A Study on Synthesizing State Machines from Multiple Communication Diagrams

Toshiki Kinoshita[1] and Toshiyuki Miyamoto[2]
[1,2]Graduate School of Engineering, Osaka University
2-1 Yamada-oka, Suita, Osaka 565-0871 Japan
E-mail : [1]kinoshita@is.eei.eng.osaka-u.ac.jp, [2]miyamoto@eei.eng.osaka-u.ac.jp

Abstract: For a service-oriented architecture based system, the problem of synthesizing a concrete model for each service configuring the system from an abstract specification, which is referred to as choreography, is known as the choreography realization problem. In this paper, we assume that choreography is given by two acyclic relations. We study the conditions for acyclic relations to synthesize concrete models.

1. Introduction

The internationalization of business activities and information and communication technology have intensified competition among companies. Companies under the pressure have to respond to the change of business needs, and the time frame for making changes to existing business and launching new businesses should be shortened. Therefore, the need to quickly change or build information systems has been increasing. Under such circumstances, service-oriented architecture(SOA)[1] has been attracting attention as the architecture of information systems. In SOA, an information system is built by composing independent software units called peers.

In this paper, we consider the problem of synthesizing a concrete model from an abstract specification. The concrete model describes the behavior of peers and the abstract specification describes how peers interact with each other. It is not easy for designers to design a concrete model directly from requirements because huge gaps exist between requirements and concrete models. In contrast, defining an abstract specification is relatively simple, and we can verify its accuracy easily. Therefore, if we can automatically synthesize a concrete model from an accurate abstract specification, the designer's workload would decrease significantly and product quality would improve.

In SOA, the problem of synthesizing a concrete model from an abstract specification is known as the choreography realization problem (CRP)[2]. The abstract specification, called *choreography*, is defined as a set of interactions among peers, which are given by a dependency relation among messages; the concrete model is called *service implementation*, which defines the behavior of the peer. This paper uses the communication diagram and the state machine of Unified Modelling Language (UML) 2.x[3] to describe the choreography and service implementation, respectively.

Miyamoto et al. proposed a method for synthesizing hierarchical state machines from the choreography given in communication diagrams called the Construct State machine Cutting Bridges (CSCB) method[2]. In [4], a new notion called re-constructible decomposition of acyclic relations was introduced; a necessary and sufficient condition for a decomposed relation to be re-constructible was shown. It has already shown that the condition for the behavioral model is given by lower and upper bounds of the acyclic relation. The CSCB method assumes that choreography is defined by only one communication diagram. However, this is restrictive because a system must work well in a variety of cases. In this paper, we study the CRP when the choreography is defined by two communication diagrams.

The remainder of this paper is organized as follows. In Sect.2, a notion called re-constructible decomposition of acyclic relations and a subset of UML called *subset of UML for formally describing choreography and behavioral feature*(cbUML) are introduced. In Sect.3, we introduced CSCB method and Sect.4 shows the main result of this paper.

2. Preliminaries

2.1 Re-constructible Decomposition

Let Σ be a finite set and $\mathcal{R}$ be a relation on Σ. The transitive closure and reduction of $\mathcal{R}$ is denoted by $\mathcal{R}^+$ and $\mathcal{R}^-$, respectively. A relation $\mathcal{R}$ is called cyclic if e_1 and $e_2 \in \Sigma$ exist such that $(e_1, e_2) \in \mathcal{R}$ and $(e_2, e_1) \in \mathcal{R}^+$; otherwise it is called acyclic. Hereinafter, we assume that every relation is acyclic.

The set of all topological sorts of an acyclic directed graph$(\Sigma, \mathcal{R})$ is denoted by $\mathfrak{L}(\mathcal{R})$. A topological sort is called a word and their set is called a language.

Let $\mathcal{C}$ be a set, $\{\Sigma_c\}$ be a partition of Σ wrt $c \in \mathcal{C}$. Let $\mathcal{R}_c$ be a relation on Σ_c and their set be $\{\mathcal{R}_c\} = \{\mathcal{R}_c \subseteq \Sigma_c^2 \mid \Sigma_c \in \{\Sigma_c\}\}$. A relation $\mathcal{R}_{\rm com} \subseteq \mathcal{R} \setminus (\bigcup_c \Sigma_c^2)$ is called a communal relation of $\mathcal{R}$.

Definition 1 (Re-constructible Decomposition) Given a set $\{\mathcal{R}_c\}$ and a communal relation $\mathcal{R}_{\rm com}$, the relations $\{\mathcal{R}_c\}$ are re-constructible to $\mathcal{R}$ if $\mathfrak{L}(\mathcal{R}_{\rm com} \cup \bigcup_c \mathcal{R}_c) = \mathfrak{L}(\mathcal{R})$.

Relations $\mathcal{R}_c^{\min}, \mathcal{R}_c^{\max}, \mathcal{R}^{\min}$, and $\mathcal{R}^{\max}$ are defined as follows: $\mathcal{R}_c^{\min} = \Sigma_c^2 \cap \mathcal{R}^-, \mathcal{R}_c^{\max} = \Sigma_c^2 \cap \mathcal{R}^+, \mathcal{R}^{\min} = \mathcal{R}_{\rm com} \cup (\bigcup_c \mathcal{R}_c^{\min})$, and $\mathcal{R}^{\max} = \mathcal{R}_{\rm com} \cup (\bigcup_c \mathcal{R}_c^{\max})$, where $\mathcal{R}_c^{\min}, \mathcal{R}_c^{\max}, \mathcal{R}^{\min}$, and $\mathcal{R}^{\max}$ are acyclic because they are sub-relations of $\mathcal{R}$.

Under Assumption 1, Theorem 1 holds [4].

Assumption 1 $\mathfrak{L}(\mathcal{R}) = \mathfrak{L}(\mathcal{R}^{\min})$.

Theorem 1 $\{\mathcal{R}_c\}$ is re-constructible iff $\forall c : \mathcal{R}_c^{\min} \subseteq \mathcal{R}_c \subseteq \mathcal{R}_c^{\max}$.

2.2 cbUML

Let us introduce a subset of UML called cbUML. The complete set of cbUML is described in [5]. This section shows a

simplified version of cbUML, which is sufficient for the discussion of this paper.

Definition 2 (cbUML) A cbUML model is a tuple $(\mathcal{C}, \mathcal{M}, \mathcal{A}, \mathcal{CD}, \mathcal{SM})$, where $\mathcal{C}$ is the set of classes, $\mathcal{M}$ is the set of messages, $\mathcal{A}$ is the set of attributes, $\mathcal{CD}$ is the set of communication diagrams, $\mathcal{SM}$ is the set of state machines.

One class exists for each peer, and a state machine defines its behavior. A communication diagram describes a scenario, which is an interaction of peers.

2.2.1 Messages

The set of messages is partitioned by the type of messages: $\mathcal{M} = \mathcal{M}_{sop} \cup \mathcal{M}_{aop} \cup \mathcal{M}_{rep}$, where $\mathcal{M}_{sop}$ is the set of synchronous messages generated by synchronous calls, $\mathcal{M}_{aop}$ is the set of asynchronous messages generated by asynchronous calls, and $\mathcal{M}_{rep}$ is the set of reply messages to synchronous messages. Let $\mathcal{M}_s = \mathcal{M}_{sop}$ and $\mathcal{M}_a = \mathcal{M}_{aop} \cup \mathcal{M}_{rep}$. Correspondings between the synchronous call and its reply is given by the function $ref : \mathcal{M} \to \mathcal{M} \cup \{nil\}$, such that $\forall m \in \mathcal{M}_{sop} : ref(m) \in \mathcal{M}_{rep}, \forall m \in \mathcal{M}_{rep} : ref(m) \in \mathcal{M}_{sop}, \forall m \in \mathcal{M}_{aop} : ref(m) = nil$ and $\forall m \in \mathcal{M}_{sop} \cup \mathcal{M}_{rep} : ref(ref(m)) = m$.

The peers behave differently during interactions depending on the type of message, as follows. In the case of a synchronous call, the caller's execution is suspended until the caller receives a reply from the callee. However, in the case of an asynchronous call, the caller can continue to operate, regardless of behavior of the callee.

In UML, each message has two events: a *send event* and a *receive event*. For a synchronous message, the receive eevent occurs immediately after the send event. However, for a discussion that occurs subsequently, we need two events that occur sequentially. Therefore, we define that each synchronous message has two events: a *preparation event* for message sending and a *send-receive event* where the preparation event is a caller's event and the send-receive event is a callee's event. The preparation event and the send-receive event of a synchronous message $m \in \mathcal{M}_s$ are denoted by $\$m$ and $!m$, respectively. For an asynchronous or a reply message $m \in \mathcal{M}_a$, the send and receive events are denoted by $!m$ and $?m$, respectively. Hereafter, an *active event* is the send-receive event of a synchronous message or the send event of an asynchronous or a reply message. The set Σ of message events and set Δ of active events are defined as follows:

$$\Sigma = \{\$m, !m \mid m \in \mathcal{M}_s\} \cup \{!m, ?m \mid m \in \mathcal{M}_a\} \quad (1)$$

$$\Delta = \{!m \mid m \in \mathcal{M}\} \quad (2)$$

The acyclic relation $\Rightarrow_{\mathcal{M}}$ on the order of the caller's and callee's events for each message is defined as follows:

$$\Rightarrow_{\mathcal{M}} = \{(\$m, !m) \mid m \in \mathcal{M}_s\} \cup \{(!m, ?m) \mid m \in \mathcal{M}_a\} \quad (3)$$

2.2.2 Communication Diagram

Definition 3 (Communication Diagram) A communication diagram $cd \in \mathcal{CD}$ is a tuple $cd = (\mathcal{C}^{cd}, \mathcal{M}^{cd}, Conn^{cd}, line^{cd}, D^{cd})$, where $\mathcal{C}^{cd} \subseteq \mathcal{C}$ is the set of classes, which are called lifelines and correspond to peers; $\mathcal{M}^{cd} \subseteq \mathcal{M}$ is the set of messages; $Conn^{cd} \subseteq \mathcal{C}^{cd} \times \mathcal{C}^{cd}$ is the set of connectors, which is given as a symmetric relation on $\mathcal{C}^{cd}$; $line^{cd} : \mathcal{M}^{cd} \to Conn^{cd}$ assigns a connector for each message; and $D^{cd} \subseteq \Delta \times \Delta$ indicates a dependency relation among active events, where D^{cd} must be acyclic.

A *conversation* is a sequence of messages exchanged among peer[4]. The set of conversations defined by a communication diagram cd is denoted by $\mathfrak{C}(cd) \subseteq \mathcal{M}^*$, where $\mathcal{M}^*$ is the set of all sequences of distinct messages.

Definition 4 A conversation $\sigma = m_1 m_2 \ldots m_n$ is in $\mathfrak{C}(cd)$ if and only if $\sigma \in \mathcal{M}^*$ and the corresponding sequence $\gamma = !m_1 !m_2 \ldots !m_n$ of active events satisfy $\forall i, j \in [1..n] : (!m_i, !m_j) \in D^{cd} \Rightarrow i < j$.

If there exists a communication diagram $cd \in \mathcal{CD}$ such that $\sigma \in \mathfrak{C}(cd)$, then $\sigma \in \mathfrak{C}(\mathcal{CD})$.

2.2.3 State Machine

Definition 5 A state machine is a tuple $sm = (V, R, r^t, \Theta, \Phi, E, C, B)$, where V is the set of vertices, R is the set of regions, $r^t \in R$ is the top region, Θ is an ownership relation between vertices and regions, Φ is the set of transitions, E is the set of events, C is the set of constraints, and B is the set of behaviors.

In UML state machines, although there are various kinds of states and pseudo-states, only *simple states, composite states, final states*, and *initial pseudo-states* are used in this paper. Therefore, the set V of vertices is partitioned into the following types of subsets: $V = SS \cup CS \cup FS \cup IS$, where SS is the set of simple states, CS is the set of composite states, FS is the set of final states, and IS is the set of initital pseudo-states.

Let Pro be a mapping that translates a word of acyclic relation $\mathcal{R}$ on Σ to a conversation. A conversation $\sigma \in \mathrm{Pro}(\mathfrak{L}(\mathcal{R}))$ is obtained by removing non-active events and replacing each active event with corresponding message from a word $w \in \mathfrak{L}(\mathcal{R})$. A word w is accepted by the set $\mathcal{SM}$ of state machines if every state machine is in the final state in the top region after occurring all events in w. Let $\mathcal{R}$ be an acyclic relation, and let the language of $\mathcal{R}$ and the language accepted by $\mathcal{SM}$ be equivalent. Then, the set of all conversations for $\mathcal{SM}$, denoted by $\mathfrak{C}(\mathcal{SM})$, satisfies the following equation:

$$\mathfrak{C}(\mathcal{SM}) = \mathrm{Pro}(\mathfrak{L}(\mathcal{R})).$$

3. Choreography Realization Problem

Problem 1 (CRP) For a given set $\mathcal{CD}$ of communication diagrams, is it possible to synthesize the set $\mathcal{SM}$ of state machines that satisfy $\mathfrak{C}(\mathcal{CD}) = \mathfrak{C}(\mathcal{SM})$? If possible, obtain the set of state machines.

In the case of un-realizable choreography, it is preferred that state machines that mimic the choreography as closely as possible are synthesized. A set of state machine that satisfy $\mathfrak{C}(\mathcal{CD}) \supseteq \mathfrak{C}(\mathcal{SM})$ is called a *weak realization* of the given choreography. Since a set of empty state machine, in which $\mathfrak{C}(\mathcal{SM}) = \emptyset$, is a weak realization for any choreography, the set of state machines whose $\mathfrak{C}(\mathcal{SM})$ is maximal is expected.

Definition 6 If a weak relization $\mathcal{SM}$ that satisfies $\mathfrak{C}(\mathcal{SM}^*) \subseteq \mathfrak{C}(\mathcal{SM})$ and $\mathcal{SM}^* \neq \mathcal{SM}$ does not exist, $\mathcal{SM}^*$ is called a maximal weak realization.

Definition 7 $\mathcal{SM}$ is a fair realization of $\mathcal{CD}$ if $\forall cd \in \mathcal{CD}$: $\mathfrak{C}(cd) \cap \mathfrak{C}(\mathcal{SM}) \neq \emptyset$.

3.1 CSCB Method

Miyamoto et al. proposed the CSCB method that synthesizes state machines from a communication diagram in [2]. Due to space limitations, the details of the algorithm are omitted here. State machines are synthesized as follows:

(1) Construct an acyclic relation $\Rightarrow$ on the set of events.

For each peer c, perform the following steps.

(2) Derive an acyclic relation $\Rightarrow_c$ from $\Rightarrow$.

(3) Construct a state machine from $\Rightarrow_c$.

Recall that we assume Assumption 1 for $\Rightarrow_c$.

Because the acyclic relation D is a relation on active events, we have to extend it to the relation on active and non-active events. The acyclic relation $\Rightarrow \subseteq \Sigma^2$ on the set of events is obtained by augmenting D, as follows:

$$\begin{aligned}\Rightarrow = D \cup \{(?m_1, !m_2) \mid m_1 \in \mathcal{M}_a, m_2 \in \mathcal{M}_a, \Omega(?m_1, !m_2)\} \\ \cup \{(?m_1, \$m_2) \mid m_1 \in \mathcal{M}_a, m_2 \in \mathcal{M}_s, \Omega(?m_1, \$m_2)\} \\ \cup \{(!m_1, \$m_2) \mid m_1 \in \mathcal{M}_s, m_2 \in \mathcal{M}_s, \Omega(!m_1, \$m_2)\} \\ \cup \Rightarrow_{\mathcal{M}} \cup \{(!m, e) \mid m \in \mathcal{M}_s, \Omega(\$m, e)\}, \quad (4)\end{aligned}$$

where $\Omega(e_1, e_2)$ is true when both events e_1 and e_2 occur in the same peer and $(!e_1, !e_2) \in D$, where $!e_1$ and $!e_2$ are the corresponding active events for events e_1 and e_2, respectively.

The communal relation for decomposition is given as follows:

$$\Rightarrow_{\mathrm{com}} = \Rightarrow_{\mathcal{M}} \cup \{(!m, e) \mid m \in \mathcal{M}_s, \Omega(\$m, e)\}, \quad (5)$$

where $\{(!m, e) \mid m \in \mathcal{M}_s, \Omega(\$m, e)\}$ implies that an event e that follows a preparation event $\$m$ of a synchronous message and occurs in the same peer follows the send-receive event $!m$ of the message. As stated before, a caller of a synchronous message waits for the occurrence of callee's receive event. Therefore, $!m$ precedes e. In the case of state machines of cbUML, any event following a preparation event follows the send-receive event. Therefore, the order given by $\Rightarrow_{\mathrm{com}}$ is kept when multiple state machines are executed in parallel.

The relation Y_c for a peer c is given as follows:

$$\begin{aligned}Y_c = \Rightarrow_c^{\max} \cup \{(?ref(m), e) \mid \\ m \in \mathcal{M}_s, e \neq ?ref(m), (\$m, e) \in \Rightarrow_c^{\max}\} \quad (6)\end{aligned}$$

The first set is the projected relation of the transitive closure of $\Rightarrow$ on the set of events of peer c. The second set adds the additional constraints so that only the receive event $?ref(m)$ of the reply message of a synchronous m is the direct successor of the preperation event $\$m$. Next, the acyclic relation $\Rightarrow_c$ for a peer c is obtained by transitively reducing Y_c, as follows:

$$\overline{\Rightarrow_c} = Y_c^- \quad (7)$$

However, Theorem 1 shows that transitive reduction of $\Rightarrow$ is sufficient to derive $\Rightarrow_c$. Thus, a $\underline{\Rightarrow_c}$ is calculated as follows:

$$\begin{aligned}Y_c^{'} = \Rightarrow_c^{\min} \cup \{(ref(m), e) \mid \\ m \in \mathcal{M}_s, e \neq ?ref(m), (\$m, e) \in \Rightarrow_c^{\min}\} \quad (8)\end{aligned}$$

$$\underline{\Rightarrow_c} = Y_c^{'-} \quad (9)$$

Definition 8 Given a set $\{\mathcal{R}_c\}$ of relations, $\mathcal{SM}$ which is synthesized from $\{\mathcal{R}_c\}$ is denoted as $\mathcal{SM}_{\{\mathcal{R}_c\}}$.

Theorem 2 [4] If $\{\mathcal{R}_c\}$ is re-constructible to $\Rightarrow$, then $\mathcal{SM}_{\{\mathcal{R}_c\}}$ is a strong realization of cd.

4. Synthesizing State Machines from Two Acyclic Relations

In this paper, we assume that choreography is composed of two communication diagrams: $cd1$ and $cd2$. We also assume that we use same message set for all communication diagrams. Therefore, let $cd1$ and $cd2 \in \mathcal{CD}$ be

$$\begin{aligned}cd1 &= (\mathcal{C}^{cd1}, \mathcal{M}^{cd1}, Conn^{cd1}, line^{cd1}, D^{cd1}) \text{ and} \\ cd2 &= (\mathcal{C}^{cd2}, \mathcal{M}^{cd2}, Conn^{cd2}, line^{cd2}, D^{cd2}),\end{aligned}$$

respectively, then $\mathcal{C}^{cd1} = \mathcal{C}^{cd2}, \mathcal{M}^{cd1} = \mathcal{M}^{cd2}, Conn^{cd1} = Conn^{cd2}, line^{cd1} = line^{cd2}$, and $D^{cd1} \neq D^{cd2}$. We also assume that D^{cd1} and D^{cd2} are acyclic.

For each communication diagram and peer c, we obtain $\overline{\Rightarrow_c}$ from Eq. (7) and let us denote them by $\overline{\Rightarrow_c^{cd1}}$ and $\overline{\Rightarrow_c^{cd2}}$, respectively. We also obtain $\underline{\Rightarrow_c}$ from Eq. (9) and let us denote them by $\underline{\Rightarrow_c^{cd1}}$ and $\underline{\Rightarrow_c^{cd2}}$, respectively. From Theorem 1 and Theorem 2, $\mathcal{SM}_{\{\Rightarrow_c\}}$ that satisfies

$$\forall c : \underline{\Rightarrow_c^{cd1}} \subseteq \Rightarrow_c \subseteq \overline{\Rightarrow_c^{cd1}} \quad (10)$$

is a strong realization of $cd1$; $\mathcal{SM}_{\{\Rightarrow_c\}}$ that satisfies

$$\forall c : \underline{\Rightarrow_c^{cd2}} \subseteq \Rightarrow_c \subseteq \overline{\Rightarrow_c^{cd2}} \quad (11)$$

is a strong realization of $cd2$.

We put the following assumption on relation $\Rightarrow_{\mathrm{com}}, \underline{\Rightarrow_c^{cd1}}$ and $\underline{\Rightarrow_c^{cd2}}$.

Assumption 2 $\Rightarrow_{\mathrm{com}} \cup \left\{\bigcup_c (\underline{\Rightarrow_c^{cd1}} \cup \underline{\Rightarrow_c^{cd2}})\right\}$ is acyclic.

Lemma 1 Under Assumption 2, $\mathcal{SM}_{\{\Rightarrow_c\}}$ is a strong realization of $\mathcal{CD}$ iff there exists $\{\Rightarrow_c\}$ that satisfies following condition.

$$\forall c \in \mathcal{C} : (\underline{\Rightarrow_c^{cd1}} \cup \underline{\Rightarrow_c^{cd2}}) \subseteq \Rightarrow_c \subseteq (\overline{\Rightarrow_c^{cd1}} \cap \overline{\Rightarrow_c^{cd2}}) \quad (12)$$

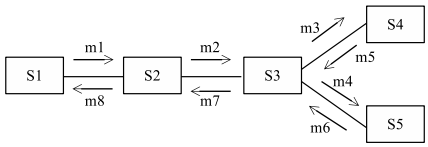

Figure 1. A choreography with five peers

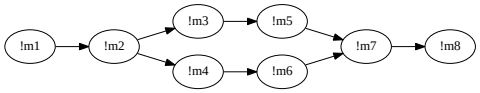

Figure 2. D^{cd1}

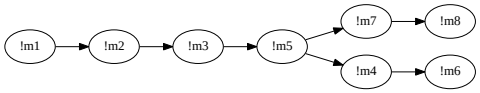

Figure 3. D^{cd2}

If $\mathfrak{C}(cd1) \subseteq \mathfrak{C}(cd2)$, a strong realization of $cd2$ is a strong realization of $\mathcal{CD}$. However, this situation is useless, so that we put the following assumption.

Assumption 3 $\mathfrak{C}(cd1) \nsubseteq \mathfrak{C}(cd2)$ and $\mathfrak{C}(cd1) \not\supseteq \mathfrak{C}(cd2)$.

Lemma 2 Under Assumption 2 and Assumption 3, $cd1$ and $cd2$ that satisfy following condition do not exist.

$$\forall c \in \mathcal{C} : (\underline{\Rightarrow_c^{cd1}} \cup \underline{\Rightarrow_c^{cd2}}) \subseteq (\overline{\Rightarrow_c^{cd1}} \cap \overline{\Rightarrow_c^{cd2}}) \qquad (13)$$

Theorem 3 Under Assumption 2 and Assumption 3, $\{\Rightarrow_c\}$ that synthesizes a strong realization $\mathcal{SM}_{\{\Rightarrow_c\}}$ of the $\mathcal{CD}$ does not exist.

Then, we consider the way to realize weakly and fairly. We can obtain the following lemma and theorem.

Lemma 3 $\mathfrak{C}(cd1) \cap \mathfrak{C}(cd2) \neq \emptyset$ iff Assumption 2 is satisfied.

Theorem 4 Under Assumption 2, a strong realization of $cd1$ or $cd2$ is a fair weak realization of $\mathcal{CD}$.

Trivially, Theorem 3 holds when choreography is given by more than two communication diagrams. Lemma 3 holds for more than two communication diagrams cases under Assumption 2, so that Theorem 4 also holds.

Example

Figure 1 Shows choreography for a system composed of five peers. Figure 2 and Figure 3 show the dependency relations on messages in $cd1$ and $cd2$, respectively. For simplicity, we assume that all messages are asynchronous. These communication diagrams satisfy all assumptions. Figure 4 and Figure 5 are acyclic relations on peer S3, and show dissatisfaction of Eq. (12).

We can synthesize $\mathcal{SM}$ which satisfies $\mathfrak{C}(\mathcal{SM}) = \mathfrak{C}(cd1)$ by using the algorithm in [2]. Then, $\mathcal{SM}$ is a fair weak realization, since $\mathfrak{C}(\mathcal{SM}) \cap \mathfrak{C}(cd2) = \{\mathrm{m1m2m3m5m4m6m7m8}\}$.

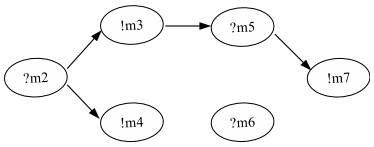

Figure 4. $\overline{\Rightarrow_{S3}^{cd1}} \cap \overline{\Rightarrow_{S3}^{cd2}}$

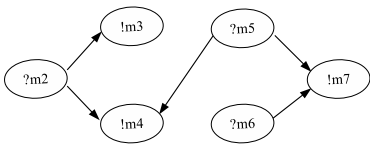

Figure 5. $\underline{\Rightarrow_{S3}^{cd1}} \cup \underline{\Rightarrow_{S3}^{cd2}}$

5. Conclusion

In this paper, choreography was given by two communication diagrams. The result can be applied for more than two cases. We tried to extend the CSCB method by using lower and upper bounds of acyclic relation. However, we showed that we cannot obtain any strong realization in this way. Considering new method to achieve strong realization will be a future work.

Acknowledgement This work was supported by KAKENHI (26330083).

References

[1] T.Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall Professional Technical Reference, 2005.

[2] T. Miyamoto, Y. Hasegawa, and H. Oimura, "An Approach for Synthesizing Intelligible State Machine Models from Choreography Using Petri Nets," *IEICE Transaction on Information and Systems*, vol.E97.D, no.5, pp.1171-1180, May 2014.

[3] Object Management Group, "OMG Unified Modeling Language(OMG UML), superstructure," ver.2.4.1, http://www.uml.org, Aug.2011.(accessed Oct. 31, 2013).

[4] T. Miyamoto, "Choreography Realization by Reconstructible Decompositionof Acyclic Relations," *IEICE Transactions on Information and Systems*, vol.E99.D, no.5, pp.1420-1427, 2016.

[5] Y. Hasegawa, H. Niimura, and T. Miyamoto, "A UML subset for design and vertification of systems based on SOA," IEICE Technical Report, vol.111, no.293, pp.89-94, Nov. 2011. (in Japanese).