Fault-Tolerance Verification in a Distributed Collective Collaborative Robotic System

Sungeetha Dakshinamurthy , Dr.Vasumathi Narayanan

Abstract- In this paper, we model a collective collaborative robotic system which acts as a distributed system, solving the problem which a single robot cannot. A system consisting of *n* processes is modeled by a respective set of *n* communicating finite-state machines (CFSMs). Robotic processes often run concurrently and communicate with each other to accomplish a common goal. We begin from a specification of a set of robotic tasks in the form of CFSMs. As opposed to the traditional product automaton, built from a given specification of CFSMs, whose state-space explodes, we build a state-compressed model out of CFSMs. The model is composed by simulating the specified set of CFSMs in a global environment into a corresponding set of what are defined as communicating minimal prefix machines (CMPMs). The states of CMPMs form a well-founded, partial order. This model truly represents sequence, choice/non-determinism and concurrency exhibited by the concurrent robotic system tasks. The model provides a sound platform for performing state exploration/model-checking without exponential state explosion to verify both safety and liveness properties of the given set of robotic tasks.

Index Terms— CCR, CFSMs, CMPMS, Safety, Liveness, exponential state explosion

1 INTRODUCTION

1.1 COLLECTIVE AND COLLABORATIVE ROBOTIC SYSTEM.

In this paper, we model a collective collaborative robotic system (CCRS) which acts as a distributed system solving the problem which a single robot cannot. Robotic processes often run concurrently and communicate with each other to accomplish a common goal. This model truly represents sequence, choice and concurrency, exhibited by the concurrent robotic system tasks. Here, we propose how collective collaboration in a simple reactive can be obtained through the exploitation of synchronous local and global communication of entire group of robots. It is significantly faster than a product-based model and its minimal set of parameters allows identifying the effect of characteristics of individual robots on the team performance. This model provides a sound platform for performing state exploration/model-checking without exponential state explosion, and also to verify fault-tolerant properties, namely safety and liveness, of the given set of robotic tasks.

- *Sungeetha Dakshinamurthy is Research scholar in Sathyabama University (e-mail:sungeetha5@yahoo.com).*
- Dr.Vasumathi Narayanan is Professor with St.Joseph's college of Engineering (e-mail: vasumathin@yahoo.com).

1.2 WHAT IS A CCRS?

CCRS has been defined as a novel approach to the coordination of few numbers of robots and as the study of how few numbers of robots can be designed to obtain a desired collective behavior which emerges from local interactions among agents representing those robots and between the agents and the environment.

The main characteristics of a CCRS are as follows:

- robots are autonomous and few in number;
- robots do not have access to centralized control;
- robots are reactive and interactive;
- robots are non-deterministic;
- robots may or may not be identical.

We use the above characteristics to discriminate a CCRS from a swarm robotic system with hundreds or thousands of identical robots.

Guaranteeing safety, liveness, predictability, adaption and reliability of robots and their high-level behaviors is crucial [1]. Human beings can perform sophisticated coordinated tasks due to inherent behavior of perception and inter-personal understanding, whereas robots perform sophisticated tasks in controlled environments without such understanding.

1.3 FAULT TOLERANCE IN CCRS

The aim of fault tolerance in distributed systems is to provide proper solutions to the system faults upon their occurrence and make the system more dependable by increasing its reliability. The main motivation of collective robotics research is the coordination of several systems (Gerkey and Matari'c, 2002; Agassounon et al., 2001;

Melhuish, 1999; Flocchini et al., 2000) and the robustness that can be achieved by the redundancy of the whole system (Parker, 1998; Goldberg and Matari'c, 2002; Fukuda et al., 1999). An increasing number of applications, such as space robotic missions (Chien et al., 2000; Earon et al., 2001) where there is a strong advantage of obtaining a more robust system, plan to exploit this type of information processing.

The reason is that some faults in individual robots may cause the whole system to fail even though there are still many fault-free robots in the system. Since there is no centralized control and global information, for identifying which robot or which component in the robot is faulty is difficult.

The fault-tolerance properties include *safety* and *liveness*. Safety properties mainly consist of guarantee of the absence of *communication deadlocks*. A *deadlocked state* is a state where there is no outgoing transition. Liveness properties consist of *eventuality* guarantee which means that eventually certain global-state vectors are *reachable* the concurrent robotic system tasks. Here, we propose how collective collaboration in a simple reactive robot can be obtained through the exploitation of synchronous local and global communications of an entire group. It is significantly faster than a product-based model and its minimal set of parameters allows identifying the effect of characteristics of individual robots on the team performance. This model provides a sound platform for performing state exploration/model-checking without exponential state explosion, and also to verify fault-tolerant properties, namely safety and liveness, of the given set of robotic tasks.

2. MOTIVATION

Some tasks may be fatal to human beings. On a larger scale, robots could play a part in military, for search and rescue operations, in forests, lakes, for mining detection and cleaning, acting together in areas where it would be too dangerous or impractical for humans to go. But the field work is expected to be done by collective robots. A collaborative task is a scenario where collective robots work together to complete a task that is beyond the capabilities of any of its individual robots. It can be more useful than a unique specialized robot, mainly because of the robustness of the CCRS. Though one robot fails, the rest of the robots continue working. In the case of a single robot, this is not possible. CCRS has many potential advantages over single-robot systems, including increased speed of task completion through parallelism; improved solutions for tasks that are inherently distributed in space, time or functionality ; cheaper solutions for complex applications that can be addressed with multiple specialized robots, rather than all-capable monolithic entities and increased robustness and reliability through redundancy (Parker 2008).

In recent research of robotics, much attention has been paid on utilizing reinforcement learning for designing robot controllers. However, there still exist difficulties; one of them is the well known state explosion problem. As the state space for a learning system becomes continuous and high dimensional, its combinatorial state space exponentially explodes and the learning process becomes time-consuming [3]. In this paper, we address this problem also by proposing a computational model of what are called *Communicating Minimal-Prefix Machines* (CMPMs) that cut down the state space of the modeled distributed CCRS drastically.

2.1 DISTRIBUTED CCRS

The distributed control collective robotics can be divided into movement and formation control, distributed learning, and coordination and task completion. The distributed hardware can provide strong robustness to failures, which is an advantage for self-organizing robots. Collective robotics is a field of multi-robotics in which few number of robots are coordinated in a distributed and decentralized way, where the robots are autonomous in their decision, and there is no leader. It is noted, however, that our proposed distributed system can only contribute to general theoretical foundation and that further progress is needed for the application of such methods to the CCRS.

3. GENERATION OF CMPMS MODEL FOR REDUCTION OF COMPUTATIONAL STATE SPACE

This computational model is developed from a given specification consisting of a set of CFSMs. Each CFSM models a *collaborative robot*. We take up an example of a CCRS consisting of *five* different robots each modeled by a CFSM and represented by a corresponding *state-transition graph*. We process this specification consisting of five CFSM graphs into a corresponding set of five *unfolded trees* whose *leaves* correspond to what are defined as *cutoff states*. Different CFSMs communicate by *synchronous message passing* upon synchronous/global actions. An event is an instance of an action. An action can be completely asynchronous/local to CFSM or a synchronous one involved by a set of two or more CFSMs from a given set. Each unfolded CFSM is called a CMPM for the following reason: Each CMPM state represents an instance of not only its corresponding local CFSM state, but also a vector of non-local CFSM states that are its *causal predecessors due to synchronization* in most recent past. This vector forms the synchronous *environment* of the concerned CMPM state, called its *Minimal-Prefix (MP) vector* unfolded from its corresponding CFSM.

Figure 1 shows a set of *five specified CFSMs*. The *states* are labeled with numbers and *global, synchronous actions* are labeled with upper case letters. The *local. asynchronous actions* local within CFSMs are not labeled. Figure 2 shows the flow chart of each robot. Figure 3 shows the synchronous actions performed while different CFSMs communicate by synchronous message passing, performing identical synchronous actions.

Figure 4 lists the table showing the local and global transitions. Figure 5 shows the corresponding set of CMPMs which are nothing but the *simulated CFSMs* in global environment. In CMPM M1, the *initial state vector* is represented by (1a, 5a, 9a, 13a, 15a) where state (1a) represents the *local component* and (5a, 9a, 13a,15a) represents the *MP vector* of (1a) comprising the *synchronous environment* of the latter state in CMPM M1. Similarly in CMPM M2, the *initial state vector* is represented again by (1a,5a,9a,13a,15a) where (5a) represents the *local component* and (1a, 9a, 13a,15a) represents its *MP vector* comprising the *synchronous environment* of the local component (5a) of CMPM M2 and so on for CMPMs M3, M4 and M5 as well. Similarly, the *CMPM events* A0, B0, C0, etc., of Fig. 4 are instances of the corresponding CFSM actions A, B, C, etc., respectively, of Fig. 1.

The *leaves* of the five CMPM *trees* represent the *cut-off* states. State (1b) is a *cut-off state* since the *Mp vector* of (1b) which is equal to (5b, 9b,13b,15b) and the *Mp vector* of (1a) which is (1a,5a,9a,13a,18a) are instances of the same FSM-state vector (1,5,9,13,18). Therefore, the state and transition instances of the *descendents* of (1b,5b,9ba,13b,15b) will be identical to those of its *ancestor* state vector (1a,5a,9a,13a,15a) and therefore need not be repeated and thus (1b,5b,9ba,13b,15b) forms the cut-off state by virtue of its ancestor (1a,5a,9a,13a,15a) represented as a *leaf state* represented by CMPM tree M1.

Since the CMPM structures simulate the given set of CFSMs, the set of successor states and transitions from a given CMPM state depend upon the CFSM vector it represents.

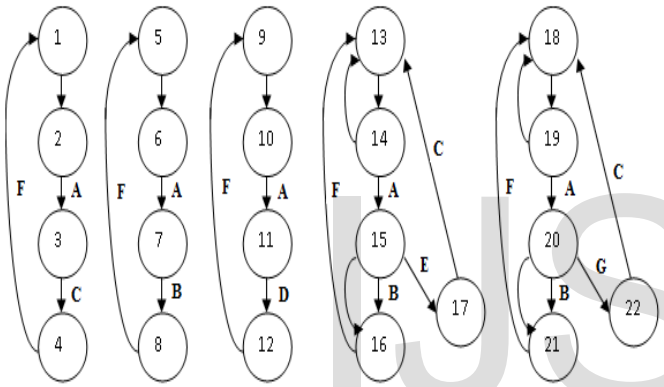


Fig. 1. Set of five CFSM graphs depicting a CCRS

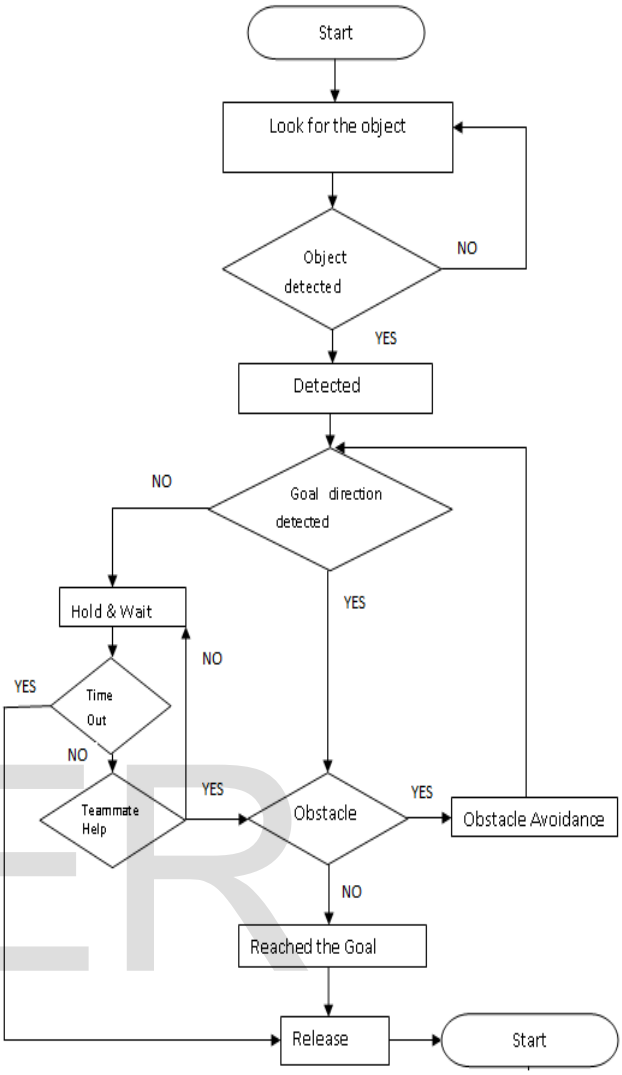


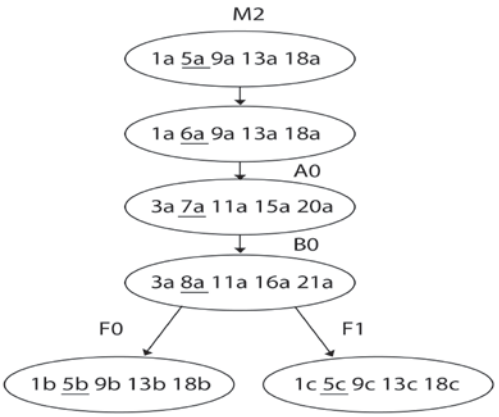
Fig. 2. Flow chart of each robot's controller

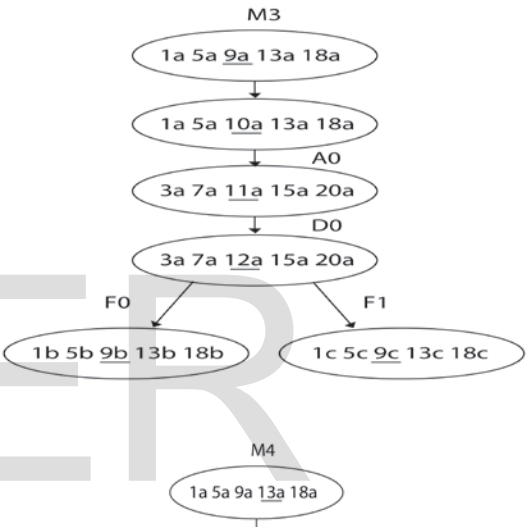
ACTION LABELS	ACTIONS
A	**Searching**
B	**Grabbing**
C	**Obstacle Avoidance**
D	**Target**
E	**Depositing**
G	**Resting**
F	**Reset.**

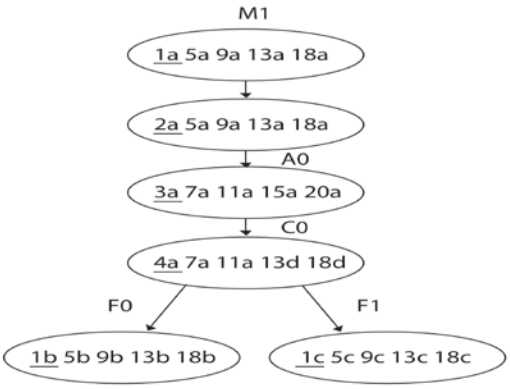
Fig. 3. Actions and action labels

TRANSITION	ACTION	LOCAL	GLOBAL	SYNC HRONOUS	ASYNC HRONOUS
CFSM 1					
1-2		√			√
2-3	A		√	CFSMs:1,2,3,4,5	
3-4	C		√	CFSMs:1,4,5	
4-1	F		√	CFSMs:1,2,3,4,5	
CFSM 2					
5-6		√			√
6-7	A		√	CFSMs:1,2,3,4,5	
7-8	B		√	CFSMs:2,4,5	
8-5	F		√	CFSMs:1,2,3,4,5	
CFSM 3					
9-10		√			√
10-11	A		√	CFSMs:1,2,3,4,5	
11-12	D		√		√
12-9	F		√	CFSMs:1,2,3,4,5	
CFSM 4					
13-14		√			√
14-13		√			√
14-15	A		√	CFSMs:1,2,3,4,5	
15-16	B		√	CFSMs:2,4,5	
15-16		√			√
17-13	C		√	CFSMs:1,4,5	
15-17	E		√		√
16-13	F		√	CFSMs:1,2,3,4,5	
CFSM 5					
18-19		√			√
19-18		√			√
19-20	A		√	CFSMs:1,2,3,4,5	
20-21	B		√	CFSMs:2,4,5	
20-21		√			√
22-18	C		√	CFSMs:1,4,5	
20-22	G		√		√
21-18	F		√	CFSMs:1,2,3,4,5	

Fig. 4. Table listing global- and local-state transitions







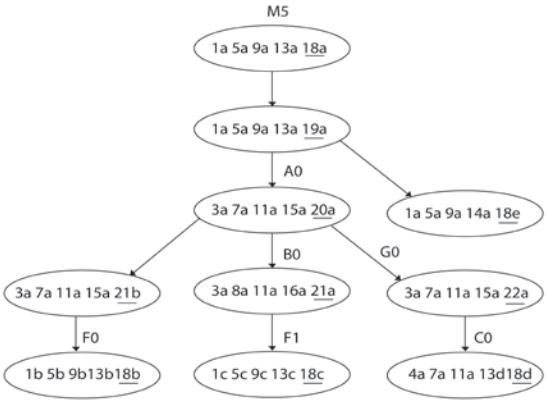


Fig. 5. (M1, M2, M3, M4, M5-CMPMs)

4. MODEL-CHECKING

Model-checking is the appropriate technique where there are many different scenarios of interaction between components in a system. The most effective and successful formal technique is model-checking. This is a highly automated approach used to verify that a formal model of a system satisfies the set of desired properties.

The complexity of model-checking is that it involves checking all the states at most once. Thus, the complexity is linear in the number of total CMPM states, *N*.

Our model-checking consists of reachability analysis of the state vectors. The CMPMs model has distributed the synchronous global-state vectors into *n* interactive components. Model-checking provides an approach to check the *safety* and *liveness* properties of finite-state systems. The global-state transition graph of traditional *product-based model* [11] is viewed as a single *finite Kripke structure*. The main difficulty in using the model-checking approach with single Kripke structure is the *state-explosion problem* which is avoided using our CMPMs model which represents a *concurrent set of finite Kripke structures* which can be parallely explored, thus reducing the state-space explosion.

4.1 MODEL-CHEKING OF FAULT TOLERANCE IN CCRS

As explained in the introduction, the fault-tolerance properties consist of *safety* and *liveness*. Safety properties mainly consist of guarantee of the absence of *communication deadlocks*. A *deadlocked state* is a state where there is no outgoing transition. Liveness properties consist of *eventuality* guarantee which means that eventually certain global-state vectors are *reachable*. A *reachable state* is a state such that there exists a path from the initial state to the state in question in the CMPM tree.

We assume that all the interesting global-state vectors of scrutiny are synchronous global-state vectors whose reachability can be analysed in a parallel fashion by making depth-first analysis of the CMPM trees independently and thus concurrently.

4.2 THE MODEL-CHECKING ALGORITHM

Detection of communication deadlocks

/* Parallel checking of all *n* CMPM-trees M_i, i= 1..n to check the reachability of *leaf-states* that are *non-cutoff states*. */
/*S_{fi} Set of states
R_{fi} Set of transition relations of the form (s_{fi}, a_{fi}, s'_{fi})
S_{0fi} Set of initial CFSM states

S_i Set of instances of states
E_i set of events which are instances of actions
R_i set of transitions of the form (s_i, e_i, s'_i)
S_{0i} set of initial states*/

deadlock_state_list$_i$: List of deadlocked states from M_i, i= 1..n;/*set of n trees*/
find_dead_states$_i$(s_i) /*s_i set of instances of states*/
{
if s_i is a (*leaf-state* $\wedge$ not (cut_off_state))
{
add (s_i, dead_state_list$_i$);
return;
}
else if s_i is a leaf-state
return;
for all next_states s'_i of s_i
/* environment of O/P (s'_i) */

return(find_dead_states$_i$(s'_i));
}

Main()
{
Deadlock_state_list$_i$:= Null, for all i=1..n;
Par begin
For i=1..n do find_dead_states$_i$(s_{0i});
/*S_{0i} set of initial states*/
Par end;
}

Detection of Liveness Property

/* This involves checking all the k CMPM trees Mi, i=1..k
For the reachability of the given synchronous state vector.*/

Chk_tree$_i$(s_i, s_f) /*Set of state in CFSM*/
/* s_f := Fsm_vector(s_i)*/
{
if (id(s_i) = s_{fi} $\wedge$ id(env$_j$(s_i)) = s_{fj}
/*I/P is equal to id of s_i in tree*/
/*(s_{fi})=initial state of CFSM */

for all j = 1..k, k$\neq$ i)
return(true);
else if s_i is a leaf state
return(false) ;
else for all next states s'_i

```
        such that: (s_i R_i s'_i) is a transition do
    {
      success_i:= Chk_tree_i(s'_i, s_f);
                              if (success_i) return(true);
                              else continue;
    }
}/*Chk_tree_i() */

Main()
{
    Par begin
      for i = 1..k do
              success_i := Chk_tree_i(s_0i, s_f);
    Par end;
 }/*Main()*/
```

4.3 Complexity of the Model-Checking Algorithms

Since the procedure of distributed model-checking is *recursive*, the proof of correctness can be done by *inductive generation* of state-vector space.

The time complexities of both the algorithms involve *depth-first* recursive *search* of at most all *n* CMPM trees in parallel, checking all the states of each CMPM tree at most once. Thus, they are linear in the number of total states, N, of all the component CMPMs.

5. CONCLUSIONS AND FUTURE WORK

The structure theory aims at overcoming the state-space explosion problem, inherent to the analysis of concurrent systems, by bridging structural and behavioral properties.

We have proposed a couple of model-checking algorithms based on CMPMs model to verify the fault-tolerant properties namely safety and liveness. Safety involves the detection of communication deadlocked states. Liveness property involves the eventual occurrence of certain required synchronous global-state vectors.

We have provided the generation and model-checking algorithms using a CCRS. The future work we propose is a *swarm intelligent robotic system*, which consists of *hundreds* of *identical* robots working concurrently on a common goal unlike our CCRS with a *few, not-necessarily identical* robots.

References

[1] Kondo T., Ito K., "A reinforcement learning using adaptive state space construction strategy for real autonomous mobile robots", 41st SICE Annual Conference, Vol 5, pp3139-3144, Aug 2002.

[2] Vasumathi, K. Narayanan, "A state-oriented, Partial-order model and Logic for Distributed Systems Verification, Ph.D. Thesis, Concordia University, Montreal, 1997.

[3] R.Kurshan et al.,"Lessons Learned from Model-checking a NASA robot controller", Formal Methods in System Design, Vol-25, No:2-3, pp241-270.

[4] E.Teruel, M.Silva et al, "Choice-free Petri nets: a model for deterministic concurrent systems with bulk services and arrivals" ,IEEE Transactions on Systems, Man and Cybernetics", Vol 27, Jan 1997, pp73-83.

[5] Bernardi S, Campos J., "Computation of performance of Real-time systems using Timed Petri-nets, IEEE Transactions on Industrial Informatics", May 2009, Vol 5, No:2, pp168-180.

[6] D.A. Stuart et al., "Simulation-Verification: Biting at the state-explosion problem", IEEE Transactions on SE, Vol 27, No:7, July 2001.

[7] Peter Bokor et al, "Efficient modelchecking of fault-tolerant distributed protocols" DSN 2011, pp.73-84.

[8] Edmund Clarke et al, "Progress on the state-explosion problem in model-checking", Informatics 2001, pp. 176-194.

[9] K.L. McMillan, "Symbolic Model Checking: An approach to the state explosion problem" Ph.D. Thesis, May, 1992 CMU-CS-92-131.

[10] E. M. Clarke and E. A. Emerson. "Synthesis of synchronization skeletons for branching time temporal logic", In Logic of Programs:Workshop, LNCS, 1981.

[11] E. M. Clarke, E. A. Emerson, and A. P. Sistla. "Automatic verification of finite-state concurrent system using temporal logic", In Proceedings of the Tenth Annual ACM Symposium on Principles of Programming Languages (POPL), January 1983.

[12] E. M. Clarke Jr., E. A. Emerson, and A. P. Sistla. Automatic verification of finite-stateconcurrent systems using temporal logic specifications. ACM TOPLAS, 8(2):244–263, Apr.1986.

[13] E. Clarke, O. Grumberg, and D. Peled. "Model Checking", MIT Publishers, 1999.

[14] E. Clarke and H. Schlingloff. Model checking. In J. Robinson and A. Voronkov, editors,Handbook of Automated Reasoning. Elsevier, 2000.

[15] J.Park et al,"A Theorem Prover for Boolean BI",ACM International conference on Principles of Programming Language, POPL 2013.

[16] C.A.R.Hoare, "Communicating Sequential Processes", Prentice Hall 1984.

[17] Sungeetha Dakshinamurthy and Vasumathi Narayanan, "A fully-distributed checkpointing-protocol for fault-tolerance in real-time distributed systems", in National IETE Conf., 2012.

[18] Sungeetha Dakshinamurthy and Vasumathi Narayanan, "A parallel algorithm for model-transformation of interactive state machine specification",Inetrnational Journal of Wisdom Based Computing ,Vol. 2 No:1,pp 52-57,Apr 2012 .

[19] Sungeetha Dakshinamurthy and Vasumathi Narayanan, "A Model-checking Algorithm for Formal verification of Peer-to-peer Fault-Tolerant Networks", (LNIT ISSN 2301-3788) Lecture Notes on Information Theory, Vol. 1 No:3,pp 128-131,Sep 2013 .

[20] Sungeetha Dakshinamurthy and Dr.Vasumathi Narayanan "A Component-based Approach to Verification of Formal Software Models to Check Safety Properties of Distributed Systems" (LNSE ISSN 2301-3559) Lecture Notes on Software Engineering,Vol. 1 No:2,pp 186-189,Apr 2013 .