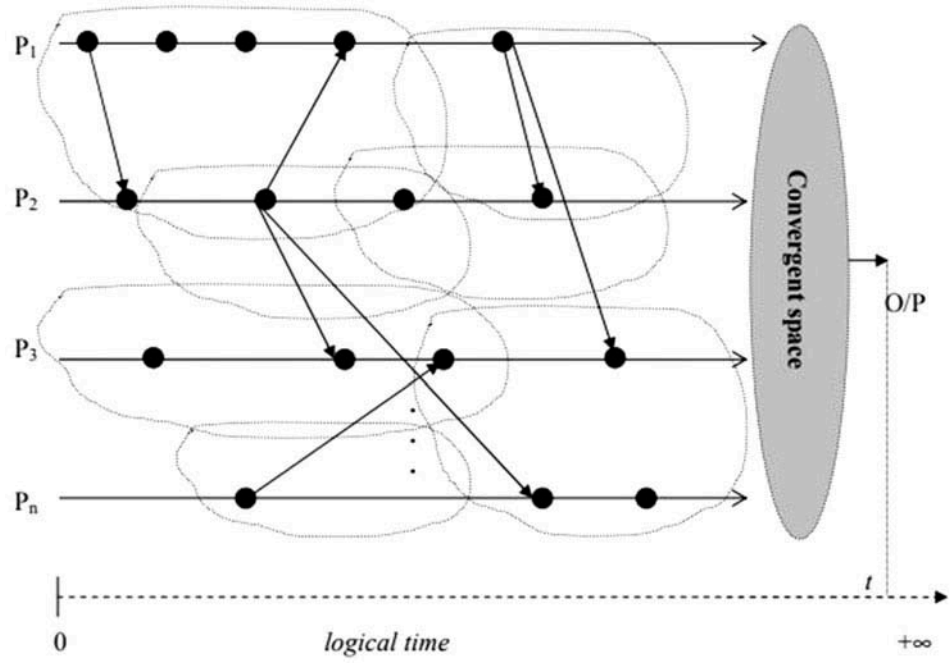
Formal analysis of control and termination of distributed computation in weaker spaces

Susmit Bagchi

cogent engineering



Schematic representation of topological aspects of distributed computing.

COMPUTER SCIENCE | RESEARCH ARTICLE

Formal analysis of control and termination of distributed computation in weaker spaces

Susmit Bagchi

Cogent Engineering (2018), 5: 1475033

Bagchi, *Cogent Engineering* (2018), 5: 1475033
https://doi.org/10.1080/23311916.2018.1475033

cogent engineering

COMPUTER SCIENCE | RESEARCH ARTICLE

Formal analysis of control and termination of distributed computation in weaker spaces

Susmit Bagchi[1]*

Received: 13 February 2018
Accepted: 08 May 2018
First Published: 24 May 2018

*Corresponding author: Susmit Bagchi, Department of Aerospace and Software Engineering (Informatics) Gyeongsang National University, Jinju, Republic of Korea
E-mail: profsbagchi@gmail.com

Reviewing editor:
Paolo Zampieri, Universita degli Studi di Padova, Italy

Additional information is available at the end of the article

Abstract: The control and termination detection of a distributed computation involving large-scale distributed database is difficult in the presence of concurrency, random network delays, and varying reliability of computing nodes. The formal modeling and analysis of controllability as well as termination detection of such systems are required to design reliable and dependable systems involving distributed computations and distributed datasets. The modeling and analysis of distributed computation can be performed by using combinatorial topology by forming simplexes, which imposes a set of relatively rigid geometric structures. This article proposes the modeling and analysis of observable and controllable distributed computation in weaker topological spaces. The proposed model enhances structural flexibility in metrized monotone spaces. The metrizability of the space and convergence property of the computation are analyzed. Axiomatic termination detection and fault mode analysis are formulated. A distributed algorithm is designed to observe and to determine varying control states of computation. The article includes a detailed comparative analysis of a set of models in the domain.

Subjects: Mathematics & Statistics; Computer Science; Engineering & Technology

Keywords: distributed computing; termination detection; monotone; topology; lattice; stable predicate

1. Introduction

The applications of distributed computing systems involving databases are pervasive in nature. These hybrid systems encompass LAN-based large-scale distributed systems having static network topology and wireless network-based distributed embedded systems having dynamic network

Susmit Bagchi

ABOUT THE AUTHOR

Susmit Bagchi has received B.Sc. (Honours) in 1993 from Calcutta University, B. E. in Electronics Engineering in 1997 from Nagpur University, M.E. in Electronics and Telecommunication Engineering in 1999 from Bengal Engineering and Science University (presently, IIEST). He obtained Ph.D. (Engineering) in Information Technology in 2008 from IIEST. Currently, he is Associate Professor in Department of Aerospace and Software Engineering (Informatics), Gyeongsang National University, Jinju, South Korea. His research interests are in Distributed Computing and Systems.

PUBLIC INTEREST STATEMENT

The formulation of topological model of distributed computing is a relatively new approach offering additional analytical insights. However, analysis of termination and control of distributed computation become analytically rigorous if topological space is weaker having monotone property and metrizability. This article proposes a novel topological model of distributed computation control and termination detection, which is analytically rigorous in nature offering better insight into the complex systems.

topology. In recent times, the applications of large-scale distributed database systems are accelerated due to the emergence of cloud computing platforms (Acosta, Lopez Dominguez, Gomez Castro, Pomares Hernandez, & Medina Nieto, 2015; Friday & Davies, 1995; Li & Lu, 2016; Pereira, Morais, & Freitas, 2018). The stable performance of large-scale distributed systems involving databases is dependent on the concurrency control, network delays, and reliability of nodes. The network delay behaves as a random variable and it affects overall control of distributed computations at nodes involving distributed datasets (Li & Lu, 2016; Rhode, Presicce, Simeoni, & Taentzer, 1999). The development of formal model of observation as well as control algorithm of asynchronous distributed computations is required to design and analyze large-scale systems in the presence of transient network partitioning (Bagchi, 2017; Rhode et al., 1999; Soneoka & Ibaraki, 1994). In general, distributed systems and databases are modeled by employing graph theoretical formalisms and relational algebraic operators (Dubois, Kaaouachi, & Petit, 2015; Kuhn, Lynch, & Oshman, 2010; Rhode et al., 1999; Sitohang, 2002). The traditional distributed systems involving databases consider asynchronous and iterative computation based on shared data (Borowsky & Gafni, 1997; Garg & Mittal, 2008). The iterative asynchronous computation and concurrent updates to shared objects or files in shared memory invite race conditions requiring the strict data consistency protocols. However, the distributed processes in a system cannot be synchronized employing any globally consistent clock (Garg & Mittal, 2008). Observation and control of such asynchronous iterative distributed computational systems are challenging in case of large-scale systems. The applications of concepts of topological spaces in modeling distributed computations facilitate the determination of consistency in concurrent executions (Fajstrup, Rauben, & Goubault, 2006; Saks & Zaharoglou, 2000). For example, the higher-dimensional topological structures and simplexes are applied to model and analyze distributed computing systems (Conde & Rajsbaum, 2012). However, the traditional combinatorial topological models (simplicial complexes) of distributed computation impose relatively rigid geometric structures on computational states. The simplicial complexes form a fixed set of graph structures related to states of processes consistent to distributed computation and it is assumed that a distributed computation should conform to such structures enhancing rigidity. A more flexible algebraic model is required to enhance observability and controllability of distributed computing systems in heterogeneous environments involving distributed databases. Furthermore, termination detection and fault mode analysis of distributed computation are required to be analyzed in order to determine stability of computation.

This article proposes the design and analysis of observable and controllable distributed computation within metrized and weaker (monotone) topological spaces. The proposed formal model considers that (a) distributed computation is asynchronous in nature and (b) the underlying computational space is monotone having convergence. The proposed formal model considers generalized distributed computing framework and does not make any assumption about network topology. This article presents a set of axiomatic and analytical properties of the stronger and weaker forms of observable control states of an asynchronous distributed computation. Furthermore, it is illustrated that the Birkhoff's representation theorem can be preserved in the proposed model. The properties of monotone slicer are constructed. Finally, the termination detection and fault modes are analyzed in axiomatic form. A detailed comparative analysis is presented.

2. Application aspects and motivation

A large-scale distributed database system is comprised of geo-distributed clusters of databases connected by Internet (Li & Lu, 2016). The local clusters containing large datasets are comprised of multiple computing nodes having large storage spaces. The clusters may implement either data replication or partitioned datasets depending upon the requirements and design principles aiming particular application. The schematic representation of large scale geo-distributed databases is illustrated in Figure 1.

The formulation of analytical models of observation and control of large-scale distributed computations are essential requirements to maintain consistency under asynchrony as well as concurrency (Lindstrom, 2004). However, in order to determine control states of asynchronous

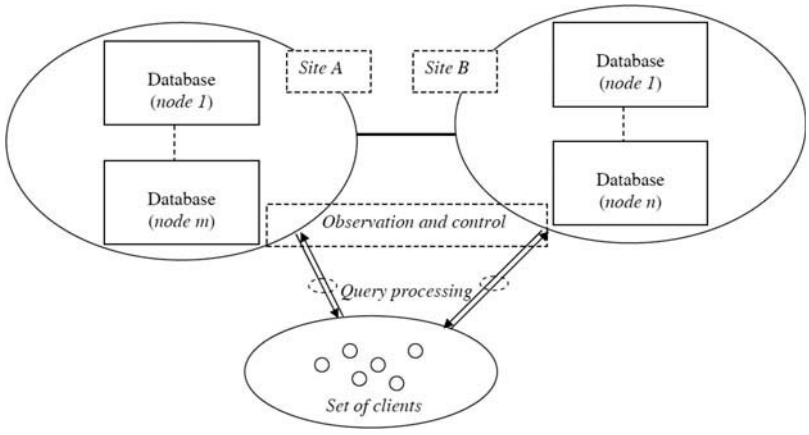


Figure 1. Schematic representation of large-scale databases with control interface.

distributed computation, the modeling of observability of computational states is required to be formulated. The modular graph models and relational algebraic models do not adequately formalize observability and control of a distributed computing involving distributed database. The maintenance of data consistency as well as observation and control of distributed computations are difficult under asynchronous model of computations (Bagchi, 2017; Garg & Mittal, 2008). Applications of mathematical formalisms to model distributed computation can be found in modeling process migration in distributed systems (Khaneghah, Ghahroodi, & ShowKatAbad, 2018). The concepts of algebraic topology and homotopy theory can be applied to model and analyze distributed asynchronous computing (Goubault, 2003). The distributed shared memory (DSM)-based iterative distributed computations are modeled by using concepts of topological simplexes (Borowsky & Gafni, 1997; Conde & Rajsbaum, 2012). The directed homotopy theory can be employed to model concurrency control in local processes (Fajstrup et al., 2006). However, the directed homotopy theory may not be directly applicable to model distributed computations and it needs to be adapted for applications in distributed computing systems in the absence of synchronized global clock. In general, monotone spaces are the generalized as well as weaker forms of topological spaces with the provision of ending property (Ghosh & Dasgupta, 2004). The modeling and analysis of distributed computation in monotone spaces are facilitated if the space is metrizable, where ending property helps to incorporate finiteness of computation in the respective model. This article proposes the model of asynchronous distributed computation in metrized topological spaces having monotone property. The properties of consistent slicer and convergence of distributed computation are analyzed. The axiomatic determinations of observable and controllable distributed computation in monotone spaces are formulated. It is illustrated that the Birkhoff's representation theorem can be preserved in the proposed model of asynchronous distributed computation in monotone spaces. The main contributions of this article are as follows.

- Formulation of an execution state-space model of distributed computation in metrized topological spaces having monotone property.
- Formulation of a consistent monotone slicer of distributed computations preserving Birkhoff's representation theorem under noncommutative composition.
- Axiomatic determination of observation of asynchronous distributed computations in monotone spaces.
- Designing of a distributed algorithm to determine control states of computation.
- Analysis of metrizability conditions as well as convergence of global state transition function.
- Axiomatic determination of termination and fault modes of distributed computation in weaker spaces.
- Comparative analysis and evaluation of a set of models.

The rest of the article is organized as follows. Section 3 represents related work. Section 4 presents formulation of state-space model of asynchronous distributed computation in monotone spaces. Section 5 describes designing of sliceable, observable, and controllable asynchronous distributed computation in monotone spaces. Section 6 illustrates designing of algorithm and formulation of associated properties related to determination of control state of computation in monotone spaces. Section 7 presents a set of analytical properties associated to the model. Section 8 presents convergence analysis and formation of relationship between the spaces. Section 9 presents termination detection and fault mode analysis in axiomatic forms. A detailed comparative analysis of a set of models is presented in Section 10. Finally, Section 11 concludes the article.

3. Related work

The applications of distributed systems along with databases are pervasive and hybrid in nature, which include cloud platforms (Acosta et al., 2015; Li & Lu, 2016). The modeling and analysis of distributed systems and databases are performed by using modular graph theory and relational algebraic operators considering that the underlying network topology is static (Garg & Mittal, 2001; Rhode et al., 1999; Sitohang, 2002). Researchers have proposed to model hybrid distributed systems by using dynamic graphs (Kuhn et al., 2010). In another direction, the combinatorial topological models of distributed computing are proposed by forming higher-dimensional geometrical objects and higher-dimensional automata (Armstrong, 1983; Edelsbrunner & Harer, 2010; Goubault & Jensen, 1992; Herlihy, Kozlov, & Rajsbaum, 2014; Herlihy & Rajsbaum, 1999). Furthermore, the DSM-based computing model is constructed utilizing combinatorial algebraic topological concepts (Borowsky & Gafni, 1997; Conde & Rajsbaum, 2012). Although the combinatorial topological models of computation address concurrency issues, the comparatively rigid geometrical structures are imposed on computational states. Following the similar direction, the concepts of persistent homology are employed to formulate the model of distributed computation in topological spaces (Bauer, Kerber, & Reininghaus, 2014; Zomorodian & Carlsson, 2005). It is proposed that homotopy theory can be applied to model the complexity of mutual exclusion of concurrently executing programs (Fajstrup et al., 2006; Goubault, 2003; Gunawardena, 1994). It is illustrated that the semaphore objects (for mutual exclusions) can be formed in topological spaces having partial ordering and it can be extended to analyze deadlock and serializability properties of concurrent processes (Carson & Reynolds, 1987). However, the application of homotopy theory cannot be made in distributed systems without considering directional property. This is because the general homotopy does not prevent time-reversal in a computing system. Hence, the directed homotopy is required to model distributed computing systems preserving the concept of monotonically increasing logical clocks.

In general, the combinatorial topological objects form simplicial complexes to model distributed processes and protocols (Fraigniaud, Rajsbaum, & Travers, 2011; Herlihy et al., 2014; Herlihy & Shavit, 1999). The connectivity properties of higher-dimensional topological objects are employed to model and analyze the computability issues in distributed computing systems (Borowsky & Gafni, 1993; Herlihy & Rajsbaum, 1999). The simulation of dynamics of topological objects is proposed to analyze the impossibility results in distributed computing systems, which can be reduced to manifolds (Borowsky & Gafni, 1993). Furthermore, the topology theory is employed to compute the time complexity bounds of determining approximate agreement in iterated immediate snapshot model (Hoest & Shavit, 2006). The object-oriented distributed systems are a class of designing distributed systems. The characterizations of object-based distributed systems are developed by using algebraic topological properties (Duarte, 2011). The modeling of liveness and safety properties of concurrency in distributed systems are constructed by employing concepts of topological spaces (Alpern & Schneider, 1985). The monotone spaces are the generalized topological spaces having ending property (Ghosh & Dasgupta, 2004). The applications of monotone spaces can be made in asynchronous distributed computing systems in order to determine consistency. The concept and mechanism of designing slicer of a distributed computation can be modeled in monotone spaces along with applications of Birkhoff's

representation theorem (Bagchi, 2017; Birkhoff & Kiss, 1947; Garg & Mittal, 2001). Moreover, the observation of state of control in an asynchronous distributed computation can be made in monotone spaces.

4. Distributed computation in monotone spaces

The applications of topological spaces can be made in modeling distributed computation. The distributed computation in topological spaces is modeled as simplexes, where simplexes are a set of complex in topological spaces (Fraigniaud et al., 2011; Herlihy et al., 2014). However, the asynchronous distributed computation can be modeled in monotone topological spaces in a more general form. The monotone space is a topological space having certain properties (Ghosh & Dasgupta, 2004). The concept and mechanism of designing slicer of a distributed computation can be modeled in monotone spaces along with applications of Birkhoff's representation theorem (Birkhoff & Kiss, 1947; Garg & Mittal, 2001). The observation of state of control of an asynchronous distributed computation can be made by formulating model of computation in monotone spaces.

4.1. Preliminary concepts

This article proposes model of distributed computation based on discrete topological spaces in order to cover all possible combinatorial execution forms under monotony. In this section, the basic definition of discrete topological spaces is described and the construction of monotone property is explained. If W is a point set and $u \subseteq \Omega(W)$, where $\Omega(.)$ is power set, then the topological space is (W, u) having properties as follows:

(a) $W, \phi \in \Omega(W)$,

(b) $\forall u_a, u_b \in \Omega(W), u_a \cup u_b \in \Omega(W)$, and

(c) $\forall u_a, u_b \in \Omega(W), u_a \cap u_b \in \Omega(W)$.

Property (a) indicates that the empty set and the entire set are parts of topological space whereas properties (b) and (c) indicate that topological space is closed under arbitrary union and intersection. The discrete topology of W is (W, u), where $u = \Omega(W)$. Moreover, W, ϕ are open; $\forall u_a, u_b \in u, u_a \cap u_b$ and $u_a \cup u_b$ are open. The simplicial complexes are the higher-dimensional analogues of the graphs representing the distributed processes (Conde & Rajsbaum, 2012; Fajstrup et al., 2006; Herlihy & Shavit, 1999). Following the concepts of topological spaces, simplicial complexes $W_C = \{w_n : n > 0 \wedge n \in Z^+\}$, where w_n is simplex and W_C is finite having properties as follows: $\forall w_n, w_m \in W_C, w_n \cap w_m \in W_C, w_n \cup w_m \in W_C$.

A simplicial map between simplexes (W_{c1} and W_{c2}) is given by $\partial : W_{c1} \rightarrow W_{c2}$.

If $g : \Omega(W) \rightarrow \Omega(W)$ is considered to be a monotone function such that $g(\phi) = \phi, \forall A \in \Omega(W), A \subset g(A)$ and $\forall A, F \in \Omega(W), A \subset F \Rightarrow g(A) \subset g(F)$ then, (W, g) is a monotone space. Any arbitrary intersections of closed sets in monotone spaces are closed, where finiteness of computation is maintained by closed finite intersection of sets.

4.2. Definitions and model

Let a distributed computation be represented as D comprised of a set of distributed processes given by $P = \{p_j : 1 \leq j \leq N \wedge j \in Z^+\}$. All the distributed processes are considered as finite state machines (FSM) and a process p_j has a set of deterministic execution states denoted by S_j, where $\phi \notin S_j$. Hence, D can be identified by,

$$S(D) = \bigcup_{j=1}^{N} S_j \tag{1}$$

It indicates that the state-based topological model of distributed computation considers a set of global states encompassing all the processes creating a state space of computation. The distributed processes execute as FSM with inter-process communications, and as a result, state transitions occur in a process. Considering the processing of inputs to a process is the internal computation, the overall dynamics of a set of processes can be modeled by using simplified and system-wide state transition function given by

$$\begin{aligned} &f : S(D) \to S(D), \\ &f(s_j \in S_j) \in S_j \backslash \{s_j\} \end{aligned} \tag{2}$$

The state transition function ensures that the FSM of distributed processes are free from self-loops at nodes resulting in a simple graph. The distributed computation is considered to be deterministic if the system-wide state transition function is having convergence property. Thus, the system dynamics converge to a space and such converging space of D is defined as

$$\overline{S(D)} = \bigcap_{j=1}^{N} S_j \tag{3}$$

The set of all possible cuts on distributed computation D is denoted by $C^{\otimes}(D) \subset \Omega(S(D))$.

4.2.1. Definition of boundary elements
The set of boundary elements B_j of a process p_j is defined as, $\forall p_j \in P$,

$$f(B_j \subset S_j) \in \overline{S(D)} \tag{4}$$

The concept of boundary elements represents the recognition of regions of convergences in state space of a distributed computation.

4.2.2. Definition of computation in D
If $B_j \subset S_j$ then a computation in D is defined as

$$\vec{D} = S(D) \backslash f\left(\bigcup_{j=1}^{N} B_j\right) \tag{5}$$

This indicates that the executions in a distributed computation consider global states excluding the convergent region. However, it includes the boundary state space.

4.2.3. Definition of monotone distributed computing
A monotone space over $\vec{D}$ is $(\vec{D}, g)$ having the following properties:

$$\begin{aligned} &g : \Omega(\vec{D}) \to \Omega(\vec{D}) \\ &g(\phi) = \phi \\ &\forall A \in \Omega(\vec{D}), A \subset g(A) \\ &\forall A, B \in \Omega(\vec{D}), A \subset B \Rightarrow g(A) \subset g(B) \end{aligned} \tag{6}$$

This definition represents the monotone property of a distributed computation and the mapping of structural forms within the state space of distributed computation under consideration.

4.2.4. Concept of topological cut
In this section, the concept of topological cut is introduced. Let, X_P be a point set and a topological space over X_P is (X_P, τ_X). If $d_X : X_P \times X_P \to \Re$, where $d_X \in [0, +\infty)$ is a distance metric in X_P then (X_P, d_X, τ_X) is a metric space having topological structure. The space is segmented by a topological cut $S_X \subset X_P$ such that the following axioms are satisfied,

$$
\begin{aligned}
&S_X \neq \phi,\\
&\exists X_1 \subset X_P, \exists X_2 \subset X_P : X_1 \cap X_2 = \phi,\\
&X_P \backslash S_X = X_1 \cup X_2,\\
&\forall x_1 \in X_1, \forall x_2 \in X_2, \forall s \in S_X :\\
&d_X(x_1, x_2) \leq d_X(x_1, s) + d_X(x_2, s)
\end{aligned} \tag{7}
$$

The set S_X is a topological cut of (X_P, d_X, τ_X). The cut S_X is called topologically symmetric cut of (X_P, d_X, τ_X) if the following axioms are satisfied,

$$
\begin{aligned}
&A \subset X_1, B \subset X_2 :\\
&(a \in A \wedge b \in B) \Rightarrow (\exists s \in S_X : d_X(a, s) = d_X(b, s)),\\
&|A| = |B| = |S_X|
\end{aligned} \tag{8}
$$

The computed set of distances of all elements of set X_1 and X_2 with respect to $\forall s \in S_X$ is given by

$$
\begin{aligned}
&\underset{\forall s \in S_X}{\Delta^1} = \{d_X(s, x) : \forall x \in X_1\}\\
&and,\\
&\underset{\forall s \in S_X}{\Delta^2} = \{d_X(s, x) : \forall x \in X_2\}
\end{aligned} \tag{9}
$$

The corresponding extremals of a topological cut S_X are as follows:

$$
\begin{aligned}
&\omega(X_1) = \left\{x : x \in X_1 \wedge d_X(s, x) = \max\left(\underset{\forall s \in S_X}{\Delta^1}\right)\right\}\\
&and,\\
&\omega(X_2) = \left\{x : x \in X_2 \wedge d_X(s, x) = \max\left(\underset{\forall s \in S_X}{\Delta^2}\right)\right\}
\end{aligned} \tag{10}
$$

The extremals of a topological cut are unique if $|\omega(X_1)| = |\omega(X_2)| = 1$.

4.2.5. Definition of $||$

In this case, it is assumed that the distributed processes are executing as FSM in a distributed system and the entire state space is $X_P = S(D)$. A partial ordering relation $(\xrightarrow{m})$ is defined in global state space of the respective distributed system by the following axioms (where s_{i-1} and s_{i+1} denote previous/next states of p_i before/after state s_i due to state transition, respectively),

$$
\begin{aligned}
&\forall p_i, p_j \in P : \xrightarrow{m} \subset (S_i \times S_j) \cup (S_j \times S_i),\\
&\forall s_i \in S_i, \forall s_j \in S_j :\\
&[(s_i, s_j) \in \xrightarrow{m}] \Rightarrow [(s_j, s_i) \notin \xrightarrow{m}],\\
&[(s_i, s_j) \in \xrightarrow{m} \vee (s_{j+1}, s_{i+1}) \in \xrightarrow{m}] \Rightarrow [s_{i+1} = f(s_i)] \wedge [s_{j+1} = f(s_j)]
\end{aligned} \tag{11}
$$

A relation $|| \subset S(D)^2$ between any two states within state-space of the distributed computation under consideration is defined by following axioms,

$$
\begin{aligned}
&\forall p_i \in P : s_{i-1} \in A \subset X_1, s_{i+1} \in B \subset X_2 : (s_i = f(s_{i-1})) \wedge (s_{i+1} = f(s_i)),\\
&\forall (s_i || s_j) \Rightarrow [(s_i, s_j) \notin \xrightarrow{m}] \wedge [(s_j, s_i) \notin \xrightarrow{m}],\\
&\forall (s_i || s_j) \Rightarrow \neg[\exists s_{j+1} \in B : (s_{j+1}, s_i) \in \xrightarrow{m}],\\
&(s_i || s_j) \Leftrightarrow (s_j || s_i)
\end{aligned} \tag{12}
$$

Thus, the relation $||$ maintains the consistency of cuts in execution state space of distributed processes.

5. Consistent slicing in monotone spaces

In this section, the models of cuts and distributive lattice of a distributed computation are formulated. In order to formulate an observable distributed computation, it is necessary to restrict the dynamics of distributed computation in monotone topological spaces. The restriction forms a

partition between sets of consistent states and inconsistent states in the sequences of executions. Let $A_j \in \Omega(\overrightarrow{D})$ and $A_j \subset g(A_j)$. Set of cuts $C \subset C^{\otimes}(D)$ in $\vec{D}$ for $K < \left|\Omega(\vec{D})\right|$ is given by $C = \bigcup_{j=1}^{K}\{g(A_j)\}$ such that $\forall c \in C, |c| = N$ and $\forall p_u, p_v \in P, [\forall x \forall y : x, y \in c] \Rightarrow [x \in S_u \wedge y \in S_v]$, where $u \neq v$. Hence, a cut is comprised of states of processes spanning the entire set of processes in the distributed system in monotone spaces. Each element in a set of cuts is not consistent and the consistency of cuts is formed by using lattice model maintaining consistency property of process states maintaining $||$ relation. The set of consistent cuts forms a lattice $(L, \leq)$ in $\vec{D}$, where $L \subset C$ and $\forall c \in L, [\forall x \forall y : x, y \in c] \Rightarrow (x||y)$. A set of lattice join-irreducible elements of L is given by $J(L) \subset L$. A distributed computation in monotone topological spaces is consistent if the run R in $\vec{D}$ is defined as a sequence given in the following, where $A_k, A_m \in \Omega(\vec{D})$, such that $(g(A_k), g(A_m)) \in \leq$ and $m = k + 1$,

$$R = \langle A_k : A_k \in \Omega(\vec{D}), k = 0, 1, 2, ...\rangle \tag{13}$$

Let X be a set of Boolean variables given by $X = \{x_n : n = 1, 2, 3, ...I\}$, where $x_n \in \{0, 1\}$.

The predicate $\Gamma(X|c) \in \{0, 1\}$ is assumed to be computable at cut state $c \in C$ in the corresponding distributed system. Thus, the slice of a distributed computation is defined as

$$J(L|\Gamma) = \{c : (\Gamma(X|c) = 1) \wedge (c \in J(L))\} \tag{14}$$

The lean slice of a distributed computation is $l = \{c : c \in J(L|\Gamma)\}$, where $|l| = 1$. A state e is called critical state if $\exists\{e\} \in \Omega(\vec{D})$ and $l = g(\{e\})$. The elements of $g(.)$ satisfying stable Boolean predicate $\Gamma(.)$ in a valid distributed computation are verified by,

$$(g(x), \Gamma) \Rightarrow [(x \in \Omega(\vec{D})) \wedge (g(x) \in L) \wedge (\Gamma(X|g(x)) = 1)] \tag{15}$$

Hence, the set of consistent and stable executions in a distributed computation can be found if the corresponding sequences of executions satisfy Equation (15).

5.1. Consistent observation of computation

The consistent observation about state of control in a distributed sliceable computation requires assignment of observable distributed variables to processes. The state of control is classified into two levels, namely, weak control state and strong control sate. The strong control state ensures maintenance of consistency of observable states throughout the lattice chain, whereas weak control state is a more relaxed form where not all observable states of processes are required to be always consistent. The dynamics of variations of observable distributed variables (local to processes) represent the combinatorial execution sequences preserving strong or weak control states in a distributed computing system. Let the set of Boolean variables be fixed to property $|X| = N$ and the elements are assigned to set of distributed processes P such that $\bar{X} = \{x_j : \forall p_j \in P, x_j \in X\}$, representing a set of observations associated to respective distributed processes and E_j be a set of local events of $p_j \in P$. Furthermore, let $\mathrm{H}(x_j, E_j) \in \{0, 1\}$ be a Boolean predicate (local to a process) defined over x_j considering local events set of the respective process in the distributed system under consideration. The global set of events in the distributed system is given by $\bar{E} = \bigcup_{j=1}^{N} E_j$. The function $r : \bar{E} \rightarrow \mathrm{Z}^{+}$ is deterministic following the finite state machine model of executions of distributed processes. The function r imposes a total order of execution of events within the processes (locally), whereas the global sequence of execution of events in the distributed system is a partial order following the asynchronous model of computation.

Let $\exists e_{xj}, e_{yj} \in E_j$ such that $r(e_{xj}) < r(e_{yj})$, then the following axiomatic implications determine stable and consistent observation of $\mathrm{H}(x_j, E_j)$,

$$\begin{aligned}&\forall p_j \in P, \forall r(e_{zj} \in E_j) \in [r(e_{xj}), r(e_{yj})] : \\ &((\mathrm{H}(x_j, e_{xj}) = 1) \wedge (\mathrm{H}(x_j, e_{yj}) = 1)) \Rightarrow (\mathrm{H}(x_j, e_{zj}) = 1) \\ &\text{and} \\ &((\mathrm{H}(x_j, e_{xj}) = 0) \wedge (\mathrm{H}(x_j, e_{yj}) = 0)) \Rightarrow (\mathrm{H}(x_j, e_{zj}) = 0)\end{aligned} \tag{16}$$

The events satisfying Equation (16) in the processes are denoted by $\forall p_j \in P : \langle e_{xj}, e_{yj} \rangle$. Furthermore, the segregated events for respective processes are computed as

$$\forall p_j \in P : Q = \left\{ e_{xj} : \langle e_{xj}, e_{yj} \rangle \right\}, T = \left\{ e_{yj} : \langle e_{xj}, e_{yj} \rangle \right\}$$

The anti-symmetric ordering relation $\bar{R}$ is defined over $\bar{R} \subset Q \times T$ such that

$$\begin{aligned}&\forall p_i \forall p_j \in P : \\ &(e_{xi}, e_{yj}) \in \bar{R} \Rightarrow (e_{xi} \in E_i) \wedge (e_{yj} \in E_j)\end{aligned} \tag{17}$$

It indicates that the ordering relation generates a partial order on associated events in combinatorial forms while maintaining the stability of local predicate within a process. The determination of controllability depends on the valid set of combinatorial execution sequences covering the global set of events in a system. The consistency-invariant execution depends upon the lattice of consistent cuts of the distributed computation under consideration.

5.2. Preserving Birkhoff's representation theorem

The Birkhoff's representation theorem reconstructs a lattice isomorphism using a multi-valued mapping function. The incorporation of isomorphic lattice cover in an observable distributed computation is an important factor in order to detect consistent execution sequences in lattice structure. It is possible to incorporate the Birkhoff's lattice isomorphism in the model proposed in this article as

$$\begin{aligned}&\beta : L \rightarrow C, \\ &\beta(c \in L) = \{x : ((x, c) \in \leq) \wedge (x \in J(L))\}\end{aligned} \tag{18}$$

Following Equation (18), the properties of Birkhoff's lattice cover and isomorphism can be preserved in the proposed slicer model of distributed computation in monotone topological spaces.

5.3. Topological cut in event-based distributed systems

We consider that X_P is an entire event space of the distributed computation under consideration having $X_P = \bar{E}$ and the corresponding logical clock function is $r : \bar{E} \rightarrow \mathrm{Z}^+$. Thus, the distance metric of (X_P, d_X, τ_X) in distributed computational model can be defined as

$$\begin{aligned}&\forall a \in \bar{E}, \forall b \in \bar{E} : d_X(a, b) = |r(a) - r(b)|, \\ &d_X(a, b) \in [0, +\infty)\end{aligned} \tag{19}$$

The partial ordering relation ($\xrightarrow{h}$) is defined in event space of distributed system under consideration as

$$\forall a \in \bar{E}, \forall b \in \bar{E} : [(a, b) \in \xrightarrow{h}] \Rightarrow [r(a) < r(b)] \wedge [d_X(a, b) > 0] \tag{20}$$

The topological cut S_X is consistent in the point-set topological model of distributed computation if the following axioms are satisfied over partial ordering relation,

$$\begin{aligned}&\forall a \in S_X, \forall b \in S_X : (\neg(a \xrightarrow{h} b)) \wedge (\neg(b \xrightarrow{h} a)) \\ &\text{and} \\ &|S_X| = N\end{aligned} \tag{21}$$

6. Determining controllability in monotone spaces

The set of all possible combinatorial execution sequences in the distributed computation under consideration can be counted by permutation $n!$, where $n = |\bar{E}|$. On the contrary, every combinatorial execution sequences are not feasible exhaustively due to data-dependency and logic

sequences between distributed processes (i.e. lattice of consistent cuts cannot be maintained in all combinations). Let the exhaustive and possible combinatorial sequences of executions for each process be computed as a finite set of permutations given by

$$\sigma(12) = E_1 \times E_2 \times E_3 \times \ldots \times E_N,$$
$$\sigma(13) = E_1 \times E_3 \times E_2 \times \ldots \times E_N,$$
$$\sigma(21) = E_2 \times E_1 \times E_3 \times \ldots \times E_N$$

and so on until the last term of permutation involving all set of events of every distributed processes. Hence, the entire combinatorial space of execution sequences in the distributed computation is computed as

$$U = \bigcup_{j=1}^{N} \sigma(jm) \text{ where, } m \leq N \tag{22}$$

The function recognizing the relation within a combinatorial sequence of execution is defined as $\mu_g : U \rightarrow \Omega(\bar{R}) \backslash \{\phi\}$ maintaining Equations (16) and (17). Thus, the filtered set of combinatorial executions (i.e. consistent set of combinatorial executions of processes) is given by

$$U_P = \left\{ u : u \in U \wedge \mu_g(u) \subseteq \bar{R} \right\} \tag{23}$$

It is important to note that all combinatorial executions in Equation (23) may not be always controllable and a stronger axiom is required. The stronger axiomatic form for observing controllability of a distributed computation by maintaining predicate throughout the execution lattice chain in the combinatorial execution space can be formulated as

$$\begin{aligned} &\forall u \in U_P : \bar{A} = \mu_g(u), \\ &C_{AS}(u) := \forall (e_{xi}, e_{yj}) \in \bar{A} : (\mathrm{H}(x_i, e_{xi}) = 1) \wedge (\mathrm{H}(x_j, e_{yj}) = 1) \end{aligned} \tag{24}$$

Hence, the set of all strongly observable and controllable combinatorial distributed computations is derived as

$$\overline{U_{PS}} = \{ u : u \in U_P \wedge C_{AS}(u) \} \tag{25}$$

However, the consistent observation and controllability of a distributed computation can be relaxed to a weaker form given by following axiom,

$$\begin{aligned} &\forall u \in U_P : \bar{A} = \mu_g(u), \\ &C_{AW}(u) := \forall (e_{xi}, e_{yj}) \in \bar{A} : (\mathrm{H}(x_i, e_{xi}) = 1) \vee (\mathrm{H}(x_j, e_{yj}) = 1) \end{aligned} \tag{26}$$

The corresponding weakly observable and controllable distributed computation is given by

$$\overline{U_{PW}} = \{ u : u \in U_P \wedge C_{AW}(u) \} \tag{27}$$

If the Boolean predicate evaluation function $\mathrm{H}(.)$ denotes the mutual exclusion detection on a shared data, then the consistency property can be maintained by refining the controllability axiom as

$$\forall (e_{xi}, e_{yj}) \in \bar{A} : \mathrm{H}(x_i, e_{xi}) \oplus \mathrm{H}(x_j, e_{yj}) \tag{28}$$

Furthermore, the singleton duration of distributed lock (mutex) γ_j for a process $p_j \in P$ realizing global consistency is derived as

$$\exists p_j \in P : (\langle e_{xj}, e_{yj} \rangle \wedge \mathrm{H}(x_j, e_{xj})) \Rightarrow (\gamma_j = r(e_{yj}) - r(e_{xj})) \tag{29}$$

Hence, the axiomatic determination of different degrees of observation and controllability of sliceable distributed computations can be measured and the safety property of distributed mutual exclusion can be maintained in monotone space.

Figure 2. Algorithm for observing state of control.

$\forall p_j \in P$::
array V_j[N] = {0};
variables current_event = null;
Boolean a, x_j = false, strong_control = false, weak_control = false;

compute (current_event, x_j);
if (current_event ∈ $[e_{xj}, e_{yj}] \wedge < e_{xj}, e_{yj} >$) send (H($x_j$, e_{xj}), P);
a = receive ($p_i \in P$);
$V_j[i] = a$;
if ($\forall i$: $V_j[i] = 1$, $1 \leq i \leq N$) strong_control = true;
else if ($\exists i$: $V_j[i] = 0$, $1 \leq i \leq N$) weak_control = true;
End p_j

6.1. The algorithm

The state of control of an asynchronous distributed computation can be algorithmically determined by following the strong and weak forms of axioms. The pseudo-code representation of the algorithm in simplified form for observing the state of control of an asynchronous distributed computation is presented in Figure 2.

Each process in a system maintains a local vector storing the instantaneous values of distributed Boolean variables associated to the processes. The initial control states are marked as false by the processes. The distributed processes compute while generating or absorbing events in the system, and during computation, the value of variables may change depending on the conditions such as event ordering. A process determines the intervals for observing stable predicate, and in the end of such interval, it sends the value of the predicate evaluation of associated variable to other processes. On receiving the values of the predicate from other distributed processes, a process will determine the global state of control in the system. The state of control can be strong or weak depending upon the values of predicate as evaluated by distributed processes. The worst case message complexity of the algorithm is $O(N^2)$ maintaining strong control, because processes exchange values of local variables in the system under consideration.

7. Analytical properties

The determinations of consistency of a distributed computation and its observation as well as controllability are important factors in the runs of concurrent interacting processes. Identification of critical element or state of computation and its property are required to form lean slice. This section constructs a set of analytical properties and proposes the existence of generating function of a slice in the execution lattice.

Furthermore, the properties of functional composition of Birkhoff's isomorphism and monotone spaces are identified in a distributed computation.

7.1. Theorem 1

$g(.)$ is strictly consistent if $\forall s_k \in \vec{D}, \exists A_k \in \Omega(\vec{D}), g(A_k) \in L$.

Proof: Let in a distributed computation be $\forall s_k \in \bigcup_{j=1}^{N} S_j, \exists A_k \in \Omega(\vec{D})$ such that $g(A_k) \in L$. Thus, $\exists s_m \in \bigcup_{j=1}^{N} S_j, m = k+1$, where $(g(A_k), g(A_m)) \in \leq$. Hence, the run R is always consistent in the corresponding distributed computation. Hence, $g(.)$ is strictly consistent in distributed computation in monotone spaces.

Descriptions: A distributed computation in monotone spaces is consistent within the state space of distributive lattice of consistent cuts of the corresponding computation.

7.2. Theorem 2

If $V \subset \Omega(\vec{D})$ such that $g(V) = C$, then $g(V)\backslash L$ may not be consistent.

Proof: Let $V \subset \Omega(\vec{D})$ and $\exists\{x\} \in V$ such that $g(\{x\}) \in C$. Now, $g(V)\backslash L \subset C$ and if $g(\{x\}) \in L$ then $g(\{x\}) \notin g(V)\backslash L$. Thus, $\forall\{x\} \in V, [g(\{x\}) \in L] \Rightarrow [g(\{x\}) \notin g(V)\backslash L]$. So, $g(V)\backslash L$ is not consistent.

Descriptions: The set of consistent cuts generated in monotone spaces are strictly within the distributive lattice of distributed computation. However, the set of all and every possible cuts of a distributed computation may not provide execution consistency in a run.

7.3. Theorem 3

If $A_k \in \Omega(\vec{D})$ such that $g(A_k) \in L$, then the composition $\beta \circ g \in J(L)$.

Proof: Let $A_k \in \Omega(\vec{D})$, where $A_k \subset g(A_k)$ and $g(A_k) \in L$. Thus, $\exists A_m \in \Omega(\vec{D})$ such that the following property holds:

$$[g(A_k), g(A_m) \in L] \Rightarrow [((g(A_k), g(A_m)) \in \leq) \oplus ((g(A_m), g(A_k)) \in \leq)] \tag{30}$$

Let $(g(A_m), g(A_k)) \in \leq$ in the distributed computation under consideration. Hence, if $g(A_m) \in J(L)$, then $g(A_m) \in \beta(g(A_k))$. Thus, $\beta \circ g \in J(L)$.

Descriptions: The Birkhoff's representation forms a noncommutative composition in monotone spaces of distributed computation. The isomorphism of the lattice of consistent cuts is preserved under the composition.

7.4. Theorem 4

$g(.)$ is a generating function of $(L, \leq)$.

Proof: Let $E \subset \Omega(\vec{D})$ such that $\forall x \in E, |x| < N$. This indicates $\exists g(a \in E), \exists g(b \in E)$ such that $(g(a), g(b)) \in \leq$. If $F \subset E$ such that $\forall x \in F, g(x) \in L$, then $g(F)$ is a generating function of lattice $(L, \leq)$.

Lemma: $g(.)$ is a generating function of $J(L)$.

Proof: Let $M \subset C$ and $L \subset M$. If $\exists A_k \in \Omega(\vec{D})$ such that $g(A_k) \in J(L)$, then the following implication holds:

$$[g(A_k) \in J(L)] \Rightarrow [(g(A_k) \in L) \wedge (g(A_k) \in M)] \tag{31}$$

Hence, $g(.)$ is a generating function of $J(L)$.

Descriptions: The function $g(.)$ in monotone spaces of distributed computation is a generating function of a slice in lattice of consistent cuts of the corresponding distributed computation. The function can generate a lean slice if such exists in the distributed computation.

7.5. Theorem 5

If $E \subset \Omega(\vec{D})$ such that $\forall x \in E, |x| < N$ and $g(x) \in L$, then $(g(F), \Gamma)$ induces a sub-lattice where $F \subset E$.

Proof: Let $E \subset \Omega(\vec{D})$ such that $\forall x \in E, |x| < N$ and $g(x) \in L$. Let $F \subset E$ such that $\forall x \in F, (g(x), \Gamma)$ holds resulting in the following implication:

$$[(g(x \in F), \Gamma)] \Rightarrow [\neg(g(x \in E\backslash F), \Gamma)] \tag{32}$$

However, $\forall x \in E$, the following condition is satisfied in the distributed computation:

$$[(g(x) \in L) \wedge (L = g(E))] \Rightarrow [(g(E), \leq)] \tag{33}$$

Moreover, $g(x \in F) \subset L$. Hence, $(g(F), \Gamma)$ induces a sub-lattice of $(L, \leq)$.

Descriptions: The consistent cuts of distributed computation in monotone spaces form a lattice where every run satisfies a stable predicate. Thus, the satisfying predicate in the corresponding induced lattice of distributed computation is invariant.

7.6. Theorem 6

If e is a critical state of a distributed computation in monotone spaces, then $(g(\{e\}), \Gamma) \Rightarrow [\exists y \in L : g(\{e\}) \in \beta(y)]$.

Proof: Let $e \in \vec{D}$ be a critical state of a distributed computation in monotone spaces. Thus, it is valid that $g(\{e\}) \in J(L|\Gamma)$ for the distributed computation. Let $A \subset L$ such that $J(L) = \beta(A)$. However, the following implication holds in the distributed computation under consideration,

$$[J(L|\Gamma) \subset J(L)] \Rightarrow [g(\{e\}) \in J(L)] \tag{34}$$

Moreover, in the distributed computation in monotone spaces satisfying a stable predicate indicates that the following biconditional exists:

$$[g(\{e\}) \in J(L|\Gamma)] \Leftrightarrow (g(\{e\}), \Gamma) \tag{35}$$

Thus, combining the above equations and considering the condition that $J(L|\Gamma) \subset J(L)$, one can conclude that

$$(g(\{e\}), \Gamma) \Rightarrow [g(\{e\}) \in \beta(A)] \tag{36}$$

Furthermore, in the respective distributed computation in monotone spaces, stable predicate is satisfied such that $(g(\{e\}), \Gamma) \Rightarrow [\exists y \in A : g(\{e\}) \in \beta(y)]$.

Hence, $(g(\{e\}), \Gamma) \Rightarrow [\exists y \in L : g(\{e\}) \in \beta(y)]$.

Descriptions: If a critical state exists in a distributed computation in monotone spaces satisfying a stable predicate, then the corresponding consistent cut in lattice is covered by Birkhoff's representation.

7.7. Theorem 7

If $A \in \Omega(S(D))$ such that $\exists s_i, s_j \in A : (s_i, s_j) \in \xrightarrow{m}$ then $\exists s_i, s_j \in g(A) : (s_i, s_j) \in \xrightarrow{m}$.

Proof: Let $A \in \Omega(S(D))$ such that $\exists s_i, s_j \in A$, where $(s_i, s_j) \in \xrightarrow{m}$. However, $A \subset g(A)$ indicating that $\{s_i, s_j\} \subset A \Rightarrow \{s_i, s_j\} \subset g(A)$. Hence, $\exists s_i, s_j \in g(A)$ such that $(s_i, s_j) \in \xrightarrow{m}$.

8. Structural forms

The state-space-based monotone topological model of distributed computation incorporates the set of cuts of distributed computation. The set of cuts forms a topological space of cuts. The associated subspaces can also be determined. Considering $C \subset C^{\otimes}(D)$ to be set of all possible cuts in distributed computation, the corresponding discrete topological space of cuts is $(G, \Omega(G))$, where $G = \bigcup_{\forall W \in C} W$. The discrete topological subspace (G_L, τ_{GL}) formed by consistent cuts is given by following axioms:

$$\begin{aligned} &G_L \subset G : L \subset \Omega(G_L), \\ &\tau_{GL} = \{G_L \cap Y : \forall Y \in \Omega(G)\} \end{aligned} \tag{37}$$

Thus, the structure of lattice can be embedded within the topological subspace formed by set of consistent cuts of a distributed computation.

8.1. Metrizability of computing spaces

Let V be any arbitrary set such that V is not metrizable. However, V can be metrized into (V,d) under existence of a one-dimensional real-valued function $h(.)$ and corresponding two-dimensional real-valued function $\lambda(.)$ with a distance metric $d(.)$ if the following axiom is satisfied:

$$\begin{aligned}&h:V\to\Re,\lambda:\Re^2\to(\Re^+\cup\{0\}),\\&\forall x\forall y\in V,d(x,y)=\lambda(h(x),h(y))\in[0,+\infty),\\&(x=y)\Leftrightarrow d(.)=0,\\&d(x,y)=d(y,x)\end{aligned}\tag{38}$$

For example, $h(.)$ can be a monotonically increasing function such that $\forall u,v\in V:[(u,v)\in\leq]\Rightarrow[h(v)>h(u)]$ and $\lambda(a\in\Re,b\in\Re)=|b-a|$.

In the proposed model of distributed computing, $S(D)$ represents the set of entire states of computation by multiple distributed processes. However, $S(D)$ is not directly metrizable.

The real-valued monotonically increasing function $\chi:S(D)\to\Re^+$ along with state transition function $f(.)$ enforce metrizability of $S(D)$ into $(S(D),d_s)$, where $d_s:S(D)^2\to\Re$ is a distance metric on $S(D)$. Hence, the definition of $d_s(.)$ is given by

$$\begin{aligned}&\forall s_i\forall s_j\in S(D),n\in Z^+,d_s(.)\in[0,+\infty),\\&d_s(s_i,s_j)=\left|\chi(s_i)-\chi(s_j)\right|,\\&\forall s_i\in S(D),d_s(s_i,s_i)=0,\\&\forall p_i\in P,n\geq1:d_s(s_i,f^n(s_i))>0\end{aligned}\tag{39}$$

This indicates that distributed computing has metrizability property under the existence of suitable real-valued function, where state transitions in a process form a chain as execution lattice.

Furthermore, if a distributed system is designed and analyzed based on events, then metrizability becomes easier. Considering E_i to be the set of events generated by $p_i\in P$ and $\bar{E}=\bigcup_{i=1}^{N}E_i$, the corresponding monotone logical clock function is given by $r:\bar{E}\to Z^+$. The distance metric function $d_e:\bar{E}\times\bar{E}\to(Z^+\cup\{0\})$ forms a metrized computing space in distributed systems by the following axioms:

$$\begin{aligned}&\forall e_i\forall e_j\in\bar{E},d_e(.)\in[0,+\infty),\\&d_e(e_i,e_j)=\left|r(e_i)-r(e_j)\right|,\\&\forall e_i\in\bar{E},d_e(e_i,e_i)=0\end{aligned}\tag{40}$$

This indicates that the state-based as well as event-based distributed computing systems are metrizable if it is equipped with suitable functions, respectively.

8.2. Convergence analysis

It is indicated earlier that $f(.)$ is a global state transition function controlling global dynamics of distributed computation. Let $\forall p_i\in P,\exists i_f(.)$ such that $i_f:S_i\to S_i$, where $S_i\subset S(D)$. The property of $i_f(.)$ is given as

$$i_f(S_i)\subset f(S(D))\tag{41}$$

Hence, if one considers $F_D=\{i_f(s_i):\forall p_i\in P\}$, then (F_D,d_s) is a function space defined over the distributed computation under consideration. The distributed computation is stable and deterministic if $f(.)$ is convergent satisfying the following axioms:

$$\begin{aligned}&\forall p_i \in P, n_i \in Z^+, s_i \in S_i,\\&i_f^{o2}(s_i) = i_f(i_f(s_i)),\\&\exists n_i \in (1, +\infty) : i_f^{on_i}(s_i) \in \overline{S(D)}\end{aligned} \tag{42}$$

Furthermore, it is possible to view the global state transition function as a discrete variety of single variable Banach contraction mapping (Latif, 2014) with finite ending property if the following axiom is satisfied:

$$\begin{aligned}&\forall p_i \in P, \exists s_i \exists s_\otimes \in S_i :\\&[i_f^{on_i}(s_i) = s_\otimes] \Rightarrow [f(s_i) \in \overline{S(D)} \wedge s_\otimes \in \overline{S(D)}]\end{aligned} \tag{43}$$

8.3. Formation of sub-lattice chain by $g(.)$

Let $A \in \Omega(S(D))$ such that $\exists s_i, s_j \in A : (s_i, s_j) \in \xrightarrow{m}$. Let in a rigid monotone be $B = g(A)$ and $|B| = |A| + 1$. Furthermore, if $\exists x \in B$ such that $(s_j, x) \in \xrightarrow{m}$, then $(B, \leq)$ forms a sub-lattice chain if $B \backslash A = \{x\}$. Next, the rigidity of monotone is relaxed such that $|B| = |A| + n$, where $n > 1, n \in Z^+$. If $\bar{B} \subset g(A)$ such that $(\bar{B}, \leq)$ is the only sub-lattice chain, then $\forall s_i, s_j \in B \backslash \bar{B}$, it is true that $s_i || s_j$. Hence, a sub-lattice chain is formed by monotone function $g(.)$.

9. Termination detection analysis

It is mentioned earlier that the global system-wide state transition function $f(.)$ is convergent in nature. The convergence property determines the termination of a distributed computation. In the following section, different failure modes of determination of termination of a distributed computation are analyzed. It is considered that state space of distributed computation is metrizable as $(S(D), d_s)$ under real-valued metric function, $d_s : S(D) \times S(D) \rightarrow [0, +\infty)$ maintaining standard metric axioms inclusive of triangular inequalities.

9.1. Shifting cluster point theorem

In a distributed computation D, if $\exists p_i \in P, \exists n_i \in Z^+$ and $\{s_i, s_i^\otimes\} \subset S_i, n_i \in (0, +\infty)$ such that $d_s(f^{n_i}(s_i), s_i^\otimes) = 0$ and $d_s(s_i^\otimes, s_\otimes \in \overline{S(D)}) \neq 0$, then D cannot terminate globally.

Proof: Let a distributed computation D be globally terminable where $\exists p_i \in P, \exists n_i \in Z^+$ such that $n_i \in (0, +\infty)$ and $\exists s_i \exists s_i^\otimes \in S_i : d_s(f^{n_i}(s_i), s_i^\otimes) = 0$. Furthermore, let there be $\exists s_\otimes \in \overline{S(D)} : d_s(s_i^\otimes, s_\otimes) = \varepsilon > 0$. Thus, $s_i^\otimes$ is a cluster point of $S_i \subset S(D)$. The globally convergent termination of a distributed computation requires that

$$\begin{aligned}&\forall p_i \in P, \exists s_i^\otimes \in S_i :\\&[d_s(f^{n_i}(s_i), s_i^\otimes) = 0] \Rightarrow [\varepsilon = 0]\end{aligned} \tag{44}$$

However, due to shifting of cluster point due to any computational fault, the following condition will be satisfied:

$$[\varepsilon > 0] \Rightarrow [s_\otimes \in \bigcap_{j=1}^{N-1} S_j] \tag{45}$$

This is a contradiction because, in this case $\overline{S(D)} = \phi$, and a distributed computation cannot converge into a null space.

Hence, if $d_s(f^{n_i}(s_i), s_i^\otimes) = 0$ and $d_s(s_i^\otimes, s_\otimes) > 0$, then the corresponding distributed computation cannot terminate globally.

9.2. Oscillatory execution theorem

If a state transition function of $p_i \in P$ is oscillatory such that $\exists k \in Z^+ : d_s(f^{n_i}(s_i), f^{(n_i+k+1)}(s_i)) = 0$, and, $d_s(f^{n_i}(s_i), f^{(n_i+k)}(s_i)) = \varepsilon > 0$, then the distributed computation in D will not terminate.

Proof: Let $p_i \in P$ in a distributed computation $\exists k, n_i \in Z^+$ such that $\exists s_i \in S_i$: $d_s(f^{n_i}(s_i), f^{(n_i+k)}(s_i)) = \varepsilon > 0$ and $d_s(f^{n_i}(s_i), f^{(n_i+k+1)}(s_i)) = 0$. This indicates that $\lim_{n_i \to +\infty} f^{n_i}(s_i) \in X_i$, where $X_i \subset S_i$. Thus, $|X_i| > 1$ and $f(S_i)$ is not convergent.

Hence, distributed computation in D will not terminate.

9.3. Termination forcing

It is often required in a distributed computing system that termination of a computation is enforced on the event of detection of a shifting fault. In general, the simplification in fault handling is implemented by realizing Fail-Stop protocol. However, an improved mechanism would be to employ a guarding function, which would act as a supervisory process. The function of such process is to monitor state transitions in a process and enforcing gradual termination on the event of fault.

Let a guard transition function $\theta : S(D) \to S(D)$ be defined under noncommutative composition $(\theta_o f)(.)$ as,

$$\begin{aligned} &\forall u, v \in S_i, \exists s_\otimes \in \overline{S(D)} : \\ &[d_s(f(u), s_\otimes) < d_s(u, s_\otimes)] \Rightarrow [(\theta_o f)(u) = f(u)], \\ &[d_s(f(u), s_\otimes) \geq d_s(u, s_\otimes)] \Rightarrow [(\theta_o f)(u) = v : d_s(v, s_\otimes) < d_s(u, s_\otimes)] \end{aligned} \tag{46}$$

Hence, following the Banach contraction property, it can be derived further as

$$\begin{aligned} &n_i \in Z^+, \forall s_i \in S_i : \\ &\exists n_i \in (1, +\infty), d_s((\theta_o f)^{n_i}(s_i), s_\otimes) = 0 \end{aligned} \tag{47}$$

Thus, the noncommutative composition $(\theta_o f)(.)$ enforces termination in a faulty system.

10. Comparative analysis and evaluation

This section presents a detailed comparative analysis and evaluation of the proposed monotone topological model of distributed computation control (termed as MTDC) considering a diverse set of other formal models such as Aspect KLAIM (Yang, Hankin, Nielson, & Nielson, 2013), supervisory control of distributed computation (SCON) (Komenda & Masopust, 2016), MSDC (Slotine, 2003), and DTA (Mattern, 1987). The parametric comparative evaluation of different models is presented in Table 1.

Researchers have proposed language-based modeling of control of execution of processes in mobile distributed computing systems, which is termed as AspectKLAIM (Yang et al., 2013). The AspectKLAIM formalism proposes a mathematical construct for programming of distributed processes involving datasets realizing predictive access control. The formalism supports distributed process synchronization in intervals as well as Inter-Process Communication (IPC) involving

Table 1. Comparative analysis and evaluation

Parameters \models	Interval sync/ p-join	Multi-process IPC	Sequencing (S)/ Termination (T)	Commutativity of relation ||	Events (E)/ states (SS)	Contraction/ projection
AspectKLAIM	Yes	Yes	S	Yes	Not applicable	Not applicable
SCON	No	Yes	S	Yes	E	Projection
MSDC	No	Yes	Not applicable	Not applicable	SS	Contraction (continuous)
DTA	No	Yes	T	Not applicable	E	Not applicable
MTDC	No	Yes	S,T	Yes	E,SS	Contraction (discrete)

multiple processes, which is inline with the proposed MTDC model. Furthermore, both the AspectKLAIM formalism and MTDC model employ commutative relation in different contexts. However, the main difference between AspectKLAIM and MTDC model is that AspectKLAIM formalism recognizes sequences of executions, whereas MTDC model can recognize sequences enforcing global termination within a system by contraction map.

In another approach, the automata-based SCON is proposed (Komenda et al., 2016). The automata-based control proposes a formal model of decomposed distributed execution control based on distributed events. The formalism supports synchronization and commutativity of || relation, which represents synchronous product of languages involving projection maps. Unlike the MTDC model, the SCON formalism uniquely depends on distributed event-based systems. The determination of modular stability in distributed computation as well as control of stabilization (MSDC: Modular Stability of Distributed Computation) is formulated using contraction theory in continuous functional domain (Slotine, 2003). The MSDC formalism is based on state vectors and the control function is in continuous time domain. The similarity between AspectKLAIM, MSDC, and proposed MTDC models is that all of them allow parallel combinations of processes or subsystems. However, the main difference between MSDC and MTDC models is that MSDC employs state-based contraction using continuous function, whereas the MTDC model realizes state-based contraction using discrete variety of Banach contraction.

The FIFO-order independent and finite time termination detection algorithm (DTA) is proposed for distributed systems (Mattern, 1987). The DTA does not employ interval synchronization or process join primitives. However, it supports IPC involving multiple concurrent processes, which are similar to proposed MTDC model. The distinguishing feature of DTA and MTDC is that unlike MTDC model, the DTA does not employ contraction or projection principles while determining termination in distributed systems. Furthermore, DTA requires adaptation to varying logical topology.

11. Conclusions

The observation and control of distributed computation involving distributed database systems are important requirements to maintain consistency under asynchrony. The applications of concepts of topological spaces in formulating distributed computation facilitate determination of consistency and state of executions. The simplicial complexes impose a relatively rigid structure on computation, whereas greater flexibility can be incorporated by employing monotone space with metrizability. The formal modeling of observability and controllability of a distributed computation helps in determining control states of respective computation. A corresponding distributed algorithm is designed to compute the state of controls in stronger form or in weaker form in an event-based distributed system. Further analysis illustrates that the corresponding computing space is metrizable and global system-wide state-transition function is convergent to stability. The determination of termination of computation and fault mode analysis are formulated in weaker spaces in axiomatic forms. The article includes a comparative analysis considering a set of approaches in the domain.

Funding
The authors received no direct funding for this research.

Author details
Susmit Bagchi[1]
E-mail: profsbagchi@gmail.com
[1] Department of Aerospace and Software Engineering (Informatics) Gyeongsang National University, Jinju, Republic of Korea.

Citation information
Cite this article as: Formal analysis of control and termination of distributed computation in weaker spaces, Susmit Bagchi, *Cogent Engineering* (2018), 5: 1475033.

Cover Image
Source:

References

Acosta, M. A. M., Lopez Dominguez, E., Gomez Castro, G., Pomares Hernandez, S. E., & Medina Nieto, M. A. (2015). Two-level software architecture for context-aware mobile distributed systems. *IEEE Latin America Transactions*, *13* (4), 1205–1209. doi:10.1109/TLA.2015.7106376

Alpern, B., & Schneider, F. B. (1985). Defining liveness. *Information Processing Letters*, *21*(4), 181–185. doi:10.1016/0020-0190(85)90056-0

Armstrong, M. A. (1983). *Basic topology*. New York, NY: Springer-Verlag. ISBN: 978-1-4757-1793-8.

Bagchi, S. (2017). Lattice based consistent slicer and topological cut for distributed computation in monotone spaces. *CCIS*, *716*, Springer, 202–211.

Bauer, U., Kerber, M., & Reininghaus, J. (2014). *Distributed computation of persistent homology*. In *Proceedings of Meeting on Algorithm Engineering and Experiments* (pp. 31–38). SIAM, USA.

Birkhoff, G., & Kiss, S. A. (1947). A ternary operation in distributive lattices. *Bulletin of the American Mathematical Society*, *53*(1), 749–752. doi:10.1090/S0002-9904-1947-08864-9

Borowsky, E., & Gafni, E. (1993). *Generalized FLP impossibility result for t-resilient asynchronous computations. Proceedings of the 25th annual ACM Symp. on Theory of computing*. California: ACM.

Borowsky, E., & Gafni, E. (1997). *A simple algorithmically reasoned characterization of wait-free computation*. In *Proceedings of the sixteenth annual ACM Symp. on Principles of distributed computing* (pp. 189–198). California: ACM.

Carson, S. D., & Reynolds, P. F., Jr. (1987). The geometry of semaphore programs. *ACM Transactions on Programming Languages and Systems*, *9*(1), 25–53. doi:10.1145/9758.9759

Conde, R., & Rajsbaum, S. (2012). An introduction to topological theory of distributed computing with safe-consensus. *Electronic Notes in Theoretical Computer Science*, *283*, Elsevier, 29–51. doi:10.1016/j.entcs.2012.05.004

Duarte, C. H. C. (2011). Mathematical models of object-based distributed systems. *LNCS*, *7000*, Springer, 57–73.

Dubois, S., Kaaouachi, M. H., & Petit, F. (2015). *Enabling minimal dominating set in highly dynamic distributed systems. 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2015), LNCS 9212* (pp. 51–66). Edmonton: Springer.

Edelsbrunner, H., & Harer, J. (2010). Computational topology: An introduction. *American Mathematical Society*. ISBN: 978-0-8218-4925-5

Fajstrup, L., Rauben, M., & Goubault, E. (2006). Algebraic topology and concurrency. *Theoretical Computer Science*, *357*(1–3), Elsevier, 241–278. doi:10.1016/j.tcs.2006.03.022

Fraigniaud, P., Rajsbaum, S., & Travers, C. (2011).*Locality and checkability in wait-free computing. 25th International Symposium on Distributed Computing (DISC 2011), LNCS* (Vol. 6950, pp. 333–347). Rome: Springer.

Friday, A., & Davies, N. (1995). Distributed systems support for mobile applications. *IEE Colloquium on Mobile Computing and Its Applications, IEE*. doi:10.1049/ic:19951395

Garg, V. K., & Mittal, N. (2001). *On slicing a distributed computation*. 21st International Conference on Distributed Computing Systems (ICDCS'01) (pp. 322–329). Washington, DC: IEEE.

Garg, V. K., & Mittal, N. (2008). Time and state in asynchronous distributed systems. In *Wiley encyclopedia of computer science and engineering*. Wiley. doi:10.1002/9780470050118.ecse436

Ghosh, S. R., & Dasgupta, H. (2004). Connectedness in monotone spaces. *Bulletin of the Malaysian Mathematical Sciences Society*, *27*(2), 129–148.

Goubault, E. (2003). Some geometric perspectives in concurrency theory. *Homology, Homotopy and Applications*, *5*(2), 95–136. doi:10.4310/HHA.2003.v5.n2.a5

Goubault, E., & Jensen, T. P. (1992). *Homology of higher dimensional automata*. In *Proceedings of CONCUR'92, LNCS* (Vol. 630). New York, NY: Springer.

Gunawardena, J. (1994). Homotopy and concurrency. *Bulletin of the EATCS*, *54*, 184–193.

Herlihy, M., Kozlov, D., & Rajsbaum, S. (2014). *Distributed computing through combinatorial topology*. Elsevier. Imprint: Morgan Kaufmann. ISBN: 978-0-12-404578-1.

Herlihy, M., & Rajsbaum, S. (1999). *New perspectives in distributed computing*. In *Proceedings of the 24th International Symp. on Mathematical Foundations of Computer Science, LNCS* (Vol. 1672, pp. 170–186). Poland: Springer.

Herlihy, M., & Shavit, N. (1999). The topological structure of asynchronous computability. *Journal of ACM*, *46*, 858–923. doi:10.1145/331524.331529

Hoest, G., & Shavit, N. (2006). Toward a topological characterization of asynchronous complexity. *SIAM Journal on Computing*, *36*(2), 457–497. doi:10.1137/S0097539701397412

Khaneghah, E. M., Ghahroodi, R. N., & ShowKatAbad, A. R. (2018). A mathematical multi-dimensional mechanism to improve process migration efficiency in peer-to-peer computing environments. *Cogent Engineering Journal*, *5*(5), Taylor & Francis, 1458434.

Komenda, J., & Masopust, T. (2016). Distributed computation of supremal conditionally controllable sublanguages. *International Journal of Control*, *89*(2), Taylor & Francis. doi:10.1080/00207179.2015.1079736

Kuhn, F., Lynch, N., & Oshman, R. (2010). *Distributed computation in dynamic networks*. Proceedings of the forty-second ACM symposium on Theory of computing (STOC'10) (pp. 513–522). Massachusetts, MA: ACM.

Latif, A. (2014). Banach contraction principle and its generalizations. In *Topics in fixed point theory* (pp. 33–64) ISBN: 978-3-319-01585-9. Switzerland: Springer.

Li, Y., & Lu, Y. (2016). *A two-layer cloud database model and its bidirectional conversion algorithms*. 7th International Conference on Software Engineering and Service Science (ICSESS) (pp. 289–294). Beijing: IEEE.

Lindstrom, J. (2004). Performance of distributed optimistic concurrency control in real-time databases. *LNCS*, *3356*, Springer, 243–252.

Mattern, F. (1987). Algorithms for distributed termination detection. *Distributed Computing Journal*, *2*, Springer-Verlag, 161–175. doi:10.1007/BF01782776

Pereira, D. A., Morais, W. O., & Freitas, E. P. (2018). NoSQL real-time database performance comparison. *International Journal of Parallel, Emergent and Distributed Systems*, *33*(2), Taylor & Francis, 144–156. doi:10.1080/17445760.2017.1307367

Rhode, M. G., Presicce, F. P., Simeoni, M., & Taentzer, G. (1999). *Modeling distributed systems by modular graph transformation based on refinement via rule expressions*. International Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE), LNCS, volume 1779, (pp. 31–45). Netherlands: Springer-Verlag. doi:10.1007/3-540-45104-8_3

Saks, M., & Zaharoglou, F. (2000). Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing*, *29*(5), 1449–1483. doi:10.1137/S0097539796307698

Sitohang, B. (2002). *Parallel execution of relational algebra operator under distributed database systems*. 2002 International Conference on Information Technology: Coding and Computing (pp. 207–211). Las Vegas: IEEE.

Slotine, J. J. E. (2003). Modular stability tools for distributed computation and control. *International Journal of Adaptive Control and Signal Processing*, *17*(6), Wiley. doi:10.1002/acs.754

Soneoka, T., & Ibaraki, T. (1994). Logically instantaneous message passing in asynchronous distributed systems. *IEEE Transactions on Computers*, *43*(5), 513–527. doi:10.1109/12.280800

Yang, F., Hankin, C., Nielson, F., & Nielson, H. R. (2013). Predictive access control for distributed computation. *Science of Computer Programming*, *78*, Elsevier, 1264–1277. doi:10.1016/j.scico.2012.05.008

Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete and Computational Geometry*, *33*(2), 249–274. doi:10.1007/s00454-004-1146-y