Constraint-Oriented Model for Describing Distributed Cooperative Systems and Efficient Deadlock Detection Using Symmetries

Takaaki Umedu†, Hirozumi Yamaguchi†, Keiichi Yasumoto††,
Akio Nakata† and Teruo Higashino†

† : Department of Informatics and Mathematical Science, Osaka University
1-3 Machikaneyama-cho, Toyonaka Osaka 560-8531, Japan
{umedu,h-yamagu,nakata,higashino}@ics.es.osaka-u.ac.jp

†† : Department of Information Processing and Management
1-1-1 Bamba, Hikone Shiga 522-8522, Japan
yasumoto@biwako.shiga-u.ac.jp

Abstract

In this paper, we introduce a formal model for designing distributed cooperative systems (concurrent systems) with symmetries and propose an efficient deadlock detection method on this model. In our method, we describe a specification of a system by a set of coloured Petri-nets and synchronization among them. Each coloured Petri-net is either the specification of a participant's behavior or the constraint about the temporal ordering of multiple participants' behavior. For this specification, if given constraints are inconsistent with each other, the total system enters a deadlock state. In general, the reachability analysis for such systems may cause the state explosion problem depending on the size of the system. However, there are a lot of cases that multiple participants carry out the same behavior in distributed cooperative systems such as network meeting. In such symmetric specifications, by merging equivalent states in a given specification, we can reduce the cost necessary for the reachability analysis. Here, we propose an efficient reachability analysis method using symmetries. We have also developed a verification tool based on the method and shown the usefulness of the method using some examples.

1 Introduction

According to the progress of high-speed networks in recent years, many distributed cooperative systems such as network meeting, remote lecturing and distributed multimedia authoring have been developed. In such distributed cooperative systems, the number of participants is often changed, and various constraints are added depending on the number of participants and network environment. However, if given constraints are inconsistent with each other, there may not exist executable behavior satisfying all the constraints, i.e., the given system may enter a deadlock state. In order to develop high reliable distributed systems, in this paper, we propose a model for specifying such a distributed cooperative system with unspecific number of participants and an efficient reachability analysis method for detecting the deadlocks in the model.

To specify such distributed cooperative systems simply and hierarchically, we use a constraint-oriented style, which is close to the description styles in [1, 2]. Such a constraint-oriented style is also used in the formal specification language LOTOS [3]. In the proposed constraint-oriented style, we describe a specification of a distributed cooperative system (concurrent system) as (A) a set of coloured Petri-nets and (B) synchronization among them. Each Petri-net describes either (A-1) each participant's behavior or (A-2) the temporal ordering of multiple participants' actions and/or constraints among those participants. Here, we call the above (A-1) and (A-2) descriptions as a *behavior net* and a *constraint net*, respectively. In a behavior net, each participant's behavior is specified independently of other participants' behavior. Many and unspecific participants' behavior can be specified as a coloured Petri-net with multiple coloured tokens if those participants' behavior is essentially the same. In constraint nets, mutual exclusion among behavior nets and/or constraints as the total system can be specified. In the description of synchronization in the above (B), by letting an action in a behavior net and the same action in a constraint net be executed simultaneously, we can make the temporal ordering of the actions in behavior nets satisfy all the given constraints. Moreover, we can specify not only *one-to-one* synchronization but also *n-to-k* synchronization where arbitrary k processes in given n processes satisfying the constraints can synchronize with each other.

If we use this model to describe the specification of a total system, by changing the constraint nets, we can easily modify the total behavior of the system. However, the system may have the possibility to reach a deadlock state if we specify inconsistent constraints simultaneously. Since general reachability analysis techniques for coloured Petri-nets can detect such deadlocks if the number of states is finite, we impose a restriction on our model that limits the number of reachable states to finite. But if we use such techniques simply the cost for the verification still becomes large and the state explosion problem may occur.

In order to reduce the verification costs, Ref. [4] propose techniques to merge equivalent states into one and make a reduced size's reachability graph called *OS graph* [5] from the original reachability graph. Ref. [6, 7] propose another kind of efficient reachability analysis techniques using *symbolic reachability graph*. Ref. [8, 9] use invariants for reducing the verification costs. Ref. [10, 11] use stochastic Petri-nets, and Ref. [12] uses compositional high level Petri-nets for efficient reachability analysis.

There have not been proposed general techniques for finding equivalence relation between two reachable states mechanically from given specifications. However, in distributed cooperative systems, there are a lot of cases that multiple participants carry out essentially the same behavior and they do not cause different results for reachability analysis. For example, in a simple network meeting system where only one of multiple participants can be a speaker at each moment, the number of the participants does not affect the reachability analysis of the system specification, since the behavior of those participants can be regarded as the same, i.e., the specification has symmetries.

Here, we propose an efficient reachability analysis method where equivalence relation between two reachable states is found mechanically from a given specification by using the information about the symmetries. Moreover, we can reduce the number of reachable states, by some reduction rules based on the symmetries applied for CPN before reachablity analysis. We have also developed a verification tool and shown the usefulness of the method using some examples of network meeting systems.

The paper is organized as follows. Section 2 explains our constraint-oriented model and the details of coloured Petri-nets. Section 3 describes the reachability analysis method using symmetries. The experimental results are also given. In Section 4, we give the conclusion.

2 Specification of Distributed Cooperative Systems in Proposed Model

2.1 Petri Net and Coloured Petri Net

A Petri-net (PN in short) is a weighted directed graph that consists of two types of nodes, *places* and *transitions*. Each directed edge a between a place and a transition is called an *arc* where an integer called *weight* (denoted as $W(a)$) is associated. Each place may have *tokens*, and an assignment of tokens to places is called a *marking*. A marking m assigns $m(p)$ tokens to place p. A transition t may fire *iff* each its input place p of t connected by an arc a has $W(a)$ tokens. If t fires, $W(a)$ tokens are taken from each input place p, and $W(a')$ tokens are added to each output place p' of t connected by an arc a'.

A coloured Petri-net (CPN in short) is a high level Petri net[13] where each token has a value. The values are distinguished by types called *colours*. The colours are, for example, integers, real numbers and characters. A colour is associated with each token of CPN, The weight $W(a)$ of an arc a in CPN is a multi-set of *binding variables*. And the assignment of tokens $m(p)$ represented by a marking m to a place p is also a multi-set of tokens. Here, a multi-set is a set that may include more than one identical element. For each binding variable, a colour is associated and a token with the same colour can be assigned to the binding variable. Hereafter, an assignment of a multi-set of tokens to a multi-set of binding variables is simply called a *binding*. For each transition t, a boolean function of binding variables that appear in the weights of the incoming arcs of t is associated and called a *guard*.

A transition t may fire *iff* (i) each input place p of t connected by an arc a has a multi-set of tokens that can be assigned to $W(a)$ and (ii) the value of the guard of t on this binding is true. If t fires, the multi-set of tokens are taken from each input place p of t, and the multi-set of tokens $W(a')$ that are determined by each binding to $W(a)$ are added to each output place p' of t connected by an arc a'. Note that the binding variables in the

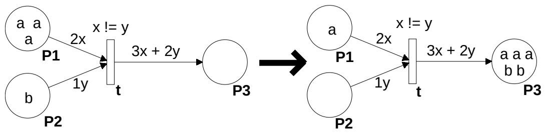

Figure 1. Firing of Transition in CPN

weight $W(a')$ of each outgoing arc a' must appear in at least one of the weights of the incoming arcs of t.

Fig. 1 shows an example. Suppose that x and y are binding variables and a and b are values of the same colours as x and y, respectively. $kx + hy$ represents a multi-set of x and y where the numbers of x and y are k and h, respectively. In this case, since a binding $[x = a, y = b]$ satisfies the guard "$x! = y$"$(x \neq y)$ of t true, t can fire on this binding. If t fires, tokens a and b are taken from places $P1$ and $P2$ respectively, and the multi-set of tokens $3a + 2b$ is added to place $P3$.

2.2 CPN Specification in Constraint-Oriented Style

CPN is a suitable model to describe specifications of distributed cooperative systems. This is mainly because the same behavior of participants (*e.g.* the behavior of students in remote lecturing), can be described as a single net where each coloured token represents an individual participant. However, for the efficient design of distributed cooperative systems where the temporal ordering of actions of participants should be specified, we introduce the concept of constraint-oriented style into CPN.

A specification of a distributed cooperative system written in CPN in constraint-oriented style consists of a set of *behavior nets*, a set of *constraint nets* and *synchronization.* Each behavior net includes tokens that represent participants. Each constraint net specifies the temporal ordering of transitions in behavior nets. Each synchronization associates each transition of a constraint net with one of the transitions in behavior nets. For each synchronization, a boolean function of binding variables of the two transitions can be specified as a guard.

The CPN specification of n participants' system (for simplicity, participants are denoted as integers $1..n$ hereafter) with k different types of behavior ($k \leq n$) consists of k behavior nets, h constraint nets ($h \geq 0$) and l synchronization ($l \geq 0$). We assume that the specification must be described in the following style.

- Hereafter, each behavior net is denoted as BN_i $(1 \leq i \leq k)$. BN_i contains a set of tokens t_i of the colour "person", an enumerative type with elements $1..n$. Here (a) $\cup_{1 \leq j \leq k} t_j = \{1..n\}$ and $t_j \cap t_{j'} = \emptyset$ must hold, (b) the set of existing tokens in BN_j is always equals to t_j. These indicate that each token in BN_i represents a participant. Moreover, each transition of BN_i has guard "true".
- Hereafter, each constraint net is denoted as CN_j $(1 \leq j \leq h)$. The colours allowed to use in constraint nets are "person", the limited number of integers and "e", which represents an empty colour (tokens that have no value). Here, considering the need for specifying "arbitrary number of participants", we introduce a new colour "variant" for the binding variables of constraint nets. A binding variable of this colour is a special variable where any multi-set of tokens can be assigned. Each CN_j must be bounded. Moreover, each transition of CN_j has guard "true".
- Hereafter, each synchronization is denoted as $sync_x$ $(1 \leq x \leq l)$. For each $sync_x$ that associates t_v of CN_j with t_u of BN_i, a boolean function of binding variables used in the weights of the incoming arcs of t_u or t_v can be specified as a guard. Here, we introduce a special function $\#(v)$ of binding variable v of colour "variant". This function returns the number of tokens that are assigned to v.

2.3 Example Specification

Fig. 2 and Table 1 shows an example specification of a simple network meeting system written in CPN in this style, where one presenter and three audiences participate in the meeting. BN_1 and BN_2 are behavior nets that represent the behavior of the presenter and the audiences, respectively. CN_1 and CN_2 are constraint nets. Six synchronization with guards $sync_1$, .. and $sync_6$ are specified in Table 1 and represented as arrows in Fig. 2.

In BN_1, two actions "PRESENT_START" (start presentation) and "PRESENT_END" (end presentation) are specified for the presenter. In BN_2, two actions "QUESTION_START" (start a question) and "QUESTION_END" (end a question) are specified for the three audiences. x and y are binding variables of colour "person". For these behavior nets, the two constraint nets and synchronization specify the temporal ordering of their actions. They represent the following constraints.

(i) Questions must not be started before the presentation is started.

(ii) Each audience may ask a question only once before the presentation is stopped.

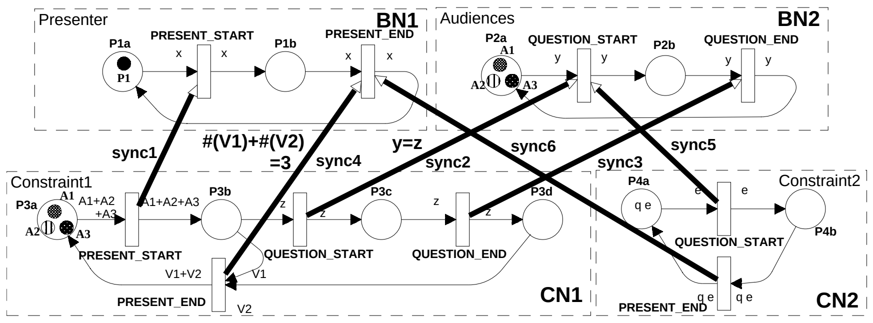

Figure 2. CPN Specification of Simple Network Meeting System in Constraint Oriented Style

	Behavior Net		Constraint Net		Guard
	Net	Transition	Net	Transition	
$sync_1$	BN_1	PRESENT_START	CN_1	PRESENT_START	true
$sync_2$	BN_2	QUESTION_START	CN_1	QUESTION_START	$y = z$
$sync_3$	BN_2	QUESTION_END	CN_1	QUESTION_END	true
$sync_4$	BN_1	PRESENT_END	CN_1	PRESENT_END	$\#(V1) + \#(V2) = 3$
$sync_5$	BN_2	QUESTION_START	CN_2	QUESTION_START	true
$sync_6$	BN_1	PRESENT_END	CN_2	PRESENT_END	true

Table 1. Description of Synchronization

(iii) q questions must be asked before the presentation is stopped.

The constraint (i) is represented by CN_1 and two synchronization $sync_1$ and $sync_2$. In $sync_1$, two transitions "PRESENT_START" of BN_1 and CN_1 synchronize, and in $sync_2$, two transitions "QUESTION_START" of BN_2 and CN_1 synchronize. According to CN_1, "QUESTION_START" cannot be executed without the presence of tokens of colour "person" in place P_{3b} of CN_1. Those tokens are produced by the firing of "PRESENT_START". Therefore, "QUESTION_START" cannot fire before the firing of "PRESENT_START".

The constraint (ii) is represented by CN_1 and three synchronization $sync_2$, $sync_3$ and $sync_4$. In order to execute "PRESENT_END", the value of the guard of $sync_4$ must be true. The guard includes two binding variables $V1$ and $V2$ of colour "variant" where any multi-set of tokens can be assigned. Here, each token in place P_{3b} represents an audience who has not asked a question yet. On the other hand, each token in place P_{3d} represents an audience who has already asked a question by firing of "QUESTION_START" and "QUESTION_END". These tokens are removed if "PRESENT_END" fires because the guard of $sync_4$ "$\#(V1) + \#(V2) = 3$" needs all the three tokens in P_{3b} and P_{3d} to be assigned to $V1$ and $V2$. This means that each audience who has already asked a question is never allowed to ask a question once again before the firing of "PRESENT_END".

The constraint (iii) is represented by CN_2 and synchronization $sync_5$ and $sync_6$. By each firing of "QUESTION_START" in CN_2, token "e" is produced in place P_{4b}. The tokens in place P_{4b} represent the number of questions that have been asked after the presentation has been started. In order to end the presentation by the firing of "PRESENT_END", there must be q tokens of "e" in P_{4b}.

In Section 3, we explain how a specification described in this model is transformed into a pure CPN where the reachability analysis can be performed. The state space reduction by using the symmetries of CPN is also explained.

3 Reachability Analysis

In our model, since we specify a system specification as a set of behavior nets, constraint nets and synchronization among them, the total system may include deadlock states due to some constraints inconsistent with each other. Generally we can check whether a system includes a deadlock state or not by constructing the reachability graph of the system. Here, we adopt a policy to check the deadlock-free property or liveness property of a given system by constructing an occurrence graph[5], which is known as a kind of abstraction of the reachability graph. To construct an occurrence graph, we transform a given specification (consisting of several CPNs and synchronization among them) into

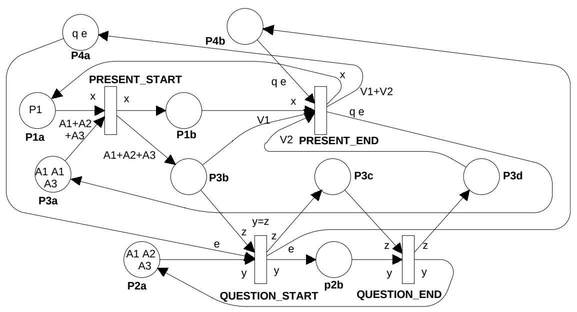

Figure 3. Specification of Total System Derived from Fig.2

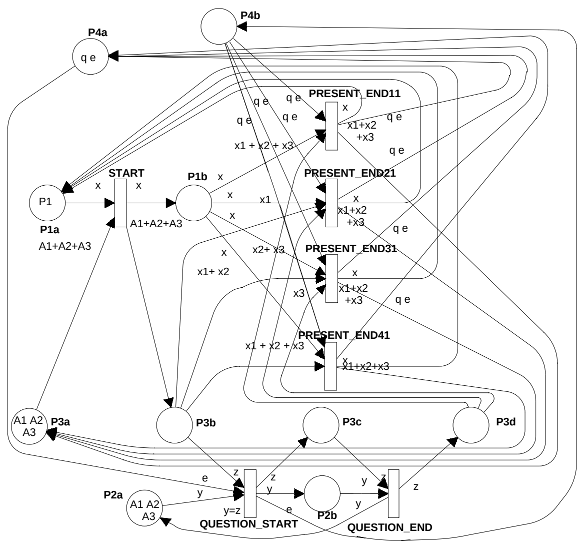

Figure 4. Transformed Specification of Total System

the equivalent single CPN. Then for the derived CPN, we construct the corresponding occurrence graph so that the number of nodes in the graph is reduced using symmetries.

3.1 Derivation of Single CPN from CPN Specification in Constraint-Oriented Style

The proposed transformation technique is as follows.

(i) Each pair of transitions in a constraint net and a behavior net that are specified to synchronize is merged into a single transition. If a guard is specified in the synchronization, the guard should be added as the guard function of the transition (Fig.3).

(ii) Each transition with variant type variables is replaced by a set of transitions without those variables (Fig.4). This replacement is carried out as follows.

   (a) Enumerate the possible bindings for the variant type variables, where each binding satisfies the specified guard functions.

   (b) For each possible binding, generate a new transition with appropriate tokens and variables for the binding.

To make sure that (a) is always possible, we suppose that the number of possible bindings for each variant type variable is finite. Fig.4 shows the CPN derived from the specification in Fig.2. As an example, the transitions PRESENT_END1,..,PRESENT_END4 in Fig.4 are derived by applying the step (ii). These transitions are generated from the combination of transitions PRESENT_END of BN_1 and CN_1 in Fig.2 where the variant type variables of PRESENT_END of CN_1 are replaced by the variables $x1$,..,$x3$. From the guard "$\#(V1) + \#(V2) = 3$", the possible bindings of $V1$ and $V2$ are the following four patterns: $\{(\#(V1), \#(V2))\} = \{(0,1),(1,2),(2,1),(3,0)\}$. Transitions PRESENT_END1,..,PRESENT_END4 correspond to those patterns, respectively. For example, variable $V1$ in PRESENT_END2 is replaced by variable $x3$ and variable $V2$ is replaced by $x1 + x2$ as a result of the binding. In this way, a set of concurrent finite CPNs including variant type variables are transformed into a single finite CPN.

3.2 Reachability Analysis with OS Graph

Let us suppose that a network conferencing system consisting of a chairman, k presenters and n audiences is modeled as a set of $k + n + 1$ CPNs in general. In this case, if we strictly distinguish the $k + n + 1$ types of tokens from each other, a quite large *occurrence graph* (reachability graph) will be derived from the given specification.

However, if we allow any audience to ask a question in a given system specification, we can regard that the global state transition of the system is the same whoever asks a question. In such a case, we need not distinguish each individual audience from the others in constructing the occurrence graph. Instead, we can construct the corresponding *OS graph* (occurrence graph with symmetries) where the number of possible states is reduced using symmetries, and carry out the reachability analysis efficiently. The OS graph is a reachability graph where each node corresponds to an equivalent class of states classified depending on the given symmetries. Since in the OS graph, a node can represent multiple states, the size of reachability graph can be reduced and we can efficiently carry out the reachability analysis.

The occurrence graph is defined as a pair (M, A). $M = \{m_1, .., m_n\}$ is the set of all possible markings and $A \subseteq M \times M$ represents a state transition. Here $(m_i, m_j) \in A$ holds if and only if marking m_j is reachable by the firing of one transition from marking m_i. The OS graph is a graph where the number of states is reduced by unifying "equivalent" states into a single state. For a given equivalence relation E, the OS graph is represented as a pair $(\mathbf{M}, \mathbf{A})$, where $\mathbf{M}$ is a family of sets of equivalent markings and $\mathbf{A}$ is a set of state transitions on $\mathbf{M}$. Hereafter, $[m]$ denotes the set of markings that are equivalent with m. $\mathbf{M} = \{[m_1], .., [m_m]\}$ is a family of sets of equivalent markings calculated by classifying M by E. $\mathbf{A}$ is defined such that

$$\forall m_i, m_j : (m_i, m_j) \in A \rightarrow ([m_i], [m_j]) \in \mathbf{A} \quad (1)$$

Here, if the following condition holds, it is known that $(\mathbf{M}, \mathbf{A})$ is a deadlock free OS graph if and only if the original occurrence graph is deadlock free[5].

$$\begin{aligned} &([m_i], [m_j]) \in \mathbf{A} \\ &\rightarrow \forall m'_i \in [m_i] : \exists m'_j \in [m_j] : (m'_i, m'_j) \in A \quad (2) \end{aligned}$$

Intuitive Proof : From the condition (1), if m_j is reachable from m_i in the graph (M, A), $[m_j]$ is also reachable from $[m_i]$ in the graph $(\mathbf{M}, \mathbf{A})$. From the condition (2), if $[m_j]$ is reachable from $[m_i]$ in the graph $(\mathbf{M}, \mathbf{A})$, for any marking $m'_i \in [m_i]$, there exists a marking $m'_j \in [m_j]$ where m'_j is reachable from m'_i. □

3.3 Sufficient Condition of Symmetries

There are no general methods for finding an equivalence relation satisfying the above (1) and (2) automatically. Here, we would like to give a sufficient condition for finding such an equivalence relation automatically. Here, we consider a set of *symmetric* tokens. The word "a set S of *symmetric* tokens" means that for the initial marking, weight and guard in a given specification, either the following (i) or (ii) holds;

(i) any colour of tokens contained in S is not specified.

(ii) all the colours of tokens contained in S are specified.

For example, for the specification in Fig.2, let S denote the set of tokens $\{A1, A2, A3\}$. In Fig.2, the initial marking contains either all elements in S or no elements in S. For each weight and guard, the same property holds. So, $S = \{A1, A2, A3\}$ can be regarded as a set of *symmetric* tokens.

For a given CPN specification in constraint-oriented style, by checking all initial marking, weights and guards step-by-step, we can extract the sets of tokens appeared in the initial marking, weights and guards. For each set of tokens, we can mechanically check whether either the above (i) or (ii) holds. Then, we can mechanically find a set of symmetric tokens for a given CPN specification if there exists such a set.

Hereafter, we will propose a method to generate an equivalence relation E satisfying the above conditions (1) and (2) mechanically from the derived set of symmetric tokens. Note that for a set S of symmetric tokens such as $S = \{A1, A2, A3\}$, let p_1 and p_2 denote two different lists containing all the tokens in S (for example, $p_1 = [A1, A2, A3]$ and $p_2 = [A2, A1, A3]$). Then, we say that p_1 is a *permutation* of p_2, vise versa.

Here, for a given set S of symmetric tokens, let us consider a relation E between two reachable markings m_i and m'_i where m'_i is obtained from m_i by replacing S in m_i with a *permutation* of S and vise versa. For example, for the set of symmetric tokens $S = \{A1, A2, A3\}$ in Fig. 2, two reachable markings $m_i = (\emptyset, P1, A2 + A3, A1, A2 + A3, A1, \emptyset, 2e, e)$ and $m'_i = (\emptyset, P1, A1 + A3, A2, A1 + A3, A2, \emptyset, 2e, e)$ satisfy the relation E since m'_i is obtained from m_i by replacing $[A1, A2, A3]$ in m_i with one of its permutations $[A2, A1, A3]$ and vise versa. In our method, this relation E is an equivalence relation.

Intuitively this is clear because all the symmetric tokens move in the same way and make no difference between two markings m_i and m'_i where a token proceeds in m_i and one of its symmetric tokens proceeds in m'_i. The sketch of proof is given as follows.

Assume that two reachable markings m_i and m'_i satisfy the relation E based on the set S of symmetric tokens, where m'_i is obtained by replacing S in m_i with one of its permutations (this permutation is denoted as p hereafter). Also assume a marking m_j reachable from m_i by the firing of a transition T on a binding B. Here, since the tokens of S are symmetric, those tokens in binding B can be replaced with the permutation p, and this replacement makes a new binding B'. Obviously, since m'_i is obtained by the same permutation p, there exists a state transition from m'_i to a marking m'_j by the firing of the same transition T on binding B'. Then we can say that m_j and m'_j satisfy the relation E by the same permutation p, since permutation p replaces the tokens of S in m_j that are in each input (or output) place of T with the ones in the same input (or output) place in m'_j. Consequently, for every marking $m'_i \in [m_i]$, there exist a marking $m'_j \in [m_j]$ and a state transition from m'_i to m'_j $((m'_i, m'_j) \in A)$. Therefore, the sufficient condition for equivalence relation in

Section 3.2 holds.

Now we can say that since the two markings $m_i = (\emptyset, P1, A2 + A3, A1, A2 + A3, A1, \emptyset, 2e, e)$ and $m_i' = (\emptyset, P1, A1 + A3, A2, A1 + A3, A2, \emptyset, 2e, e)$ in Fig.2 satisfy the above equivalence relation E, they are merged in the corresponding OS graph.

3.4 Further Reduction of CPN and Omitting Colour Information

In our method, we use the following techniques for reducing the size of CPNs so that the reachability analysis can be efficiently carried out.

- A consecutive sequence of transitions is transformed into one transition. The transitions that will obviously fire sequentially and are not specified to synchronize with other CPNs are replaced by one transition.
- If there are consecutive transitions t_1, t_2 in a net and consecutive transition t_1', t_2' in another net, and if t_1 and t_2 synchronize with t_1' and t_2', respectively, then the two synchronization relation can be merged into one synchronization.
- The synchronization guard checking the colours of tokens can be deleted if the guard is always true in checking the condition of firing for any reachable marking and there is no more synchronization that is specified with guard in a given specification.
- The colour information of a set of tokens can be omitted, if the colours of these tokens are never checked anywhere. To do so, we introduce a new token that represents all of the tokens belonging to the set, and replace the existing tokens by the new token.

For example, in Fig.2 the synchronization 2 and 3 are sequential and their guards are the same. So, each QUESTION_START and QUESTION_END can be combined into one transition T. As a result, a set of tokens $\{A1, A2, A3\}$ is obviously always placed on the input place of the transition T of BN2. And other tokens except tokens $A1$, $A2$ and $A3$ are never placed on the input place of transition T of CN1. So, if a token is on the place of CN1, this token must be also placed on the place of BN2 and the guard $y = z$ holds. And since no other synchronization between BN2 and CN1 is specified, this guard of synchronization can be deleted. As a result, since there is no specification that distinguishes tokens $A1$, $A2$ and $A3$, they can be regarded as the same token. So we can replaces them by a new token A for reachability analysis.

3.5 Reachability Analysis System

In order to evaluate our method, we have developed a system to derive a single CPN from a given specification consisting of behavior nets, constraint nets and synchronization. This system derives a single CPN by coupling the transitions of the behavior nets and constraint nets specified to synchronize for some specific cases. And then it reduces the size of CPN by picking up the symmetric tokens and by omitting the colour information needless to distinguish. Then we check the deadlock-free property with a general formal model checker for CPN.

3.6 Experimental Result

We have used a tool to design and simulate Petri-nets called Design/CPN[14] in our reachability analysis system. We have measured the time to examine the deadlock-free property of the example of Fig.2. For this example, we have checked the example as changing the number of audiences and the number of questions. Table 2 shows the results. In this table, the size of the reachability graph and the time consumed for the calculation are shown for the following three cases: (i) the case of generating occurrence graph without considering symmetries; (ii) the case of making the OS graph by using symmetries; and (iii) the case of making the OS graph after omitting some colour information. In case (ii), the size of the graph is very small compared with case (i), while the consumed time may increase due to calculation of symmetries. In case (iii), the consumed time is substantially reduced, especially in large specifications.

4 Conclusion and Future Work

We have proposed a constraint-oriented model for developing distributed cooperative systems with symmetries. In our model, we describe the specification of a system by a set of coloured Petri-nets and synchronization among them. A specification of each node is described independently and the interactions among them are specified using constraints and synchronization among them.

We have also proposed a method for efficient reachability analysis for this model. In our method, the symmetries are automatically detected from a given specification and the reachability analysis is quickly carried out by making an OS graph using the symmetries. We have adopted this method for an example, and we can reduce the size of reachability graphs and the required time for reachability analysis.

	(i) Occurrence Graph			(ii) OS Graph			(iii) OS Graph (Omit Colour Information)		
Audience	#Nodes	#Arcs	Time	#Nodes	#Arcs	Time	#Nodes	#Arcs	Time
3	28	61	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
4	66	177	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
5	132	451	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
6	234	1333	7 Sec.	11	14	3 Sec.	11	14	1 Sec.
7	380	6063	212 Sec.	11	14	586 Sec.	11	14	1 Sec.
15							11	14	3 Sec.
16							11	14	11 Sec.
17							11	14	17 Sec.

Table 2. Experimental Result

To extend this model so that we can specify time constraints and to develop a method of efficient reachability analysis for such a model are our future work.

References

[1] T. Bolognesi : "Toward Constraint-Object Oriented Development", *IEEE Trans. on Software Eng.*, Vol. 26, No. 7, pp. 594 – 616 (2000).

[2] C. A. Vissers, G. Scollo and M. v. Sinderen : "Architecture and Specification Style in Formal Descriptions of Distributed Systems", *Proc. 8th Int. Symp. on Protocol Specification, Testing, and Verification (PSTV-VIII)*, pp. 189–204 (1988).

[3] ISO : "Information Processing System, Open Systems Interconnection, LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", IS 8807 (1989).

[4] J. B. Jorgensen and L. M. Kristensen : "Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Colored Petri Nets and Occurrence Graphs with Symmetries", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 10, No. 7, pp.714–722 (1999).

[5] K. Jensen : "Coloured Petri Nets", *EATCS Monographs in Theoretical Computer Science* Vol. 2., Springer-Verlag, (1997).

[6] N. Hameurlain and C. Sibertin-Blanc : "Finite Symbolic Reachability Graphs for High-Level Petri Nets", *Proc. of 4th Asia-Pacific Software Engineering and Int. Computer Science Conference (APSEC '97 / ICSC '97)*, pp. 150–159 (1997).

[7] J. Cortadella : "Combining Structural and Symbolic Methods for the Verification of Concurrent Systems", *Proc. of Int. Conf. on Application of Concurrency to System Design (CSD '98)*, pp. 152–157 (1998).

[8] T. Miyamoto and S. Kumagai : "Calculating Place Capacity for Petri Nets Using Unfoldings", *Proc. of Int. Conf. on Application of Concurrency to System Design (CSD '98)*, pp. 143–151 (1998).

[9] M. Nakamura, Y. Kakuda and T. Kikuno : "Analyzing Non-determinism in Telecommunication Services Using P-invariant of Petri-Net Model", *Proc. of INFOCOM '97*, pp.1253–1259 (1997).

[10] L. Capra, G. Franceschinis, C. Dutheillet and J. M. Ilie : "Towards Performance Analysis with Partially Symmetrical SWN", *Proc. of 7th Int. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 148–155 (1998).

[11] R. Gaeta : "Efficient Discrete-Event Simulation of Colored Petri Nets", *IEEE Trans. on Software Engineering*, Vol. 22, No. 9, pp. 629–639 (1996).

[12] B. Grahlmann and H. Fleischhack : "Towards Compositional Verification of SDL Systems", *Proc. of 31st Hawaii Int. Conf. on System Sciences (HICSS'98)*, pp. 404–414 (1998).

[13] K. Jensen and G. Rozenberg (eds.) : "High-level Petri Nets. Theory and Application", Springer-Verlag, (1991).

[14] CPN group at the University of Aarhus, Denmark : "Design/CPN", Ver. 4.0.4, *http://www.daimi.aau.dk/designCPN/*