IEICE TRANS. INF. & SYST., VOL.E103–D, NO.8 AUGUST 2020

LETTER Special Section on Formal Approaches

# A Study on Re-Constructibility of Event Structures

**Marika IZAWA**[†], *Nonmember and* **Toshiyuki MIYAMOTO**[†a)], *Senior Member*

**SUMMARY** The choreography realization problem is a design challenge for systems based on service-oriented architecture. In our previous studies, we studied the problem on a case where choreography was given by one or two scenarios and was expressed by an acyclic relation of events; we introduced the notion of re-constructibility as a property of acyclic relations to be satisfied. However, when choreography is defined by multiple scenarios, the resulting behavior cannot be expressed by an acyclic relation. An event structure is composed of an acyclic relation and a conflict relation. Because event structures are a generalization of acyclic relations, a wider class of systems can be expressed by event structures. In this paper, we propose the use of event structures to express choreography, introduce the re-constructibility of event structures, and show a necessary condition for an event structure to be re-constructible.

***key words:*** *SOA, model-based development, event structure, re-constructibility, choreography realization problem*

## 1. Introduction

Service-Oriented Architecture (SOA) is an information system architecture [1]. In SOA, an information system is constructed by combining independent software units called peers. In SOA, the problem of designing a concrete model from an abstract specification is called the Choreography Realization Problem (CRP) [2], [3]. The abstract specification defines the interaction between peers and is expressed by message dependencies. This is called a scenario, and a set of scenarios is called choreography. The concrete model is called a service implementation and defines the behavior of each peer.

We have studied the CRP on a case where choreography was given by an acyclic relation of events [4]. Miyamoto introduced the concept of re-constructible decomposition of acyclic relations, and showed the necessary and sufficient conditions for decomposed acyclic relations to be re-constructible [5]. In [5], it is assumed that choreography is defined by one scenario. Kinoshita et al. showed the necessary and sufficient conditions for choreography to be realizable when it is defined by two scenarios [6]. In these studies, it is assumed that choreography can be expressed by an acyclic relation on events. However, for example, cases exist where choreography is defined by three or more scenarios, or where different message sets are used in multiple scenarios. In these cases, choreography cannot be expressed by an acyclic relation.

In this paper, we consider a case where choreography consists of multiple scenarios where each scenario is expressed by an acyclic relation and, however, where choreography cannot be expressed by an acyclic relation. We propose using the event structures as a modeling language of abstract specifications and concrete models [7], [8]; we discuss the re-constructible decomposition of event structures.

An event structure is defined by a set of events, an acyclic relation, and a conflict relation. For the re-constructibility of event structures, the non-existence of any relation across decomposed event structures is a necessary condition. However, some redundant relations can be deleted by reduction. A reduction of an acyclic relation can be easily achieved by the transitive reduction, however, to the best of our knowledge, no study on the reduction of conflict relations exists. In this paper, we give necessary and sufficient conditions for deleting redundant conflict relations and give a necessary condition for event structures to be re-constructible.

## 2. Re-Constructibility of Event Structures

### 2.1 Event Structure

In this paper, to express relations between events, event structures are defined as follows:

**Definition 1** (Event Structure): An event structure is a 3-tuple $\mathbb{F} := (\Sigma, \#, \Rightarrow)$, where $\Sigma$ is a set of events, $\# \subseteq \Sigma \times \Sigma$ is a reflective and symmetric conflict relation, and $\Rightarrow \subseteq \Sigma \times \Sigma$ is a flow relation.

When two events $e_1, e_2 \in \Sigma$ occur in an execution, $e_1$ must occur before $e_2$ if $e_1 \Rightarrow e_2$; $\Rightarrow^+$ denotes the transitive closure of $\Rightarrow$, and $\Rightarrow^*$ denotes the reflective and transitive closure of $\Rightarrow$. If $e_1 \# e_2$, $e_1$ and $e_2$ cannot occur in the same execution. However, there may be a pair $(e_1, e_2)$ such that $(e_1, e_2) \notin \#$ and they do not occur in any same execution. In this case, $e_1$ and $e_2$ are called in potential conflict. A closure $\hat{\#}$ of conflict relation $\#$ is a conflict relation that includes all potential conflict relations. The closure can be calculated by Algorithm 1.

A configuration of an event structure is defined as follows:

**Definition 2** (Configuration): A configuration of an event structure $\mathbb{F}$ is a set of events $C = \{e_1, \cdots, e_n\} \subseteq \Sigma$, and satisfies the following conditions: a) $\forall e_1, e_2 \in C : (e_1, e_2) \notin \hat{\#}$,

Manuscript received October 3, 2019.
Manuscript revised February 6, 2020.
Manuscript publicized March 27, 2020.

[†]The authors are with the Graduate School of Engineering, Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: miyamoto@eei.eng.osaka-u.ac.jp

DOI: 10.1587/transinf.2019FOL0002

**Algorithm 1** Calculate closure conflict relation $\hat{\#}$

**Require:** $\mathbb{F} = (\Sigma, \#, \Rightarrow)$
  $\hat{\#} \leftarrow \#$
  tsort[]$\leftarrow$ an array of events such that $\forall i, j \in \{0, \ldots, |\Sigma| - 1\} : i < j \Rightarrow (\text{tsort}[j], \text{tsort}[i]) \notin \Rightarrow^+$
  **for** $i \leftarrow 0$ to $|\Sigma| - 1$ **do**
    $e_i \leftarrow \text{tsort}[i]$
    $G_i(V_i, E_i) \leftarrow$ new graph such that $V_i \leftarrow \{e \mid (e, e_i) \in \Rightarrow\}$, $E_i \leftarrow \{(e_1, e_2) \in V^2 \mid (e_1, e_2) \notin \hat{\#}\}$
    $K_{e_i} \leftarrow \{V'_i \subseteq V_i \mid V'_i \text{ is a clique of } G_i\}$
    **for** $j \leftarrow 0$ to $i$ **do**
      $e_j \leftarrow \text{tsort}[j]$
      $G_j(V_j, E_j) \leftarrow$ new graph such that $V_j \leftarrow \{e \mid (e, e_j) \in \Rightarrow\}$, $E_j \leftarrow \{(e_1, e_2) \in V^2 \mid (e_1, e_2) \notin \hat{\#}\}$
      $K_{e_j} \leftarrow \{V'_j \subseteq V_j \mid V'_j \text{ is a clique of } G_j\}$
      **if** $(e_j, e_i) \notin \Rightarrow^+ \wedge (e_j, e_i) \notin \hat{\#}$ **then**
        **if** $(\forall k_{e_j} \in K_{e_j}, \exists e \in k_{e_j} : (e_i, e) \in \hat{\#}) \vee (\forall k_{e_i} \in K_{e_i}, \exists e' \in k_{e_i} : (e', e_j) \in \hat{\#})$ **then**
          $\hat{\#} \leftarrow \hat{\#} \cup \{(e_i, e_j), (e_j, e_i)\}$
        **end if**
      **end if**
    **end for**
  **end for**

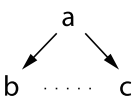


**Fig. 1** An example of event structure

and b) for all events $e_2 \in C$ and $e_1 \notin C$ such that $(e_1, e_2) \in \Rightarrow$, it holds that there is an event $e_3 \in C$ such that $(e_1, e_3) \in \hat{\#}$ and$(e_3, e_2) \in \Rightarrow$.

The set of all configurations of an event structure $\mathbb{F}$ is denoted by Conf($\mathbb{F}$).

The set of events that appear in a sequence $w = e_1 e_2 \cdots e_n$ of events is called a support of $w$ and is denoted by sup($w$). A word of an event structure is a sequence $w$ of events and satisfies the following conditions: a) $\text{sup}(w) \in \text{Conf}(\mathbb{F})$, b) $\forall i, j \in \{1, \ldots, n\}, i < j : (e_j, e_i) \notin \Rightarrow^+$, and c) a word $w'$ such that $w$ is a substring of $w'$ does not exist. The set of all words in an event structure is denoted by $\mathfrak{L}(\Sigma, \#, \Rightarrow)$ or $\mathfrak{L}(\mathbb{F})$.

Figure 1 shows an example of event structures, where a, b, and c are events. The acyclic relation is represented by arrows, and the conflict relation is represented by dotted lines. For the event structure shown in Fig. 1, Conf($\mathbb{F}$) = {∅, {a}, {a, b}, {a, c}} and $\mathfrak{L}(\mathbb{F})$ = {ab, ac}.

Let us define three families, $\mathbb{C}_a$, $\mathbb{T}_a$, and $\mathbb{I}_a$, for an event $a$:

$$\mathbb{C}_a = \{C = \{e_1, \ldots, e_n\} \in \text{Conf}(\mathbb{F}) \mid \forall i \in \{1, \ldots, n\} : (e_i, a) \in \Rightarrow^*\} \quad (1)$$

$$\mathbb{T}_a = \{C \in \mathbb{C}_a \mid a \in C\} \quad (2)$$

$$\mathbb{I}_a = \{I \subseteq \Sigma \backslash \{a\} \mid \forall T \in \mathbb{T}_a : T \cap I \neq \emptyset\} \cup \{a\} \quad (3)$$

$\mathbb{C}_a$ is a family of configurations composed of $a$ and ancestor events of $a$; $\mathbb{T}_a$ is a family of configurations when $a$ occurs for the first time in $\mathbb{C}_a$. For certain $I \in \mathbb{I}_a$, $a$ does not occur if the occurrence of all the events in $I$ is prohibited.

## 2.2 Re-Constructibility of Event Structures

Let $\mathcal{P}$ be a set of peers. $\mathbb{F}_p = (\Sigma_p, \#_p, \Rightarrow_p)$ is a partition of the event structure w.r.t. $p \in \mathcal{P}$ on $\mathbb{F} = (\Sigma, \#, \Rightarrow)$, where $\#_p \subseteq \Sigma_p^2$ and $\Rightarrow_p \subseteq \Sigma_p^2$. A flow relation between peers $\Rightarrow_{\text{com}} \subseteq \Rightarrow \backslash (\bigcup_p \Sigma_p^2)$ is called a communal relation. The re-constructible decomposition of event structures can be defined as follows:

**Definition 3** (Re-constructible Decomposition): For a set $\{(\Sigma_p, \#_p, \Rightarrow_p)\}$ of event structures and a communal relation $\Rightarrow_{\text{com}}$, $\{(\Sigma_p, \#_p, \Rightarrow_p)\}$ is re-constructible to $(\Sigma, \#, \Rightarrow)$ when the following equation holds:

$$\mathfrak{L}\left(\Sigma, \bigcup_p \#_p, \Rightarrow_{\text{com}} \cup \bigcup_p \Rightarrow_p\right) = \mathfrak{L}(\Sigma, \#, \Rightarrow). \quad (4)$$

When there exists a re-constructible $\{(\Sigma_p, \#_p, \Rightarrow_p)\}$, it is called that $(\Sigma, \#, \Rightarrow)$ has re-constructibility.

When two events in a conflict relation exist in different peers, it is impossible to directly control so that only one of the events occurs. However, in some cases, only one event can be allowed to occur if we control the ancestors of both events. In other words, the existence of conflict relations across peers does not mean that the event structure does not have re-constructibility. Therefore, we can delete unnecessary conflict relations as long as the deletion does not affect the overall behavior. Let $\mathbb{F}\backslash_{(a\#b)}$ be the event structure obtained by deleting the conflict relation between $a$ and $b$ from $\mathbb{F}$. That is, for $\mathbb{F} = (\Sigma, \#, \Rightarrow)$, $\mathbb{F}\backslash_{(a\#b)}$ is given by:

$$\mathbb{F}\backslash_{(a\#b)} = (\Sigma, \#\backslash\{(a, b)\}, \Rightarrow). \quad (5)$$

If $\mathfrak{L}(\mathbb{F}\backslash_{(a\#b)}) = \mathfrak{L}(\mathbb{F})$, then a conflict relation $(a, b) \in \#$ can be deleted. Obviously, the necessary condition for an event structure to have re-constructibility are that no conflict relation exists across peers in the partitioned set of event structures after deleting all possible conflict relations.

The conflict relation between sets $X$ and $Y$ is defined as follows:

$$X \# Y \quad \Leftrightarrow \quad \forall x \in X, \forall y \in Y : (x, y) \in \hat{\#}. \quad (6)$$

Then, Lemma 1 holds.

**Lemma 1:** $\exists I_a \in \mathbb{I}_a, \exists I_b \in \mathbb{I}_b : I_a \# I_b, (I_a \neq \{a\} \vee I_b \neq \{b\}) \Leftrightarrow \mathfrak{L}(\mathbb{F}) = \mathfrak{L}(\mathbb{F}\backslash_{(a\#b)})$.

**Proof :** For the events $a$ and $b$, it always holds that $\{a\} \in \mathbb{I}_a$ and $\{b\} \in \mathbb{I}_b$. Since events $a$ and $b$ are in the conflict relation, $I_a$ and $I_b$ that satisfy $I_a \# I_b$ always exists. To eliminate this case, $(I_a \neq \{a\} \vee I_b \neq \{b\})$ is in the condition. In the following, $(I_a \neq \{a\} \vee I_b \neq \{b\})$ is assumed to be satisfied.

($\Rightarrow$) Obviously, $\mathfrak{L}(\mathbb{F}) \subseteq \mathfrak{L}(\mathbb{F}\backslash_{(a\#b)})$. Suppose that $\mathfrak{L}(\mathbb{F}\backslash_{(a\#b)})\backslash\mathfrak{L}(\mathbb{F}) \neq \emptyset$. Then, all words in $\mathfrak{L}(\mathbb{F}\backslash_{(a\#b)})\backslash\mathfrak{L}(\mathbb{F})$ include both $a$ and $b$.

Now, we define $\mathbb{S}_e = \{C \in \text{Conf}(\mathbb{F}) \mid e \in C\}$; then it

holds that:

$$(e_1, e_2) \in \hat{\#} \Leftrightarrow \mathbb{S}_{e_1} \cap \mathbb{S}_{e_2} = \emptyset. \quad (7)$$

$I_a \# I_b$ means $\forall e_i \in I_a, \forall e_j \in I_b : (e_i, e_j) \in \hat{\#}$, so it holds that:

$$\bigcup_i \mathbb{S}_{e_i} \cap \bigcup_j \mathbb{S}_{e_j} = \emptyset. \quad (8)$$

Also, we define $\mathbb{S}_{a,e_i} = \{C \in \mathrm{Conf}(\mathbb{F}) \mid a \in C, e_i \in C\}$; then for all $e_i \in I_a$, it holds that $\mathbb{S}_{e_i} \supseteq \mathbb{S}_{a,e_i}$. Therefore,

$$\bigcup_i \mathbb{S}_{e_i} \supseteq \bigcup_i \mathbb{S}_{a,e_i} \quad (9)$$

holds.

Suppose that $\bigcup_{e_i \in I_a} \mathbb{S}_{a,e_i} \neq \mathbb{S}_a$. Therefore, $\bigcup_{e_i \in I_a} \mathbb{S}_{a,e_i} \subset \mathbb{S}_a$, so $\mathbb{S}_a \setminus \bigcup_i \mathbb{S}_{a,e_i} \neq \emptyset$ holds. That is, there is a configuration $C$ that has $a$ and does not have any $e_i \in I_a$ as an element. Then, for a configuration $C_1 \in \mathbb{S}_a \setminus \bigcup_i \mathbb{S}_{a,e_i}$, there exists a configuration $C_2 \in \mathbb{S}_a \setminus \bigcup_i \mathbb{S}_{a,e_i}$ such that $C_2 \subseteq C_1$ and $(e, a) \in \Rightarrow^*$ for all $e \in C_2$.

$C_2$ does not have any elements $e_i \in I_a$, so

$$C_2 \cap I_a = \emptyset \quad (10)$$

holds. For $C_2$, $a \in C_2$ and $(e, a) \in \Rightarrow^*$ for all $e \in C_2$, so $C_2 \in \mathbb{T}_a$. However, Eq. (10) is inconsistent with the definition of $\mathbb{I}_a$. Thus, there is no configuration that satisfies $C \in \mathbb{S}_a \setminus \bigcup_i \mathbb{S}_{a,e_i}$. Therefore, it holds that

$$\bigcup_{e_i \in I_a} \mathbb{S}_{a,e_i} = \mathbb{S}_a. \quad (11)$$

From Eqs. (9) and (11), $\bigcup_i \mathbb{S}_{e_i} \supseteq \mathbb{S}_a$. Similarly, for the event $b$, $\bigcup_j \mathbb{S}_{e_j} \supseteq \mathbb{S}_b$ holds. Also, from Eq. (8), $\bigcup_i \mathbb{S}_{e_i} \cap \bigcup_j \mathbb{S}_{e_j} = \emptyset$, so $\mathbb{S}_a \cap \mathbb{S}_b = \emptyset$. Thus, from Eq. (7), there is no case where both $a$ and $b$ exist in all words of $\mathbb{F}$.

From the above, $\mathfrak{L}(\mathbb{F}_{\setminus(a\#b)}) \setminus \mathfrak{L}(\mathbb{F}) = \emptyset$, so $\mathfrak{L}(\mathbb{F}) = \mathfrak{L}(\mathbb{F}_{\setminus(a\#b)})$ holds.

($\Leftarrow$) We prove the contrapositive, i.e., $\forall I_a \in \mathbb{I}_a, \forall I_b \in \mathbb{I}_b, \exists e_a \in I_a, \exists e_b \in I_b : (e_a, e_b) \notin \hat{\#} \vee (I_a = \{a\} \wedge I_b = \{b\}) \Rightarrow \mathfrak{L}(\mathbb{F}) \neq \mathfrak{L}(\mathbb{F}_{\setminus(a\#b)})$. Suppose that $\forall I_a \in \mathbb{I}_a, \forall I_b \in \mathbb{I}_b, \exists e_a \in I_a, \exists e_b \in I_b : (e_a, e_b) \notin \hat{\#} \vee (I_a = \{a\} \wedge I_b = \{b\})$ and $\mathfrak{L}(\mathbb{F}) = \mathfrak{L}(\mathbb{F}_{\setminus(a\#b)})$. Now, the word $w$, which contains both $a$ and $b$, does not exist in $\mathfrak{L}(\mathbb{F})$. Thus, there exists an event that conflicts with $b$ in all $T_a \in \mathbb{T}_a$. Let $I'_a = \{e_1, e_2, \cdots, e_n\}$ be a set of such events. By the definition of $\mathbb{I}_a$, $I'_a$ is also included in $\mathbb{I}_a$. However, since $\{b\} \in \mathbb{I}_b$, an $e_i$ that satisfies $(e_i, b) \notin \hat{\#}$ should exist in $I'_a$. This is inconsistent. Therefore, the contrapositive holds. □

For an event structure $\mathbb{F}$, let $\mathbb{F}_{\mathrm{red}} := (\Sigma, \#_{\mathrm{red}}, \Rightarrow)$ be the event structure that is obtained by deleting all possible conflict relations by Lemma 1. Then, Theorem 1 holds.

**Theorem 1:** For an event structure $(\Sigma, \#, \Rightarrow)$ and its partition $\{(\Sigma_p, \#_p, \Rightarrow_p)\}$, if $\{(\Sigma_p, \#_p, \Rightarrow_p)\}$ is re-constructible to $(\Sigma, \#, \Rightarrow)$, it holds that:

$$\#_{\mathrm{red}} \subseteq \bigcup_p \#_p. \quad (12)$$

**Proof :** Consider the contrapositive of the proposition. Suppose that $\#_{\mathrm{red}} \subsetneq \bigcup_p \#_p$, then, there exists a pair of events in the conflict relation and in different peers. Let $(a, b)$ be the pair of events, then it holds that $(a, b) \in \#_{\mathrm{red}}$ and $(a, b) \notin \bigcup_p \#_p$. That is, $\mathfrak{L}(\Sigma, \#_{\mathrm{red}}, \Rightarrow)$ does not contain both $a$ and $b$, and $\mathfrak{L}(\Sigma, \bigcup_p \#_p, \Rightarrow_{\mathrm{com}} \cup \bigcup_p \Rightarrow_p)$ contains both. Therefore, $\mathfrak{L}(\Sigma, \#_{\mathrm{red}} \Rightarrow) = \mathfrak{L}(\Sigma, \#, \Rightarrow) \neq \mathfrak{L}(\Sigma, \bigcup_p \#_p, \Rightarrow_{\mathrm{com}} \cup \bigcup_p \Rightarrow_p)$. □

## 3. Application to CRP

In SOA, the problem of synthesizing a concrete model from an abstract specification is known as the choreography realization problem (CRP), and is given as follows:

**Problem 1** (CRP): Let $\mathfrak{C}(\mathcal{CD})$ be the behavior defined by the set of communication diagrams $\mathcal{CD}$, and $\mathfrak{C}(\mathcal{SM})$ be the behavior defined by the set of state machines $\mathcal{SM}$. For a set $\mathcal{CD}$ of communication diagrams, is it possible to synthesize the set $\mathcal{SM}$ of state machines to satisfy $\mathfrak{C}(\mathcal{CD}) = \mathfrak{C}(\mathcal{SM})$? If possible, obtain the set of state machines.

Consider a system consisting of four peers. $\mathcal{CD} = \{cd1, cd2, cd3\}$, and each communication diagram is shown in Figs. 2–4. Figures 5–7 show the message dependencies in each communication diagram. The event structure, which is constructed from the dependencies in Figs. 5–7, is shown in Fig. 8. We will not describe the details on how to construct the event structure due to space limitations.

As an example, consider whether the conflict relation between ?m4 and ?m3 can be deleted. In the event structure of Fig. 8, $\mathbb{I}_{?\mathrm{m4}}$ and $\mathbb{I}_{?\mathrm{m3}}$ are as follows:

$$\mathbb{I}_{?\mathrm{m4}} = \{\{!\mathrm{m1}\}, \{?\mathrm{m1}\}, \{!\mathrm{m4}\} \cdots\}, \text{ and}$$
$$\mathbb{I}_{?\mathrm{m3}} = \{\{!\mathrm{m1}\}, \{?\mathrm{m1}\}, \{!\mathrm{m3}\} \cdots\}.$$

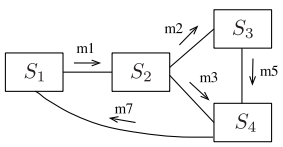


**Fig. 2** $cd1$

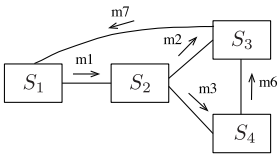


**Fig. 3** $cd2$

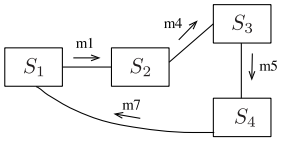


**Fig. 4** $cd3$

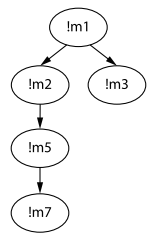


**Fig. 5** $D^{cd1}$

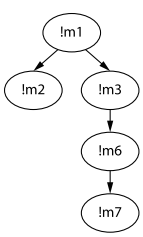


**Fig. 6** $D^{cd2}$

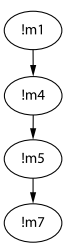


**Fig. 7** $D^{cd3}$

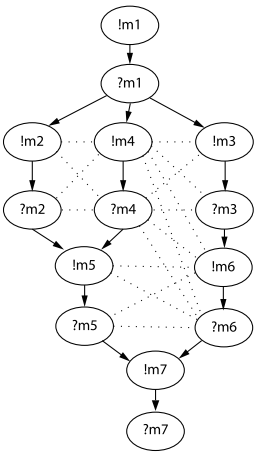


**Fig. 8** $\mathbb{F}$

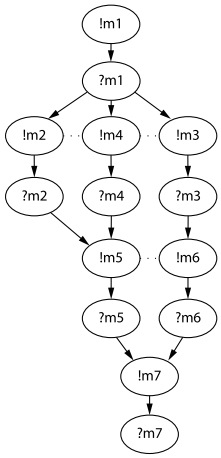


**Fig. 9** $\mathbb{F}_{\text{red}}$

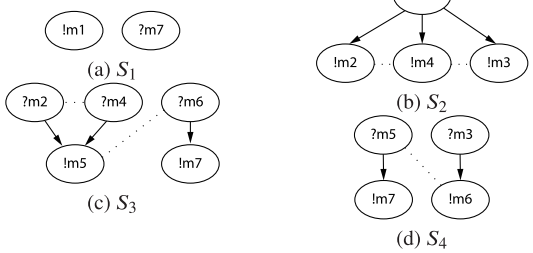


**Fig. 10** Decomposition of $\mathbb{F}$

Because !m4 and !m3 are in the conflict relation, ?m4 and ?m3 satisfy the conditions in Lemma 1. Therefore, the conflict relation between ?m4 and ?m3 can be deleted. By deleting all conflict relations, the event structure shown in Fig. 9 is obtained. Therefore, $\#_{\text{red}} = \{(!m2, !m4), (!m4, !m2), (!m4, !m3), (!m3, !m4), (!m5, !m6), (!m6, !m5)\}$. Then, $(\Sigma_p, \#_p, \Rightarrow_p)$, which is decomposed from $\mathbb{F}$, is obtained as $\#_p = \# \cap \Sigma_p^2$ and $\Rightarrow_p = \Longrightarrow \cap \Sigma_p^2$ as shown in Fig. 10. Thus, conflict relations in each peer are as follows:

$$\#_{S_1} = \emptyset$$
$$\#_{S_2} = \{(!m3, !m4), (!m4, !m3), (!m4, !m2), (!m2, !m4)\}$$
$$\#_{S_3} = \{(!m5, ?m6), (?m6, !m5), (?m2, ?m4), (?m4, ?m2)\}$$
$$\#_{S_4} = \{(?m5, !m6), (!m6, ?m5)\}$$

Therefore, Eq. (12) in Theorem 1 is not satisfied. That is, the event structures are not re-constructible to $\mathbb{F}$. In addition, $\mathbb{F}$ does not have re-constructibility because it is impossible to delete further conflict relations. In this case, we need to redesign the choreography.

When an event structure is partitioned into peers, being re-constructible to the original event structure for the decomposed event structures implies that the state machines synthesized from the set of event structures satisfies $\mathfrak{C}(\mathcal{CD}) = \mathfrak{C}(\mathcal{SM})$. Since Eq. (4) in Definition 3 uses language equivalence, the amount of computation for checking re-constructibility is exponential. On the other hand, the condition of Theorem 1 can be checked on the event structure. If we can check the conditions of Lemma 1 in polynomial time, we can expect a reduction in the amount of computation for the re-constructibility test. The deleting method for the conditions of Lemma 1 will be left for our future study.

## 4. Conclusion

In this paper, we proposed the use of event structures when choreography is given by multiple communication diagrams with different message sets. Since the class of event structures includes acyclic relations, the class of choreography that can be described has expanded.

In order to realize the choreography, event structures partitioned into peers need to be equivalent to the original event structure. Therefore, we introduced re-constructibility as the condition for event structure equivalence and derived a necessary condition for it to be re-constructible. This makes it possible to check realizability using the event structure composed of choreography. However, it may be realizable even if the condition is not satisfied, because it is a necessary condition. Deriving necessary and sufficient conditions will be the focus of a future study.

In addition, it is necessary to study the method of deriving an event structure from the choreography, and the method for synthesizing state machines from event structures. We would like to consider these as future tasks.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP17K00100.

### References

[1] T. Erl, Service-oriented architecture: Concepts, technology, and design, Prentice Hall Professional Technical Reference, 2005.
[2] J. Su, T. Bultan, X. Fu, and X. Zhao, "Toward a theory of web service choreographies," Proc. 4th Intl. Conf. on Web Services and Formal Methods, pp.1–16, 2007.
[3] T. Bultan and X. Fu, "Specification of realizable service conversations using collaboration diagrams," Service Oriented Computing and Applications, vol.2, no.1, pp.27–39, 2008.
[4] T. Miyamoto, Y. Hasegawa, and H. Oimura, "An approach for synthesizing intelligible state machine models from choreography using Petri nets," IEICE Trans. Inf. & Syst., vol.E97-D, no.5, pp.1171–1180, May 2014.
[5] T. Miyamoto, "Choreography realization by re-constructible decomposition of acyclic relations," IEICE Trans. Inf. & Syst., vol.E99-D, no.6, pp.1420–1427, June 2016.
[6] T. Kinoshita and T. Miyamoto "Realizability of choreography given by two scenarios," IEICE Trans. Fundamentals, vol.E101-A, no.2 pp.345–356, Feb. 2018.
[7] M. Nielsen, G.D. Plotkin, and G. Winskel, "Petri nets, event structures and domains, Part I," Theor. Comput. Sci., vol.13, no.1, pp.85–108, 1981.
[8] A. Polyvyanyy, A. Armas-Cervantes, M. Dumas, and L. Garcá-Bañuelos, "On the expressive power of behavioral profiles," Formal Aspects of Computing, vol.28, no.4, pp.597–613, 2016.