International Journal of Control and Automation
Vol. 10, No. 12 (2017), pp.209-220
http//dx.doi.org/10.14257/ijca.2017.10.12.19

Modeling Slicer and Control Algorithm for Distributed Computation in Metrized Weaker Topological Spaces

Susmit Bagchi

Department of Aerospace and Software Engineering (Informatics), Gyeongsang National University, Jinju, South Korea
profsbagchi@gmail.com

Abstract

A distributed system is essentially a networked system having controlled execution under observation. The present day distributed systems require hybridization with distributed databases to support heterogeneous data-intensive applications. The control of computation of such hybrid geo-distributed systems is difficult in the presence of randomly varying network delays and unpredictable occurrence of partitions in network topology. In order to gain insight to controllability of a hybrid geo-distributed system and to design control algorithm, the formal analytical model of the system is required. In general, lattice and algebraic topological concepts are employed to model distributed systems, which impose rigid geometric structures. However, flexibility of the model can be enhanced in weaker topological spaces. This paper proposes the model of observable distributed computation in weaker topological spaces having monotone and metrizability properties. A corresponding distributed control algorithm is designed. The distributed algorithm can detect various classes of state of control of distributed computation.

Keywords*: Distributed computing, monotone, topology, lattice, control algorithm, metric space*

1. Introduction

The present day distributed systems encompass peer-to-peer systems as well as mobile geo-distributed systems involving heterogeneous networks [1, 26]. A cloud computing platform is essentially a hybrid system comprised of geo-distributed systems and distributed databases storing massive datasets [30]. In such systems, the multiple server-groups (SGs) are formed by clustering a set of nodes and the SGs are connected by Internet. The nodes in SGs store large datasets and carryout computation. The inter-SG and intra-SG nodes coordinate by messages over network. In general, the clusters employ data replication of full datasets or partitioned datasets depending upon the requirements of a particular application. However, the performance of geo-distributed systems involving distributed databases is highly dependent on the concurrency control and stability of network topology. The limits of distributed computations and conditions of controllability are often required to be formulated in order to avoid waste of computing resources and to maintain stability. In order to model observable distributed computation and to design associated control algorithm, the formal model of asynchronous distributed computation is required [29, 30].

In the view of graph theory, a distributed computing system can be modeled as a graph having fixed topology if the underlying network is static [2, 28]. The computational model of such distributed system involves iterative execution and shared memory [10]. It is important to note that, a distributed computation cannot be rigidly synchronized based on any global clock due to random network delays [27]. Thus, the observation and control

Received (October 20, 2017), Review Result (December 7, 2017), Accepted (December 8, 2017)

ISSN: 2005-4297 IJCA
Copyright © 2017 SERSC Australia

of geo-distributed hybrid systems become more challenging in the presence of heterogeneous networks. Interestingly, the concepts of algebraic topology can be applied to model distributed computation [5, 6, 9]. However, the traditional algebraic topological models of distributed computation impose rigid geometric structures on computation reducing flexibility. A model with enhanced flexibility is required to design and analyze observability and controllability of geo-distributed computing systems in heterogeneous environments. The monotone space is a generalized weaker form of topological spaces [7]. Hence, the applications of monotone spaces with metrizability property in modeling controllability of distributed systems are promising approach. This paper proposes the design and analysis of observable and controllable distributed computation within weaker (monotone) topological spaces. The metrizability of the computing space is analyzed. A distributed algorithm is designed to control the respective distributed computation. This paper presents a set of axiomatic and analytical properties of the stronger and weaker forms of observable states of control of a distributed computation.

1.1. Motivation

The observation and control of hybrid geo-distributed computations are the essential requirements to maintain data consistency under replication, concurrency as well as asynchrony [31]. The observation and control of computing states of large scale distributed systems require formal models [27, 30, 31]. Traditionally, the graph theory and relational algebra are employed to model distributed computation and databases. However, the modular graphs and relational algebraic models do not adequately formalize observability and control of a hybrid geo-distributed computing in the presence of heterogeneous network, dynamic topology and, stability of nodes. The hybrid geo-distributed computation is essentially asynchronous in nature [27, 29]. The formal models of asynchronous distributed systems are constructed by employing algebraic topology and homotopy theory [5, 6, 10, 15]. The algebraic topological models impose a set of very rigid and fixed geometric structures, whereas a more flexible structure would benefit the modeling of asynchronous distributed systems exposing finer states of control. The formulation of computational model in monotone spaces would enhance structural flexibility. The metrizability of such space would enhance the observability of control states in topological spaces. This paper proposes the model of observable and controllable asynchronous distributed computation in metrized monotone topological spaces. The axiomatic determinations of observable and controllable distributed computation in monotone spaces are formulated. The main contributions of this paper are as follows.

- Formulation of an observable and controllable model of distributed computations in metrized monotone topological spaces.
- Formulation of a distributed algorithm to identify computational states in weaker topological spaces.
- Analysis of metrizability and convergence properties of distributed computational space.

Rest of the paper is organized as follows. Section 2 represents related work. Section 3 presents formulation of model of asynchronous distributed computation in monotone topological spaces. Section 4 describes designing of monotone slicer and control algorithm of asynchronous distributed computation in monotone spaces. Section 5 illustrates analytical properties of the model. Finally, Section 6 concludes the paper.

2. Related Work

The applications of large scale distributed systems are ranging from cloud computing platforms to distributed embedded systems. In recent times, the distributed systems

 Copyright © 2017 SERSC Australia

having hybrid architecture have emerged involving wireless network based mobile platforms [1]. Traditionally, the distributed computing systems are modeled by employing discrete structural elements with assumptions that network topology is static [25]. However, the hybrid distributed systems are dependent on dynamic network topology, which are modeled by using dynamic graphs [28].

In recent time growing number of attempts are made to apply the concepts of algebraic and combinatorial topology to model and analyze distributed computing systems. The concepts of algebraic topology theory are employed to model and analyze synchronous as well as asynchronous distributed computing systems [3, 11, 21]. The structures of algebraic topology deal with higher dimensional geometrical objects resulting in the formation of higher dimensional automata representing concurrent computation [17, 18, 24]. In general, the algebraic topological objects form simplicial complexes to model distributed processes and protocols [8, 21]. The connectivity properties of higher-dimensional topological objects are employed to model and analyze the computability issues in distributed computing systems [11, 13]. The modeling and analysis of distributed systems employing asynchronous iterated shared memory (DSM) model are formulated utilizing combinatorial algebraic topological concepts [5, 10]. Furthermore, the concepts of persistent homology are employed to formulate the model of distributed computation in topological spaces [20, 23]. Researchers have proposed to employ homotopy theory to model the complexity of mutual exclusion involving concurrently executing programs [6, 16]. Interestingly, the stability properties of distributed concurrent processes can be analyzed by applying homotopy theory [15]. 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 [14]. 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 reversal of time in a computing system.

Often, the distributed computing systems require iterative computation of immediate snapshot. The topology theory is employed to compute the time complexity bounds of determining approximate agreement in iterated immediate snapshot model [12]. Interestingly, the simplicial complexes can be reduced to manifolds in case of immediate snapshot model in order to reduce structural complexities [13]. However, generally the topological structures are very complex having rigid structural geometries. Following the combinatorial computing approach, distributed systems are modeled in combinatorial topological spaces. In such cases, the combinatorial relations in topological spaces are defined to construct the model of asynchronous processes having wait-free computation [8]. These combinatorial relations are in static form without considering the interleaving computations in distributed systems. The object-oriented distributed systems are a class of designing distributed systems. The characterizations of object-based distributed systems are formulated by using topological structures [19]. The modeling of liveness and safety properties of concurrency in distributed systems are constructed by employing concepts of topological spaces [22].

3. Distributed Computation in Monotone Spaces

The simplicial complexes are the higher-dimensional objects analogous to structures of graphs representing the distributed processes [5, 6, 8]. Following the concepts of algebraic topological spaces, simplicial complexes are $W_C = \{w_n : n > 0 \wedge n \in Z^+\}$ where, w_n is simplex and W_C is finite having properties: $\forall w_n, w_m \in W_C, w_n \cap w_m \in W_C, w_n \cup w_m \in W_C$. A simplicial map between simplexes (W_{c1} and W_{c2}) is given by, $\partial : W_{c1} \to W_{c2}$.

Copyright © 2017 SERSC Australia

Let W be a point set and, $\Omega(W)$ is power set. If $g:\Omega(W)\to\Omega(W)$ 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.

3.1. Construction of Model

Let a distributed computation be symbolically represented as D comprised of a set of distributed processes given by $P=\{p_j:1\le j\le N\wedge j\in Z^+\}$. All the distributed processes are considered as finite state machines 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,

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

The distributed processes execute as state machines 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 *global* dynamics of a set of processes can be modeled by using a simplified and *system-wide* state transition function defined as,

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

The distributed computation is considered to be deterministic if the system-wide state transition function is having convergence property within global state-space. Thus the overall system dynamics converge to a state-space and such converging space of D is defined as,

$$\begin{aligned} &\overline{S(D)}=\bigcap_{j=1}^{N}S_j,\\ &\overline{S(D)}\neq\phi \end{aligned} \tag{3}$$

The set of all possible cuts on distributed computation D is denoted by $C^{\otimes}(D)\subset\Omega(S(D))$, where $\Omega(.)$ generates power set. 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. If $B_j\subset S_j$ then a computation in D is defined as,

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

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

A monotone space over $\vec{D}$ is $(\vec{D},g)$ having following properties:

 Copyright © 2017 SERSC Australia

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

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.

3.2. Inter-process Communication

The set of distributed processes communicates through messages over the network in order to maintain synchrony and coordination. A partial ordering relation ($\xrightarrow{m}$), representing message passing, is defined in global state-space of the respective distributed system by 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),

$$
\begin{aligned}
&\forall p_i,p_j\in P:\xrightarrow{m}\subset(S_i\times S_j)\cup(S_j\times S_i),\\
&\forall s_i\in S_i,\forall s_j\in S_j:[(s_i,s_j)\in\xrightarrow{m}]\Rightarrow[(s_j,s_i)\notin\xrightarrow{m}],\\
&[(s_i,s_j)\in\xrightarrow{m}\vee(s_{j+1},s_{i+1})\in\xrightarrow{m}]\Rightarrow[s_{i+1}=f(s_i)]\wedge[s_{j+1}=f(s_j)]
\end{aligned}
\tag{7}
$$

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

$$
\begin{aligned}
&\forall p_i\in P:s_{i-1}\in A\subset S(D),s_{i+1}\in B\subset S(D):(s_i=f(s_{i-1}))\wedge(s_{i+1}=f(s_i)),\\
&\forall(s_i\|s_j)\Rightarrow[(s_i,s_j)\notin\xrightarrow{m}]\wedge[(s_j,s_i)\notin\xrightarrow{m}],\\
&\forall(s_i\|s_j)\Rightarrow\neg[\exists s_{j+1}\in B:(s_{j+1},s_i)\in\xrightarrow{m}],\\
&(s_i\|s_j)\Leftrightarrow(s_j\|s_i)
\end{aligned}
\tag{8}
$$

Thus, the relation $\|$ maintains the consistency of cuts in execution state-space of distributed processes by following traditional concepts of consistent cuts [25].

3.3. Convergence of Global Transition Function

Let, $\forall p_i\in P,\exists i_f(.)$ such that, $i_f:S_i\to S_i$ and, $i_f(S_i)\subset f(\mathrm{S(D)})$. The distributed computation is stable and deterministic if $f(.)$ is convergent transition function satisfying following axioms, where $i_f^{o2}(s_i)=i_f(i_f(s_i))$,

$$
\begin{aligned}
&\forall p_i\in P,\exists n_i\in Z^+,s_i\in S_i,\\
&n_i\in(1,+\infty),\\
&i_f^{on_i}(s_i)\in\overline{S(D)}
\end{aligned}
\tag{9}
$$

Copyright © 2017 SERSC Australia

If $\forall p_i \in P, \exists s_i \exists s_\otimes \in S_i : [i_f^{on_i}(s_i) = s_\otimes] \Rightarrow [f(s_i) \in \overline{S(D)} \wedge s_\otimes \in \overline{S(D)}]$ then, the global transition function is a discrete variety of Banach contraction map with finite ending.

4. Consistent Slicing and Control of Computation

In this section, the model of consistent slicing of distributed computation in monotone spaces is formulated. Furthermore, the axiomatic determination of controllability of a distributed computation is presented.

4.1. Monotone Slicing Model

In order to formulate an observable distributed computation, it is necessary to restrict the dynamics of distributed computation in monotone topological spaces. Let $A_j \in \Omega(\vec{D})$ and $A_j \subset g(A_j)$. Set of cuts $C \subset C^{\otimes}(D)$ in $\vec{D}$ for $K \lhd |\Omega(\vec{D})|$ is given by, $C = \bigcup_{j=1}^{K} \{g(A_j)\}$ such that, $\forall c \in C, |c| = N$ and, $\forall p_u, p_v \in P, [\forall x \forall y : x, y \in c] \Rightarrow [x \in S_u \wedge y \in S_v \wedge u \neq v]$. 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 is formed by using lattice model. The set of consistent cuts forms a lattice $(L, \leq)$ in $\vec{D}$ where, $L \subset C$ and, $\forall c \in L, [\forall x \forall y \in c \Rightarrow (x \| y)]$. 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 below where, $A_k, A_m \in \Omega(\vec{D})$ such that, $(g(A_k), g(A_m)) \in \leq$ and, $m = k + 1$,

$$R = \langle A_k : A_k \in \Omega(\vec{D}), k = 0,1,2,\ldots. \rangle \tag{10}$$

Let X be a set of Boolean observation variables given by, $X = \{x_n : n \in Z^+ \wedge x_n \in \{0,1\}\}$. The predicate $\Gamma(X \mid c) \in \{0,1\}$ 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,

$$J(L \mid \Gamma) = \{c : (\Gamma(X \mid c) = 1) \wedge (c \in J(L))\} \tag{11}$$

The lean slice of a distributed computation is $l = \{c : c \in J(L \mid \Gamma)\}$ 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,

$$(g(x), \Gamma) \Rightarrow [(x \in \Omega(\vec{D})) \wedge (g(x) \in L) \wedge (\Gamma(X \mid g(x)) = 1)] \tag{12}$$

Hence, the set of consistent and stable executions in a distributed computation can be found if the corresponding sequences of executions satisfy Equation (12). The Birkhoff's representation theorem formalizes corresponding lattice isomorphism structure [4]. It is possible to incorporate the Birkhoff's lattice isomorphism in the monotone slicing model by maintaining following axiom,

$$\begin{aligned} &\beta : L \to C, \\ &\beta(c \in L) = \{x : ((x, c) \in \leq) \wedge (x \in J(L))\} \end{aligned} \tag{13}$$

 Copyright © 2017 SERSC Australia

Thus, the proposed model of computation preserves Birkhoff's representation theorem within the corresponding space of distributed computation.

4.2. Controllability of Computation

The determination of controllability of computation in monotone spaces requires observation of control states within topological spaces. It is denoted earlier that X is a set of observation variables given by, $X = \{x_n : n \in Z^+ \wedge x_n \in \{0,1\}\}$ and, let $| X |= N$. The elements of the set are assigned to different distributed processes such that, $\overline{X} = \{x_j : \forall p_j \in P, x_j \in X\}$. Let E_j be a set of local events of $p_j \in P$ and, $\overline{E} = \bigcup_{j=1}^{N} E_j$. 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 set of events of the respective process in the distributed system under consideration. The integer valued function $r : \overline{E} \to \mathrm{Z}^+$ imposes a total order of execution of events within the processes (locally). 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 $\mathrm{H}(x_j, E_j)$,

$$\begin{aligned} &\forall p_j \in P, \forall r(e_{zj} \in E_j) \in [r(e_{xj}), r(e_{yj})]: \\ &((\mathrm{H}(x_j, e_{xj}) = 1) \wedge (\mathrm{H}(x_j, e_{yj}) = 1)) \Rightarrow (\mathrm{H}(x_j, e_{zj}) = 1), \\ &((\mathrm{H}(x_j, e_{xj}) = 0) \wedge (\mathrm{H}(x_j, e_{yj}) = 0)) \Rightarrow (\mathrm{H}(x_j, e_{zj}) = 0) \end{aligned} \tag{14}$$

The events satisfying Equation (14) 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, $\forall p_j \in P : Q = \{e_{xj} : \langle e_{xj}, e_{yj} \rangle\}, T = \{e_{yj} : \langle e_{xj}, e_{yj} \rangle\}$. The anti-symmetric ordering relation $\overline{R}$ is defined over $\overline{R} \subset Q \times T$ such that,

$$\begin{aligned} &\forall p_i \forall p_j \in P: \\ &(e_{xi}, e_{yj}) \in \overline{R} \Rightarrow (e_{xi} \in E_i) \wedge (e_{yj} \in E_j) \end{aligned} \tag{15}$$

The set of all possible combinatorial execution sequences in the distributed computation under consideration can be counted by permutation $n!$ where, $n = | \overline{E} |$. On the contrary, every combinatorial execution sequences are not feasible exhaustively due to data-dependency between distributed processes. Let, the exhaustive and possible combinatorial sequences of executions for each process be computed as a finite set of permutations denoted by, $\sigma(jm)$ where $\sigma(jm) = E_j \times E_1 \times E_2 \ldots\ldots \times E_m$. Hence, the entire combinatorial space of execution sequences in the distributed computation is computed as,

$$U = \bigcup_{j=1}^{N} \sigma(jm) \text{ where, } m \le N \tag{16}$$

Copyright © 2017 SERSC Australia

The function recognizing the relation within a combinatorial sequence of execution is defined as, $\mu_g : U \to \Omega(\overline{R}) \setminus \{\phi\}$. Thus, the filtered set of combinatorial executions (*i.e.* consistent set of combinatorial executions of processes) is given by,

$$U_P = \{u : u \in U \wedge \mu_g(u) \subseteq \overline{R}\} \tag{17}$$

It is important to note that, all combinatorial executions in Equation (17) may not be always controllable and a stronger axiom is required. The stronger axiomatic form for observing controllability of a distributed computation in the combinatorial execution-space can be formulated as,

$$\begin{aligned} &\forall u \in U_P : \overline{A} = \mu_g(u), \\ &C_{AS}(u) := \forall (e_{xi}, e_{yj}) \in \overline{A} : (\mathrm{H}(x_i, e_{xi}) = 1) \wedge (\mathrm{H}(x_j, e_{yj}) = 1) \end{aligned} \tag{18}$$

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

$$\overline{U_{PS}} = \{u : u \in U_P \wedge C_{AS}(u)\} \tag{19}$$

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

$$\begin{aligned} &\forall u \in U_P : \overline{A} = \mu_g(u), \\ &C_{AW}(u) := \forall (e_{xi}, e_{yj}) \in \overline{A} : (\mathrm{H}(x_i, e_{xi}) = 1) \vee (\mathrm{H}(x_j, e_{yj}) = 1) \end{aligned} \tag{20}$$

The corresponding weakly observable and controllable distributed computation is given by,

$$\overline{U_{PW}} = \{u : u \in U_P \wedge C_{AW}(u)\} \tag{21}$$

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,

$$\forall (e_{xi}, e_{yj}) \in \overline{A} : \mathrm{H}(x_i, e_{xi}) \oplus \mathrm{H}(x_j, e_{yj}) \tag{22}$$

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.

4.3. The Control 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 1.

 Copyright © 2017 SERSC Australia

$\forall p_j \in P$::
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

Figure 1. Algorithm for Observing State of Control

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 message complexity of the algorithm is $O(N^2)$, because each process has to share the local values to other processes in the distributed system under consideration.

5. Analytical Properties

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 $B \setminus A = \{x\}$. Next, the rigidity of monotone is relaxed such that, $|B| = |A| + n$ where, $n > 1, n \in Z^+$. If $\overline{B} \subset g(A)$ such that, $(\overline{B}, \leq)$ is the only sub-lattice chain, then $\forall s_i, s_j \in B \setminus \overline{B}$ it is true that $s_i \parallel s_j$. Hence, a sub-lattice chain is formed by the monotone function $g(.)$.

5.1. Critical State Theorem

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, $g(\{e\}) \in J(L \mid \Gamma)$ for the distributed computation. Let $A \subset L$ such that, $J(L) = \beta(A)$. However, following implication holds in the distributed computation under consideration,

$$[J(L \mid \Gamma) \subset J(L)] \Rightarrow [g(\{e\}) \in J(L)] \tag{23}$$

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

Copyright © 2017 SERSC Australia

$$[g(\{e\}) \in J(L \mid \Gamma)] \Leftrightarrow (g(\{e\}), \Gamma) \tag{24}$$

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

$$(g(\{e\}), \Gamma) \Rightarrow [g(\{e\}) \in \beta(A)] \tag{25}$$

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.

5.2. Message Passing Theorem

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, $(s_i, s_j) \in \xrightarrow{m}$.

5.3. Metrizability Property

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,

$$\begin{aligned} &\forall s_i \forall s_j \in S(D), n \in Z^+, d_s(.) \in [0, +\infty), \\ &d_s(s_i, s_j) = \mid \chi(s_i) - \chi(s_j) \mid, \\ &\forall s_i \in S(D), d_s(s_i, s_i) = 0, \\ &\forall p_i \in P, n \geq 1 : d_s(s_i, f^n(s_i)) > 0 \end{aligned} \tag{26}$$

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 be the set of events generated by $p_i \in P$ and $\overline{E} = \bigcup_{i=1}^{N} E_i$, the corresponding monotone logical clock function is given by

 Copyright © 2017 SERSC Australia

$r : \overline{E} \to Z^+$. The distance metric function $d_e : \overline{E} \times \overline{E} \to (Z^+ \cup \{0\})$ forms a metrized computing space in distributed systems by following axioms,

$$\begin{aligned} &\forall e_i \forall e_j \in \overline{E}, d_e(.) \in [0,+\infty), \\ &d_e(e_i, e_j) = \mid r(e_i) - r(e_j) \mid, \\ &\forall e_i \in \overline{E}, d_e(e_i, e_i) = 0 \end{aligned} \tag{27}$$

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

6. Conclusions

The topological models of distributed computation offer a new insight to the inherent properties of distributed computation. The algebraic topological complexes impose rigid geometric structures on computing models needing greater flexibility. The enhanced flexibility facilitates determination of slice and observability of fine-grained control states of computations. The model of asynchronous distributed computation in metrized monotone topological spaces allows greater flexibility in designing observable and controllable distributed computation. The distributed algorithm in metrized monotone topological spaces can identify the variations in state of control in an asynchronous distributed computation.

References

[1] M. A. M. Acosta, "Two-Level Software Architecture for Context-Aware Mobile Distributed Systems", IEEE Latin America Transactions, vol. 13, issue 4, **(2015)**, pp. 1205 - 1209.
[2] S. Dubois, M. H. Kaaouachi and F. Petit, "Enabling Minimal Dominating Set in Highly Dynamic Distributed Systems", 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2015), LNCS 9212, Springer, **(2015)**, pp. 51-66.
[3] P. Fraigniaud, S. Rajsbaum and C. Travers, "Locality and Checkability in Wait-Free Computing", 25th International Symposium on Distributed Computing (DISC 2011), Springer LNCS, vol. 6950, **(2011)**, pp. 333-347.
[4] G. Birkhoff and S. A. Kiss, "A ternary operation in distributive lattices", Bulletin of the American Mathematical Society, vol. 53, no. 1, **(1947)**, pp. 749–752.
[5] R. Conde and S. Rajsbaum, "An introduction to topological theory of distributed computing with safe-consensus", Electronic Notes in Theoretical Computer Science, Elsevier, vol. 283, **(2012)**, pp. 29-51.
[6] L. Fajstrup, M. Rauben and E. Goubault, "Algebraic topology and concurrency", Theoretical Computer Science, vol. 357, Issue 1-3, Elsevier, **(2006)**, pp. 241-278.
[7] S. R. Ghosh and H. Dasgupta, "Connectedness in Monotone Spaces", Bulletin of the Malaysian Mathematical Sciences Society, vol. 27, no. 2, **(2004)**, pp. 129-148.
[8] M. Herlihy and N. Shavit, "The topological structure of asynchronous computability", Journal of ACM, vol. 46, **(1999)**, pp. 858–923.
[9] M. Saks and F. Zaharoglou, "Wait-free k-set agreement is impossible: The topology of public knowledge", SIAM Journal on Computing, vol. 29, issue 5, **(2000)**, pp. 1449–1483.
[10] E. Borowsky and E. Gafni, "A simple algorithmically reasoned characterization of wait-free computation", In Proceedings of the sixteenth annual ACM Symp. on Principles of distributed computing, **(1997)**, pp. 189–198.
[11] M. Herlihy and S. Rajsbaum, "New perspectives in distributed computing", In Proceedings of the 24th International Symp. on Mathematical Foundations of Computer Science, LNCS, Springer, vol. 1672, **(1999)**, pp. 170-186.
[12] G. Hoest and N. Shavit, "Toward a topological characterization of asynchronous complexity", SIAM Journal on Computing, vol. 36, no. 2, **(2006)**, pp. 457-497.
[13] E. Borowsky and E. Gafni, "Generalized FLP impossibility result for t-resilient asynchronous computations", In Proceedings of the 25th annual ACM Symp. on Theory of computing, ACM, **(1993)**.
[14] S. D. Carson and P. F. Jr. Reynolds, "The geometry of semaphore programs", ACM Trans. Programming Languages Systems, ACM, vol. 9, no. 1, **(1987)**, pp. 25–53.
[15] E. Goubault, "Some geometric perspectives in concurrency theory", Homology Homotopy Appl., vol. 5, no. 2, **(2003)**, pp. 95–136.

Copyright © 2017 SERSC Australia

[16] [16] J. Gunawardena, "Homotopy and concurrency", Bulletin of the EATCS, vol. 54, **(1994)**, pp. 184-193.
[17] M. A. Armstrong, "Basic topology, Springer-Verlag", ISBN: 978-1-4757-1793-8, **(1983)**.
[18] E. Goubault and T. P. Jensen, "Homology of higher dimensional automata", In Proceedings of CONCUR'92, LNCS, Springer, vol. 630, **(1992)**.
[19] C. H. C. Duarte, "Mathematical Models of Object-Based Distributed Systems", LNCS, Springer, vol. 7000, **(2011)**, pp. 57-73.
[20] U. Bauer, M. Kerber and J. Reininghaus, "Distributed Computation of Persistent Homology", In proceedings of Meeting on Algorithm Engineering and Experiments, SIAM, USA, **(2014)**, pp. 31-38.
[21] M. Herlihy, D. Kozlov and S. Rajsbaum, "Distributed Computing Through Combinatorial Topology", Elsevier, ISBN: 978-0-12-404578-1, **(2014)**.
[22] B. Alpern and F. B. Schneider, "Defining liveness", Information Processing Letters, vol. 21, issue 4, **(1985)**, pp. 181-185.
[23] A. Zomorodian and G. Carlsson, "Computing persistent homology, Discrete and Computational Geometry", vol. 33, issue 2, **(2005)**, pp. 249-274.
[24] H. Edelsbrunner and J. Harer, "Computational topology: An introduction, American Mathematical Society", **(2010)**.
[25] V. K. Garg and N. Mittal, "On Slicing a Distributed Computation", 21st International Conference on Distributed Computing Systems (ICDCS'01), IEEE, **(2001)**, pp. 322-329.
[26] A. Friday and N. Davies, "Distributed systems support for mobile applications", IEE Colloquium on Mobile Computing and its Applications, IEE, DOI: 10.1049/ic: 19951395, **(1995)**.
[27] V. K. Garg and N. Mittal, "Time and State in Asynchronous Distributed Systems", Wiley Encyclopedia of Computer Science and Engineering, Wiley, DOI: 10.1002/9780470050118.ecse436, **(2008)**.
[28] F. Kuhn, N. Lynch and R. Oshman, "Distributed Computation in Dynamic Networks", Proceedings of the forty-second ACM symposium on Theory of computing (STOC'10), ACM, **(2010)**, pp. 513-522.
[29] T. Soneoka and T. Ibaraki, "Logically Instantaneous Message Passing in Asynchronous Distributed Systems", IEEE Transactions on Computers, vol. 43, issue 5, **(1994**), pp. 513-527.
[30] Y. Li and Y. Lu, "A two-layer cloud database model and its bidirectional conversion algorithms", 7th International Conference on Software Engineering and Service Science (ICSESS), IEEE, **(2016)**, pp. 289-294.
[31] J. Lindstrom, "Performance of distributed optimistic concurrency control in real-time databases", LNCS, Vol. 3356, Springer, **(2004)**, pp. 243-252.

Author

Susmit Bagchi, he has received B. E.in Electronics Engineering in 1997, M.E. in Electronics and Telecommunication Engineering in 1999 and, Ph.D. (Engineering) in Information Technology in 2008. Currently, he is Associate Professor in Department of Aerospace and Software Engineering (Informatics), Gyeongsang National University, South Korea. His research interests are in Distributed Computing.

 Copyright © 2017 SERSC Australia