Preservation of Dynamic Behaviours in a Hierarchical Event Structure

Weidong Tang[1,2], Jinzhao Wu[1,2*], Meiling Liu[1, 3]

[1] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China.
[2] Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041, China.
[3] Science Computing and Intelligent Information Processing of Guang Xi Higher Education Key Laboratory, Nanning 530023, China.

* Corresponding author. Tel.: 008607713260599; email: himrwujz@126.com
Manuscript submitted March 25, 2016; accepted July 22, 2016.
doi: 10.17706/jsw.11.8.712-732

Abstract: Event structure is a method of modelling and verification for concurrent system, and action refinement is the core operation in event structure. This paper researches on what conditions action refinement must satisfy, such that some structural properties and dynamic properties of contour model, such as connectivity, liveness, fairness and regression, can also be preserved in its detailed model. In researching on the dynamic properties of event structure, it is difficult to apply event structure model to represent infinite events, this is because each action can be executed infinite times, thus it will produce infinite events. There has been little research on the problem. Therefore, we propose the concept of action structure model. The differences between the action structure model and event structure model include: (1)The events in event structure model are replaced by actions, and the same actions are merged into one action; (2) The inheritance of causality and the inheritance of conflict relation are no longer preserved; (3) When an action is executed even times, it will not appear in the configuration; and when an action is executed odd times, it will appear in the configuration. The use of action structure model can easily solve the difficult problem of representation of infinite events which encountered in the event structure model. This paper gives an example to demonstrate the application of action structure model.

Key words: Action structure model, action refinement, reachability, dynamic behaviour.

1. Introduction

As a semantic model of process algebra, event structure is a method of modelling and verification for concurrent system. Using event structure, it is convenient to describe the sequence, concurrency, conflict and synchronization of processes and components in system. Compared to labelled transition system model, event structure has the powerful advantage of appropriately describing the true concurrency.

The process algebra was proposed in 1980s, among which Hoare [1], Milner [2] and Park [3] are the most famous ones. The event structure as a semantic model of process algebra was proposed in Winskel [4]. When process algebra and event structure theories are explored, model checking is studied as well. model checking has attracted more attention in the North American academia while process algebra and event structure have gained wide supports in Europe.

As a system model, event structure can not only describe the structure of system, but also describe the

dynamic behaviors of system. Event structure has both intuitive graphical representation and mathematical methods to analyze system properties. For complex system modelling, hierarchical modelling and stepwise refinement methods can be used to construct the system model. In the first step, contour model is constructed. In the contour model, some subsystems can be represented by only one basic element (event). To further show the details of system. Sub-event structure systems modelling subsystems are used to replace basic element in contour model, this process is called action refinement. For complex system, we can use the hierarchical modeling method. What condition should action refinement meet, can some structural properties and dynamic properties (such as connectivity, liveness, fairness, regression) in contour model N be preserved in detailed model N'. The question is: what conditions should action refinement satisfy, can some structural properties and dynamic properties (such as connectivity, liveness, fairness, regression) in contour model N be preserved in detailed model N'?

2. Event Structure and Configuration

A basis element of a behaviour is called as an action. One occurrence of an action is called an event. Assume that Act is a set of actions .

Definition 2.1[5]-[7] A event structure Σ is a 5-tuple $(E,<,\#,\Diamond,l)$, which consists of

- E is the set of events;
- $<\subseteq E\times E$ is a irreflexive partial relation and satisfies the rule of finite causes such that $\forall e\in E,\{d\in E\,|\,d<e\}$ is finite; In addition, $d<e$ can be expressed as $e>d$.
- $\#\subseteq E\times E$ is irreflexive conflict relation, and satisfies the rule of inheriting of conflict such that $\forall e_1,e_2,e_3\in E: e_1<e_2\wedge e_1\#e_3\Rightarrow e_2\#e_3$;
- $\Diamond\subseteq E\times E$ is irreflexive independent relation and satisfies that $e_1\Diamond e_2\Leftrightarrow\neg(e_1=e_2\vee e_1<e_2\vee e_2<e_1\vee e_1\#e_2)$;
- $l:E\to Act$ is a label function of actions.

Definition 2.2 Let $\sum=(E,\ <,\#,\Diamond,\ l)$ be an event structure. let $R_1,R_2\subseteq E\times E$ .The composite operation between R_1 and R_2 is $R_1\circ R_2=\{(e_x,e_y)\,|\,\exists e_t((e_x,e_t)\in R_1\wedge(e_t,e_y)\in R_2)\}$. Reflexive closure, symmetric closure, transitive closure in R_1 is $r(R_1)=R_1\cup R_1^0$, $s(R_1)=R_1\cup R_1^{-1}$ and $t(R_1)=R_1\cup R_1^2\cup R_1^3\cup\cdots$, respectively, where $R_1^0=\{(e,e)\,|\,e\in E\}$, $R_1^{-1}=\{(e_x,e_y)\in E\times E\,|\,(e_y,e_x)\in R_1\}$, $R_1^{i+1}=R_1^i\cdot R_1,\ i\in\mathbb{N}$.

According to Definition 2.1 and Definition 2.2, we can easily get the following property of an event structure.

Theorem 3.1 Let $\sum=(E,\ <,\#,\Diamond,\ l)$ be an event structure, then

(1) $<=t(<)$; (2) $\#=rt(>)\circ s(\#)$; (3) $\Diamond=s(\Diamond)$.

By theorem 3.1, we can recursively compute to acquire three kinds of relations $<,\#\ and\ \Diamond$ in an event structure.

Definition 2.3 [5]-[7] Let $\sum=(E,\ <,\#,\Diamond,\ l)$ be an event structure, $X\subseteq E$.

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

(2) X is conflict-free iff $\sum|_X=(E|_X,\ <|_X,\#|_X,\Diamond|_X,\ l|_X)$ is conflict-free;

(3) X is called as a configuration if X is not only left closed but also conflict-free.

Here, let $C(\Sigma)$ represent the set of all configurations in event structure Σ.

A configuration X ($X \in C(\Sigma)$) is called successfully terminated configuration iff $\forall e \in E : e \notin X \Rightarrow \exists d \in X : d \# e$.

The event structure is also often represented with graph, where a circle repects an event, and solid line with arrows, dotted line and arrow lines connected with an arc stand for casual relation, immediately conflict relation, casual relation t that two or more conditions are simultaneously met, respectively. concurrent relation is not explicitly expressed.

Example 2.1 [5]-[7] The system $\mathcal{P} = (a \,|||\, b) + (c; b; d)$, $a, b, c, d \in Act$, executing either a, b Concurrently or c, b and d sequentially, can be discribed by the event structure Σ with events e_1, e_2, e_3, e_4, e_5 with $l(e_1) = a$, $l(e_3) = c$, $l(e_2) = l(e_4) = b$, $l(e_5) = d$,where $e_1 \Diamond e_2, e_3 < e_4 < e_5$, each of e_1, e_2 is in conflict with each of e_3, e_4, e_5. The graphical representation of the event structure is shown in Fig. 1.

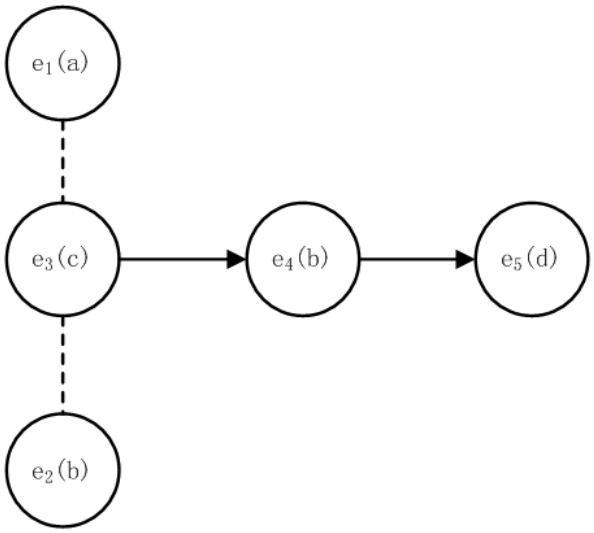

Fig. 1. The event structure model Σ.

Its configurations are $\varnothing, \{e_1\}, \{e_2\}, \{e_1, e_2\}, \{e_3\}, \{e_3, e_4\}, \{e_3, e_4, e_5\}$,where $\{e_1, e_2\}$, $\{e_3, e_4, e_5\}$ are terminated configurations.

3. Dynamic Properties [4]-[11] of Event Structure

In general, using event structure to construct the model for an actual system, it needs not only basic knowledge of event structure, but also the professional knowledge about the system domain. Only by understanding the relation between components (events and processes) in an actual system, and knowing how to describe the relation by using event structure, can we construct a proper event structure model for an actual system.

Reachability is the most basic dynamic property of event structure, and other properties can be defined by reachability.

Definition 3.1 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure. If there exists $t \in \text{Act}$,

X, $X' \in C(\sum)$, such that $X \xrightarrow{t} X'$, then X' is called to be immediately reachable from X. If there exist some action sequences $t_1, t_2, \cdots, t_k$ and configuration sequences $X_1, X_2, \cdots, X_k$ such that

$$X \xrightarrow{t_1} X_1 \xrightarrow{t_2} X_2 \cdots X_{k-1} \xrightarrow{t_k} X_k \tag{3.1}$$

Then X_k is called to be reachable from X. The set of all configurations reachable from X is denoted by $R(X)$. Assume $X \in R(X)$, if action sequence $t_1, t_2, \cdots, t_k$ is notated as σ, then formula (3.1) is also denoted by $X \xrightarrow{\sigma} X_k$.

When an actual system is modelled by event structure $\sum = (E, <, \#, \Diamond, l)$, the set of all possible states (i.e, configurations) appearing during the system's running can be represented by $R(\Sigma)$. The formal definition and the properties deduced of $R(\Sigma)$ are as follows.

Definition 3.2 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure and X_0 be an initial configuration. The set $R(\Sigma)$ of reachable configurations of $\sum$ is defined as the smallest set satisfying the following two conditions:

1) $X_0 \in R(\Sigma)$;

2) if $X \in R(\Sigma)$, and there exists $t \in Act$ such that $X[t > X'$, then $X' \in R(\Sigma)$.

Theorem 3.1 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, then

1) For any $X \in R(\Sigma)$, there exists $R(X) \subseteq R(\Sigma)$;

2) For any $X_1, X_2 \in R(\Sigma)$, $R(X_1) = R(X_2)$, if and only if $X_1 \in R(X_2)$ and $X_2 \in R(X_1)$.

Proof 1) Because $X \in R(\Sigma)$, then $\forall X' \in R(X)$: $X' \in R(\Sigma)$, therefore $R(X) \subseteq R(\Sigma)$.

3) $\Leftarrow$) Because $X_1 \in R(X_2)$, then $R(X_1) \subseteq R(X_2)$; Because $X_2 \in R(X_1)$, then $R(X_1) \subseteq R(X_2)$, therefore $R(X_1) = R(X_2)$.

$\Rightarrow$) Because $X_1 \in R(X_1)$, $R(X_1) = R(X_2)$, therefore $X_1 \in R(X_2)$; Similarly, $X_2 \in R(X_1)$.

Definition 3.3 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $t \in Act$. If for any $X \in R(\Sigma)$, there exists $X' \in R(X)$ such that $X'[t >$, then the action t is called to be live. If each $t \in Act$ is live, then $\sum$ is called to be a live event structure.

Example 3.1 Let the event structure of Figure 3.1(a) be $\sum 1 = (E, <, \#, \Diamond, l)$, where $E = \{e_i \mid i \in \mathbb{N}\}$, $< = t(\{(e_{6*m+1}, e_{6*m+2}), (e_{6*m+1}, e_{6*m+4}), (e_{6*m+2}, e_{6*m+3}), (e_{6*m+4}, e_{6*m+5}), (e_{6*m+3}, e_{6*(m+1)}), (e_{6*m+5}, e_{6*(m+1)}), (e_{6*(m+1)}, e_{6*(m+1)+1}) \mid m \in \mathbb{Z}_{\geq 0}\})$, $\# = \varnothing$, $\Diamond = \{(e_{6*m+2}, e_{6*m+4}), (e_{6*m+2}, e_{6*m+5}), (e_{6*m+3}, e_{6*m+4}), (e_{6*m+3}, e_{6*m+5}) \mid m \in \mathbb{Z}_{\geq 0}\}$, and $l(e_{6*m+k}) = t_k$, $m \in \mathbb{Z}_{\geq 0}, 1 \leq k \leq 6$, where $t(.)$ means the transitive closure in Definition 2.2. Analyzing execution of the event structure $\sum 1$, we can easily discover that at any configuration during the process of execution, each action will once again obtain the right of occurrence by means of executing a sequence of actions. This shows that each action is live, and the whole event structure also is live.

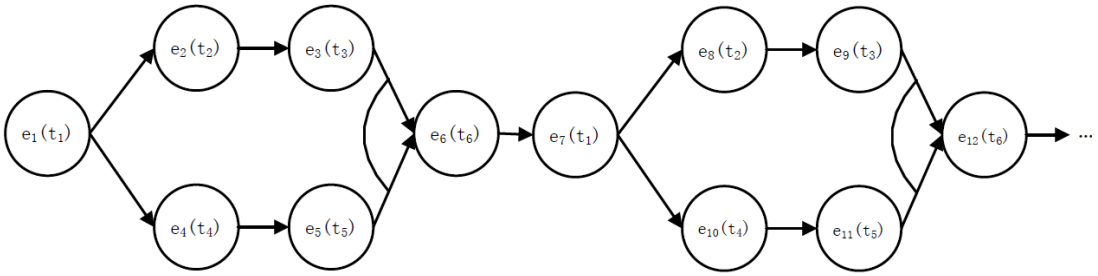

Fig 2(a). The event structure model $\Sigma 1$.

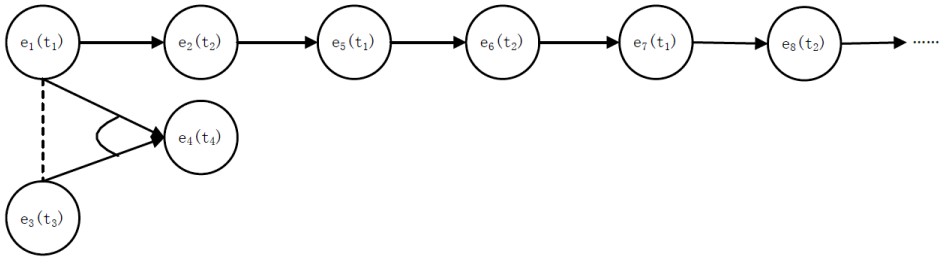

Fig 2(b). The event structure model $\Sigma 2$.

Let the event structure in Fig. 2(b) be $\sum 2 = (E, <, \#, \Diamond, l)$, where $E = \{e_i \mid i \in \mathbb{N}\}$, $<= \{(e_1, e_2), (e_1, e_4), (e_2, e_5), (e_3, e_4), (e_m, e_{m+1}) \mid m \in \mathbb{Z}_{\geq 5}\}$, $\# = \{(e_1, e_3)\}$, $\Diamond = \{(e_2, e_4), (e_4, e_m) \mid m \in \mathbb{Z}_{\geq 5}\}$, and $l(e_1) = l(e_{2*m+1}) = t_1, l(e_2) = l(e_{2*m+2}) = t_2, m \in \mathbb{Z}_{\geq 2}$, $l(e_3) = t_3, l(e_4) = t_4$. Considering the event structure $\sum 2$, the action t_1 and t_2 are live, but t_3, t_4 are not live, therefore the whole event structure is not live.

From Example 3.1, we can know that, if an action is live, then this action at any current configuration will be executed sometime, that is to say, this action can be executed infinitely. If an event structure is live, then every action in the event structure can be executed infinitely, thus it can produce infinite events. Therefore, it is difficult to represent infinite events by using event structure. In this paper, we propose the concept of action structure model.

The change between action structure model and event structure model are as follow:

1) The events in event structure model are replaced with the corresponding actions, and the same actions are merged into one action.
2) The inheritance of causality and inheritance of conflict relation in the original event structure are no longer be preserved.
3) When an action is executed even times, it will not appear in the configuration; and when an action is executed odd times, it will appear in the configuration.

Fig. 3 gives the corresponding action structure models of the event structure models $\sum 1$ and $\sum 2$ in Fig. 32, respectively.

The use of action structure model can easily solve the difficult problem of representation of infinite events which encountered in the event structure model.

Putting forward the liveness concept of event structure is derived from research on whether or not there exists the deadlock problem in the execution of the actual system. However, the liveness defined in Definition 3.3 is stronger than the deadlock-free property. The definition more similar to deadlock-free concept in the actual system is given in the following.

Definition 3.4 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure. If there exists $X \in R(\Sigma)$ such that

$\forall t \in Act : \neg X[t>$, then X is called to be a dead configuration of $\sum$. If there doesn't exist dead configuration in $\sum$, then $\sum$ is called to be non-dead or weak live.

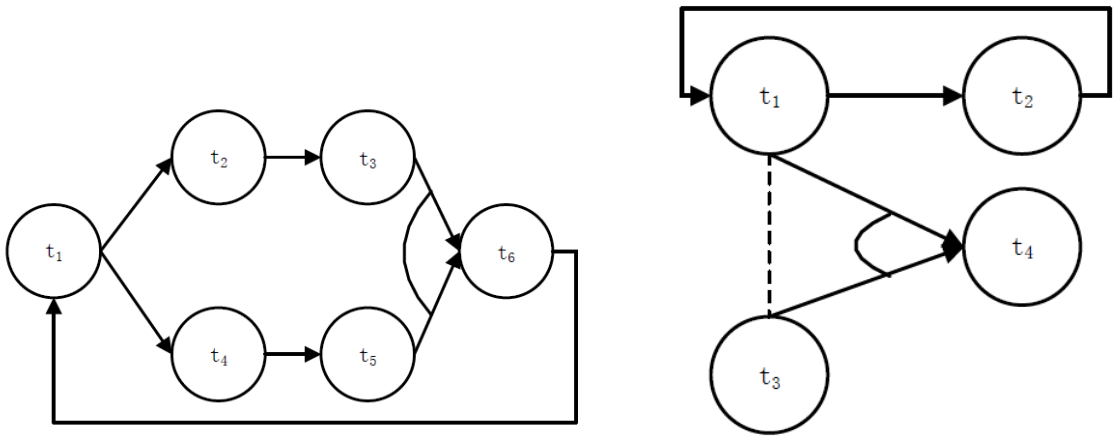

Fig 3(a). The corresponding action structure model of $\Sigma 1$. (b). The corresponding action structure model of $\Sigma 2$.

Theorem 3.2 If there exists an action which is live in the event structure $\sum = (E, <, \#, \Diamond, l)$, then $\sum$ is weak live.

Proof: It is proved by using reduction to absurdity.

Suppose that $\sum$ is not weak live, then there must exist a dead configuration $X \in R(\Sigma)$, that is $\forall t \in Act : \neg X[t>$, thus there doesn't exist $X' \in R(X)$ such that $X'[t>$. That is, any action is not live, this contradicts the premise condition.

The inverse proposition of Theorem 3.2 is not tenable. This will be illustrated in the following example.

Example 3.2 Let the event structure in Figure 3.3 be $\sum 3 = (E, <, \#, \Diamond, l)$, where $E = \{e_i \,|\, i \in \mathbb{N}\}$, $< = \{(e_{2m+1}, e_{2m+3}), (e_{2m+2}, e_{2m+4}) \,|\, m \in \mathbb{Z}_{\geq 0}\}$, $\# = \{(e_{2m+1}, e_{2n+2}) \,|\, m, n \in \mathbb{Z}_{\geq 0}\}$, $\Diamond = \varnothing$, and $l(e_1) = t_1$, $l(e_2) = t_4, l(e_{4*m+3}) = t_2, l(e_{4*m+4}) = t_5$, $l(e_{4*m+5}) = t_3, l(e_{4*m+6}) = t_6, m \in \mathbb{Z}_{\geq 0}$.

Fig. 4 gives a weak live event structure $\sum 3$, because in any reachable configuration of the system, there has at least one action which has the authority of occurrence.

However, there has no one action which is live in the system. In fact, in the initial configuration, if t_1 happens, then in the later reachable configuration, it is impossible for t_4, t_5 and t_6 to have the authority of occurrence. Conversely, in the initial configuration, if t_4 happens, then in the later reachable configuration, t_1, t_2 and t_3 will not gain the authority of occurrence.

From Theorem 3.2 and Example 3.2, we can see that the concept "there has one live action in an event structure" is stronger than "an event structure is weak live". To analyze in detail the possible situation of action execution in the system operation, we further discuss the classification of action liveness.

Definition 3.5 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $t \in \text{Act}$.

(1) If $\forall X \in R(\Sigma) : \neg X[t>$, then the action t is called to be dead.

(2) If $\exists X \in R(\Sigma) : X[t>$, then the action t is called to be one-level live.

(3) If for any integer n, $\exists \sigma \in Act^*, X \in R(\Sigma) : X[\sigma>$, and $\#(t/\sigma) \geq n$, then the action t is called to

be two-level live, where $\#(t/\sigma)$ represents the number of times that the action t appears in the action sequence σ.

(4) If there exists one infinitely long action transition sequence σ, such that $\varnothing[\sigma>$, and the action t appears infinite times in action sequence σ, then the action is called to be three-level live.

(5) If $\forall X \in R(\Sigma)$, and the action t is one-level live in the system $\sum$, then the action t is called to be four-level live in the system $\sum$.

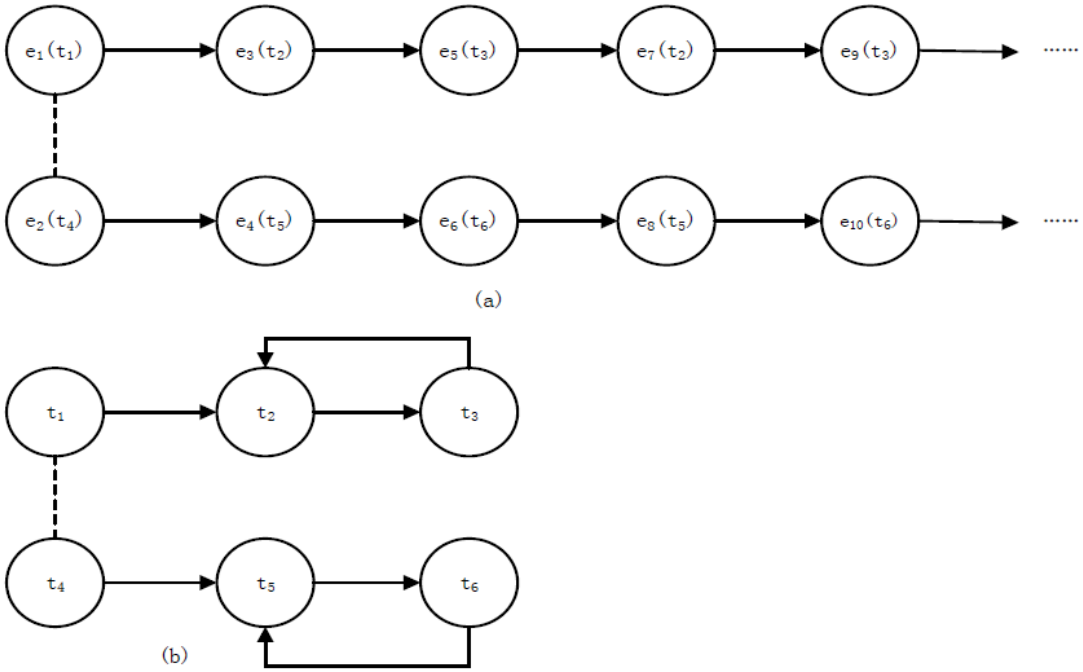

Fig. 4(a). The event structure model $\Sigma 3$ (b). The corresponding action structure model of $\Sigma 3$.

Definition 3.6 An event structure is live if and only if all of its action transition is four-level live.

Introducing the concept of fairness in the event structure, this aims at discussing the relationship between two transitions (transition groups). This relationship reflects the non-starvation problem occurred in the resource competition among each part of the simulated system.

Definition 3.7 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $t_1, t_2 \in \mathrm{Act}$. If there exists an integer k, such that $\forall X \in R(\Sigma)$ 和 $\forall \sigma \in Act^*: X[\sigma >$, there always has

$$\#(t_i/\sigma) = 0 \Rightarrow \#(t_j/\sigma) \le k, \quad i, j \in \{1, 2\} \text{ and } i \ne j \tag{3.2}$$

then t_1 and t_2 is called to be in fair relation.

If any two transitions in $\sum$ are both in fair relation, then $\sum$ is called to be a fair event structure.

We can easily know that, if there doesn't exist infinite transition sequences which are to occur, then for any $t_1, t_2 \in \mathrm{Act}$, there always exists the integer *k* in formula (3.2). Therefore, such event structure is always fair. When discussing the fairness of event structure, we aim generally at those event structures where there exists infinite running sequences.

Example 3.3 In the event structure $\Sigma 3$ shown in Fig. 4, the transition t_2 and t_5 are not in fair relation. Because there exists an infinite sequence $\sigma = t_1 t_2 t_3 t_2 t_3 \cdots$, $\Sigma[\sigma>$, such that for any integer *k*, there has

$\#(t_5/\sigma)=0$, but $\#(t_2/\sigma)>>k$. It can easily be verified that t_2 and t_3 are in fair relation.

Example 3.4 In the event structure $\sum 1$ shown in Fig. 2(a), any two transitions are both in fair relation. Therefore this is a fair event structure.

Theorem 3.3 The fair relation among transitions in event structure is an equivalence relation.

Proof The reflexivity and symmetry of fair relation is evident. The transitivity of fair relation is proved in the following.

Suppose that t_1 and t_2 are in fair relation, that is, there exists k_1, such that $\forall X\in R(\Sigma),\ \sigma\in Act^*:\ X[\sigma>$, then there has

$$\#(t_1/\sigma)=0\Rightarrow\#(t_2/\sigma)\le k_1,\qquad \#(t_2/\sigma)=0\Rightarrow\#(t_1/\sigma)\le k_1$$

Suppose that t_2 and t_3 are in fair relation, that is, there exists k_2, such that $\forall X\in R(\Sigma),\ \sigma\in Act^*:\ X[\sigma>$, then there has

$$\#(t_2/\sigma)=0\Rightarrow\#(t_3/\sigma)\le k_2,\qquad \#(t_3/\sigma)=0\Rightarrow\#(t_2/\sigma)\le k_2$$

Taking the value of k, let $k=\max\{(k_1+1)k_2,(k_2+1)k_1\}$, then $\forall X\in R(\Sigma),\ \sigma\in Act^*:\ X[\sigma>$. If $\#(t_1/\sigma)=0$, because $\#(t_2/\sigma)\le k_1$, then σ can be written as:

$$\sigma=\sigma_0t_2\sigma_1t_2\sigma_2\cdots\sigma_{j-1}t_2\sigma_j,\ \ j\le k_1$$
$$\#(t_2/\sigma_i)=0,\ \ i=0,\ 1,\ \cdots,\ j$$

Because $\#(t_2/\sigma_i)=0$, and we can know $\#(t_3/\sigma_i)\le k_2$ from the fair relation of t_2 and t_3, therefore,

$$\#(t_3/\sigma)\le k_2(j+1)\le k_2(k_1+1)\le k$$

Similarly, if $\#(t_3/\sigma)=0$, then it can deduce

$$\#(t_1/\sigma)\le k_1(k_2+1)\le k$$

This shows that t_1 and t_3 are also in fair relation, thus it is proved that fairness system has the property of transitivity.

Sometimes, we require that two group of transitions (rather than two transitions) are in fair relation. In order to describe this phenomenon, we introduce the fair relation between transition groups in the event structure and the concept of group fairness event structure in the following.

Definition 3.8 Let $\sum=(E,\ <,\#,\ \Diamond,\ l)$ be an event structure, $T_1,\ T_2\subseteq \mathrm{Act}$ and $T_1\bigcap T_2=\varnothing$. If there exists an integer k, such that $\forall X\in R(\Sigma)$ and $\forall\sigma\in Act^*:X[\sigma>$, there always has

$$\sum_{t\in T_i}\#(t/\sigma)=0\Rightarrow\sum_{t\in T_j}\#(t/\sigma)\le k,\ \ i,j\in\{1,\ 2\}\text{ and }i\ne j \tag{3.3}$$

Then the transition group T_1 and T_2 are in fair relation in Σ.

Definition 3.9 Let $\sum=(E,\ <,\#,\ \Diamond,\ l)$ be an event structure. If there exists transition subsets $T_1,\ T_2,\ \cdots,\ T_q$, such that

1) $T_1\bigcup T_2\bigcup\cdots\bigcup T_q=Act$;

2) $\forall i, j \in \{1, 2, \cdots, q\}, i \neq j \Rightarrow T_i \bigcap T_j = \varnothing$;

3) $\forall i, j \in \{1, 2, \cdots, q\}, i \neq j \Longrightarrow T_i$ and T_j are in fair relation in Σ.

Then Σ is called to be a fair grouped event structure with respect to the partition $\{T_1, T_2, \cdots, T_q\}$.

Example 3.5 In the event structure $\Sigma 1$ shown in Fig. 2(a), $\Sigma 1$ is a fair grouped event structure with respect to the partition $Act / f = \{T_1, T_2, T_3, T_4\}$, where $T_1 = \{t_1\}, T_2 = \{t_2, t_3\}, T_3 = \{t_4, t_5\}, T_4 = \{t_6\}$.

Definition 3.10 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure and X_0 be an initial configuration. The event structure Σ is regressive if and only if $\forall X \in R(\Sigma), \exists \sigma \in \text{Act}^*: X[\sigma > X_0$.

Example 3.6 The event structure $\Sigma 1$ shown in Fig. 2(a) is regressive. However, the event structure $\Sigma 3$ shown in Fig. 4 is not regressive, because there doesn't exist σ which contains t_1, t_2, t_3, such that $X_0[\sigma > X_0$.

One of the outstanding advantages of event structure is that they are convenient to describe concurrence and conflict relation.

Definition 3.11 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $e_1, e_2 \in E$, X is one configuration of Σ. If

1) $X[e_1 >$ but $\neg X[e_2 >$;

2) $X[e_1 > X_1 \rightarrow X_1[e2 >$.

Then the event e1 and e2 has sequential relation.

Definition 3.12 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $e_1, e_2 \in E$. If $\exists X \in C(\sum)$ such that $X[e_1 >$ and $X[e_2 >$, then

1) If $X[e_1 > X_1 \rightarrow X_1[e2 >$ and $X[e_2 > X_2 \rightarrow X_2[e_1 >$, then e_1 and e_2 is called to be concurrent in X, denoted by $X[\{e_1, e2\} >$.

2) If $X[e_1 > X_1 \rightarrow \neg X_1[e2 >$ and $X[e_2 > X_2 \rightarrow \neg X_2[e_1 >$, then e_1 and e_2 is called to be conflict in X.

Definition 3.13 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $li \subseteq E$. If

$$\forall x, y \in li: x < y \ \vee \ y < x$$

Then li is called to be a line set of Σ. If l is a line set of Σ, and any $li' \supset li$ is not a line set of Σ, then li is called to be a line of Σ.

Definition 3.14 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $u \subseteq E$. If

$$\forall x, y \in u: (x, y) \notin < \ \wedge \ (y, x) \notin <$$

Then u is called to be a cut set of Σ. If u is a cut set of Σ, and any $u' \supset u$ is not a cut set of Σ then u is called to be a cut of Σ.

Suppose u_1, u_2 are two cuts of Σ. If $\forall x \in u_1, \exists y \in u_2: (x, y) \in <$, then expressed as $u_1 < u_2$.

Theorem 3.4 Let $\sum = (E, <, \#, \Diamond, l)$ be an event structure, $e_1, e_2 \in E$. The events e_1 and e_2 has causality if and only if there exists a line li of Σ, such that $e_1 \in li, e_2 \in li$; The events e_1 and e_2 has

concurrence relation if and only if there eixists a cut u of Σ, such that $e_1 \in u,\ e_2 \in u$.

4. Behavior Expression [12]-[14]

Definition 4.1 Let Act be a finite character set, Act^* be the set of strings in Act, P(Act*) which is the power set of Act* be called to be step set, P(Act*)* be the set of step string in P(Act*). Then for any $L \subseteq P(Act^*)^*$, L is called to be a concurrent language in Act. For $\alpha \in P(Act^*)$, if there is only a string in α, then α is called to be a single step in Act, if there are several strings in α, then α is called to be a concurrent step in Act, for $\alpha = \varnothing$, it is called to be an empty step.

Example 4.1 Let $\Sigma = \{a,\ b,\ c\}$. $\alpha = \{a\}\{b\}\{c\}$ is a single step, denoted by $\alpha = abc$; $\beta = \{ab,\ c\}$ is a concurrent step, denoted by $\beta = \begin{pmatrix} ab \\ c \end{pmatrix}$. If $L = \left\{ abb\begin{pmatrix} a \\ c \end{pmatrix},\ abc\begin{pmatrix} ab \\ c \end{pmatrix},\ a \right\}$, then L is a concurrent language in Σ where for a step string $abb\begin{pmatrix} a \\ c \end{pmatrix}$, a, b, b are its three single steps and $\begin{pmatrix} a \\ c \end{pmatrix}$ is its a concurrent step. In fact, for a step string $abb\begin{pmatrix} a \\ c \end{pmatrix}$, abb is also its one single step, or ab, b are its two single step, because it is no essential difference how to divide a step string into single steps. For the sake of convenience, in the following a single step is thought as a character.

Definition 4.2 Let L1 and L2 be two concurrent languages in Σ. Linguistic operators are defined in the following:

(1) join operator "$\circ$": $L_1 \circ L_2 = \{\alpha_1 \circ \alpha_2 \mid \alpha_1 \in L_1 \wedge \alpha_2 \in L_2\}$;

(2) alternative operator "+": $L_1 + L_2 = \{\alpha \mid \alpha \in L_1 \vee \alpha \in L_2\}$;

(3) concurrent operator "$\Diamond$": $L_1 \Diamond L_2 = \left\{ \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \middle| \alpha_1 \in L_1 \wedge \alpha_2 \in L_2 \right\}$;

(4) closure operator "*": $L_1^* = \bigcup_{i=0}^{\infty} L_1^{\,i}$, where $L_1^{\,0} = \{\varnothing\},\ L_1^{\,i} = L_1 \circ L_1^{\,i-1},\ i \geq 1$;

Example 4.2 Let $\alpha_1 = \begin{pmatrix} ab \\ c \end{pmatrix},\ \alpha_2 = \begin{pmatrix} d \\ e \end{pmatrix}$, then $\alpha_1 \Diamond \alpha_2 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} ab \\ c \end{pmatrix} \\ \begin{pmatrix} d \\ e \end{pmatrix} \end{bmatrix} = \begin{bmatrix} ab \\ c \\ d \\ e \end{bmatrix}$.

Definition 4.3 Let Σ be a finite character set, the behavior expression in Σ and the sets it stands for be defined recursively in the following:

(1) X is a behavior expression, it represents an empty set;

(2) $\varnothing$ is a behavior expression, it represents the set $\{\varnothing\}$;

(3) For each single step serial α in $P(\Sigma^*)^*$, α is a behavior expression, it represents the set $\{\alpha\}$;

(4) For two single step serials α_1, α_2 in $P(\Sigma^*)^*$, the behavior expression of concurrent steps $\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$ composed of the two single steps serials is $\alpha_1 \Diamond \alpha_2$, it represents the set $L(\alpha_1) \Diamond L(\alpha_2)$;

(5) If α, β respectively represents the behavior expressions of the concurrent language $L(\alpha)$ and $L(\beta)$,then $\alpha \circ \beta$, $\alpha + \beta$, $\alpha \lozenge \beta$ and α^* respectively represents the behavior expression of the set $L(\alpha) \circ L(\beta)$, $L(\alpha) + L(\beta)$, $L(\alpha) \lozenge L(\beta)$ and $L(\alpha)^*$.

The following properties of behavior expression can easily be verified.

Lemma 4.1 Let α, β, γ be behavior expression, then

(1) $\alpha \circ (\beta \circ \gamma) = (\alpha \circ \beta) \circ \gamma$, $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$, $\alpha \lozenge (\beta \lozenge \gamma) = (\alpha \lozenge \beta) \lozenge \gamma$;

(2) $\alpha + \beta = \beta + \alpha$, $\alpha \lozenge \beta = \beta \lozenge \alpha$;

(3) $\alpha \circ (\beta + \gamma) = \alpha \circ \beta + \alpha \circ \gamma$, $(\alpha + \beta) \circ \gamma = \alpha \circ \gamma + \beta \circ \gamma$;

(4) $(\alpha^*)^* = \alpha^*$, $(\alpha^* + \beta^*)^* = (\alpha^* \circ \beta^*)^* = (\alpha + \beta)^*$.

Definition 4.4 Let γ be the behavior expression in Σ, then the language $L(\gamma)$ is called to be a behavior expression language in Σ.

Lemma 4.2 The behavior expression language in Σ is closed to join operator, alternative operator, concurrent operator, and closure operator.

Proof The conclusion is proved to be correct according to Definition 4.3.

5. Action Refinement

It is well known that the design and analysis of complicated system is difficult, action refinement[14]-[18] is an effective way to solve the problem. In this paper, we study the refinement operation of event structure on the goal of the behavior of event structure; on this premise, we discuss the preservation of connectivity, liveness, fairness, regression; and we propose a behavior-oriented method of hierarchical modeling of concurrent system[12], [14].

Definition 5.1 Let $\sum = (E, <, \#, \lozenge, l)$ and $\sum_s = (E_s, <_s, \#_s, \lozenge_s, l_s)$ be event structure, then after performing the refinement operation $ref(\bar{t}, \Sigma_s)$, the event structure Σ_{ref} obtained is shown in the following:

(1) $E_{ref} = E \cup Es - \{\bar{e} \mid l(\bar{e}) = \bar{t}\}$;

(2) $<_{ref} = < \cup <_s \cup \{(e, e_s) \mid e < \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\} \cup \{(e_s, e) \mid \bar{e} < e,\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\}$
$- \{(e, \bar{e}) \mid e < \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E\} - \{(\bar{e}, e) \mid \bar{e} < e,\ l(\bar{e}) = \bar{t},\ e \in E\}$;

(3) $\#_{ref} = \# \cup \#_s \cup \{(e, e_s) \mid e \# \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\} \cup \{(e_s, e) \mid \bar{e} \# e,\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\}$
$- \{(e, \bar{e}) \mid e \# \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E\} - \{(\bar{e}, e) \mid \bar{e} \# e,\ l(\bar{e}) = \bar{t},\ e \in E\}$;

(4) $\lozenge_{ref} = \lozenge \cup \lozenge_s \cup \{(e, e_s) \mid e \lozenge \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\} \cup \{(e_s, e) \mid \bar{e} \lozenge e,\ l(\bar{e}) = \bar{t},\ e \in E,\ e_s \in E_s\}$
$- \{(e, \bar{e}) \mid e \lozenge \bar{e},\ l(\bar{e}) = \bar{t},\ e \in E\} - \{(\bar{e}, e) \mid \bar{e} \lozenge e,\ l(\bar{e}) = \bar{t},\ e \in E\}$;

(5) $l_{ref} = l \cup l_s - l|_{t=\bar{t}}$

Fig. 5 represents the idea of $ref(\bar{t}, \Sigma_s)$ [14].

Example 5.1 Continue with Example 2.1. Assuming that $ref(b) = (b_1 < b_2) \# b_3$, the event structure after action refinement is expressed as Fig. 6. In the process of action refinement, each event e labeled by b is replaced by a disjoint copy, Σ_e, of $ref(b)$, i.e. the event e2 is replaced by $(e_{22} < e_{23}) \# e_{21}$ and the event e4

is replaced by $(e_{41}<e_{42})\#e_{43}$. The causality and conflict structure is inherited from Σ: all events which were casually before e will be casually before all events of Σ_e, every event which casually followed e will casually follow all events of Σ_e, and all events in conflict with e will be in conflict with all the events of Σ_e [6], [7], [14].

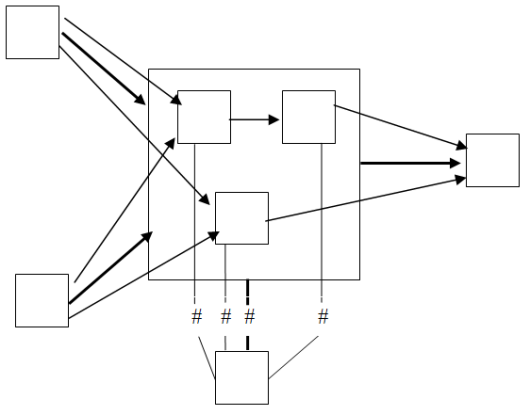

Fig. 5. Action refinement model.

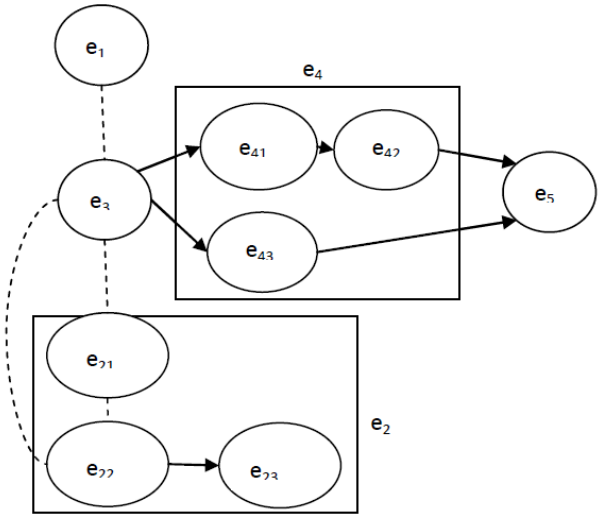

Fig. 6. The event structure after action refinement.

Theorem 5.1 Let $\Sigma=(E, <, \#, \Diamond, l)$ and $\Sigma_s=(E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structures. Σ_{ref} is the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ, then there exists a map $f: L(\Sigma) \to L(\Sigma_{ref})$, such that

$$f(\gamma)=\begin{cases}\gamma, & \text{if } \bar{t} \in Act-\|\gamma\| \\ \gamma', & \text{if } \bar{t} \in \|\gamma\|\end{cases},$$

where γ is the step sequence of Σ, $\|\gamma\|$ is the set of action transitions in γ, γ' is the new step sequences after refining the action $\bar{t}$ in γ into Σ_s.

Proof $\forall \gamma \in L(\Sigma)$, denoting $\gamma=(\alpha_1, \alpha_2, \cdots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \cdots, \alpha_r)$, and

$$X_0[\alpha_1 > X_1[\alpha_2 > \cdots X_{i-2}[\alpha_{i-1} > X_{i-1}[\alpha_i > X_i[\alpha_{i+1} > X_{i+1} \cdots X_{r-1}[\alpha_r > X_r$$

(1) if $\bar{t} \notin \| \gamma \|$, then $f(\gamma) = \gamma \in L(\Sigma_{ref})$;

(2) if $\bar{t} \in \alpha_i \subseteq \| \gamma \|$, then $X_0[\alpha_1\alpha_2 \cdots \alpha_{i-1} > X_{i-1}[\alpha_i > X_i[\alpha_{i+1} \cdots \alpha_r > X_r$ is correct.

From the definition of $ref(\bar{t}, \Sigma_s)$, we can obviously see that

$X_0 \cdots X_{i-1}[\alpha_i \bigcup L(\Sigma_s) - \{\bar{t}\} > X'[\alpha_{i+1} > X_{i+1} \cdots X_r$ and $\exists \sigma_s \in L(\Sigma_s)$.

Suppose that $\alpha_i' = \alpha_i \bigcup \sigma_s - \{\bar{t}\}$, then the occurrence sequences of Σ_{ref} is

$X_0[\alpha_1 > X_1[\alpha_2 > \cdots X_{i-2}[\alpha_{i-1} > X_{i-1}[\alpha_i' > X_i'[\alpha_{i+1} > X_{i+1} \cdots X_{r-1}[\alpha_r > X_r$,

its corresponding action is

$$\begin{aligned} \gamma' &= \alpha_1 \circ \alpha_2 \circ \cdots \circ \alpha_{i-1} \circ \alpha_i' \circ \alpha_{i+1} \circ \cdots \circ \alpha_r \\ &= \alpha_1 \circ \alpha_2 \circ \cdots \circ \alpha_{i-1} \circ (\alpha_i \bigcup \sigma_s - \{\bar{t}\}) \circ \alpha_{i+1} \circ \cdots \circ \alpha_r \\ &= \alpha_1 \circ \alpha_2 \circ \cdots \circ \alpha_{i-1} \circ \begin{pmatrix} \alpha_i - \{\bar{t}\} \\ \sigma_s \end{pmatrix} \circ \alpha_{i+1} \circ \cdots \circ \alpha_r \in L(\Sigma_{ref}) \end{aligned}$$

According to (1) (2), theorem 5.1 is proved.

Theorem 5.1 shows the behavior relation between before and after performing refinement operation $ref(\bar{t}, \Sigma_s)$.

Theorem 5.2 Let $\sum = (E, <, \#, \Diamond, l)$ and $\sum_s = (E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structure, Σ_{ref} is the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ. If Σ and Σ_s are both live, then Σ_{ref} is also live.

Proof $\forall t' \in Act_{ref}$, $\forall X' \in R(\Sigma_{ref})$, then $t' \in Act - \{\bar{t}\}$ or $t' \in Act_s$.

Four conditions are discussed in the following:

(1) If $t' \in Act - \{\bar{t}\}$,

According to the liveness of Σ, for $X \in R(X_0)$, $\exists \bar{X} \in R(X)$, such that $\bar{X}[t' >$,

denoting $X_0[\beta_0 > X[\beta > \bar{X}[t' >$.

(1.1) if $\bar{t} \notin \| \beta_0 \| \cup \| \beta \|$, according to the definition of refinement operation, then

$$X_0' = [X_0, \varnothing],\ X' = [X, \varnothing],\ \bar{X}' = [\bar{X}, \varnothing],$$

Therefore, $X_0'[\beta_0 > X'[\beta > \bar{X}'[t' >$, that is, t' is live in Σ_{ref}.

(1.2) if $\bar{t} \in \| \beta_0 \| \cup \| \beta \|$, suppose that if $\bar{t} \in \| \beta \|$, and denoted by

$$X_0[\beta_0 > X[\beta_1 \bar{t} \beta_2 > \bar{X}[t' >,$$

According to the refinement operation and the liveness of Σ_s, then $X_0'[\beta_0 > X'[\beta_1 \sigma_s \beta_2 > \bar{X}'[t' >$,

where, $X_0' = [X_0, \varnothing],\ X' = [X, \varnothing],\ \bar{X}' = [\bar{X}, X_{1s}],$ σ_s is the step serials in Σ_s.

Therefore, t' is still live in Σ_{ref}.

(2)if $t' \in Act_s$,

according to $X_0'[\beta > X'$, that is $[X_0, \varnothing][\beta > [X_2, X_s]$, denoting $X_{0s}[\sigma_{0s} > X_s$.

(2.1) if $t' \notin \| \sigma_{0s} \|$, then according to the liveness of Σ_s, we can know $\exists \bar{X}_s \in R(X_s)$, such that

$$X_{0s}[\sigma_{0s} > X_s[\sigma_s > \bar{X}_s[t' >,$$

Considering the liveness of Σ, therefore,

$[X_0, \varnothing][\beta_0 > [X_2, X_{0s}][\sigma_{0s} > [X_2, X_s][\sigma_s > [X_2, \bar{X}_s][t' >$

Let $\bar{X}' = [X_2, \bar{X}_s]$, then

$$X_0'[\beta_0\sigma_{0s} > X'[\sigma_s > \bar{X}'[t' >,$$

Therefore, t' is live in Σ_{ref}.

(2.2) if $t' \in \| \sigma_{0s} \|$,then$[X_0, \varnothing][\beta_0 > [X_1, X_{0s}][\sigma_{0s} > [X_1, X_{1s}][\beta_1 > [X_2, X_{1s}]$,

According to the liveness of Σ, for X_2, $\exists X_3$, such that $X_2[\beta_2 > X_3,\ X_3[\bar{t} >$,

there has

$$[X_2, X_{1s}][\beta_2 > [X_3, X_{1s}],\ [X_3, X_{1s}][\bar{t} >.$$

According to the liveness of Σ_s, for X_{1s}, $\exists X_{2s}$, such that $X_{1s}[\sigma_s > X_{2s},\ X_{2s}[t' >$,

there has

$$[X_3, X_{1s}][\sigma_s > [X_3, X_{2s}],\ [X_3, X_{2s}][t' >,$$

where, σ_s is the prefix substring of σ_{0s}. That is,

$[X_0, \varnothing][\beta_0 > [X_1, X_{0s}][\sigma_{0s} > [X_1, X_{1s}][\beta_1 > [X_2, X_{1s}][\beta_2 > [X_3, X_{1s}][\sigma_s > [X_3, X_{2s}][t' >$,

that is

$$X_0'[\beta_0\sigma_{0s}\beta_1 > X'[\beta_2\sigma_s > \bar{X}'[t' >,$$

Therefore, t' is still live in Σ_{ref}.

According to (1) and (2), then t' is live in Σ_{ref}. According to the arbitrariness of t', then Σ_{ref} is live.

Theorem 5.3 Let $\sum = (E, <, \#, \Diamond, l)$ and $\sum_s = (E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structures, Σ_{ref} be the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ. If $X_0[\bar{t} >$, then Σ_{ref} is live if and only if Σ and Σ_s are both live.

Proof $\Leftarrow$) It is proved according to Theorem 5.2.

$\Rightarrow$) $X_0'[\bar{t} >$ is given.

(1) for $\forall t' \in Act_s,\ \forall X_s \in R(X_{0s}),\ X' = [X_0, X_s]$,

According to the liveness of Σ_{ref}, then $\exists\ \bar{X}' = [X_0, \bar{X}_s] \in R(X')$, such that

$[X_0, X_{0s}][\sigma 1 > [X_0, X_s][\sigma 2 > [X_0, \bar{X}_s][t' >$, where σ_1, σ_2 is the step serials of Σ_s.

That is, $X_{0s}[\sigma 1 > X_s[\sigma 2 > \bar{X}_s][t' >$, Therefore, t' is live in Σ_s.

Hence, according to the arbitrariness of t', we can know Σ_s is live.

(2) for $\forall t' \in Act,\ \forall X' = [X, X_s] \in R(X_0')$,

According to the liveness of Σ_{ref}, then $\exists \bar{X}' = [\bar{X}, \bar{X}_s] \in R(X')$, such that

$$X_0'[\sigma_1\beta_1\sigma_2 > X'[\sigma_3\beta_2\sigma_4 > \bar{X}'[t' >,$$

where $\sigma_i,\ i = 1,2,3,4$ is the step serials of Σ, $\beta_i, i = 1,2$ is the step serials of Σ_s.

That is, $[X_0, X_{0s}][\sigma_1\beta_1\sigma_2 > [X, X_s][\sigma_3\beta_2\sigma_4 > [\bar{X}, \bar{X}_s][t' >$,

According to the liveness and refinement operation of Σ_s, there has

$$X_0[\sigma_1\bar{t}\sigma_2 > X[\sigma_3\bar{t}\sigma_4 > \bar{X}[t' >,$$

Therefore, t' is live in Σ. Hence, according to the arbitrariness of X and t′, we can know Σ is live.

Theorem 5.4 Let $\sum = (E, <, \#, \Diamond, l)$ and $\sum_s = (E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structures, Σ_{ref} be the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ, then Σ_{ref} is fair if and only if Σ and Σ_s are both fair.

Proof $\Rightarrow$) is proved by using reduction to absurdity.

(1) if Σ is not fair, then supposing that $t_1,\ t_2 \in Act$, t1 and t2 are in unfair relation.

(1.1) if $t_1 = \bar{t}$, then $t_2 \in Act - \{\bar{t}\}$, according to the definition of refinement operation, we can infer that $\forall t_s \in Act_s$ and t2 are in unfair relation. Therefore, Σ_{ref} is not fair, this is contradictory.

(1.2) if $t_1,\ t_2 \in Act - \{\bar{t}\} \subseteq Act_{ref}$, then $t1$ and $t2$ are in unfair relation. Therefore Σ_{ref} is unfair, this is contradictory.

(2) if Σ_s is unfair, then supposing that $t_1,\ t_2 \in Act_s$, $t1$ and $t2$ are in unfair relation. According to the assumption, we can easily know that Σ_{ref} is unfair, this is contradictory.

All the analysis shows that Σ and Σ_s are both fair, the necessity is proved.

$\Leftarrow$) is also proved by using reduction to absurdity.

If Σ_{ref} is unfair, then supposing that $t_1,\ t_2 \in Act_{ref}$, $t1$ and $t2$ are in unfair relation.

(1) if $t_1,\ t_2 \in Act - \{\bar{t}\} \subseteq Act_{ref}$, then t1 and t2 are in unfair relation. Therefore Σ is unfair, , this is contradictory.

(2) if $t_1,\ t_2 \in Act_s \subseteq Act_{ref}$, then $t1$ and $t2$ are in unfair relation. According to the assumption, we can easily know that Σ_s is unfair, this is contradictory.

(3) if $t_1 \in Act_s$, then $t_2 \in Act - \{\bar{t}\}$, according to the definition of refinement operation, we can infer that $\bar{t}$ and t2 are in unfair relation. Therefore, Σ is not fair, this is contradictory.

All the analysis shows that Σ_{ref} is fair, the sufficiency is proved.

Similar to the proof of Theorem 5.2 and Theorem 5.3, we can obtain the following conclusion:

Theorem 5.5 Let $\sum = (E, <, \#, \Diamond, l)$ and $\sum_s = (E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structures, Σ_{ref} be the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ. If Σ and Σ_s are both

regression, then Σ_{ref} is also regression.

Theorem 5.6 Let $\sum=(E, <, \#, \Diamond, l)$ and $\sum_s=(E_s, <_s, \#_s, \Diamond_s, l_s)$ be event structures, Σ' be the event structure after performing the refinement operation $ref(\bar{t}, \Sigma_s)$ on Σ. If $X_0[\bar{t}>$, then Σ' is regressive if and only if Σ and Σ_s are also regressive.

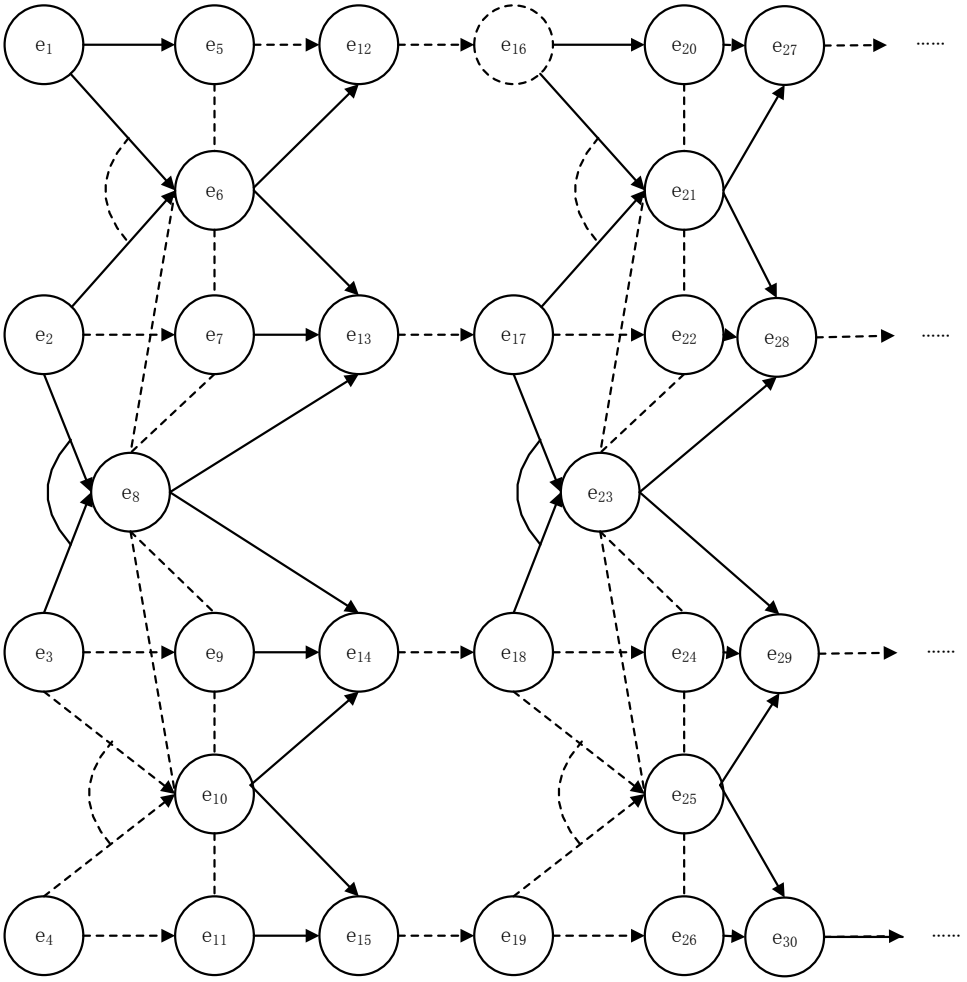

Fig. 7. The event structure model in first layer.

6. Application: System Modeling of Robotic Hand with Grasping Objects

We suppose that there are four robotic hands R1, R2, R3 and R4, three types of objects including type A, type B and type C. All robotic hands can work independently when moving small objects of type A. Either the group named R12 with R1 and R2 or the other group named R34 with R3 and R4 must work in cooperation and coordination when moving big objects of type B. Only the group named R23 with R2 and R3 is able to complete the task when moving big objects of type C.

A set of all actions Σ is expressed as

$$\Sigma=\{a_i, b_i, c_i, d_i, r_i, f_{1,2}, g_{1,2}, f_{3,4}, g_{3,4}, h, k\}, \ i=1,2,3,4;$$

where, a_i, b_i, c_i, d_i and r_i represent that R_i arrives at the position of objects, R_i is grasping objects, R_i moves a small object of type A, R_i puts down a small object of type A and R_i returns back to the

original position, respectively; $f_{1,2}$ and $g_{1,2}$ represent that R_1 and R_2 move a big object of type B together, R_1 and R_2 put down a big object of type B together, respectively; $f_{3,4}$ and $g_{3,4}$ represent that R_3 and R_4 move a big object of type B together, R_3 and R_4 put down a big object of type B together, respectively; h and k represent that R_2 and R_3 move a big object of type C together, R_2 and R_3 put down a big object of type C together, respectively.

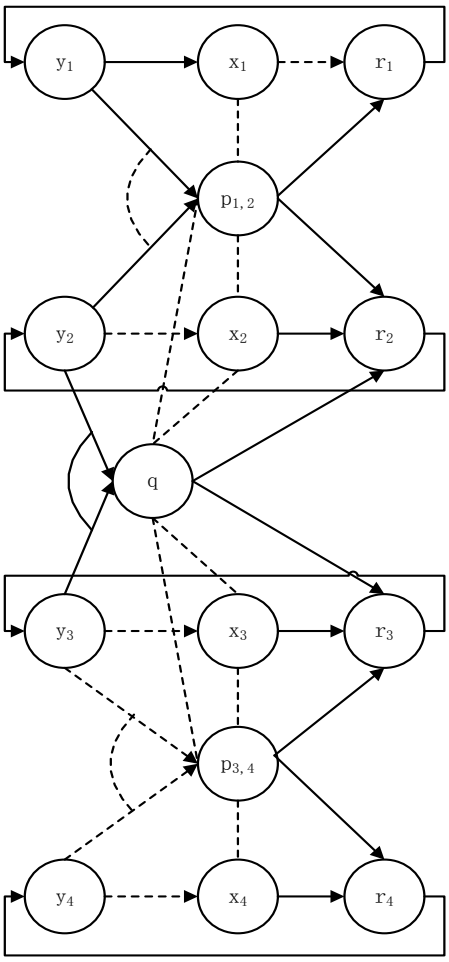

Fig. 8. The action structure model corresponding to the event structure model in first layer.

When this system was considered, the actions in the first layer can be concluded

$$\begin{pmatrix} (((y_1 \circ x_1 \circ r_1)^* \Diamond (y_2 \circ x_2 \circ r_2)^*) + ((y_1 \Diamond y_2) \circ p_{1,2} \circ (r_1 \Diamond r_2))) \Diamond \\ (((y_3 \circ x_3 \circ r_3)^* \Diamond (y_4 \circ x_4 \circ r_4)^*) + ((y_3 \Diamond y_4) \circ p_{3,4} \circ (r_3 \Diamond r_4))) + \\ (((y_1 \circ x_1 \circ r_1)^* \Diamond (y_4 \circ x_4 \circ r_4)^*) \Diamond ((y_2 \Diamond y_3) \circ q \circ (r_2 \Diamond r_3))) \end{pmatrix}^*,$$

where,

x_i represents that R_i is operating on a small object of type A;

y_i represents the preparation work before R_i moves;

$p_{i,j}$ represents that R_i and R_j are operating on a big object of type B together;

q represents that R_2 and R_3 are operating on a big object of type C together.

Therefore, the event structure model in the first layer and its corresponding action structure model are constructed, which are shown in Fig. 7 and Fig. 8 respectively. From comparison of the two graphs, we can see that the action structure model has obvious advantages.

In the second layer, the action of x_i, y_i, $p_{i,j}$ and q can be refined as $c_i \circ d_i$, $a_i \circ b_i$, $f_{i,j} \circ g_{i,j}$ and $h \circ k$, which is shown in Fig. 9.

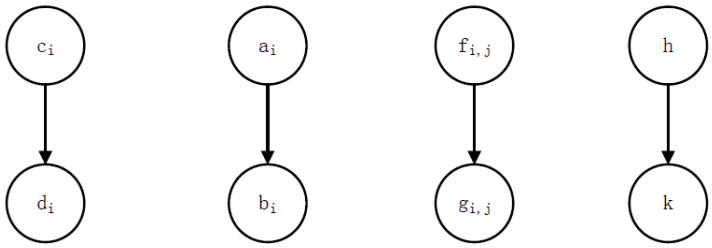

Fig. 9. (1) ref(x_i) (2) ref(y_i) (3) ref($P_{i,j}$) (4) ref(q).

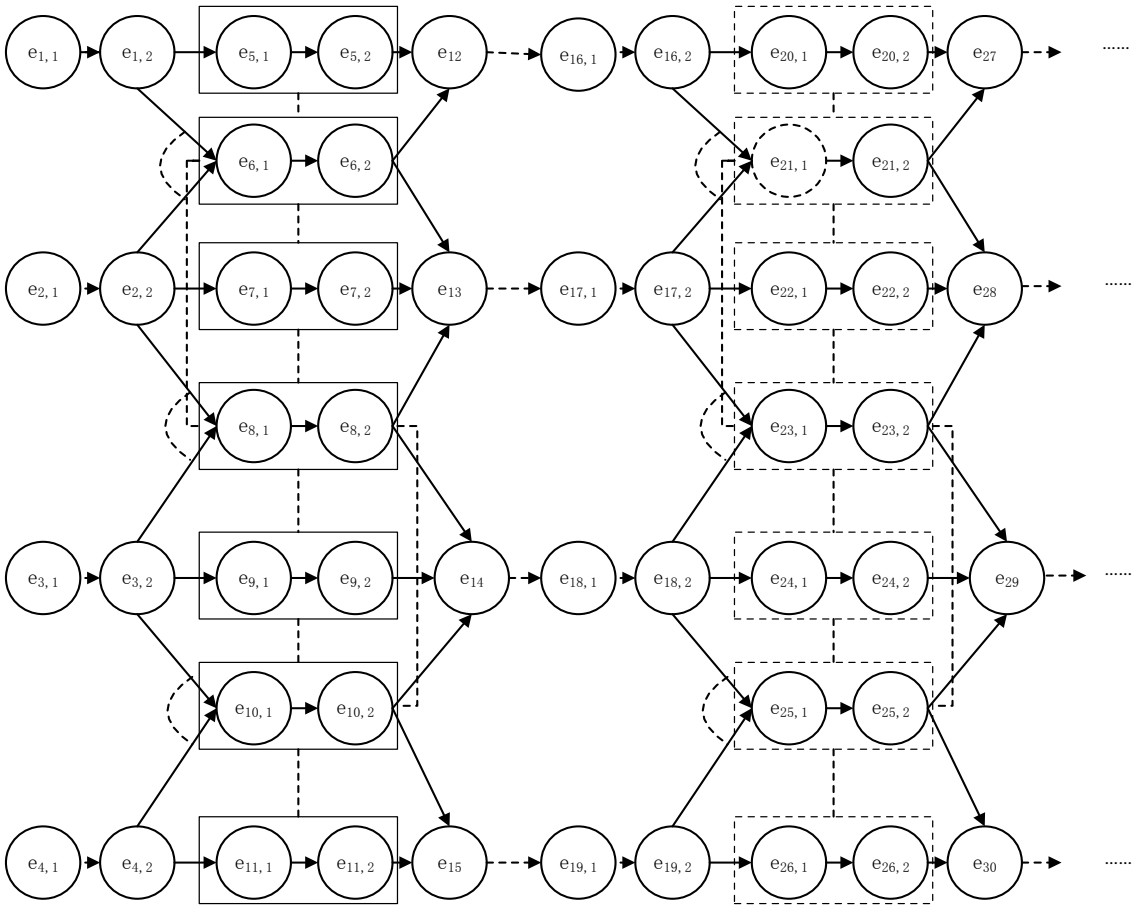

Fig. 10. The event structure model in second layer.

It can be easily verified that event structure prototype and the corresponding sub event structures of Σx_i, Σy_i, $\Sigma p_{i,j}$ and Σq are all live and regressive, but not fair. Thus after performing refinement

operation $ref(x_i, \Sigma x_i)$, $ref(y_i, \Sigma y_i)$, $ref(p_{i,j}, \Sigma p_{i,j})$ and $ref(q, \Sigma q)$, the event structure and its corresponding action structure in the second layer shown in Fig. 10 and Fig. 11 respectively are also live and regressive, but not fair. In Fig. 11, the group consisting of $f_{1,2}$ and $g_{1,2}$ is conflict with the one consisting of c_1 and d_1, the one consisting of c_2 and d_2, and the one consisting of h and k; the group consisting of $f_{3,4}$ and $g_{3,4}$ is conflict with the one consisting of c_3 and d_3, the one consisting of c_4 and d_4, and the one consisting of h and k.

We can see that liveness reflects that the system would not appear the deadlock state; and regression means that each state in the system can return to initial state, so it can work periodically and repeatedly; unfairness exactly reflects the independence when two robotic hands operate on small objects and the cooperativeness when two robotic hands operate on big objects, this is just the desired goal of the system. Similarly, in contrast to Fig. 10, Fig. 11 is more concise and clear, and shows stronger expressive ability.

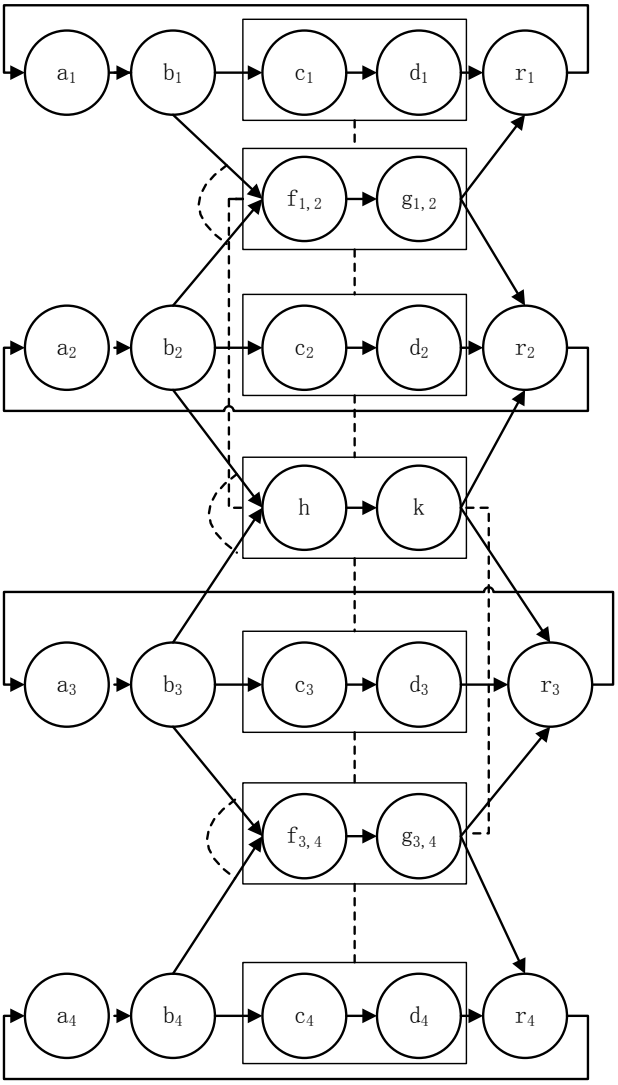

Fig. 11. The action structure model corresponding to the event structure model in second layer.

7. Conclusion

The use of action structure model can easily solve the difficult problem of representation of infinite events which encountered in the event structure model. We can adequately illustrate this point by modeling for Robotic Hand with Grasping Objects by using the event structure model method and the action structure model method respectively.

Acknowledgements

This work is partly supported by the Scientific Research Project No. KY2015ZD039 from Guangxi Education Department, Grant No. HCIC201306 of Guangxi HCIC lab Open Fund, the National Natural Science Foundation of China under Grant No. 11371003 and No. 11461006, 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 (GXSCIIP201201).

References

[1] Hoare, C. A. R. *Communicating Sequential Processes*. Prentice-Hall.

[2] Milner. R. (1995). *Communication and Concurrency*. Prentice-Hall.

[3] Park, D. (1981). Concurrency and automata on infinite sequences. *Proceedings of the 5th GI-Conference Theoretical Computer Science* (pp. 167-183).

[4] Winskel, G. (1989). *An Introduction to Event Structures*. Berlin: Springer.

[5] Van, G. R. J. (1990). *The Linear Time-Branching Time Spectrum*. Springer Berlin Heidelberg.

[6] Tang, W. D., Wu, J. Z., & Liu, M. L. (2014). Step semantics and action refinement in event structures. *Computer Modelling and New Technologies*, *18(5)*, 94-101.

[7] Tang, W. D., Wu, J. Z., & Liu, M. L. (2014). Interleaving semantics and action refinement in event structures. *Computer Modelling and New Technologies*, *18(6)*, 44-51.

[8] Louchka, P. Z. (2013). *Time and Petri nets*. Springer Berlin Heidelberg.

[9] Wu. Z. H. (2006). *Introduction to Petri net*. Chinese Mechanical Industry Press.

[10] Kurt, J., & Jonathan, B. (2009). Transactions on Petri nets and other models of concurrency III. *Lecture Notes in Computer Science*.

[11] Colom, J. M., & Jörg, D. (2013). Application and theory of Petri nets and concurrency. *Proceedings of the 34th International Conference Lecture Notes in Computer Science*..

[12] Changjun, J. (2008). *Discrete Event Dynamic System Theory of PN Machine*. Chinese Science Press.

[13] Glabbeek, R. V., & Goltz, U. (1990). Refinement of actions in causality based models. *Lecture Notes in Computer Science*, *430*, 267-300.

[14] Glabbeek, R. V., & Goltz, U. (2001). Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, *37*, 229-327.

[15] Tang, W. D., Wu, J. Z., & Wei, Z. D. (2014). On fuzzy rough sets and their topological structures. *Mathematical Problems in Engineering*.

[16] Majster-Cederbaum, M., & Wu, J. Z. (2003). Towards action refinement for true concurrent real time. *Acta Informatica*, *39(8)*, 531 - 577.

[17] Majster-Cederbaum, M., & Wu, J. H. (2006). Refinement of actions for real-time concurrent systems with causal ambiguity. *Acta Informatica*, *42(6-7)*, 389-418.

[18] Wu, J. (2001). Action refinement in timed LOTOS. *Proceedings of the ASCM'01, World Scientific Publ.*

Weidong Tang is an associate professor. He studies computer software and theory. His research interests are symbolic computation, formal verification

Jinzhao Wu is a professor. He studies are computer software and theory. His research interests symbolic computation, automated reasoning, formal methods.

Meiling Liu is an associate professor. She studies computer software and theory. Her research interests data mining, formal verification.