PAPER Special Section on Mathematical Systems Science and its Applications

Conflict Reduction of Acyclic Flow Event Structures

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

SUMMARY Event structures are a well-known modeling formalism for concurrent systems with causality and conflict relations. The flow event structure (FES) is a variant of event structures, which is a generalization of the prime event structure. In an FES, two events may be in conflict even though they are not syntactically in conflict; this is called a semantic conflict. The existence of semantic conflict in an FES motivates reducing conflict relations (i.e., conflict reduction) to obtain a simpler structure. In this paper, we study conflict reduction in acyclic FESs. A necessary and sufficient condition for conflict reduction is given; algorithms to compute semantic conflict, local configurations, and conflict reduction are proposed. A great time reduction was observed in computational experiments when comparing the proposed with the naive method.

key words: *flow event structures, reduction of conflict relation, semantic conflict, local configuration*

1. Introduction

Event structures [1] are a well-known modeling formalism for concurrent systems with causality and conflict relations. An event structure is composed of a set of events and two dependency relations on events: causality and conflict relations. If events a and b are in a causality relation, event a must occur before b. If two events are in a conflict relation, the occurrence of one is prohibited by the other.

The most fundamental class of event structures is the prime event structure (PES) [1]. Since its creation, many other variants have been proposed: the asymmetric event structure (AES) [2], flow event structure (FES) [3], [4], bundle event structure (BES) [5], and context-dependent event structure (CDES) [6].

Event structures have been used in studies to check the properties of concurrent/distributed systems. Armas-Cervantes et al. use PES and AES for the diagnosis of business processes [7]. Garcia-Bañuelos et al. present a method for checking the conformance of business processes using PES [8]. Izawa and Miyamoto use an FES for the choreography realization problem [9], [10] in service-oriented architecture (SOA) in [11]. Van Beest et al. describe an approach that represents different business process variants in a PES and provide a method to subsequently derive variability rules [12]. De León et al. propose a test generation algorithm for concurrent systems using a PES [13]. Kienzle et al. use a PES for model composition in model-driven engineering [14].

Our research uses FES as a modeling formalism for distributed systems, where multiple peers are processing independently and interacting via communication among them [11]. Suppose that an FES defines the specification of a distributed system. If a conflict exists between two events in different peers, the specification is not realizable without introducing an additional peer for the control of exclusive occurrence or by modifying the specification. Therefore, the non-existence of conflicts over peers is a necessary condition for the realizability of the specification. In the FES, two events may be in conflict. In other words, there exists no computation that contains both events even though they are not syntactically in conflict; such a conflict is called a semantic conflict. The existence of semantic conflict in an FES means the existence of a minimal conflict relation; this requires a reduction of conflict relations (conflict reduction) in the FES for the realizability checking of a specification.

The objective of this paper is to derive a conflict-reduction algorithm for FESs. Some research on the reduction or minimization of event structures already exists. Armas-Cervantes et al. [15] address the problem of reducing the size of an event structure for an AES and FES, where the reduction is done by merging events to reduce their number. Baldan and Raffaetà have studied the notion of folding [16]. This research focuses on reducing event structures by merging events. In contrast, conflict reduction in this paper is accomplished by removing conflicts, which are syntactically in conflict and remain in semantic conflict when they are removed, from the FES.

The rest of this paper is organized as follows. Section 2 recalls the PES and FES. The motivation for conflict reduction is described in Sect. 3. Section 4 provides a necessary and sufficient condition for conflict reduction and algorithms to compute the semantic conflict and conflict reduction. Computational experiments that were conducted are described in Sect. 5. Finally, Sect. 6 draws some conclusions.

2. Event Structures

2.1 Prime Event Structures

Most studies on event structures use the PES [1], whereas

Manuscript received March 1, 2022.
Manuscript revised August 1, 2022.
Manuscript publicized October 26, 2022.

[†]The author is with the Faculty of Information Science and Technology, Osaka Institute of Technology, Hirakata-shi, 573-0916 Japan.
[††]The author was with the Graduate School of Engineering, Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: toshiyuki.miyamoto@oit.ac.jp
DOI: 10.1587/transfun.2022MAP0002

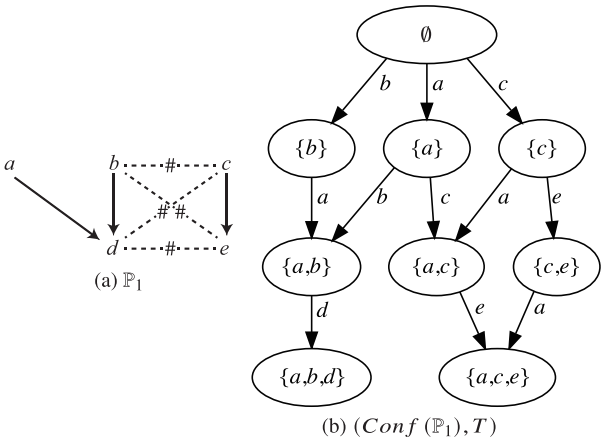


Fig. 1 PES $\mathbb{P}_1$ (a) and its configuration space (b).

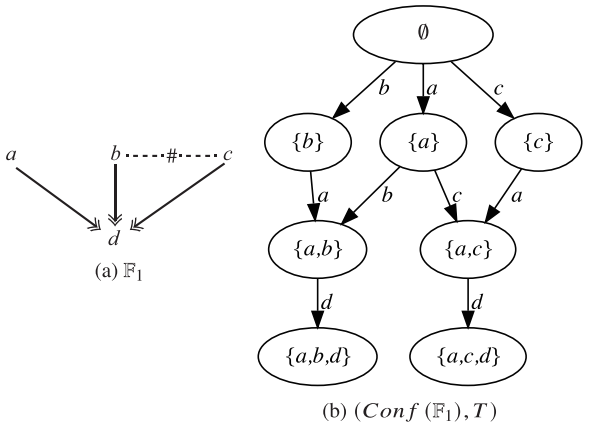


Fig. 2 FES $\mathbb{F}_1$ (a) and its configuration space (b).

this paper uses a variant of event structures: FES [3], [4]. To make the difference clear, we recall the formal definition of the PES using the notation in [15].

Definition 1 (prime event structure): A *prime event structure* (PES) is a tuple $\mathbb{P} = (E, \leq, \#)$, where E is a set of events, $\leq$ is a *causality* relation, and $\sharp$ is a *conflict* relation such that

1. the causality relation $\leq$ is a partial order, i.e., $\leq$ is reflexive, anti-symmetric, and transitive, and $\lfloor e \rfloor = \{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$;
2. the conflict relation $\#$ is irreflexive, symmetric, and hereditary with respect to causality, i.e., for all e, e', $e'' \in E$, if $e \# e' \leq e''$ then $e \# e''$.

The computations of a PES are described in terms of configurations, in other words, sets of events that are closed with respect to causality and conflict-free.

Definition 2 (configuration): The *configuration* of a PES $\mathbb{P} = (E, \leq, \#)$ is a finite set of events $C \subseteq E$ such that

1. for all $e \in C$, $\lfloor e \rfloor \subseteq C$, and
2. for all $e, e' \in C$, $\neg(e \# e')$.

The set of configurations of a PES $\mathbb{P}$ is denoted by $Conf(\mathbb{P})$.

Figure 1(a) depicts a PES $\mathbb{P}_1$. The first condition in Definition 2 means that when $e \in C$, then all the $\leq$-predecessors of the event e must be contained in C. Therefore, any configuration containing d of the PES $\mathbb{P}_1$ must contain the events a and b. The second condition in Definition 2 means that any configuration must be conflict-free. In the PES $\mathbb{P}_1$, since $b \# c$, there is no configuration that contained b and c.

Let T be the transition relation on the set $Conf(\mathbb{P})$ of configurations. A pair (C, C') of configurations C and C' is in T when the configuration changes from C to C' by the occurrence of an event. The graph $CS = (Conf(\mathbb{P}), T)$ is called a *configuration space*. The configuration space of the PES in Fig. 1(a) is depicted in Fig. 1(b).

2.2 Flow Event Structures

Definition 3 (flow event structure): A *flow event structure* (FES) is a tuple $\mathbb{F} = (E, \prec, \#)$, where E is a set of events, $\prec$ is a *flow* relation, and $\#$ is a *conflict* relation such that

1. the flow relation $\prec \subseteq E \times E$ is irreflexive;
2. the conflict relation $\# \subseteq E \times E$ is symmetric.

The $\prec$-predecessors of an event $e \in E$ are defined as $\bullet e = \{e' \mid e' \prec e\}$.

Figure 2(a) depicts an FES. A flow relation is represented with a double-headed arrow, and a conflict relation is represented with a dotted line labeled by a sharp ($\#$).

In a PES, the causality relation is transitive; all the events in $\lfloor e \rfloor \setminus \{e\}$ must precede the occurrence of e. However, the flow relation in an FES is not required to be transitive. The $\prec$-predecessors $\bullet e$ of an event e can be seen as a set of possible immediate causes for e in an FES. In an FES, conflicts can exist in $\bullet e$; the event e needs to be preceded by a maximal and conflict-free subset of $\bullet e$. The set of maximal and conflict-free $\prec$-predecessors of an event e is denoted by $\Phi(e)$. In the FES in Fig. 2(a), events b and c are in conflict; $\prec$-predecessors of event d are $\bullet d = \{a, b, c\}$ and the set of maximal and conflict-free $\prec$-predecessors of d is $\Phi(d) = \{\{a, b\}, \{a, c\}\}$. Thus, the occurrence of d must be preceded by either $\{a, b\}$ or $\{a, c\}$.

In a PES, the causality relation is a partial order; in other words, the causality relation is acyclic. In an FES, the flow relation is required to be irreflexive only; thus, it could be cyclic. An FES is called *acyclic* when the flow relation is acyclic. We study conflict reduction of FESs. For cyclic FESs, conflict reduction must be checked for each occurrence of conflict pairs. Therefore, we assume any FES to be acyclic in this paper.

The notion of configuration in an FES is defined in [3], [4], [15] as follows:

Definition 4 (configuration): The configuration of an FES $\mathbb{F}$ is a finite set of events $C \subseteq E$ such that

1. (conflict-freeness) $\neg(e \# e')$ for all $e, e' \in C$;
2. ($\prec$-closedness) $\prec^*_{|C}$ is a partial order;
3. (maximality) for all $e \in C$ and $e' \notin C$ s.t. $e' \prec e$, there exists an $e'' \in C$ such that $e' \# e'' \prec e$,

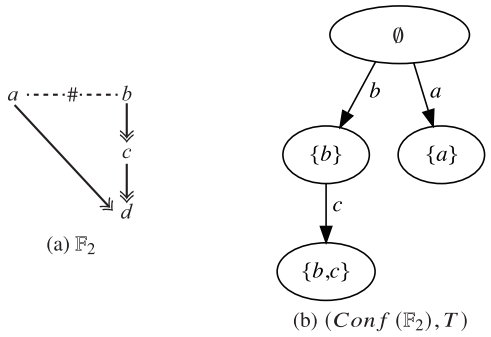


Fig. 3 An example of semantic conflict. Events a and c are in semantic conflict because no configuration that contains both a and c exists in $Conf(\mathbb{F}_2)$.

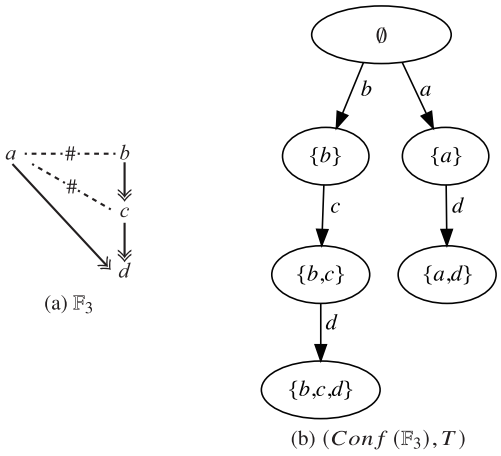


Fig. 4 An FES contains reducible conflicts.

where $\prec^*$ is the reflexive and transitive closure of $\prec$; $\prec_{|C}$ is the restriction of $\prec$ to the relation on C.

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

A configuration is a conflict-free and $\prec$-closed subset of events. The third condition means that given an event $e \in C$, for any $\prec$-predecessor $e' \prec e$, either $e' \in C$ or it is excluded by the existence of $e'' \in C$ such that $e' \# e'' \prec e$. Thus, for any $e \in C$, the configuration C must include a maximal and conflict-free subset of the $\prec$-predecessors of e. The configuration space of the FES $\mathbb{F}_1$ is depicted in Fig. 2(b).

In the FES in Fig. 3(a), events a and c are not syntactically in conflict; however, there is no configuration that contains both a and c. Thus, a and c are semantically in conflict.

Definition 5 (semantic conflict): The events e and e' in an FES $\mathbb{F}$ are in *semantic conflict*, which is denoted by $e \#_S e'$, when for all configurations $C \in Conf(\mathbb{F})$, it does not hold that $\{e, e'\} \subseteq C$.

Clearly, $\# \subseteq \#_S$. Moreover, $e \#_S e$ could be possible; in this case, e never occurs and is called *dead*.

Configurations C_1 and C_2 are in conflict, which is denoted by $C_1 \# C_2$, when there exist any conflict events in C_1 and C_2:

$$C_1 \# C_2 \quad \Leftrightarrow \quad \exists e_1 \in C_1, \exists e_2 \in C_2 : e_1 \# e_2 \tag{1}$$

3. Motivation for Conflict Reduction

The motivation for conflict reduction comes from the existence of semantic conflicts in FESs.

Consider a designing problem, which is called the choreography realization problem (CRP) [9], of a distributed system that is composed of two peers: p_1 and p_2. Suppose that the specification of the entire system is given by the FES in Fig. 4(a), which is called choreography. The objective of the CRP is to obtain the specification for each peer that acts along with the choreography.

The realizability of choreography depends on the allocation of events to peers. If two events that are in flow relation are allocated on different peers, the flow relation can be realized by inserting send and receive events of a message between the two events. On the other hand, if two events that are in conflict relation are allocated on different peers, the conflict relation cannot be realized in general.

If events a, b, and c are the events of peer p_1 and event d is the event of peer p_2, the choreography is realizable because both of the two dependency, (a, d) and (c, d), are in flow relation.

If event a is the event of peer p_1 and events b, c, and d are the events of peer p_2, the choreography is not realizable because the exclusive occurrence of a or b cannot be realized without introducing an additional peer for the control of exclusive occurrence or by modifying the choreography.

From the discussion of the two cases, one may raise the idea that non-existence of conflicts among peers is a necessary condition for realizability. However, this is not true. Consider the case where events a and b are the events of peer p_1 and events c and d are the events of peer p_2. Because a and c are in semantic conflict regardless of the presence or absence of $a \# c$, we do not have to consider $a \# c$ for the checking of realizability. In this case, the choreography is realizable.

The motivation for conflict reduction comes from the necessity of obtaining reduced conflict relation for the checking of realizability.

Note that, as shown in Fig. 3(b) and Fig. 4(b), the configuration spaces of FESs $\mathbb{F}_2$ and $\mathbb{F}_3$ are different. That means that the conflict reduction causes a change in the configuration space. To avoid this problem, we have to use the original maximal and conflict-free $\prec$-predecessors $\Phi(e)$ on checking the occurrence of event e.

4. Conflict Reduction of Acyclic Flow Event Structures

4.1 Conditions for Conflict Reduction

Let us denote the FES obtained after reducing a conflict $e \# e'$ from an FES $\mathbb{F}$ by

$$\mathbb{F}\backslash_{(e\#e')} = (E, \prec, \# \setminus \{e\#e'\}). \tag{2}$$

As we saw in Figs. 3 and 4, the configuration spaces of $\mathbb{F}$ and $\mathbb{F}\backslash_{(e\#e')}$ are different in general. Let us denote the reduced FES in which the conflict relation is replaced by the semantic conflict relation by

$$\mathbb{F}\backslash^{S}_{(e\#e')} = (E, \prec, (\# \setminus \{e\#e'\})_S). \tag{3}$$

If the configuration spaces of $\mathbb{F}$ and $\mathbb{F}\backslash^{S}_{(e\#e')}$ are isomorphic, e and e' are in semantic conflict regardless of the presence or absence of $e\#e'$. In this case, $e\#e'$ is reducible for the checking of realizability.

Definition 6 (reducible conflict): A conflict $e\#e'$ is called *reducible* from an FES $\mathbb{F}$ when $Conf(\mathbb{F}) = Conf(\mathbb{F}\backslash^{S}_{(e\#e')})$.

According to Definition 6, checking reducibility of a conflict pair requires constructing all configurations; however, it is computationally expensive. Let us give another definition of configurations using *local configurations*.

Definition 7 (local configuration): A local configuration X_e of an event e is a configuration satisfying the minimality condition in addition to the conditions in Definition 4:

4. (minimality) $\forall C \in Conf(\mathbb{F}) : C \subsetneq X_e \rightarrow e \notin C$

The notion of local configuration can be seen in the unfolding research on Petri nets [17]. A local configuration of e is a minimal and $\prec$-closed set of events for the occurrence of event e.

The set $loc(e)$ of all local configurations of an event e is inductively calculated using the local configurations of the $\prec$-predecessors as follows:

$$\begin{aligned} loc(e) = \{X_e \mid \exists K \in \Phi(e),\ \exists L \in \prod_{e'\in K} loc(e') :\\ X_e = \{e\} \cup \bigcup_{i\in\{1,\ldots,|K|\}} L[i],\\ (\forall e_1, e_2 \in X_e :\ \neg(e_1\#e_2))\} \end{aligned} \tag{4}$$

where $\prod$ stands for the Cartesian product and $L[i]$ is the i-th element in the tuple L. For a minimal event e of an FES $\mathbb{F}$, since $\Phi(e) = \{\emptyset\}$, $loc(e)$ is given as follows:

$$loc(e) = \big\{\{e\}\big\} \tag{5}$$

The local configuration of a minimal event satisfies both of the $\prec$-closedness in Definition 4 and the minimality in Definition 7. The clause $X_e = \{e\} \cup \bigcup_{e'\in K} X_{e'}, K \in \Phi(e)$ means that a local configuration is a union of local configurations of the maximal and conflict-free $\prec$-predecessors; therefore, X_e satisfies maximality in Definition 4. Using the clause, both of $\prec$-closedness and minimality of any X_e can be proved by induction. The clause $\forall e_1, e_2 \in X_e : \neg(e_1\#e_2)$ expresses conflict-freeness of X_e; therefore, X_e satisfies conflict-freeness in Definition 4. Thus, X_e in (4) is a local configuration. Because we assume that any FES is acyclic, local configurations are well defined. If no local configuration that satisfies (4) exists, the event e is dead.

For the FES $\mathbb{F}_2$ in Fig. 3, we can find the local configurations as follows:

$$loc(a) = \big\{\{a\}\big\} \tag{6}$$

$$loc(b) = \big\{\{b\}\big\} \tag{7}$$

$$loc(c) = \big\{\{b, c\}\big\} \tag{8}$$

$$loc(d) = \emptyset \tag{9}$$

Because a local configuration is a configuration, a union of local configurations is a configuration when no conflict exists in it.

Proposition 1: A finite set C of events is a configuration of an FES if and only if the two conditions:

1. $\forall e, e' \in C : \neg(e\#e')$;
2. $\forall e \in C, \exists X_e \in loc(e) : X_e \subseteq C$.

are satisfied.

The first condition exists for conflict-freeness; the second condition means that C is a union of local configurations.

The following lemma gives a condition to check semantic conflicts using local configurations.

Lemma 1: Let x and y be events of an FES $\mathbb{F} = (E, \prec, \#)$. The events x and y are in semantic conflict if and only if the following holds:

$$\forall X \in loc(x), \forall Y \in loc(y), \exists e_x \in X, \exists e_y \in Y : e_x \# e_y \tag{10}$$

Proof First, suppose that

$$\exists X \in loc(x), \exists Y \in loc(y), \forall e_x \in X, \forall e_y \in Y : \neg(e_x \# e_y)$$

holds. Let X_1 and Y_1 be local configurations that satisfy the condition, then $X_1 \cup Y_1$ becomes a configuration that contains both x and y.

Conversely, suppose that there exists a configuration C that contains both x and y while satisfying (10). From the second condition of Proposition 1, C must include the local configurations X_1 and Y_1 of x and y, respectively. From the first condition of Proposition 1, for all $e, e' \in X_1 \cup Y_1$, e and e' must not be in conflict. This contradicts the assumption. □

The following lemma gives a condition to check reducibility using local configurations.

Lemma 2: Let x and y be events of an FES $\mathbb{F} = (E, \prec, \#)$ and $x\#y$. The conflict $x\#y$ is reducible if and only if the following condition holds:

$$\begin{aligned} &\forall K_x \in \Phi(x), \forall X = \{x\} \cup \bigcup_{e_x\in K_x} X_{e_x},\\ &\forall K_y \in \Phi(y), \forall Y = \{y\} \cup \bigcup_{e_y\in K_y} Y_{e_y} :\\ &\qquad \Big|\big\{\{e_1, e_2\} \subseteq X \cup Y \mid e_1 \# e_2\big\}\Big| \geq 2, \end{aligned} \tag{11}$$

where X_{e_x} and Y_{e_y} are local configurations in $loc(e_x)$ and $loc(e_y)$, respectively.

Proof The set X (or Y) in (11) is a candidate of local configuration. If it contains a conflict pair, it cannot be a local configuration; otherwise, it becomes a local configuration. Thus, every local configuration in (10) appears in (11). Clearly, if (11) is satisfied, then (10) is satisfied, i.e., x and y are in semantic conflict.

Moreover, if (11) is satisfied, and X and Y are local configurations, then there exists another pair such that $e_1 \# e_2$, $e_1 \in X$, and $e_2 \in Y$. This holds for any pair of local configurations for x and y. Thus, x and y are in semantic conflict regardless of the presence or absence of $x \# y$.

Next, we have to ensure that new configurations do not become executable by the reduction. When $|\{\{e_1, e_2\} \subseteq X \cup Y \mid e_1 \# e_2\}| \geq 2$, the set $X \cup Y$ contains more than one conflict pair. Therefore, reducing $x \# y$ does not produce any new local configuration. Thus, if (11) is satisfied, then $x \# y$ is reducible.

Conversely, if the following condition:

$$\exists K_x \in \Phi(x), \exists X = \{x\} \cup \bigcup_{e_x \in K_x} X_{e_x},$$
$$\exists K_y \in \Phi(y), \exists Y = \{y\} \cup \bigcup_{e_y \in K_y} Y_{e_y} :$$
$$\left|\left\{\{e_1, e_2\} \subseteq X \cup Y \mid e_1 \# e_2\right\}\right| = 1 \tag{12}$$

holds, then $\left\{\{e_1, e_2\} \subseteq X \cup Y \mid e_1 \# e_2\right\} = \{x, y\}$. Therefore, $x \# y$ is not reducible because $X \cup Y$ become a new configuration when $x \# y$ is reduced. Thus, if $x \# y$ is reducible, then (11) is satisfied. □

4.2 Algorithms

4.2.1 Conflict Reduction Using Local Configurations

Let $\#^-$ be the conflict relation that is reduced by removing reducible conflicts from $\#$. The following part of this section provides an algorithm to compute the semantic conflict relations $\#_S$ and reduced conflict relations $\#^-$. Let $\#_O$ be a conflict relation such that $x \#_O y$ when x and y are in semantic conflict, regardless of the presence or absence of $x \# y$. Clearly, the semantic conflicts $\#_S$ and reducible conflict relations are given by $\# \cup \#_O$ and $\# \cap \#_O$, respectively. Let LC be the set of local configurations of an FES, and $\#_{LC}$ be the conflict relations on LC.

Consider a case when a new local configuration X of x is being added and the conflict with a local configuration Y of y is being checked. Let $X = \{x\} \cup (\cup X_e)$ and $Y = \{y\} \cup (\cup Y_e)$, respectively; the condition (1) of local configurations to be in conflict can then be given as follows:

$$\begin{aligned} X \#_{LC} Y \quad &\Leftrightarrow \quad \exists x' \in X, \exists y' \in Y : x' \# y' \\ &= \exists x' \in \{x\} \cup (\cup X_e), \exists y' \in \{y\} \cup (\cup Y_e) : x' \# y' \\ &= (x \# y) \vee (\exists x' \in (\cup X_e) : x' \# y) \\ &\quad \vee (\exists y' \in (\cup Y_e) : x \# y') \vee ((\cup X_e) \#_{LC} (\cup Y_e)) \end{aligned} \tag{13}$$

Let us define the relation $X \#'_{LC} Y$ as follows:

Algorithm 1 Compute semantic conflict

Input: $\mathbb{F} = (E, \prec, \#)$
Output: $\#_S$, $\#_O$

```
#_O ← ∅; LC ← ∅; #_LC ← ∅; #'_LC ← ∅;
E_min ← the set of minimal events of 𝔽;
tsort[] ← an array of events such that ∀i, j ∈ {0, ..., |E| − 1} : i < j ⇒ ¬(tsort[j] <^+ tsort[i]);
for i ← 0 to |E| − 1 do
    e_i ← tsort[i];
    loc(e_i) ← ∪_{K∈Φ(e_i)} {{e_i} ∪ ⋃_{X∈V'_i} X |
                V'_i is a maximal clique of G_K with size |K|};
    if e_i ∉ E_min and loc(e_i) = ∅ then
        #_O ← #_O ∪ {(e_i, e_i)};
        continue;
    end if
    LC ← LC ∪ loc(e_i);
    for all X ∈ loc(e_i) do
        for all Y ∈ LC do
            if condition (14) holds then
                #_LC ← #_LC ∪ {(X, Y), (Y, X)} ; #'_LC ← #'_LC ∪ {(X, Y), (Y, X)} ;
            end if
            y ← the maximal element in Y;
            if e_i # y then
                #_LC ← #_LC ∪ {(X, Y), (Y, X)} ;
            end if
        end for
    end for
    for j ← 0 to i − 1 do
        e_j ← tsort[j];
        if condition (11) holds then
            #_O ← #_O ∪ {(e_i, e_j), (e_j, e_i)};
        end if
    end for
end for
#_S ← #_O ∪ #;
```

$$\begin{aligned} X \#'_{LC} Y \quad &\Leftrightarrow \\ &(\exists x' \in (\cup X_e) : x' \# y) \vee (\exists y' \in (\cup Y_e) : x \# y') \\ &\vee ((\cup X_e) \#_{LC} (\cup Y_e)) \end{aligned} \tag{14}$$

The condition (13) can be rewritten as follows:

$$X \#_{LC} Y \quad \Leftrightarrow \quad (x \# y) \vee (X \#'_{LC} Y) \tag{15}$$

Note that the first two terms in (14) can be checked using the given conflict relation $\#$; the third term can be checked using the conflict relation $\#_{LC}$ for each pair of local configurations of $\prec$-predecessors.

Algorithm 1 computes the conflict relation $\#_O$ and the conflict relation $\#_{LC}$ together for an FES $\mathbb{F} = (\Sigma, \prec, \#)$. In the algorithm, $tsort[]$ is an array of events representing a topological sort with respect to the flow relation. To find the local configurations of an event e_i, we use the maximal cliques enumeration algorithm [18] on an undirected graph. Let $G_K = (V_K, E_K)$ be an undirected graph of $K \in \Phi(e_i)$, where V_K and E_K are defined as follows:

$$V_K = \left\{X \mid X \in loc(e), e \in K\right\} \tag{16}$$
$$E_K = \left\{\{X_1, X_2\} \subseteq V_K \mid \neg(X_1 \#_{LC} X_2)\right\} \tag{17}$$

Thus, each node of the graph is a local configuration of $\prec$-predecessors; each edge represents that the adjacent local

Algorithm 2 Conflict reduction using local configurations

Input: $\mathbb{F} = (E, \prec, \#)$
Output: $\#^{-}$
1: **for all** $e \in E$ **do**
2: $\quad \Phi(e) \leftarrow$ the maximal and conflict-free $\prec$-predecessors of e;
3: **end for**
4: $\#_O \leftarrow$ execute Algorithm 1;
5: **while** $\#_O \cap \# \neq \emptyset$ **do**
6: $\quad e \# e' \leftarrow$ select a conflict from $\#_O \cap \#$;
7: $\quad \mathbb{F} \leftarrow \mathbb{F}\backslash_{(e \# e')}$
8: $\quad \#_O \leftarrow$ execute Algorithm 1;
9: **end while**
10: $\#^{-} \leftarrow \#$;

configurations are not in conflict. Therefore, each clique with size $|K|$ is the set of local configurations that constructs a local configuration of e_i.

Lemma 3: Let $(E, \prec, \#)$ be an FES. If $e \in E$ and $K \in \Phi(e)$, then for every $X_e \subseteq E$, $(\exists L \in \prod_{e' \in K} loc(e') : X_e = \{e\} \cup \bigcup_{i \in \{1,\ldots,|K|\}} L[i] \wedge (\forall e_1, e_2 \in X_e : \neg(e_1 \# e_2)))$ iff there exists a maximal clique (V'_K, E'_K) of G_K with $|V'_K| = |K|$ and $X_e = \{e\} \cup \bigcup_{X \in V'_K} X$.

In Algorithm 1, local configurations of event e_i are computed by enumerating maximal cliques of the graph G_K at line 6. If the condition at line 7 holds, the event e_i is dead. In lines 12 to 22, the conflict relations $\#_{LC}$ and $\#'_{LC}$ is updated, and in lines 23 to 28, the conflict relation $\#_O$ are updated. The lines from 5 to 28 are repeatedly executed for each node along with the topological sort $tsort[]$; therefore, when the conflicts $\#_{LC}$, $\#'_{LC}$, and $\#_O$ of event e_i are computed, the conflicts of its $\prec$-ancestors have been already determined.

Algorithm 2 performs conflict reduction. The removal of a conflict from $\#$ may make other conflicts non-reducible; thus, conflict reduction must be done one by one. The set $\Phi(e_i)$ of maximal and conflict-free $\prec$-predecessors at line 6 in Algorithm 1 must be obtained from the original conflict relation $\#$; thus, it is computed at line 2. At line 4, Algorithm 1 is executed to find $\#_O$. A conflict is selected from $\#_O \cap \#$ at line 6; it is removed from $\mathbb{F}$ at line 7, and Algorithm 1 is executed again. The lines from 6 to 8 are continued while $\#_O \cap \#$ is not empty. The final event structure depends on the selection at line 6. To find an optimal event structure, an optimal selection is necessary. However, the optimization is out of the scope of this paper[†].

The worst-case time complexity of enumerating maximal cliques is exponential to the size of the graph [19]. Therefore, the worst-case time complexity of Algorithm 1 is exponential to the size of the FES. However, in many cases, it is expected that the number of local configurations of $\prec$-predecessors is much smaller than the size of the FES. Supposing that the time to enumerate maximal cliques is bounded by a constant and the number of local configurations is polynomial to the size of the FES, then the time complexity of the algorithm becomes pseudo-polynomial.

To the best of the authors' knowledge, this is the first study of conflict reduction in acyclic FESs; the computational complexity is an open problem. In 4.2.2, we show another conflict-reduction algorithm as the comparison method with the proposed method.

[†]In fact, the definition of the optimal event structure depends on its application. For example, in the context of the CRP, we want to reduce as many conflicts among peers as possible for the checking of realizability.

Algorithm 3 Configuration space construction

Input: $\mathbb{F} = (E, \prec, \#)$
Output: $CS = (Conf(\mathbb{F}), T)$
1: execute Algorithm 1 to compute local configurations;
2: $C_0 \leftarrow \emptyset$; $Conf(\mathbb{F}) \leftarrow \{C_0\}$; $queue$.add(C_0);
3: **while** $queue$ is not empty **do**
4: $\quad C \leftarrow queue$.poll();
5: $\quad$ **for all** $e \in E$ **do**
6: $\quad\quad$ **for all** $X_e \in loc(e)$ **do**
7: $\quad\quad\quad X'_e \leftarrow X_e \setminus \{e\}$;
8: $\quad\quad\quad$ **if** $X'_e \subseteq C \wedge e \notin C \wedge \forall e' \in C : \neg(e' \# e)$ **then**
9: $\quad\quad\quad\quad C' \leftarrow C \cup \{e\}$;
10: $\quad\quad\quad\quad$ **if** $C' \notin Conf(\mathbb{F})$ **then**
11: $\quad\quad\quad\quad\quad Conf(\mathbb{F}) \leftarrow Conf(\mathbb{F}) \cup \{C'\}$; $queue$.add(C');
12: $\quad\quad\quad\quad$ **end if**
13: $\quad\quad\quad\quad T \leftarrow T \cup \{(C, C')\}$;
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: **end while**

Algorithm 4 Compute semantic conflict (naive method)

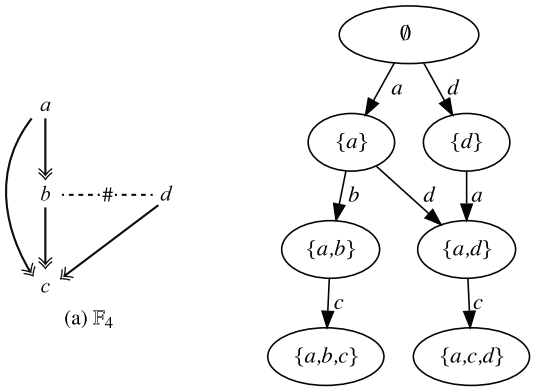
Input: $\mathbb{F} = (E, \prec, \#)$
Output: $\#_S$, $\#_O$
1: **for all** $e \in E$ **do**
2: $\quad \Phi(e) \leftarrow$ the maximal and conflict-free $\prec$-predecessors of e;
3: **end for**
4: $\#_O \leftarrow \emptyset$;
5: **for all** $(e, e') \in \#$ **do**
6: $\quad$ execute Algorithm 3 on $\mathbb{F}\backslash_{(e \# e')}$;
7: $\quad$ **if** $\nexists C \in Conf(\mathbb{F}\backslash_{(e \# e')}) : \{e, e'\} \subseteq C$ **then**
8: $\quad\quad \#_O \leftarrow \#_O \cup \{(e, e'), (e', e)\}$;
9: $\quad$ **end if**
10: **end for**
11: $\#_S \leftarrow \#_O \cup \#$;

4.2.2 Conflict Reduction Using Configuration Space

From Definition 5, we can check the reducibility of conflict $e \# e'$ using the configuration space of $\mathbb{F}\backslash_{(e \# e')}$. If no configuration exists that contains both e and e' in the configuration space, then the conflict is reducible.

Algorithm 3 computes the configuration space of an FES. At line 8, X'_e is a set of events obtained by removing event e from a local configuration X_e of e. Thus, $X'_e \subseteq C$ means that all the ancestors of e in X_e have occurred in C, $e \notin C$ means that e has not yet occurred in C, and $\forall e' \in C : \neg(e' \# e)$ means conflict freeness. Therefore, if the condition at line 8 is satisfied, event e can occur at configuration C. A new configuration C' and a transition from C to C' are then added in the configuration space at lines 9 to 13.

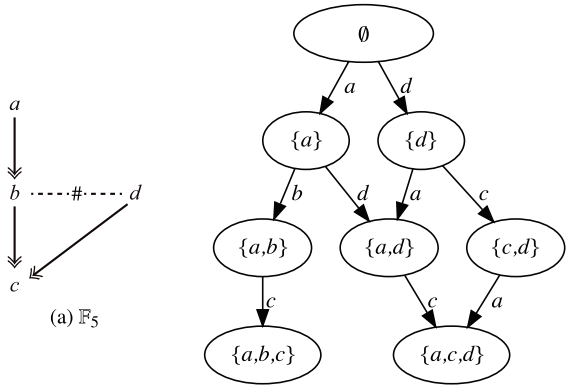
Algorithm 4 is another method (naive method) to compute the semantic conflict relations $\#_S$ and $\#_O$ using the



(a) $\mathbb{F}_4$

(b) configuration space of $\mathbb{F}_4$

Fig. 5 FES $\mathbb{F}_4$ with an irreducible flow $a \prec c$.



(a) $\mathbb{F}_5$

(b) configuration space of $\mathbb{F}_5$

Fig. 6 FES $\mathbb{F}_5$ without flow $a \prec c$.

configuration space. At line 6, a configuration space of $\mathbb{F}\backslash_{(e\#e')}$ is constructed using Algorithm 3. If any configuration does not include $\{e, e'\}$, the conflict $e \# e'$ is reducible.

Because this method uses the configuration space, its complexity is exponential to the size of the event structure.

For conflict reduction, we can obtain the proper algorithm by replacing "Algorithm 1" by "Algorithm 4" at lines 4 and 8 in Algorithm 2.

4.3 Note for Transitive Reduction of the Flow Relation

Algorithm 1 uses the maximal cliques enumeration algorithm on the graph of the $\prec$-predecessors. To shorten the computation time, the transitive reduction of the flow relation is effective. However, because of the existence of the conflict relation, the transitive reduction of the flow relation is not possible in general.

Figure 5(a) shows an FES $\mathbb{F}_4$ and (b) shows its configuration space. Figure 6(a) shows an FES $\mathbb{F}_5$ that is obtained by removing the flow $a \prec c$ from $\mathbb{F}_4$, and (b) shows its configuration space. Remember that the flow relation is not transitive; $a \prec b \prec c$ does not imply $a \prec c$. Thus, either b or d must occur before the occurrence of c; a is not required for the occurrence of c in $\mathbb{F}_5$, whereas either $\{a, b\}$ or $\{a, d\}$ must occur before c in $\mathbb{F}_4$.

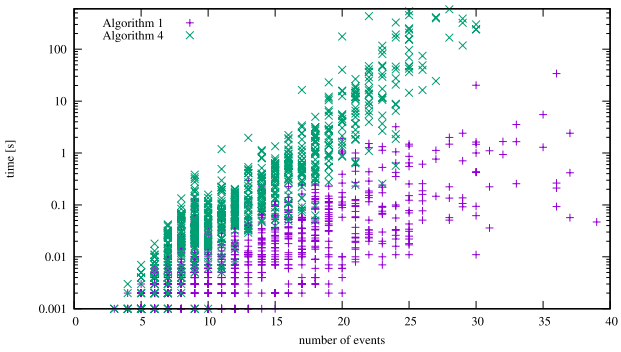


Fig. 7 Time to compute semantic conflict by Algorithms 1 and 4.

As can be seen in the figures, the configuration spaces of $\mathbb{F}_4$ and $\mathbb{F}_5$ are not isomorphic; the flow $a \prec c$ in $\mathbb{F}_4$ is not reducible. However, if no conflict exists between b and d, then the flow $a \prec c$ becomes reducible.

5. Computational Experiments

The algorithms in Sect. 4 were implemented using Java language; times to compute the semantic conflict relations by Algorithms 1 or 4 were compared. The experiments were conducted on a PC (CPU: Intel Core i7-9700, memory: 64 GB, OS: Linux 4.18). A total of 875 instances were randomly generated. The instances were pre-processed as follows: when an event was dead or isolated, it was removed; when two events were in both flow and conflict relations, the conflict was removed. The time to execute the algorithms was limited to 600 s.

The results are shown in Fig. 7. The horizontal axis is the number of events; the vertical axis is the time to run the algorithms. The vertical axis is a log scale; thus, the worst-case complexity in both algorithms was exponential to the number of events. However, clearly, the time to compute the semantic conflict relation was drastically shortened using the proposed Algorithm 1.

6. Conclusion

In this paper, we studied the conflict reduction of acyclic FESs. The computations of the event structures were described in terms of configurations. This paper provided another explanation of configurations using the notion of local configurations. The computation of conflict reduction is done using the local configurations; a great time reduction was achieved by the proposed method compared with the naive method.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP20K117460.

References

[1] M. Nielsen, G. Plotkin, and G. Winskel, "Petri nets, event structures

and domains, part I," Theoretical Computer Science, vol.13, no.1, pp.85–108, Jan. 1981.
[2] P. Baldan, A. Corradini, and U. Montanari, "Contextual Petri nets, asymmetric event structures, and processes," Information and Computation, vol.171, no.1, pp.1–49, Nov. 2001.
[3] G. Boudol and I. Castellani, "Permutation of transitions - An event structure semantics for CCS and SCCS," Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS, vol.354, pp.411–427, 1989.
[4] G. Boudol, "Flow event structures and flow nets," Semantics of Systems of Concurrent Processes, LNCS, vol.469, pp.62–95, 1990.
[5] R. Langerak, "Bundle event structures: A non-interleaving semantics for LOTOS," Formal Description Techniques V, M. Diaz and R. Groz, eds., IFIP transactions C, Communication systems, pp.331–346, North-Holland Publishing Company, 1991.
[6] G.M. Pinna, "Representing dependencies in event structures," Logical Methods in Computer Science, vol.16, no.2, pp.3:1–3:25, 2020.
[7] A. Armas-Cervantes, P. Baldan, M. Dumas, and L. Garcia-Bañuelos, "Diagnosing behavioral differences between business process models: An approach based on event structures," Information Systems, vol.56, pp.304–325, March 2016.
[8] L. Garcia-Bañuelos, N. van Beest, M. Dumas, M.L. Rosa, and W. Mertens, "Complete and interpretable conformance checking of business processes," IEEE Trans. Softw. Eng., vol.44, no.3, pp.262–290, Feb. 2018.
[9] 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, April 2008.
[10] 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.
[11] M. Izawa and T. Miyamoto, "A study on re-constructibility of event structures," IEICE Trans. Inf. & Syst., vol.E103-D, no.8, pp.1810–1813, Aug. 2020.
[12] N. van Beest, H. Groefsema, L. Garcia-Bañuelos, and M. Aiello, "Variability in business processes: Automatically obtaining a generic specification," Information Systems, vol.80, pp.36–55, Feb. 2019.
[13] H.P. de León, S. Haar, and D. Longuet, "Model-based testing for concurrent systems with labelled event structures," Software Testing, Verification and Reliability, vol.24, no.7, pp.558–590, Aug. 2014.
[14] J. Kienzle, G. Mussbacher, B. Combemale, and J. Deantoni, "A unifying framework for homogeneous model composition," Softw. Syst. Model., vol.18, no.5, pp.3005–3023, Dec. 2018.
[15] A. Armas-Cervantes, P. Baldan, and L. García-Bañuelos, "Reduction of event structures under history preserving bisimulation," Journal of Logical and Algebraic Methods in Programming, vol.85, no.6, pp.1110–1130, Oct. 2016.
[16] P. Baldan and A. Raffaetà, "Minimisation of event structures," Leibniz International Proceedings in Informatics, LIPIcs, p.15, Università Ca' Foscari Venezia, Venice, Italy, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Dec. 2019.
[17] K.L. McMillan, "Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits," Computer Aided Verification, LNCS, vol.663, pp.164–177, Jan. 1993.
[18] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph [H]," Communications of the ACM, vol.16, no.9, pp.575–577, Sept. 1973.
[19] F. Cazals and C. Karande, "A note on the problem of reporting maximal cliques," Theoretical Computer Science, vol.407, no.1-3, pp.564–568, Nov. 2008.

Toshiyuki Miyamoto received his B.E. and M.E. degrees in electronic engineering from Osaka University, Japan in 1992 and 1994, respectively. Moreover, he received Dr. of Eng. degree in electrical engineering from Osaka University, Japan in 1997. From 2000 to 2001, he was a visiting researcher in Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA. Currently, he is a Professor with the Faculty of Information Science and Technology, Osaka Institute of Technology. His areas of research interests include theory and applications of concurrent systems and multi-agent systems. He is a member of IEEE, SICE, and ISCIE.

Marika Izawa received the B.E. and M.E. degrees from Osaka University, Osaka, Japan in 2019 and 2021, respectively.