

Step semantics and action refinement in event structures

Weidong Tang^{1, 2, 3}, Jinzhao Wu^{1, 2, 3*}, Meiling Liu^{1, 4}

¹*School of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China*

²*Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041, China*

³*Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, China*

⁴*Science Computing and Intelligent Information Processing of GuangXi higher education key laboratory, Nanning 530023, China*

Received 5 May 2014, www.tsi.lv

Abstract

An event structure acts as a denotational semantic model of concurrent systems. Action refinement is an essential operation in the design of concurrent systems. However, there exists an important problem about preserving equivalence under action refinement. If two processes are equivalent with each other, we hope that they still can preserve equivalence after action refinement. In linear time equivalence and branching time equivalence spectrum, step equivalences, which include step trace equivalence and step bisimulation equivalence are not preserved under action refinement [17]. In this paper, we define a class of concurrent processes with specific properties and put forward the concept of clustered action transition, which ensures that step equivalences are able to preserve under action refinement.

Keywords: event structure, action refinement, concurrency, step equivalence, clustered equivalence

1 Introduction

In order to model concurrent systems, we hope to have formal method for hierarchical structure. Action refinement is the core operation of the hierarchical method, which interprets an action in higher abstract layer with a process in lower layer, hence reduces the level of abstraction and eventually reaches its implementation layer. In the development course from top to bottom of concurrent system, we must first build models, which depict the system with description language of top layer; subsequently, according to these descriptions, we complete its implementation. This course often requires equivalence notion to verify the correctness of implementation of system. More concretely, assuming that P represents the descriptions of system and Q represents its implementation, if P is equivalent with Q (expressed as $P \approx Q$), then this shows that Q is correct. In development, the description P of a system can be refined layer-by-layer; accordingly, its implementation Q can be converted from framework into code or electronic components. Only the description and its implementation at all levels are required to maintain equivalence so as to ensure correctness of its implementation. This leads to an important question what kind of equivalence is maintained under action refinement, that is, if two concurrent systems are equivalent with each other, we hope that they still can preserve equivalence after action refinement.

Vogler [31, 33] first raised the basic thought of preserving equivalence under action refinement. Czaja, Van Glabbeek and Goltz [34] demonstrated that if interleaving bisimulation equivalence doesn't produce choice operations or action self-concurrences after actions are refined then it can preserve equivalence under action refinement, but interleaving trace equivalence still cannot preserve. Goltz and Wehrheim [30] proved that history preserving bisimulation is consistent with global causal dependencies, but they did not further discuss about the problem how to preserve equivalence under action refinement, and did not discuss that there are other situations under environment of action independencies. At last, in paper [17] Van Glabbeek and Goltz summarized that research results of action refinement in recent ten years, gave a detailed explanation for preserving equivalence problem under action refinement, and proved that interleaving equivalence cannot preserve under action refinement in general, but did not discuss further. Moreover, no work further discusses preserving problem under action refinement of step trace equivalence and step bisimulation equivalence. In this paper, we define a class of concurrent processes with specific properties and put forward the concept of clustered action transition, which ensures that step equivalences are able to preserve under action refinement in the absence of constraints.

2 Event structures and action refinement

Assume that Act be a set of actions.

* *Corresponding author* e-mail: himrwujz@126.com

Definition 2.1 A event structure \mathcal{P} is a 5-tuple $(E, <, \#, \Delta, l)$, where

E is the set of events;

$< \subseteq E \times E$ is irreflexive partial relation, and satisfy the

rule of finite causes that $\forall e \in E: \{e1 \in E | e1 < e\}$ is finite; In addition, its inverse “<” is expressed as “>”;

$\# \subseteq E \times E$ is irreflexive and finite conflicting relation, and satisfy the rule of inheriting of conflict that $\forall e1, e2, e3 \in E: e1 < e2 \wedge e1 \# e3 \Rightarrow e2 \# e3$;

$\Delta \subseteq E \times E$ is irreflexive concurrent relation, altogether with < and # to satisfy the principle of partition that $< \cup \# \cup \Delta = E \times E$,

$e1 \Delta e2 \Leftrightarrow \neg(e1 = e2 \vee e1 < e2 \vee e2 < e1 \vee e1 \# e2)$;

$l: E \rightarrow Act$ is a label function of actions.

In this paper, let \mathbb{S} stand for the set of all event structures.

Definition 2.2 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. A relation between \mathcal{P} and \mathcal{Q} is called isomorphic (expressed as $\mathcal{P} \cong \mathcal{Q}$) iff there exists an bijection between their sets and preserves corresponding relations with <, #, Δ and same corresponding labels.

Unless specified, we do not discriminate isomorphic event structures.

The behaviour of event structure is depicted with configuration which is the set of events with specific properties. Configurations are considered as possible states of system. The following is its definition.

Definition 2.3 Let X be a subset of the set $E_{\mathcal{P}}$ of all events in event structure \mathcal{P} .

(1) X is left closed iff $\forall e1, e \in E: e \in X \wedge e1 < e \Rightarrow e1 \in X$;

(2) X is conflicted-free iff $\mathcal{P}|_X$ is conflicted-free;

(3) X is a configuration iff X is not only left closed but also conflicted-free.

Here, let $C(\mathcal{P})$ represent the set of all configurations in event structure \mathcal{P} .

A configuration $X (X \in C(\mathcal{P}))$ is called successfully terminated configuration iff $\forall e \in E: e \notin X \Rightarrow \exists e1 \in X: e1 \# e$.

The event structure is also often represented with graph, where \rightarrow, \dots stands for casual relation and immediately conflict relation in event structure respectively, inheriting conflict relation, which is not considered and independent relation is not explicitly expressed.

Example 2.1 The system $\mathcal{P} = (a|b) + (c; b; d)$, executing either a, b concurrently or c, b and d sequentially, can be described by the event structure with events $e1, e2, e3, e4, e5$ with $l(e1) = a, l(e3) = c, l(e2) = l(e4) = b, l(e5) = d$, where $e1 \Delta e2, e3 < e4 < e5$,

each of $e1, e2$ is in conflict with each of $e3, e4, e5$. This event structure is expressed as below.

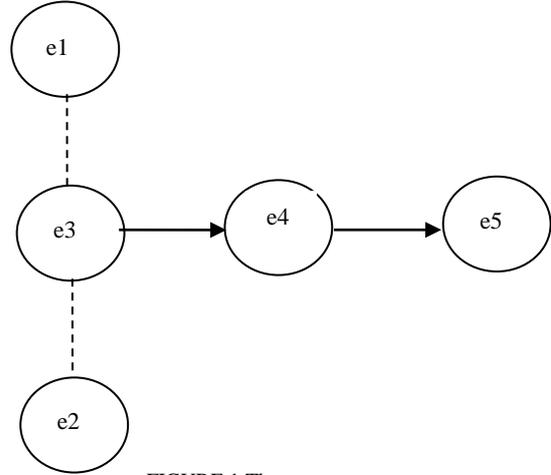


FIGURE 1 The event structure

Its configurations are $\emptyset, \{e1\}, \{e2\}, \{e1, e2\}, \{e3\}, \{e3, e4\}, \{e3, e4, e5\}$, where $\{e1, e2\}, \{e3, e4, e5\}$ are terminated configurations.

Basic thought of action refinement is: replace an action in higher layer with a process in lower layer, do it layer by layer, until get detailed design or implementation of system.

Definition 2.4 A function $ref: Act \rightarrow E - \{\emptyset\}$ is called a refinement function of event structure, iff $\forall a \in Act: ref(a)$ is not empty, finite and conflict-free. Let $\mathcal{P} \in \mathbb{S}$. $ref(\mathcal{P})$ is an event structure defined as follows:

$$E_{ref(\mathcal{P})} = \left\{ (e, e') \mid e \in E_{\mathcal{P}}, e' \in E_{ref(l_{\mathcal{P}}(e))} \right\}, \tag{1}$$

$$(e1, e1') <_{ref(\mathcal{P})} (e2, e2') \text{ iff } e1 <_{\mathcal{P}} e2 \text{ or } e1 = e2 \wedge e1' <_{ref(l_{\mathcal{P}}(e1))} e2', \tag{2}$$

$$(e1, e1') \#_{ref(\mathcal{P})} (e2, e2') \text{ iff } e1 \#_{\mathcal{P}} e2, \tag{3}$$

$$(e1, e1') \Delta_{ref(\mathcal{P})} (e2, e2') \text{ iff } e1 \Delta_{\mathcal{P}} e2 \text{ or } e1 = e2 \wedge e1' \Delta_{ref(l_{\mathcal{P}}(e1))} e2', \tag{4}$$

$$l_{ref(\mathcal{P})}(e, e') = l_{ref(l_{\mathcal{P}}(e))}(e'). \tag{5}$$

Example 2.2 Continue with Example 2.1. Assuming that $ref(b) = (b1; b2) + b3$, the event structure after action refinement is expressed as Figure 2.2.

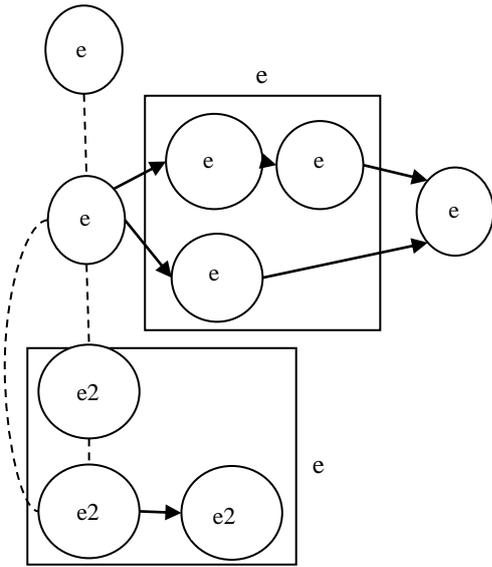


FIGURE 2 The event structure after action refinement

In the process of action refinement, each event e labelled by b is replaced by a disjoint copy, \mathcal{P}_e , of $ref(b)$, i.e. the event $e2$ is replaced by $(e22;e23)\#e21$ and the event $e4$ is replaced by $(e41;e42)\#e43$ [17]. The causality and conflict structure is inherited from \mathcal{P} : all events which were casually before e will be casually before all events of \mathcal{P}_e , every event, which casually followed e will casually follow all events of \mathcal{P}_e , and all events in conflict with e will be in conflict with all the events of all events which were casually before e will be casually before all events of \mathcal{P}_e .

3 Step equivalences

Step equivalences embody step trace equivalence and step bisimulation equivalence. To begin with, we give the definitions of single action transition and step action transition by comparison.

Definition 3.1 Let $\mathcal{P} \in \mathcal{S}$. A transition relation $X \xrightarrow{a} \mathcal{P} X'$ is called single action transition iff $a \in Act, X, X' \in C(\mathcal{P}), X \subseteq X'$, and $\exists e \in E_{\mathcal{P}} : X' - X = e, l_{\mathcal{P}}(e) = a$.

Here, $X \xrightarrow{a} \mathcal{P} X'$ denotes that the state expressed by Configuration X turns into the one expressed by Configuration X' after performing single action a in event structure $\mathcal{P} \in \mathcal{S}$.

Definition 3.2 Let $\mathcal{P} \in \mathcal{S}$. A transition $X \xrightarrow{A} X'$ is called a step action transition iff $A \in N^{Act}$ i.e., A is multiple set in $Act, X, X' \in C(\mathcal{P}), X \subseteq X', X' - X = G$, to make $\forall d, e \in G : d \Delta_{X'} e$ and

$l_{\mathcal{P}}(G) = A$, where $l_{\mathcal{P}}(G) \in N^{Act}$ is given by $l_{\mathcal{P}}(G)(a) = \{e \in G | l_{\mathcal{P}}(e) = a\}$.

Here, $X \xrightarrow{A} X'$ means that in event structure \mathcal{P} , after independently executing all actions of set A , the state expressed by configuration X is changed into the one expressed by configuration X' .

According to the above-mentioned definitions, obviously there is following proposition.

Proposition 3.1 A single action transition is also a step action transition.

Proof is omitted.

Then, we define trace and interleaving trace equivalence, step trace and step trace equivalence by comparison.

Definition 3.3 Let $\mathcal{P} \in \mathcal{S}$. A word $\mathcal{W} = a_1 \dots a_n \in Act^*$ is called a trace of event structure \mathcal{P} iff $\exists X_0, \dots, X_n \in C(\mathcal{P}) : X_0 = \emptyset$ and $X_{i-1} \xrightarrow{a_i} X_i, i = 1, \dots, n$.

Here, $trs(\mathcal{P})$ represents the set of all traces in event structure \mathcal{P} .

Definition 3.4 Let $\mathcal{P}, \mathcal{Q} \in \mathcal{S}$. A relation between \mathcal{P} and \mathcal{Q} is called interleaving trace equivalence (expressed as $\mathcal{P} \approx_{it} \mathcal{Q}$) iff $trs(\mathcal{P}) = trs(\mathcal{Q})$.

Definition 3.5 Let $\mathcal{P} \in \mathcal{S}$. A sequence $\mathcal{W} = A_1 \dots A_n (A_i \in N^{Act} (i = 1, \dots, n))$ is called a step trace of event structure \mathcal{P} iff $\exists X_0, \dots, X_n \in C(\mathcal{P}) : X_0 = \emptyset$ and $X_{i-1} \xrightarrow{A_i} X_i, i = 1, \dots, n$ is a step action transition.

Here, $steptrs(\mathcal{P})$ represents the set of all step traces of event structure \mathcal{P} .

Definition 3.6 Let $\mathcal{P}, \mathcal{Q} \in \mathcal{S}$. A relation between \mathcal{P} and \mathcal{Q} is called step trace equivalence (expressed as $\mathcal{P} \approx_{st} \mathcal{Q}$) iff $steptrs(\mathcal{P}) = steptrs(\mathcal{Q})$.

Furthermore, we define interleaving bisimulation equivalence and step bisimulation equivalence by comparison.

Definition 3.7 Let $\mathcal{P}, \mathcal{Q} \in \mathcal{S}$. A relation $R \subseteq C(\mathcal{P}) \times C(\mathcal{Q})$ is called a interleaving bisimulation between \mathcal{P} and \mathcal{Q} iff $(\emptyset, \emptyset) \in R$ and if $(X, Y) \in R$ then

$$\begin{aligned} X \xrightarrow{a} \mathcal{P} X', a \in Act &\Rightarrow \exists Y' : \\ Y \xrightarrow{a} \mathcal{Q} Y' \wedge (X', Y') \in R, \\ Y \xrightarrow{a} \mathcal{Q} Y', a \in Act &\Rightarrow \exists X' : \\ X \xrightarrow{a} \mathcal{P} X' \wedge (X', Y') \in R. \end{aligned}$$

A relation between \mathcal{P} and \mathcal{Q} is called interleaving bisimulation equivalence (expressed as $\mathcal{P} \approx_{ib} \mathcal{Q}$) iff there exists a interleaving bisimulation between them.

Definition 3.8 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. A relation $R \subseteq C(\mathcal{P}) \times C(\mathcal{Q})$ is called a step bisimulation between \mathcal{P} and \mathcal{Q} iff $(\emptyset, \emptyset) \in R$ and if $(X, Y) \in R$ then

$$\begin{aligned} X &\xrightarrow{A} \mathcal{P} X', A \in N^{Act} \Rightarrow \exists Y' : \\ Y &\xrightarrow{A} \mathcal{Q} Y' \wedge (X', Y') \in R; \\ Y &\xrightarrow{A} \mathcal{Q} Y', A \in N^{Act} \Rightarrow \exists X' : \\ X &\xrightarrow{A} \mathcal{P} X' \wedge (X', Y') \in R. \end{aligned}$$

A relation between \mathcal{P} and \mathcal{Q} is called step bisimulation equivalence (expressed as $\mathcal{P} \approx_{sb} \mathcal{Q}$) iff there exists a step bisimulation between them.

According to the definitions of trace equivalence and bisimulation equivalence, obviously there are following two propositions.

Proposition 3.2 $\mathcal{P} \approx_{ib} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{it} \mathcal{Q}$.

Proposition 3.3 $\mathcal{P} \approx_{sb} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{st} \mathcal{Q}$.

The paper [17] has showed that step bisimulation equivalence and step trace equivalence cannot preserve under action refinement. The following proposition shows that if no independency exists in event structure then step equivalences (include step trace equivalence and step bisimulation equivalence) are able to preserve under action refinement.

Proposition 3.4 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$, let ref be a refinement function. If independency between events in an event structure is such that $\Delta_{\mathcal{P}} = \Delta_{\mathcal{Q}} = \emptyset$ then

- (1) $\mathcal{P} \approx_{st} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$.
- (2) $\mathcal{P} \approx_{sb} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{sb} \text{ref}(\mathcal{Q})$.

Proof (1) Because $\Delta_{\mathcal{P}} = \Delta_{\mathcal{Q}} = \emptyset$, any transition in event structures \mathcal{P} and \mathcal{Q} is single action transition. Also, by proposition 3.1, $\mathcal{P} \approx_{st} \mathcal{Q} \Leftrightarrow \mathcal{P} \approx_{it} \mathcal{Q}$. By proposition 3.5(1) [31], $\mathcal{P} \approx_{it} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{it} \text{ref}(\mathcal{Q})$, namely, $\text{trs}(\text{ref}(\mathcal{P})) = \text{trs}(\text{ref}(\mathcal{Q}))$. In \mathcal{P} and \mathcal{Q} , every action label is exactly the same and is refined in the same way; moreover, $\Delta_{\mathcal{P}} = \Delta_{\mathcal{Q}} = \emptyset$. Hence, there exist same multiple action sets where actions are just concurrent in $\text{ref}(\mathcal{P})$ and $\text{ref}(\mathcal{Q})$. So we arrive at the conclusion that $\text{steptrs}(\text{ref}(\mathcal{P})) = \text{steptrs}(\text{ref}(\mathcal{Q}))$, namely $\text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$.

Proof (2) Because $\Delta_{\mathcal{P}} = \Delta_{\mathcal{Q}} = \emptyset$, any transition in event structures \mathcal{P} and \mathcal{Q} is single action transition. Also, by proposition 3.1, $\mathcal{P} \approx_{sb} \mathcal{Q} \Leftrightarrow \mathcal{P} \approx_{ib} \mathcal{Q}$. By proposition 3.5(2) [31], $\mathcal{P} \approx_{ib} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{ib} \text{ref}(\mathcal{Q})$. In \mathcal{P} and \mathcal{Q} , every action label is exactly the same and is refined in the same way; moreover, $\Delta_{\mathcal{P}} = \Delta_{\mathcal{Q}} = \emptyset$. Hence, there exist same multiple action sets where actions are all concurrent and branching time properties of corresponding action are just the same in $\text{ref}(\mathcal{P})$ and

$\text{ref}(\mathcal{Q})$. So we arrive at the conclusion that $\text{ref}(\mathcal{P}) \approx_{ib} \text{ref}(\mathcal{Q}) \Rightarrow \text{ref}(\mathcal{P}) \approx_{sb} \text{ref}(\mathcal{Q})$.

4 Clustered action transition

We introduce new one class action transition, where A is multiple set in action set Act and all actions within A independently perform with each other. We call this multiple set A as a clustered action and call this class transitions as clustered action transitions that is in fact one kind of special step action transition. With clustered action transitions, we construct two new types of equivalence.

Definition 4.1 Let $\mathcal{P} \in \mathbb{S}$. A transition $X \xrightarrow{A} X'$ is called a clustered action transition iff $A \in N^{Act}$ (i.e., A is multiple set in Act), $X, X' \in C(\mathcal{P})$, $X \subseteq X'$, $X' - X = G$, the events in set G satisfy:

- (1) entire independency of causes:
 $\forall d, e \in G : (d \Delta_{X'} e) \wedge (\{e_1 \in E_{\mathcal{P}} \mid e_1 \Delta_{\mathcal{P}} e\} \cup \{e\} = \{e_2 \in E_{\mathcal{P}} \mid e_2 \Delta_{\mathcal{P}} d\} \cup \{d\} = G)$ and $l_{\mathcal{P}}(G) = A$;
- (2) same causality:
 $\forall e_1 \in E_{\mathcal{P}} \setminus G, \exists e_2 \in G : (e_1 < e_2 \Rightarrow \forall e_3 \in G : e_1 < e_3)$
 $\vee (e_1 > e_2 \Rightarrow \forall e_3 \in G : e_1 > e_3)$;
- (3) same conflict relation:
 $\forall e_1 \in E_{\mathcal{P}} \setminus G, \exists e_2 \in G : (e_1 \# e_2 \Rightarrow \forall e_3 \in G : e_1 \# e_3)$.

Here, $l_{\mathcal{P}}(G) \in N^{Act}$ results from $l_{\mathcal{P}}(G)(a) = |\{e \in G \mid l_{\mathcal{P}}(e) = a\}|$.

Clustered action transition $X \xrightarrow{A} X'$ represents that after independently executing all actions of set A , the state expressed by configuration X is changed into the one expressed by configuration X' in event structure \mathcal{P} .

Example 4.1 In a simple process $K = (a \parallel b); (c \parallel d) + f$, its actions names corresponds to events e_a, e_b, e_c, e_d and e_f respectively. Assume that event structure model \mathcal{P} of process K is \mathcal{P}_K , figure 4.1 describes all single action transitions and configurations of \mathcal{P}_K , whereas figure 4.2 describes its all clustered action transitions and configurations. Find from comparison of two figures that when arriving at configuration $\{e_a, e_b, e_c, e_d\}$, there exists 10 possible transitions in figure 4.1, however there exists only 2 clustered action transitions in figure 4.2.

This example shows that clustered action transitions can simplify the system expressed by single action transitions. If applying this thought to simplify systems, a lot of good results may acquire.

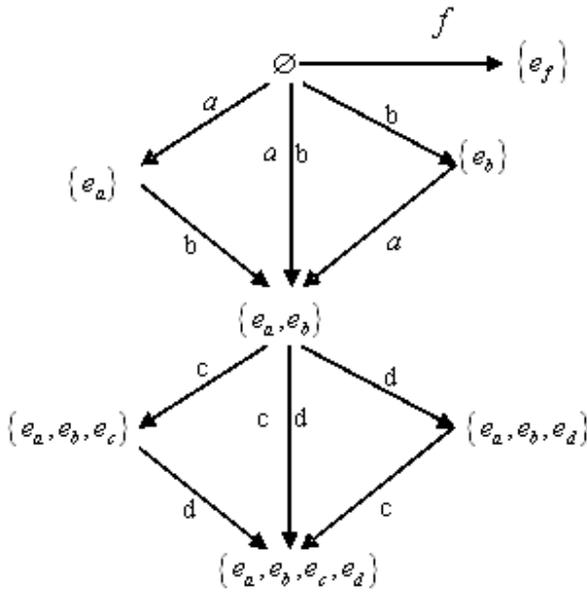


FIGURE 3 Single action transitions in a event structure

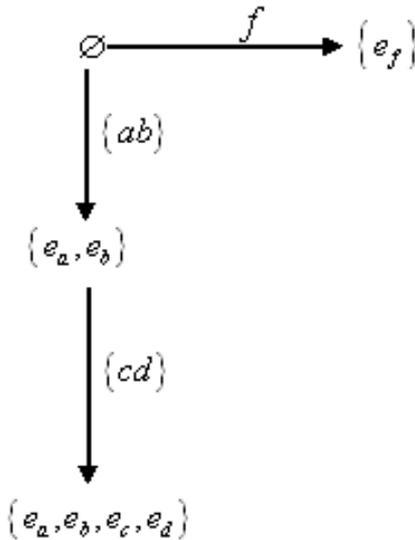


FIGURE 4 Clustered action transitions in a event structure

Definition 4.2 suppose that e is an event in event structure \mathcal{P} . Independency set of cause on e $\varphi(e)$, including itself of e , is defined as $\varphi(e) = \{e_1 \in E_{\mathcal{P}} \mid e_1 \Delta_{\mathcal{P}} e\} \cup \{e\}$.

In order to study the follow-up problems, we give a proposition in advance.

Proposition 4.1 Let $\mathcal{P} \in \mathbb{S}$. If all transitions in \mathcal{P} are clustered action transitions then

(1) $\forall d \in E_{\mathcal{P}} \Rightarrow \exists (X \xrightarrow{A} X') : \varphi(d) = G$, where $A \in N^{Act}$, $X, X' \in C(\mathcal{P}), X \subseteq X', X' - X = G$ and $l_{\mathcal{P}}(G) = A$;

(2) $\forall (X \xrightarrow{A} X') \Rightarrow \exists d \in E_{\mathcal{P}} : G = \varphi(d)$, where $A \in N^{Act}$, $X, X' \in C(\mathcal{P}), X \subseteq X', X' - X = G$ and $l_{\mathcal{P}}(G) = A$;

(3) $\bigcup_{e \in E_{\mathcal{P}}} \varphi(e) = E_{\mathcal{P}}$;

(4) $\forall e_1, e_2 \in E_{\mathcal{P}} : e_1 \neq e_2 \wedge \neg(e_1 \Delta e_2) \Rightarrow \varphi(e_1) \cap \varphi(e_2) = \emptyset$.

Proof Conclusions (1), (2) and (3) is very easy to reach, omitted here. Only give proof of (4). If $\varphi(e_1) \cap \varphi(e_2) \neq \emptyset$ then $\exists e \in \varphi(e_1) \cap \varphi(e_2)$. By definition 4.1, we obtain $\varphi(e) = \varphi(e_1) = \varphi(e_2)$, consequently obtain either $e_1 \Delta e_2$ or $e_1 = e_2$. This contradict with $e_1 \neq e_2 \wedge \neg(e_1 \Delta e_2)$. Accordingly, arrive at the conclusion that

$\forall e_1, e_2 \in E_{\mathcal{P}} : e_1 \neq e_2 \wedge \neg(e_1 \Delta e_2) \Rightarrow \varphi(e_1) \cap \varphi(e_2) = \emptyset$.

The above proposition shows: If all transitions in an event structure are clustered action transitions then all independent actions involved in a clustered action transition are seen as a “big” action, its corresponding events can be thought of as a “big” event. Hence, not only no independency between events exists in this event structure but also independency sets of cause divide set of events into different parts, which induces an equivalence relation in set of events of this event structure.

Here, clustered action transition is a special kind of step transition. The fact that there exist entire independency of cause in a clustered action transition means that an action only belongs to a certain clustered action and is not shared with other clustered action. For example, in the process $(a;b) \parallel c$, action c may belong to two clustered actions i.e. $\{a,c\}$ and $\{b,c\}$, hence the event structure corresponding to this process cannot satisfy the requirement of entire independency of cause. Thus, the transitions here are not clustered action transitions but general step action transitions. This exactly is the reason why process $(a;b) \parallel c$ is step trace

equivalence with process $(a \parallel c); b + a; (b \parallel c)$, but step trace equivalence cannot hold under action refinement. On the contrary, independent events in clustered action transition possess same causal relation and same conflict relation, so every clustered action can be seen as a “big” action. These advantages ensure that equivalence based on clustered action transition can hold under action refinement. Accordingly, we introduce concept of clustered action transition equivalence.

Definition 4.3 Let $\mathcal{P} \in \mathbb{S}$. A sequence $\mathcal{W} = A_1 \cdots A_n$ ($A_i \in N^{Act}$ ($i=1, \dots, n$)) is called a clustered trace of event structure \mathcal{P} iff $\exists X_0, \dots, X_n \in C(\mathcal{P}) : X_0 = \emptyset$ and $X_{i-1} \xrightarrow{A_i} X_i, i=1, \dots, n$ is a clustered action transition.

Here, $\text{Clusteredtrs}(\mathcal{P})$ represents the set of all clustered traces of event structure \mathcal{P} .

Subsequently, we define clustered trace equivalence.

Definition 4.4 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$, all transitions in \mathcal{P} and \mathcal{Q} be clustered transitions. Let $\text{Clusteredtrs}(\mathcal{P})$ and $\text{Clusteredtrs}(\mathcal{Q})$ be the sets of all clustered traces of \mathcal{P} , \mathcal{Q} respectively. A relation between \mathcal{P} and \mathcal{Q} is called clustered trace equivalence (expressed as $\mathcal{P} \approx_{ct} \mathcal{Q}$) iff $\text{Clusteredtrs}(\mathcal{P}) = \text{Clusteredtrs}(\mathcal{Q})$.

The following propositions show that clustered trace equivalence is accordance with step trace equivalence in given conditions.

Proposition 4.2 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then $\mathcal{P} \approx_{ct} \mathcal{Q} \Leftrightarrow \mathcal{P} \approx_{st} \mathcal{Q}$.

Proof By definition 3.2 and definition 4.1, we draw the above conclusion at once.

Proposition 4.3 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then $\mathcal{P} \approx_{ct} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{it} \mathcal{Q}$.

Proof By definition 4.1 and proposition 4.1, $\forall A_i, A_j (i=1, \dots, n, j=1, \dots, n, i \neq j)$ in a clustered trace, then all actions in A_i are independent of those in A_j . Performing of each action in A_i does not interfere with those in A_j , vice versa. Consequently, $\text{Clusteredtrs}(\mathcal{P}) = \text{Clusteredtrs}(\mathcal{Q}) \Rightarrow \text{trs}(\mathcal{P}) = \text{trs}(\mathcal{Q})$, i.e. $\mathcal{P} \approx_{ct} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{it} \mathcal{Q}$.

Provide that $W = A_1 \cdots A_n$, where $A_i \in N^{Act}$ ($i=1, \dots, n$) be a clustered trace, and $|A_i|$ stand for the number of elements, then the clustered trace W corresponds to $|A_1|! \times |A_2|! \times \cdots \times |A_n|!$ general traces. In example 4.1, the clustered trace $\{a, b\} \{c, d\}$ corresponds to 4 (namely, $|\{a, b\}| \times |\{c, d\}| = 2 \times 2! = 4$) general traces $abcd, abdc, bacd, badc$.

After discussing clustered trace equivalence, we begin with another new class of equivalence and study whether they can maintain under action refinement or not. This class of equivalence is designated clustered bisimulation equivalence. The following provides for its definition.

Definition 4.5 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$, let all transitions in \mathcal{P} and \mathcal{Q} be clustered action transitions. A relation $R \subseteq C(\mathcal{P}) \times C(\mathcal{Q})$ is called a clustered bisimulation between \mathcal{P} and \mathcal{Q} iff $(\emptyset, \emptyset) \in R$ and if $(X, Y) \in R$ then

$$\begin{aligned} X \xrightarrow{A} \mathcal{P} X', A \in N^{Act} \Rightarrow \\ \exists Y' : Y \xrightarrow{A} \mathcal{Q} Y' \wedge (X', Y') \in R; \end{aligned}$$

$$Y \xrightarrow{A} \mathcal{Q} Y', A \in N^{Act} \Rightarrow$$

$$\exists X' : X \xrightarrow{A} \mathcal{P} X' \wedge (X', Y') \in R.$$

A relation between \mathcal{P} and \mathcal{Q} is called clustered bisimulation equivalence (expressed as $\mathcal{P} \approx_{cb} \mathcal{Q}$) iff there exists a clustered bisimulation between them.

Obviously, by definition 4.5, we come to a decision that $\mathcal{P} \approx_{cb} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{ct} \mathcal{Q}$.

Proposition 4.4 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then $\mathcal{P} \approx_{cb} \mathcal{Q} \Leftrightarrow \mathcal{P} \approx_{sb} \mathcal{Q}$.

Proof is omitted.

The above proposition shows that clustered bisimulation equivalence is consistent with step bisimulation equivalence under certain conditions.

Proposition 4.5 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then $\mathcal{P} \approx_{cb} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{ib} \mathcal{Q}$.

Procedure of proof is similar to that of proposition 4.3, omitted here.

5 Preserving of step equivalences

The paper [17] has demonstrated that step equivalence cannot preserve under action refinement in the general case. However, proposition 3.4 shows that step equivalence without concurrency can preserve under action refinement. This part will extend proposition 3.4 and show that in the presence of concurrency, step equivalence may preserve under action refinement in given conditions.

Proposition 5.1 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$, let ref be a refinement function. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then

- (1) $\mathcal{P} \approx_{ct} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$;
- (2) $\mathcal{P} \approx_{st} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$.

Proof (1) By proposition 4.2, $\mathcal{P} \approx_{ct} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{st} \mathcal{Q}$.

By definition 4.1 and proposition 4.3, firstly, each clustered action transition corresponds to many single action transitions formed by interleaving performing of clustered actions; Secondly, when single action is refined, the corresponding clustered action is refined; Thirdly, Concurrent actions involved in per clustered action can be seen as one action, accordingly, their corresponding events also can be seen as one event. Therefore, by this treatment, there is not independency of cause in this event structure. Hence, we derive from proposition 3.4(1) that $\mathcal{P} \approx_{ct} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{st} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$.

(2) By proposition 4.2 and the above derivation process, we immediately reach a conclusion that $\mathcal{P} \approx_{st} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{st} \text{ref}(\mathcal{Q})$.

This proposition shows that if all transitions in an event structure are clustered action transitions then step trace equivalence can hold under action refinement.

Subsequently, we discuss the relationship between clustered bisimulation equivalence and step bisimulation equivalence under action refinement, as well as how to preserve step bisimulation equivalence under action refinement.

Proposition 5.2 Let $\mathcal{P}, \mathcal{Q} \in \mathbb{S}$, let ref be a refinement function. If all transitions in event structures \mathcal{P} and \mathcal{Q} are clustered action transitions then

$$(1) \mathcal{P} \approx_{\text{cb}} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{\text{sb}} \text{ref}(\mathcal{Q});$$

$$(2) \mathcal{P} \approx_{\text{sb}} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{\text{sb}} \text{ref}(\mathcal{Q}).$$

Proof (1) By proposition 4.4, $\mathcal{P} \approx_{\text{ct}} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{\text{st}} \mathcal{Q}$.

By definition 4.1 and proposition 4.5, firstly, each clustered action transition corresponds to many single action transitions formed by interleaving performing of clustered actions; Secondly, when single action is refined, the corresponding clustered action is refined; Thirdly, Concurrent actions involved in per clustered action can be seen as one action, accordingly, their corresponding events also can be seen as one event. Therefore, by this treatment, there are not independency of cause in this event structure. Hence, we derive from proposition 3.4(2) that $\mathcal{P} \approx_{\text{cb}} \mathcal{Q} \Rightarrow \mathcal{P} \approx_{\text{sb}} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{\text{sb}} \text{ref}(\mathcal{Q})$.

(2) By proposition 4.4 and the above derivation process, we immediately reach a conclusion that $\mathcal{P} \approx_{\text{sb}} \mathcal{Q} \Rightarrow \text{ref}(\mathcal{P}) \approx_{\text{sb}} \text{ref}(\mathcal{Q})$.

This proposition shows that if all transitions in an event structure are clustered action transitions then step bisimulation equivalence can hold under action refinement.

To sum up, if all transitions in event structures are clustered action transitions then two kinds of step equivalences, including step trace equivalence and step bisimulation equivalence, can hold under action refinement. Proposition 5.1 and proposition 5.2 extend proposition 3.4.

References

- [1] Wu J 2001 Action refinement in timed LOTOS *Proc. Of ASCM'01, World Scientific Publ.* 183-92
- [2] Aceto L, Hennessy M C B 1994 Adding action refinement to a finite process algebra *Information and Computation* **115** 179-247
- [3] Fecher H, Majster-Cederbaum M, Wu J 2002 Action refinement for probabilistic processes with true concurrency models *Lecture Notes in Computer Science* **2399** 77-94
- [4] Boudol G 1989 Atomic actions. *Bull. Eur. Ass. Theoret. Comput. Sci.* **38** 136-44
- [5] Busi N, van Glabbeek R J, Gorrieri R 1994 Axiomatizing ST bisimulation equivalence *Proceedings of the IFIP TC2/WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi* 169-88
- [6] Castellano L, De Michelis G, Pomello L 1987 Concurrency vs. interleaving: An instructive example *Bull. Eur. Ass. Theoret. Comput. Sci.* **31** 12-5
- [7] Majster-Cederbaum M, Wu J 2003 Towards action refinement for true concurrent real time *Acta Informatica* **39** 1-47
- [8] Clarke E M, Grumberg O, Minea M, Peled D 1999 State space reduction using partial order techniques *STTT* **2**(3) 279-87
- [9] Darondeau P, Degano P 1993 Refinement of actions in event structures and casual trees *Theoretical Computer Science* **118** 21-48

6 Results and Discussion

The paper has demonstrated that (1) In event structures without independency between events, step trace equivalence and step bisimulation equivalence can preserve under action refinement; (2) In event structures, if all transitions are clustered action transitions, then with clustered trace equivalence between event structures, we can reach that step trace equivalence can preserve under action refinement; likewise, with clustered bisimulation equivalence between event structures, we can reach that step bisimulation equivalence can preserve under action refinement.

Therefore, we find a class of concurrent processes with specific properties, which enable step equivalence to preserve under action refinement in the absence of constraint.

Our next work is to introduce the thought of clustered action transition into model checking so as to deal with states explosion problem in the process of system verification.

Acknowledgements

This Work is supported by Grants No. HCIC201306 of Guangxi HCIC lab Open Fund, the National Natural Science Foundation of China under Grant No. 11371003, the Natural Science Foundation of Guangxi under Grant No. 2011GXNSFA018154 and No. 2012GXNSFGA060003, the Science and Technology Foundation of Guangxi under Grant No. 10169-1, the Scientific Research Project No. 201012MS274 from Guangxi Education Department., and Science Computing and Intelligent Information Processing of GuangXi higher education key laboratory No. GXSCIP201201.

- [10] Darondeau P, Degano P 1989 Casual trees *Automata, Languages and Programming, Lecture Notes in Computer Science* **372** 234-48
- [11] Jiang J, Wu J 2005 Symmetry and autobisimulation *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and technologies. IEEE Computer Society Press* 866-70
- [12] Jiang J, Wu J, Yan W 2005 Structural reductions in process algebra languages *Proceedings of the 11th Joint International Computer Conference. World Scientific Publishing Co.* 596-600
- [13] Degano P, Gorrieri R 1995 A causal operational semantics of action refinement *Information and Computation* **122** 97-119
- [14] van Glabbeek R J, Goltz U 1989 Equivalence notions for concurrent systems and refinement of actions *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science* **379** 237-48
- [15] van Glabbeek R J 1990 The linear time-branching time spectrum *CONCUR '90, Lecture Notes in Computer Science* **458** 297-8
- [16] van Glabbeek R J, Goltz U 1990 *A deadlock-sensitive congruence for action refinement* Institut fuer Informatik, Technische Universitaet Munchen:SFB-Bericht 342/23/90 A
- [17] van Glabbeek R J, Goltz U 2001 Refinement of actions and equivalence notions for concurrent systems *Acta Informatica* **37**(4/5) 229-327

- [18] Fecher H, Majster-Cederbaum M, Wu J 2002 Refinement of actions in a real-time process algebra with a true concurrency model *Electronic Notes in Theoretical Computer Science* **70**(3) 620-40
- [19] Gorrieri R, Rensink A 2001 *Action Refinement* //Bergstra J A, Ponse A and Smolka S A, editors Handbook of Process Algebra New York: Elsevier Science 1047-147
- [20] Huhn M 1996 Action refinement and property inheritance in systems of sequential agents *Concur'96, Lecture Notes in Computer Science* **1119** 639-54
- [21] Jategaonkar L, Meyer A R 1992 Testing equivalences for Petri nets with action refinement *Concur'92, Lecture Notes in Computer Science* **630** 17-31
- [22] Majster-Cederbaum M, Wu J 2001 Action refinement for true concurrent real time *Proc. ICECCS'01, IEEE Computer Society Press* 58-68
- [23] Majster-Cederbaum M, Wu J, Yue H 2006 Refinement of actions for real-time concurrent systems with causal ambiguity *Acta Informatica* **42**(6/7) 389-418
- [24] Majster-Cederbaum M, Wu J 2003 Adding action refinement to stochastic true concurrency models *ICFEM'03 Lecture Notes in Computer Science* **2885** 226-45
- [25] Alur R, Brayton R K, Henzinger T A, Qadeer S, Rajamani S K 1997 Partial-order reduction in symbolic state-space exploration *CAV'97, Lecture Notes in Computer Science* **1254** 340-51
- [26] Sun X, Zhang W, Wu J 2004 Event-based operational semantics and a consistency result for real-time concurrent processes with action refinement *Journal of Computer Science and Technology* **19**(6) 828-40
- [27] Winskel G 1989 *An Introduction to Event structures* Berlin: Springer, LNCS **354** 364-97
- [28] Wu J 2000 Logic programming-taking advantage of symmetry *Proc. Of ASCM'00, World Scientific Publ.* 100-9
- [29] Wu J, Fecher H 2004 Symmetric structure in logic programming *Journal of Computer Science and Technology* **19**(6) 803-11
- [30] Goltz U, Wehrheim H 1996 Modelling causality by dependency of actions in branching time semantics *Information Processing Letters* **59**(4) 179-84
- [31] Vogler W 1991 *Bisimulation and action refinement* STACS'91, Lecture Note in Computer Science **480** 309-21
- [32] Tang W, Wu J *Interleaving Semantics and Action Refinement in Event Structures* To be published
- [33] Vogler W 1992 Modular construction and partial order semantics of Petri nets *Lecture Note in Computer Science* **625** 625-48
- [34] Czaja I, van Glabbeek R J, Goltz U 1992 Interleaving semantics and action refinement with atomic choice *Advances in Petri Nets, Lecture Notes in Computer Science* **609** 89-107

Authors	
	<p>Weidong Tang</p> <p>Current position, grades: Associate professor University studies: Computer Software and Theory Scientific interest: Symbolic computation, formal verification</p>
	<p>Jinzhao Wu</p> <p>Current position, grades: Professor, Ph.D. University studies: Computer Software and Theory Scientific interest: Symbolic computation, automated reasoning, formal methods</p>
	<p>Meiling Liu</p> <p>Current position, grades: Teacher, Lecturer University studies: Computer Software and Theory Scientific interest: Data Mining, formal verification</p>