $Z1$ to $Z5$. There must exist some interval I_j^L occurring in G_L and some interval I_k^R occurring in G_R, that do not extend into zone Z3. Hence, intervals I_j^L and I_k^R will not overlap in time, a contradiction because each pair of intervals must overlap in time. Thus, d_L and d_R must overlap and the invariant holds. When $h = \log |\mathcal{I}|$, all the intervals in $\mathcal{I}$ overlap at a common instant in time. This reasoning is true in all observations. Hence, $Definitely(\phi)$ holds.

Part 2 : If $Possibly(\phi)$, then some observation of the execution includes a state in which ϕ holds. Thus, in some linear extension of the execution, the strings

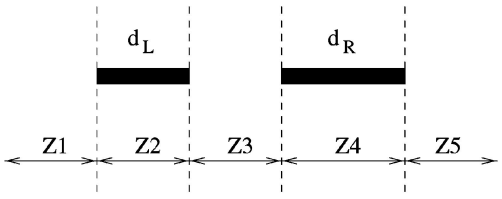

Fig. 7. Nonoverlapping of intervals leads to a contradiction. The shaded durations d_L and d_R denote the overlap durations of intervals in G_L and G_R, respectively.

$[min(X), max(X)]$ and $[min(Y), max(Y)]$ for each pair of intervals X and Y overlap. We also have that in some observation of the execution, there must exist an instant in time when $\phi = \bigwedge_i \phi_i$ is true. At this instant, the consequent must hold for the predicate decomposed pairwise, i.e., the execution must pass through this instant at which $\bigwedge_{(\forall i \in V)(\forall j \in V)} Possibly(\phi_i \wedge \phi_j)$.

We now show that $\bigwedge_{(\forall i \in V)(\forall j \in V)} [Possibly(\phi_i \wedge \phi_j)]$ implies $Possibly(\phi)$. We have that for each pair of processes, there exists a linear extension in which the corresponding interval strings $[min(X), max(X)]$ and $[min(Y), max(Y)]$ overlap. Hence, (from the properties of linear extensions), it follows that there must also exist *some* linear extension (of the same execution) in which $min(X)$ can precede $max(Y)$ for each pair of intervals X and Y. Analogous to Part (1) for $Definitely(\phi)$, in *this* linear extension, each pair of intervals overlap at some instant in time. Hence, $Possibly(\phi)$ holds.

We now have the lemma: "$Possibly(\phi)$ holds if and only if $\bigwedge_{(\forall i \in V)(\forall j \in V)} Possibly(\phi_i \wedge \phi_j)$." The expression for $\neg Definitely(\phi) \wedge Possibly(\phi)$ follows.

Part 3 : The third mutually exclusive possibility $\neg Possibly(\phi)$ must correspond exactly to the remaining cases $(\exists i \in V)(\exists j \in V) \neg Possibly(\phi_i \wedge \phi_j)$ when considering the intervals pairwise. □

By Theorem 3, given a predicate ϕ defined on any number of processes, $Definitely(\phi)$ and $Possibly(\phi)$ can be expressed in terms of the $Possibly$ and $Definitely$ modalities on predicates defined over pairs of processes. By Theorems 1 and 2, the $Possibly$ and $Definitely$ modalities on predicates defined over a pair of processes have been refined into the set $\Re$ of 40 orthogonal modalities. Therefore, $Definitely(\phi)$ and $Possibly(\phi)$, where ϕ is defined over any number of processes, can be expressed in terms of the fine-grained orthogonal set $\Re$ of modalities over predicates defined over various pairs of processes.

As an example, consider ϕ: $a_i = 2 \wedge b_j = 3 \wedge c_k = 5$. Let a_i, b_j, and c_k be 2, 3, and 5, respectively, in intervals X_i, Y_j, and Z_k, respectively, and let $ID(X_i, Y_j)$, $IV(Y_j, Z_k)$, and $IN(Z_k, X_i)$ be true. Then, by Theorems 1 and 2, we have 1) $Definitely\ (a_i = 2 \wedge b_j = 3)$, 2) $Possibly\ (b_j = 3 \wedge c_k = 5)$ and $\neg Definitely\ (b_j = 3 \wedge c_k = 5)$, and 3) $Definitely\ (a_i = 2 \wedge c_k = 5)$. By Theorem 3, we have the modality $Possibly(\phi) \wedge \neg Definitely(\phi)$. Conversely, if $Possibly(\phi) \wedge \neg Definitely(\phi)$ is known in the classical course-grained classification, the fine-grained classification gives the added information: $ID(X_i, Y_j)$, $IV(Y_j, Z_k)$, and $IN(Z_k, X_i)$.

Based on the above, the alternate definition (Definition 2) of $Possibly$ and $Definitely$ for $|V| = 2$ processes is enhanced for $|V| > 2$ processes. Let $\mathcal{I}$ be a set of intervals, one from each process, such that during these intervals, the local predicates are true.

- $Possibly(\phi)$: (For some such set $\mathcal{I}$ of intervals,) there exists a linear extension of $(E, \prec)$ such that for each pair of intervals X and Y in $\mathcal{I}$, the string $[min(X), -max(X)]$ overlaps with the string $[min(Y), max(Y)]$.
- $Definitely(\phi)$: (For some such set $\mathcal{I}$ of intervals,) for every linear extension of $(E, \prec)$, for each pair of intervals X and Y in $\mathcal{I}$, the string $[min(X), max(X)]$ overlaps with the string $[min(Y), max(Y)]$.

Without loss of generality, we now assume that ϕ can be expressed as

$$f(\phi_1, \phi_2, \ldots, \phi_k) = g_{\forall i \in V, \forall j \in V, i \neq j}(h_{i,j}(\phi_i, \phi_j)),$$

where f, g, and h map to $\{true, false\}$, and ϕ_i denotes the predicate ϕ's component variables local to process P_i. This expression of f occurs naturally when ϕ is a conjunction of local predicates, as assumed by the earlier works [6], [7], [8]. However, nonconjunctive predicates such as $(x_i + y_j = 5) \wedge (x_i + z_k = 10)$ and $x_i + y_j + z_k = 10$ are also expressible in this way (perhaps after reexpressing them using infinitely many clauses). The notation $\phi_{i,j}$ will be used as shorthand for $h_{i,j}(\phi_i, \phi_j)$. For such nonconjunctive predicates ϕ, a result analogous to Theorem 3 can be stated by replacing $\phi_i \wedge \phi_j$ by $\phi_{i,j}$ in the statement.

We remark that to use this result in any way for nonconjunctive predicates, the intervals need to be identified carefully, perhaps after examining the state lattice of the execution.

4 Tests for the Interaction Type

Each of the 29 (40) possible orthogonal interaction types in the dense (nondense) model of time can be tested using the bit-patterns for the dependent relations, as given in Table 2. Rather than test for the 12 dependent relations to determine which particular orthogonal interaction type holds, if one is interested only in determining whether a specific interaction type holds, one can test only for the prime cover for that relation [10]. Table 2 shows a prime cover in italics for all 40 orthogonal relations. Each orthogonal interaction type is uniquely identified by those dependent relations whose values are in displayed in italics. For example, $IC(X, Y)$ holds if and only if $\overline{R2}(X, Y) \wedge R3(X, Y) \wedge \overline{R4}(Y, X)$ holds. $IK(X, Y)$ holds if and only if $\overline{R4}(X, Y) \wedge \overline{R2}(Y, X) \wedge \overline{R3}(Y, X) \wedge R4(Y, X)$ holds. Note that such a prime cover need not be unique.

The tests for the relations $R1$, $R2$, $R3$, $R4$, $S1$, and $S2$ in terms of vector timestamps are given in the third column of Table 4. Intervals X and Y are assumed to occur at process P_i and P_j, respectively, and are also denoted as X_i and Y_j, respectively. We use $T(X_i^-)$ and $T(X_i^+)$ to denote the vector timestamp at process P_i at the start of interval X_i and at the end of an interval X_i, respectively. The mth component of the timestamps would be $T(X_i^-)[m]$ and $T(X_i^+)[m]$, respectively. $T(x_i)$ denotes the vector timestamp of event x_i (at process P_i). Processes can send certain timestamps of their intervals to a central server, which runs the tests in Table 4. The tests for $R1$, $R2$, $R3$, and $R4$ take one comparison each. The test for $S1$ is as follows—given $T(Y_j^-)[j]$, identify the first event x_i'' on P_i such that $T(x_i'')[j] \geq T(Y_j^-)[j]$; then, the event preceding x_i'' on P_i is event x_i'; then, $S1$ is true iff $T(x_i')[i] \not\leq T(Y_j^+)[i]$. The test for $S2$ is as follows—given $T(X_i^-)[i]$, identify the first event y_j' on P_j such that $T(y_j')[i] \geq T(X_i^-)[i]$; then, $S2$ holds iff $T(y_j')[j] \leq T(X_i^+)[j]$. Note that to test for $S1$ and $S2$, it is not sufficient to have the

TABLE 4
Timestamp-Based Tests for the Set of Six Dependent Relations R1-R4 and S1 and S2

Relation r	Expression for $r(X,Y)$	Test for $r(X,Y)$
R1	$\forall x \in X_i \forall y \in Y_j, x \prec y$	$T(Y_j^-)[i] \geq T(X_i^+)[i]$
R2	$\forall x \in X_i \exists y \in Y_j, x \prec y$	$T(Y_j^+)[i] \geq T(X_i^+)[i]$
R3	$\exists x \in X_i \forall y \in Y_j, x \prec y$	$T(Y_j^-)[i] \geq T(X_i^-)[i]$
R4	$\exists x \in X_i \exists y \in Y_j, x \prec y$	$T(Y_j^+)[i] \geq T(X_i^-)[i]$
S1	$\exists x \in X_i \forall y \in Y_j, x \not\preceq y \bigwedge y \not\preceq x$	**if** $T(Y_j^-)[j] \not\leq T(X_i^-)[j] \bigwedge T(Y_j^+)[i] \not\geq T(X_i^+)[i]$ **then** $\exists x_i' \in X_i$: $T(Y_j^-)[j] \not\leq T(x_i')[j] \wedge T(x_i')[i] \not\leq T(Y_j^+)[i]$ **else** $false$
S2	$\exists x1, x2 \in X_i \exists y \in Y_j, x1 \prec y \prec x2$	**if** $T(Y_j^+)[i] \geq T(X_i^-)[i] \bigwedge T(Y_j^-)[j] < T(X_i^+)[j]$ **then** $\exists y' \in Y_j$: $T(X_i^+)[j] \not< T(y_j')[j] \wedge T(y_j')[i] \not< T(X_i^-)[i]$ **else** $false$

Intervals X_i and Y_j occur at processes P_i and P_j, respectively.

timestamps of the start and end events of the intervals; timestamps of events in the intervals are also needed to detect the events x_i' and y_j', respectively. Also note that in the tests for $S1$ and $S2$, the "if" conditions are redundant; however, when $S1$ or $S2$ do not hold, the redundant condition provides the answer using two comparisons only.

When a predicate is defined over n (> 2) processes, a naive approach to detect $Possibly(\phi)$ and $Definitely(\phi)$, in terms of the above tests for a more fine-grained classification per pair of processes, will require $O(n^2)$ times the overhead for two processes, because C_2^n pairings of processes may have to be tested for. The axioms given in [9] can be used to reduce this overhead. It is a challenging problem to devise algorithms to detect $Possibly(\phi)$ and $Definitely(\phi)$ in terms of the fine-grained modalities per pair of processes, that have the same complexity overhead as [3], [6], [7], [8], [15], [18], [19], which simply detect $Possibly(\phi)$ and $Definitely(\phi)$. Such algorithms have been devised for conjunctive predicates [1]. The global predicate detection problem is formulated as the Detection of Orthogonal Relations (DOOR) problem as follows: "Given a relation $r_{i,j}$ from $\Re$ for each pair of processes P_i and P_j, devise an algorithm to identify the intervals, if they exist, one from each process, such that each relation $r_{i,j}$ is satisfied by the (P_i, P_j) process pair."

5 Applications

For a large class of applications, such as those involving a pair of interacting agents, a client-server interaction, or industrial process control, predicates are defined on variables local to two processes. The pairwise interaction between processes is a critical mode of communication and information exchange even in many large-scale and peer-to-peer distributed systems. Examples of such systems are sensor networks, ad-hoc mobile networks, mobile agent systems, and online collaborative motion planning and navigation systems. In a sensor network being set up dynamically, the sensors configure themselves by exchanging messages with their neighboring sensors. In the case of ad-hoc mobile networks, pair-wise information exchange with neighbors is used 1) to dynamically compute global functions such as to select routes (AODV), or 2) to dynamically compute time-dependent local functions such as the velocity of the mobile agent during navigation in a collaborative endeavor. In adaptive routing algorithms at the network layer—such as distance vector routing and link state routing—periodic information exchange with neighboring routers is a key feature of computing the global routing function. We note that the paradigm of pair-wise information exchange with each neighbor is common in other contexts also. Dynamic and online computation of local functions are used for centroidal Voronoi tessellations (a Voronoi tessellation whose generating points are the centroids of the corresponding Voronoi diagrams) with applications to problems as diverse as the following [4]—image compression, quadrature, finite difference methods, optimal placement of resources (e.g., mailboxes in a city, or cache servers in the internet), cellular biology, statistics, navigation by animals (e.g., flock of birds or school of fish), and territorial behavior of animals.

Information about the specific interaction type can provide useful information to the application. Consider a specific execution and the interaction types that map to $Possibly(\phi) \wedge \neg Definitely(\phi)$ in the dense time model—IB, IC, IG, IH, II, and their inverses (refer to Table 3). Recall that we assume intervals X and Y occur at P_i and P_j, respectively, and are denoted by X_i and Y_j, respectively. Examples of information useful to the application are the following:

- IB and II imply that even though interval Y_j may possibly have occurred concurrently, P_i's local state at the end of interval X_i is necessarily visible to the other process P_j before its interval Y_j ends, in all equivalent executions. Therefore, P_j can perform actions in interval Y_j by *taking into account* P_i's state at the end of P_i's interval X_i. This is not the case for IC, IG, and IH.
- IG implies that even though the other interval Y_j may have occurred concurrently, in all equivalent executions, neither is P_i's state at any time during interval X_i visible to the other process P_j at any time during its interval Y_j, nor is the other process P_j's state at any time during its interval Y_j visible to P_i at any time during interval X_i. Therefore, there is *no mutual knowledge* about the other interval. This is not the case for IB, IC, IH, and II.
- IG, IH, and II imply that even though interval Y_j of the other process P_j may have occurred concurrently, in all equivalent executions, the other process

P_j's execution during the interval *could not be fully controlled* by P_i's state as it existed at the start of interval X_i. This is not the case for IB and IC.

Such information (not just $Possibly(\phi) \wedge \neg Definitely(\phi)$) distinguishes each interaction type from every other, and allows processes to infer about the degree of information transfer that has occurred between the two intervals at different processes. This enables application-specific actions to be taken in and at the end of each interval. A similar analysis can reveal that the fine-grained classification for $Definitely(\phi)$, as well as for the three coarse-grained categories under the nondense time model under each of the three interpretations, provides useful information to the application. Thus, the fine-grained modality classification provides useful information that is not available under the $Possibly/Definitely$ classification.

6 Conclusions

Specifying temporal modalities on predicates in a distributed execution is an important problem. Distributed execution observation has classically used two modalities—$Possibly(\phi)$ and $Definitely(\phi)$—for predicate ϕ. Based on the temporal interactions of intervals, a complete, orthogonal set of relationships $\Re$ between pairs of intervals in a distributed execution had been identified by Kshemkalyani [9]. We leveraged this set of orthogonal interactions between pairs of intervals and mapped this rich, orthogonal classification of modalities of pairwise interval interactions, to the classical coarse-grained classification, $Possibly(\phi)$ and $Definitely(\phi)$, for specifying predicates defined on any number of processes. This significantly increases the power of expressing the temporal modalities under which predicates can be specified, beyond the current $Possibly/Definitely$ classification. We also gave timestamp-based tests for the orthogonal modalities in the refined classification.

Algorithms to detect the orthogonal relations of the fine-grained classification between pairs of processes, in addition to identifying the $Possibly$ or $Definitely$ modality, have recently been devised [1]. Future work would be to explore how the fine-grained orthogonal relations can formalize the information exchange patterns for applications such as those described in Section 5, and to integrate this framework in reasoning about applications.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under Grant No. CCR-9875617.

Figs. 1 and 2 and Tables 1 and 2 (except its italicized prime cover), are reprinted from the *Journal of Computer and System Sciences*, volume 52, 1996, Kshemkalyani, Temporal Interactions of Intervals in Distributed Systems, pages 287-298, Copyright 1996, with permission from Elsevier Science.

References

[1] P. Chandra and A. Kshemkalyani, "Algorithms for Detecting Global Predicates under Fine-Grained Modalities," Univ. of Illinois at Chicago Technical Report UIC-ECE-02-05, Apr. 2002.

[2] K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Trans. Computer Systems,* vol. 3, no. 1, pp. 63-75, 1985.

[3] R. Cooper and K. Marzullo, "Consistent Detection of Global Predicates," *Proc. ACM/ONR Workshop Parallel and Distributed Debugging,* pp. 163-173, May 1991.

[4] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Rev.,* vol. 41, no. 4, pp. 637-676, 1999.

[5] C.J. Fidge, "Timestamps in Message-Passing Systems That Preserve Partial Ordering," *Australian Computer Science Comm.,* vol. 10, no. 1, pp. 56-66, Feb. 1988.

[6] V.K. Garg and B. Waldecker, "Detection of Weak Unstable Predicates in Distributed Programs," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 3, pp. 299-307, Mar. 1994.

[7] V.K. Garg and B. Waldecker, "Detection of Strong Unstable Predicates in Distributed Programs," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 12, pp. 1323-1333, Dec. 1996.

[8] M. Hurfin, M. Mizuno, M. Raynal, and M. Singhal, "Efficient Distributed Detection of Conjunctions of Local Predicates," *IEEE Trans. Software Eng.,* vol. 24, no. 8, pp. 664-677, 1998.

[9] A.D. Kshemkalyani, "Temporal Interactions of Intervals in Distributed Systems," *J. Computer and System Sciences,* vol. 52, no. 2, pp. 287-298, Apr. 1996.

[10] A.D. Kshemkalyani, "Temporal Interactions of Intervals in Distributed Systems," IBM Technical Report 29.1933, (this includes the contents of [9] and the prime covers), 1994.

[11] A.D. Kshemkalyani, "A Framework For Viewing Atomic Events in Distributed Computations," *Theoretical Computer Science,* vol. 196, nos. 1-2, pp. 45-70, Apr. 1998.

[12] A.D. Kshemkalyani, "A Fine-Grained Modality Classification for Global Predicates," Univ. of Illinois at Chicago Technical Report UIC-EECS-00-10, 2000.

[13] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Comm. ACM,* vol. 21, no. 7, pp. 558-565, July 1978.

[14] L. Lamport, "On Interprocess Communication, Part I: Basic Formalism; Part II: Algorithms," *Distributed Computing,* vol. 1, pp. 77-85 and 86-101, 1986.

[15] K. Marzullo and G. Neiger, "Detection of Global State Predicates," *Proc. Fifth Workshop on Distributed Algorithms,* Springer-Verlag, pp. 254-272, Oct. 1991.

[16] S. Mullender, *Distributed Systems.* second ed. ACM Press, 1993.

[17] F. Mattern, "Virtual Time and Global States of Distributed Systems," *Parallel and Distributed Algorithms,* pp. 215-226, 1989.

[18] S. Stoller, "Detecting Global Predicates in Distributed Systems with Clocks," *Distributed Computing,* vol. 13, no. 2, pp. 85-98, Apr. 2000.

[19] S. Venkatesan and B. Dathan, "Testing and Debugging Distributed Programs Using Global Predicates," *IEEE Trans. Software Eng.,* vol. 21, no. 2, pp. 163-177, Feb. 1995.

Ajay Kshemkalyani received the BTech degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 1987, and he received the PhD degree in computer and information science from The Ohio State University in 1991. His research interests are in computer networks, distributed computing, algorithms, and concurrent systems. Since 2000, he has been as associate professor at the University of Illinois at Chicago. Before that, he spent several years at IBM Research Triangle Park working on various aspects of computer networks. In 1999, he received the US National Science Foundation's CAREER Award. He is a member of the ACM and a senior member of the IEEE and the IEEE Computer Society.