# Refinement of Actions for Real-Time Concurrent Systems with Causal Ambiguity

Mila Majster-Cederbaum[1], Jinzhao Wu[1,2,⋆], Houguang Yue[2], and Naijun Zhan[1]

[1] Fakultät für Mathematik und Informatik, Universität Mannheim, D7, 27, 68131 Mannheim, Germany

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

**Abstract.** We develop an approach of action refinement for concurrent systems with not only the notation of real-time but also with causal ambiguity, which often exists in real application areas. The systems are modeled in terms of a timed extension of event structures with causal ambiguity. Under a certain partial order semantics, the behavior of the refined system can be inferred compositionally from the behavior of the original system and from the behavior of the systems used to refine actions with explicitly represented start points. A variant of a linear-time equivalence termed pomset trace equivalence and a variant of a branching-time equivalence termed history preserving bisimulation equivalence based on the partial order semantics are both congruences under the refinement. The refinement operation behaves thus well and meets the commonly expected properties.

**Keywords:** Concurrency, action refinement, causal ambiguity, timed event structure with causal ambiguity.

## 1 Introduction

We consider the design of concurrent systems in the framework of approaches where the basic building blocks are actions. Refinement of actions is a core concept in the methodology of hierarchical design for concurrent systems, real-time or not. It amounts to introducing a mechanism for transforming high level actions into lower level processes until the implementation level is reached [6, 8, 16].

Refinement of actions for concurrent systems without time constraints has been thoroughly studied in the literature [6, 7, 8]. For real-time concurrent

⋆ Partially supported by National Natural Science Foundation of China(60373113).

systems it has been discussed recently by us in [4, 15, 16]. In all the previous work, the system models adopted are always under the constraint of unambiguous causality. That is, if an event has happened there must not exist ambiguity in deciding which are the causes of the event. However, causal ambiguity often exists in real application areas. Prominent examples are e.g. the design of distributed systems [20], the design and analysis of speed-independent circuits [23], and the specification of business processes like workflow management systems [21].

In this paper, we investigate how to carry out refinement of actions in concurrent systems with the notions of not only real-time but also causal ambiguity, and analyse its characteristic properties. The main practical benefit from our work is that hierarchical specification of such systems is then made possible, furthering thus the work of [4, 6, 15, 16].

The functional framework of systems is modeled in terms of event structures with causal ambiguity [13, 14], and the approach proposed in [1, 9, 10, 11, 12] to tackling time is taken, where a method to deal with urgent interaction can be incorporated. This approach is very powerful. We are unaware of any other proposal to incorporate time and timeout in a causality based setting.

Let us look at a running example to concretely motivate our paper. Figure 1 (a) shows a timed industrial inspection system observing faults, and Figure 1 (b) is a timed system used to implement action *inspect* in a more detailed level. They are both represented in timed event structures with causal ambiguity.

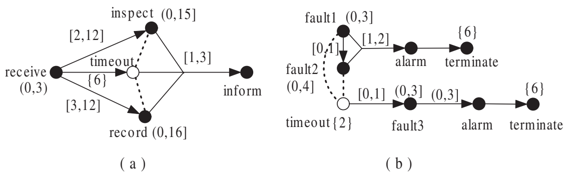

**Fig. 1.** Two timed systems with causal ambiguity

Solid dots stand for events, performing the action by which they are labeled, open dots represent timeout events, arrows from sets of events to events denote causality relations and are usually called bundles, and dashed lines denote conflict relations on the events. Two events that are in conflict cannot appear in a single system run.

Events and bundles are labeled with intervals (sets of non-negative reals). The intuitive interpretation of an interval associated to an event is that the time at which this event begins to happen is located in this interval, whereas an interval associated to a bundle has the meaning that the transition delay from the end of the causal event to the start of the result event is located in this interval. A timeout event restricts the happening time of the events in conflict with it.

It is unknown which exactly causes *inform* when both *inspect* and *record* occur. So causal ambiguity of *inform* exists in the timed system of Figure 1 (a). Similarly, causal ambiguity of *alarm* exists in the timed system of Figure 1 (b).

The following questions arise. How can action *inspect* in the timed system of Figure 1 (a) be refined by the timed system of Figure 1 (b)? How does the newly derived timed system, when obtained, behave?

The previous approach of action refinement [6, 7, 8] works for the untimed case when causal ambiguity exists [22]. However, this approach no longer applies to timed systems because the property of timeout events is violated, as shown in e.g. [4, 15], and therefore do not provide answers to the questions. To solve this, we present in this paper a new approach. Its characteristic properties are also analyzed. We consider two common problems of interest, the correctness and congruence problems.

The paper is arranged as follows. Section 2 introduces the system model timed event structures with causal ambiguity, describes partial order semantics for their behavior, and defines equivalence notions based on the semantics. Section 3 presents the approach of action refinement in the framework of this model. The correctness and the congruence results are given in Section 4. Section 5 concludes the paper.

## 2 System Model

We use event structures with causal ambiguity [3, 9, 13, 14] as the system model, and take the way of [1, 9, 10, 11, 12, 15, 16] to include time.

Assume a given set *Obs* of observable actions ranged over by $a, b, c, \ldots$, and an invisible internal action $\tau$ ($\tau \notin Obs$). Action $\surd$ ($\surd \notin Obs \cup \{\tau\}$) indicates the successful termination of a process. $Act = Obs \cup \{\tau, \surd\}$. Unlike [1, 9, 10, 11, 12], actions are viewed here as compound happenings having durations in order to have a clean notion of action refinement. Let function $k : Act \to \mathbb{R}^+$ with $k(\tau) = k(\surd) = 0$ assign durations to actions. Here, $\mathbb{R}^+ = [0, \infty)$, the set of non-negative reals, denotes the domain of time. Note that action durations defined through the function $k$ embody the intrinsic meaning of actions, and the treatment of the function $k$ as a component of a system model may result in conceptual confusion.

### 2.1 Timed Event Structures with Causal Ambiguity

A *timed event structure with causal ambiguity* (*taes* for short) $\mathcal{E}$ is a tuple $(E, \sharp, \vdash, l, \mathcal{D}, \mathcal{R}, \mathcal{U})$ with

$E$, a set of *events*;
$\sharp \subseteq E \times E$, the irreflexive and symmetric *conflict relation*;
$\vdash \subseteq 2^E \times E$, the *bundle relation*;
$l : E \longrightarrow Act$, the *action-labeling function*;
$\mathcal{D} : E \rightarrow 2^{\mathbb{R}^+}$, the *event delay function*;
$\mathcal{R} : \vdash \rightarrow 2^{\mathbb{R}^+}$, the *bundle delay function*; and
$\mathcal{U} \subseteq \{e \in E \mid l(e) = \tau\}$, the set of *urgent events*,
such that for any $X \subseteq E$, events $e \in \mathcal{U}$ and $e' \in E$,
(1) $(X \vdash e) \wedge (e \sharp e') \Rightarrow (X \vdash e') \vee (X \sharp e')$, and
(2) $\exists t \in \mathbb{R}^+ : (\mathcal{D}(e) \subseteq \{t\}) \vee (\exists Y \vdash e : (\mathcal{R}(Y, e) \subseteq \{t\}))$.

Here $2^\bullet$ denotes the power-set function, and $(X \sharp e')$ stands for $(\forall e \in X : e \sharp e')$.

The first constraint specifies a property of timeouts that are modeled by urgent events. The second constraint ensures that timeouts are enabled at a single time instant only. Notice that crucial for a taes is that for each bundle $X \vdash e$, opposite to classical event structures without causal ambiguity, the events in $X$ are not required to be in pairwise conflict as usual. Causal ambiguity thus exists in the system behavior, i.e., if an event $e$ occurs, there are alternative causes for this event.

A taes is depicted as the example shown in the introduction, where event names and delays $[0, \infty)$ are usually omitted.

**Example 2.1.1** Figure 1 (a) and (b) are two taes's. In the sequel, we abbreviate *receive* $= a$, *inspect* $= b$, *record* $= c$, *inform* $= d$, *fault1* $= b_1$, *fault2* $= b_2$, *fault3* $= b_3$, *alarm* $= b_4$. A *timeout* event is labeled with action $\tau$ since it is internal. Moreover, we assume $k(a) = 2$, $k(b) = 6$, $k(c) = 6$, $k(d) = 3$ and $k(b_1) = k(b_2) = k(b_3) = k(b_4) = 1$.

By $init(\mathcal{E})$ we denote the set of initial events of $\mathcal{E}$, and $exit(\mathcal{E})$ its set of successful termination events, i.e.,

$$init(\mathcal{E}) = \{e \in E \mid \neg(\exists X \subseteq E : X \vdash e)\}, exit(\mathcal{E}) = \{e \in E \mid l(e) = \surd\}.$$

We denote by $TAES$ the set of taes's. $\mathcal{E} = (E, \sharp, \vdash, l, \mathcal{D}, \mathcal{R}, \mathcal{U})$, possibly subscripted and/or primed, stands for a member of this set. When necessary, we also use $E_\mathcal{E}$, $\sharp_\mathcal{E}$, $\vdash_\mathcal{E}$, $l_\mathcal{E}$, $\mathcal{D}_\mathcal{E}$, $\mathcal{R}_\mathcal{E}$ and $\mathcal{U}_\mathcal{E}$ to represent the components of $\mathcal{E}$.

## 2.2 Partial Order Semantics

We first consider the functional behavior of a taes, then take the time aspect into account and define their semantics, taking care of both the functional and timing issues.

*Functional Behavior.* Let $\Phi = e_1, \cdots, e_m$ be a sequence of events (of $\mathcal{E}$), where $e_i$ and $e_j$ are distinct whenever $i \neq j$. $E(\Phi) = \{e_1, \cdots, e_m\}$. Assume $\Phi_{i-1} = e_1, \cdots, e_{i-1}$ is the $(i-1)$-th prefix of $\Phi$ $(1 \leq i \leq m)$.

In a run of a system, any two events that occur should not be in conflict, and if an event occurs in a run its causal predecessors should have occurred before. Let

$$en(\Phi_{i-1}) = \{\ e \in E \setminus E(\Phi_{i-1}) \mid (\forall e_j \in E(\Phi_{i-1}) : e \not\sharp e_j) \wedge \\ (\forall X \vdash e : X \cap E(\Phi_{i-1}) \neq \emptyset)\}.$$

$en(\Phi_{i-1})$ is then the set of events *enabled after* $\Phi_{i-1}$. If $e_i \in en(\Phi_{i-1})$ for all $1 \leqslant i \leqslant m$, then $C = E(\Phi)$ is called a *frame configuration* of $\mathcal{E}$.

The event sequence $\Phi$ in the definition of frame configurations is usually called a *trace* of $\mathcal{E}$. Traces describe actually the possible functional runs of a taes, whereas the frame configuration is the underlying event set of it. Frame configuration $C$ is usually said to be *obtained from trace* $\Phi$. By $C(\mathcal{E})$ we denote the set of all the frame configurations of $\mathcal{E}$.

A frame configuration $C$ of $\mathcal{E}$ *successfully terminates* if there exists $e \in C$ such that $l(e) = \surd$. $\mathcal{E}$ is called *well-labeled* if for each $C \in C(\mathcal{E})$, $C \cap exit(\mathcal{E})$ is empty or a singleton.

When causal ambiguity exists, as argued in [14], frame configurations or traces do not sufficiently describe the system behavior, and partial ordered sets are used to represent the causal relations between the events that occurred so far in a system run.

For $e_i, e_j \in E$, by $e_j \mapsto e_i$ we mean that $e_j$ is a *causal predecessor* of $e_i$, i.e., there exits an $X \subseteq E$ such that $e_j \in X$ and $X \vdash e_i$. Let $C \in C(\mathcal{E})$, and $\mapsto|_C$ denotes the restriction of $\mapsto$ to $C$, namely $\mapsto|_C = \mapsto \cap (C \times C)$.

Let $\mapsto_C$ be a subset of $\mapsto|_C$, such that $\overset{*}{\mapsto}_C$ is a partial order on $C$. Here, as usual $\overset{*}{\mapsto}_C$ represents the relation on $C$ by the elements of $\mapsto_C$ as generators, namely its reflexive and transitive closure. $P = \langle C, \overset{*}{\mapsto}_C \rangle$ is called a *poset of* $\mathcal{E}$. The *cause* of event $e \in C$ (in this poset) is defined as $\mathcal{A}_e = \{e' \in C \mid e' \mapsto_C e\}$. The elements of $\mathcal{A}_e$ constitute indeed all the direct predecessors of $e$ in this poset, namely the events that cause $e$ to happen. A *maximal event* in a poset $P$ is defined as an event $e \in C$, satisfying that there does not exist an event $e' \in C$ with $e \mapsto_C e'$.

*Taking Time into Account.* If event $e_j$ causes event $e_i$ in a system run of $\mathcal{E}$, and two time instants $t_j$ and $t_i \in \mathbb{R}^+$ are associated to events $e_j$ and $e_i$ respectively ($t_j \leq t_i$), then $e_j$ and $e_i$ start at time points $t_j$ and $t_i$, and $t_j$ and $t_i$ are required to be in the time instant sets labeled to $e_j$ and $e_i$, respectively. $t_i - (t_j + k(l(e_j)))$ is then the transition delay from the end of $e_j$ to the start of $e_i$. It is required to be in the time instant set attached to the corresponding bundle. Bearing this in mind, we see that if an event $e$ is enabled by its cause $\mathcal{A}_e$ in a poset $P$ then

$$T_P(e) = \mathcal{D}(e) \cap \bigcap_{e_j \in \mathcal{A}_e \wedge X \vdash e, e_j \in X} (t_j + k(l(e_j)) + \mathcal{R}(X, e))$$

consists of all the potential time instants at which event $e$ may start to happen. Here, for $t \in \mathbb{R}^+$ and $T \subseteq \mathbb{R}^+$, $t \pm T$ denotes the set $\{t \pm t_i \mid t_i \in T\} \cap \mathbb{R}^+$ respectively. $T \pm t$ is defined in a similar way, and we say $t \leq T$ if $t$ is not greater than any element of $T$. Furthermore, the fact that event $e_i$ occurs implies also that the starting time of $e_i$ must not be greater than the time at which $e_0$ may occur, where $e_0$ is an urgent event in conflict with $e_i$ and enabled by $e_j$.We define $T_P(e) = \mathcal{D}(e)$ if $\mathcal{A}_e = \emptyset$.

Formally, we assume hereafter that event sequence $\Phi = e_1, \cdots, e_m$ is a trace of $\mathcal{E}$, from which a frame configuration $C$ is obtained. For event $e \in C$, all bundles pointing to $e$ in $\mathcal{E}$ are given by $X_1 \vdash e, \cdots, X_n \vdash e$. Furthermore, $P = \langle C, \overset{*}{\mapsto}_C \rangle$ is a poset. Let $TP = T\langle C, \overset{*}{\mapsto}_C \rangle$ associate with each event $e \in C$ in poset $\langle C, \overset{*}{\mapsto}_C \rangle$ a time instant $t \in \mathbb{R}^+$, and let $time(C, e)$ denote the time instant associated to event $e$ in $TP$.

$TP = T\langle C, \overset{*}{\mapsto}_C \rangle$ is called a *timed poset* of $\mathcal{E}$, if for all $e \in C$ and $e_0 \in \mathcal{U}$ the following conditions hold:

(1) $time(C, e) \in T_P(e)$;
(2) $time(C, e) \leq T_P(e_0)$ if $e, e_0 \in en(\Phi_{i-1})$ for all $1 \leq i \leq m$ and $e \sharp e_0$.

The first condition requires that the time at which $e$ is enabled to happen cannot be smaller than the execution time of its causal events. The second condition ensures that an event in conflict with a timeout should not occur later than the timeout.

A non-empty timed poset $TP$ is often graphically denoted, where only the generators of $\overset{*}{\mapsto}_C$, i.e. the elements of $\mapsto_C$, together with the time instants associated with the events are depicted explicitly. That is, if $e_j \mapsto_C e_i$ is one of the elements of $\mapsto_C$, then an arrow from $(e_j, time(C, e_j))$ towards $(e_i, time(C, e_i))$ is drawn.

Let $Maxt(\mathcal{E})$ consist of all the time instants at which a successful termination run of the system finishes. That is,

$$Maxt(\mathcal{E}) = \{t \in \mathbb{R}^+ \mid \text{there is a timed poset } T\langle C, \mapsto_C \rangle \text{ of } \mathcal{E} \text{ and an event } e \in C \text{ with } l(e) = \surd \text{ and } t = time(C, e)\}.$$

*The Semantics.* Motivated by a large variety of applications, partial order semantics for taes's is defined on the basis of different viewpoints on the causality in a frame configuration. We only present here the definition of timed lib-posets (timed liberal posets). For the similar details of *timed bsat-posets* (timed bundle satisfaction posets), *timed min-posets* (timed minimal posets), *timed early-posets* and *timed late-posets*, as well as their relationships, the reader may consult [14].

A timed poset $T\langle C, \overset{*}{\mapsto}_C \rangle$ of $\mathcal{E}$ is said to be a *timed lib-poset*, if for all $e \in C$ the cause $\mathcal{A}_e$ of $e$ meets the conditions below:

(1) Each $e' \in \mathcal{A}_e$ occurs before $e$ in $\Phi$;
(2) $X_i \cap \mathcal{A}_e \neq \emptyset$ for all $i \in \{1, \ldots, n\}$.

Timed lib-posets model the least restrictive notion of causality. It says that each set of events from bundles pointing to event $e$ that satisfies all bundles is a cause of $e$.

From now on, we assume $x \in \{lib, bsat, min, early, late\}$, and by $TP_x(\mathcal{E})$ we denote the set of all timed x-posets of $\mathcal{E}$.

## 2.3 Equivalence Notions

For concurrency models such as timed event structures with causal ambiguity, the possible executions of systems may be represented as partially ordered multisets of actions (pomsets). Based on this consideration, a linear-time equivalence similar to pomset trace equivalence [2] is defined. A branching-time equivalence similar to history preserving bisimulation equivalence [19] can be defined as well to further record where choices are made, and the idea is to relate two events only if they have the same causal history.

For $x \in \{lib, bsat, min, early, late\}$ and $i = 1, 2$, let $TP_i = T\langle C_i, \overset{*}{\mapsto}_{iC_i}\rangle$ be timed x-posets of taes $\mathcal{E}_i$, where $C_i \in C(\mathcal{E}_i)$, and $\mapsto_{iC_i} \subseteq \mapsto_i |_{C_i}$. Moreover, let $l_i|_{C_i}$ denote the restriction of $l_i$ on $C_i$, namely $l_i|_{C_i}(e) = l_i(e)$ for $e \in C_i$.

$TP_1$ and $TP_2$ are said to be isomorphic, if they differ only in the event names, while the associated time instants, the action labels, the causality relations of events as well as the urgency of events in the two timed x-posets remain the same. Formally, $TP_1$ and $TP_2$ are *isomorphic*, denoted $TP_1 \approx TP_2$, if there exists a bijection $h : C_1 \longrightarrow C_2$ such that for arbitrary $e, e' \in C_1$,

$$\begin{array}{l}
(1)\ l_1|_{C_1}(e) = l_2|_{C_2}(h(e));\\
(2)\ e \mapsto_{1C_1} e' \text{ iff } h(e) \mapsto_{2C_2} h(e');\\
(3)\ time(C_1, e) = time(C_2, h(e));\\
(4)\ e \in \mathcal{U}_1 \text{ iff } h(e) \in \mathcal{U}_2.
\end{array}$$

By $TP_x(\mathcal{E}_1) \approx TP_x(\mathcal{E}_2)$ we mean that $\mathcal{E}_1$ and $\mathcal{E}_2$ have isomorphic timed x-posets. That is, for any $TP_1 \in TP_x(\mathcal{E}_1)$ [$TP_2 \in TP_x(\mathcal{E}_2)$] there exists $TP_2 \in TP_x(\mathcal{E}_2)$ [resp. $TP_1 \in TP_x(\mathcal{E}_1)$] such that $TP_1 \approx TP_2$.

Two taes's $\mathcal{E}_1$ and $\mathcal{E}_2$ are said to be *x-pomset trace equivalent*, denoted $\mathcal{E}_1 \cong_{px} \mathcal{E}_2$, if $TP_x(\mathcal{E}_1) \approx TP_x(\mathcal{E}_2)$.

Let $TP = T\langle C, \overset{*}{\mapsto}_C\rangle$ and $TP' = T\langle C', \overset{*}{\mapsto}_{C'}\rangle$ be two timed x-posets of taes $\mathcal{E}$, where $C, C' \in C(\mathcal{E})$, and $\mapsto_C$ and $\mapsto_{C'}$ are subsets of $\mapsto |_C$ and $\mapsto |_{C'}$, respectively. For action $a \in Act$, $t \in \mathbb{R}^+$, we say $TP \xrightarrow{a,t} TP'$ (or $\xrightarrow{a,t}_i$ when necessary) if $C' \setminus C = \{e\}$ with $l(e) = a$ and $time(C', e) = t$.

A relation $H \subseteq TP_x(\mathcal{E}_1) \times TP_x(\mathcal{E}_2) \times 2^{E_1 \times E_2}$ is called an *x-history preserving bisimulation* between $\mathcal{E}_1$ and $\mathcal{E}_2$, if $(\emptyset, \emptyset, \emptyset) \in H$, and when $(TP_1, TP_2, h) \in H$ then

$$\begin{array}{l}
(1)\ h \text{ is an isomorphism between } TP_1 \text{ and } TP_2,\\
(2)\ TP_1 \xrightarrow{a,t}_1 TP_1' \Rightarrow \exists TP_2', h' :\\
\quad TP_2 \xrightarrow{a,t}_2 TP_2', (TP_1', TP_2', h') \in H \text{ and } h'|_{C_1} = h,\\
(3)\ TP_2 \xrightarrow{a,t}_2 TP_2' \Rightarrow \exists TP_1', h' :\\
\quad TP_1 \xrightarrow{a,t}_1 TP_1', (TP_1', TP_2', h') \in H \text{ and } h'|_{C_1} = h.
\end{array}$$

Here $h'|_{C_1}$ denotes the restriction of $h'$ on $C_1$. $h$ is an isomorphism between $TP_1$ and $TP_2$ if it is the bijection in the definition that $TP_1$ and $TP_2$ are isomorphic.

Two taes's $\mathcal{E}_1$ and $\mathcal{E}_2$ are said to be x-history preserving bisimulation equivalent, denoted $\mathcal{E}_1 \cong_{bx} \mathcal{E}_2$, if there exists an x-history preserving bisimulation between $\mathcal{E}_1$ and $\mathcal{E}_2$.

x-pomset trace equivalence is coarser than x-history preserving bisimulation equivalence for any $x \in \{lib, bsat, min, early, late\}$. We refer the reader to [6] for this in the non-ambiguous case.

## 3 Refinement of Actions

We follow the methodology to treat refinement of actions as an operator. Before refining a given taes, we have first to modify the taes's that are used to implement actions.

### 3.1 Introducing Start-Events

Every system has a start point, which is usually supposed to be performing the internal silent action at time instant zero. Here, in the taes used to refine an action we introduce a new event to explicitly represent this point.

Let $r(\mathcal{E})$ be the taes obtained by adding to taes $\mathcal{E}$ a new event $e_0$ as well as new bundles from $e_0$ to all the events of $\mathcal{E}$, and transferring all the absolute time attachments of events to relative time attachments of the corresponding newly introduced bundles, where $e_0$ is labeled with the internal action $\tau$ and associated with time instant 0, imitating the start point of system, which is executing the internal silent action at time instant 0. Formally,

$$r(\mathcal{E}) = (E \cup \{e_0\}, \sharp, \mapsto_r, l \cup \{(e_0, \tau)\}, \mathcal{D}_r, \mathcal{R}_r, \mathcal{U}), \text{ where } e_0 \notin E, \text{ and}$$

$$\begin{aligned} \mapsto_r &= \mapsto \cup (\{\{e_0\}\} \times E), \\ \mathcal{D}_r &= \{(e_0, \{0\})\} \cup (E \times \{\mathbb{R}^+\}), \\ \mathcal{R}_r &= \mathcal{R} \cup \{((\{e_0\}, e), \mathcal{D}(e)) \mid e \in E\}. \end{aligned}$$

We call $r(\mathcal{E})$ the *rooted taes* associated with $\mathcal{E}$. The newly introduced event $e_0$, denoted by $o_{r(\mathcal{E})}$, is called the *start-event* of $\mathcal{E}$ or $r(\mathcal{E})$.

**Example 3.1.1** Let $\mathcal{E}_b$ be the taes of Figure 1(b) of Example 2.1.1. Then $r(\mathcal{E}_b)$ is the taes of Figure 2.

### 3.2 Refining a Taes

Let $Act_0$ be a subset of $Act$ that contains $\tau$ and $\surd$, representing the set of actions which need not or cannot be refined. Function $f : Act \setminus Act_0 \longrightarrow TAES$ is called a *refinement function*, if for any action $a \in Act \setminus Act_0$ it satisfies the following conditions:

$$\begin{array}{l}(1)\ f(a) \text{ is well-labeled;}\\(2)\ Maxt(f(a)) = \{k(a)\}.\end{array}$$

$f(a)$ is called *a refinement of action a*.

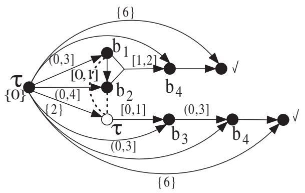

**Fig. 2.** The rooted taes associated with Figure 1(b)

**Example 3.2.1** Assume $Act \setminus Act_0 = \{b\}$, and $\mathcal{E}_b$ the taes of Figure 1(b) of Example 2.1.1. Then $f(b) = \mathcal{E}_b$ is a refinement of action $b$.

Throughout the paper, we use $f$ to denote a refinement function. The current question is how this refinement function can be applied to a given taes to obtain a refined one. Our basic idea, as illustrated in Figure 3, is that an action say $a$ is replaced by $r(f(a))$ in the original taes. For simplicity, we use in the following $rfl(e)$ and $rf(a)$ to abbreviate $r(f(l(e)))$ and $r(f(a))$, respectively.

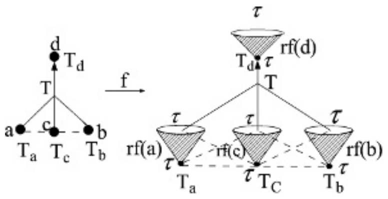

**Fig. 3.** Illustration of refining a taes

**Definition** The *refinement of taes* $\mathcal{E}$ is defined as $f(\mathcal{E}) = (E_f, \sharp_f, \vdash_f, l_f, \mathcal{D}_f, \mathcal{R}_f, \mathcal{U}_f)$, where

- *Event set*
$$E_f = \{(e, e') \mid (e \in E) \wedge (l(e) \notin Act_0) \wedge (e' \in E_{rfl(e)})\} \cup \{(e, e) \mid (e \in E) \wedge (l(e) \in Act_0)\},$$

• *Conflict relation*
for $(e_1, e_2), (e'_1, e'_2) \in E_f,\ (e_1, e_2)\sharp_f(e'_1, e'_2)$ iff
if $e_1 = e'_1$ then $e_2\sharp_{rfl(e_1)}e'_2$,
if $e_1 \neq e'_1$ then
  if $e_2 = e_1$ and $e'_2 = e'_1$ then $e_1\sharp e'_1$,
  if $e_2 \neq e_1$ and $e'_2 = e'_1$ then $e_1\sharp e'_1$ and $e_2 = o_{rfl(e_1)}$,
  if $e_2 = e_1$ and $e'_2 \neq e'_1$ then $e_1\sharp e'_1$ and $e'_2 = o_{rfl(e'_1)}$,
  if $e_2 \neq e_1$ and $e'_2 \neq e'_1$ then $e_1\sharp e'_1$, $e_2 = o_{rfl(e_1)}$ and $e'_2 \in \{o_{rfl(e'_1)}\}\cup$ $exit(rfl(e'_1))$ or $e_2 \in \{o_{rfl(e_1)}\} \cup exit(rfl(e_1))$ and $e'_2 = o_{rfl(e'_1)}$,

• *Bundle relation*
for $X \subseteq E_f$ and $(e_1, e_2) \in E_f, X \vdash_f (e_1, e_2)$ iff
if $e_2 \neq e_1$ and $e_2 \in E_{rfl(e_1)} \setminus \{o_{rfl(e_1)}\}$ then $\pi_1(X) = \{e_1\}$ and $\pi_2(X) \vdash_{rfl(e_1)} e_2$,
if $e_2 \neq e_1$ and $e_2 = o_{rfl(e_1)}$ or $e_2 = e_1$ then $\pi_1(X) \vdash e_1$ and $\pi_2(X) = \cup_{e\in\pi_1(X),l(e)\notin Act_0} exit(rfl(e)) \cup (\cup_{e\in\pi_1(X),l(e)\in Act_0})\{e\}$,

• *Action labeling function*
for $(e_1, e_2) \in E_f$ if $e_2 \neq e_1$ then
  if $e_2 \notin exit(rfl(e_1))$ then $l_f(e_1, e_2) = l_{rfl(e_1)}(e_2)$,
  if $e_2 \in exit(rfl(e_1))$ then $l_f(e_1, e_2) = \tau$,
if $e_2 = e_1$ then $l_f(e_1, e_2) = l(e_1)$,

• *Event delay function*
for $(e_1, e_2) \in E_f$ if $e_1 = e_2$ then $\mathcal{D}_f(e_1, e_2) = \mathcal{D}(e_1)$,
if $e_1 \neq e_2$ then
  if $e_2 = o_{rfl(e_1)}$ then $\mathcal{D}_f(e_1, e_2) = \mathcal{D}(e_1)$,
  if $e_2 \neq o_{rfl(e_1)}$ then $\mathcal{D}_f(e_1, e_2) = \mathbb{R}^+$,

• *Bundle delay function*
for $X \subseteq E_f$ and $(e_1, e_2) \in E_f$, if $X \vdash_f (e_1, e_2)$ then
if $e_2 \neq e_1$, $e_2 \in E_{rfl(e_1)} \setminus \{o_{rfl(e_1)}\}$ and $\pi_1(X) = \{e_1\}$ then $\mathcal{R}_f(X, (e_1, e_2)) = \mathcal{R}_{rfl(e_1)}(\pi_2(X), e_2)$,
if $e_2 \neq e_1$ and $e_2 = o_{rfl(e_1)}$ then $\mathcal{R}_f(X, (e_1, e_2)) = \mathcal{R}(\pi_1(X), e_1)$,
if $e_2 = e_1$ then $\mathcal{R}_f(X, (e_1, e_2)) = \mathcal{R}(\pi_1(X), e_1)$,

• *Urgent events*
for $(e, e') \in E_f$, $(e, e') \in \mathcal{U}_f$ iff $e \in \mathcal{U}$ or $e' \in \mathcal{U}_{rfl(e)}$ if $e' \neq e$.

Here, $\pi_1(X) = \{e \mid (e, e') \in X\}$, and $\pi_2(X) = \{e' \mid (e, e') \in X\}$.

**Example 3.2.2** Suppose that $\mathcal{E}$ is the taes of Figure 1(a) of Example 2.1.1. Let $Act \setminus Act_0 = \{b\}$, and $f(b)$ the refinement of action $b$ defined in Example 3.2.1 (Figure 1(b)). Then $rf(b)$ is the taes of Figure 2 of Example 3.1.1, and the refinement $f(\mathcal{E})$ of $\mathcal{E}$ is the taes of Figure 4.

$f(\mathcal{E})$ satisfies the definition of taes's. From this fact, the following theorem follows.

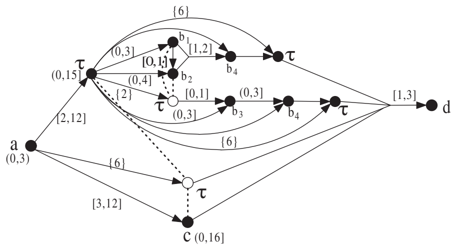

**Fig. 4.** The refinement of Figure 1(a)

**Theorem 3.2.1** Let $\mathcal{E} \in TAES$, and $f$ a refinement function. Then $f(\mathcal{E}) \in TAES$.

## 4 Correctness and Congruence Results

We show that our operation of action refinement is correct and furthermore, the equivalences defined in the preceding section are congruences under the refinement.

Assume again $x \in \{lib, bsat, min, early, late\}$. Let $TP_f = T\langle C_f, \overset{*}{\mapsto}_{fC_f}\rangle$ be a timed x-poset of $f(\mathcal{E})$, where $C_f$ is a frame configuration of $f(\mathcal{E})$, and $\mapsto_{fC_f} \subseteq \mapsto_f |_{C_f}$. Let

$$\begin{aligned}
&\pi_1(TP_f) = T\langle \pi_1(C_f), \overset{*}{\mapsto}_{\pi_1(C_f)}\rangle, \text{ where}\\
&\ \pi_1(C_f) = \{e \in E \mid (e, e_j) \in C_f\},\\
&\ \forall e \in \pi_1(C_f), time(\pi_1(C_f), e) =\\
&\quad \text{if } l(e) \in Act_0 \text{ then } time(C_f, (e, e)) \text{ else } time(C_f, (e, o_{rfl(e)})),\\
&\ \mapsto_{\pi_1(C_f)} \subseteq E \times E : e \mapsto_{\pi_1(C_f)} e' \text{ iff } \exists (e, e_1) \mapsto_{fC_f} (e', e_1').
\end{aligned}$$

Moreover, for event $e \in \pi_1(C_f)$ with $l(e) \notin Act_0$, let

$$\begin{aligned}
&\pi_e(TP_f) = T\langle \pi_e(C_f), \overset{*}{\mapsto}_{\pi_e(C_f)}\rangle, \text{ where}\\
&\ \pi_e(C_f) = \{e_j \in E_{rfl(e)} \mid (e, e_j) \in C_f\},\\
&\ \forall e_j \in \pi_e(C_f),\ time(\pi_e(C_f), e_j) = time(C_f, (e, e_j)) - time(C_f, (e, o_{rfl(e)})),\\
&\ \mapsto_{\pi_e(C_f)} \subseteq E_{rfl(e)} \times E_{rfl(e)} : e_1 \mapsto_{\pi_e(C_f)} e_2 \text{ iff } (e, e_1) \mapsto_{fC_f} (e, e_2).
\end{aligned}$$

$\pi_1(TP_f)$ is the projection of timed x-poset $TP_f$ on taes $\mathcal{E}$, and $\pi_e(TP_f)$ the projection of $TP_f$ on taes $rfl(e)$. We have then the following Lemma 4.1, which indicates that the projection of $TP_f$ on $\mathcal{E}$ is a timed x-poset of $\mathcal{E}$, and the projection of $TP_f$ on $rfl(e)$ is a timed x-poset of $rfl(e)$.

**Lemma 4.1** (1) $\pi_e(TP_f) \in TP_x(rfl(e))$; (2) $\pi_1(TP_f) \in TP_x(\mathcal{E})$.

We have also the following simple Lemma 4.2, which demonstrates that $\pi_e(TP_f)$ successfully terminates if $e$ is causally necessary for some other events. Notice that a timed x-poset is said to successfully terminate, if the frame configuration on which this timed x-poset is based successfully terminates.

**Lemma 4.2** If $e \in \pi_1(C_f)$, $l(e) \notin Act_0$ and $e$ is not maximal in $\pi_1(TP_f)$, then $\pi_e(TP_f)$ successfully terminates.

Now, we suppose $TP = T\langle C, \overset{*}{\mapsto}_C\rangle$ is a timed x-poset of $\mathcal{E}$, where $C \in C(\mathcal{E})$, and $\mapsto_C \subseteq \mapsto |_C$. Furthermore, $e \in C$ with $l(e) \notin Act_0$, and $TP_e = T\langle C_e, \overset{*}{\mapsto}_{rfl(e)C_e}\rangle$ a timed x-poset of $rfl(e)$, where $C_e \in C(rfl(e))$, $\mapsto_{rfl(e)C_e} \subseteq \mapsto_{rfl(e)} |_{C_e}$, and $C_e$ successfully terminates if $e$ is not maximal in $C$. Let

$f(TP, \rhd_e TP_e) = T\langle f(C, \cup_e C_e), \overset{*}{\mapsto}_{ff(C,\cup_e C_e)}\rangle$, where
$f(C, \cup_e C_e) = \{\ (e, e_j) \mid e \in C,\ \text{if } l(e) \in Act_0 \text{ then } e_j = e\ \text{ else } e_j \in C_e\}$,
$\forall (e, e_j) \in f(C, \cup_e C_e),\ time(f(C, \cup_e C_e), (e, e_j)) =$
  if $l(e) \in Act_0$ then $time(C, e)$ else $time(C_e, e_j) + time(C, e)$,
for $(e_1, e_1'), (e_2, e_2') \in E_f \times E_f$, $(e_1, e_1') \mapsto_{ff(C,\cup_e C_e)} (e_2, e_2')$ iff
 $e_1 \mapsto_C e_2$, and either $e_1' = e_1$ and $e_2' = e_2$, or
  if $e_1' \neq e_1$ and $e_2' = e_2$ then $e_1' \in exit(rfl(e_1))$,
  if $e_1' = e_1$ and $e_2' \neq e_2$ then $e_2' = o_{rfl(e_2)}$,
  if $e_1' \neq e_1$ and $e_2' \neq e_2$ then
   if $e_1 = e_2$ then $e_1' \mapsto_{rfl(e_1)C_{e_1}} e_2'$,
   if $e_1 \neq e_2$ then $e_1' \in exit(rfl(e_1))$ and $e_2' = o_{rfl(e_2)}$.

We call it a *refinement of TP*. By $f(TP_x(\mathcal{E}))$ we represent the set of all refinements of timed x-posets of taes $\mathcal{E}$. We have then Lemma 4.3, which states that a refinement of a timed x-poset of taes $\mathcal{E}$ is a timed x-poset of the refined taes $f(\mathcal{E})$.

**Lemma 4.3** $f(TP, \rhd_e TP_e) \in TP_x(f(\mathcal{E}))$ for $x \in \{lib, bsat, min, early, late\}$.

From Lemmas 4.1, 4.2 and 4.3, the following two theorems follow.

**Theorem 4.1** Suppose that $\mathcal{E}$ is a taes, and $f$ a refinement function. Then $TP_x(f(\mathcal{E})) = f(TP_x(\mathcal{E}))$ for $x \in \{lib, bsat, min, early, late\}$.

This theorem indicates that timed x-posets of the refined taes $f(\mathcal{E})$ can be obtained by replacing event $e$ with $e \notin Act_0$ in timed x-posets of the original taes $\mathcal{E}$ by timed x-posets of taes $rfl(e)$ used to refine action $l(e)$. The behavior of the refined taes can thus be inferred compositionally from the behavior of the original taes and from the behavior of those used to substitute actions, where the system starts are explicitly represented. The refinement keeps correct.

**Theorem 4.2** Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two taes's, and $f_1$ and $f_2$ two refinement functions. If $\mathcal{E}_1 \cong_{eq} \mathcal{E}_2$, and for any $a \in Act \setminus Act_0, f_1(a) \cong_{eq} f_2(a)$, then $f_1(\mathcal{E}_1) \cong_{eq} f_2(\mathcal{E}_2)$, where $eq \in \{px,\ bx\}$ and $x \in \{lib, bsat, min, early, late\}$.

This theorem shows that x-pomset trace and x-history preserving bisimulation equivalences are both congruences under the refinement. Our refinement notion is therefore well-defined under these equivalences.

## 5 Concluding Remarks

In this paper, we developed an approach of action refinement for real-time concurrent systems with causal ambiguity, where timed event structures with causal ambiguity are used as the system model. Furthermore, the following correctness and congruence results were certified under a certain partial order semantics:

– The behavior of the refined system can be inferred compositionally from the behavior of the original system and from the behavior of the systems used to refine actions, where new events are introduced to model the system starts.
– The variants of pomset trace and history preserving bisimulation equivalences are both congruences under the refinement.

We adopt again in this paper the basic refinement methodology proposed by us in [4, 5, 15, 16, 17]. We believe that the methodology for the probabilistic and stochastic cases proposed in [5, 17] applies to the corresponding cases when causal ambiguity exists. This is in fact our immediate future work. We also want to define a suitable process algebra to specify taes's, develop an approach of action refinement at this language level, and try to make the syntactic refinement and the semantic refinement presented in this paper conform to each other.

## References

1. H. Bowman, J-P. Katoen. A True Concurrency Semantics for ET-LOTOS. *Proceedings Int. Conference on Applications of Concurrency to System Design*, 228 - 239, 1998.
2. L. Castellano, G. Michelis, and L. Pomello. Concurrency vs Interleaving: An Instructive Example. *Bull. EATCS*, 31: 12- 15, 1987.

3. H. Fecher, M. Majster-Cederbaum, and J. Wu. Bundle Event Structures: A Revised Cpo Approach. *Information Processing Letters*, 83: 7 – 12, 2002.
4. H. Fecher, M. Majster-Cederbaum, and J. Wu. Refinement of Actions in a Real-Time Process Algebra with a True Concurrency Model. *Electronic Notes in Theoretical Computer Science*, 70(3), 2002.
5. H. Fecher, M. Majster-Cederbaum, and J. Wu. Action Refinement for Probabilistic Processes with True Concurrency Models. *Lecture Notes in Computer Science*, 2399: 77 – 94, 2002.
6. R. van Glabbeek, U. Goltz. Refinement of Actions and Equivalence Notions for Concurrent Systems. *Acta Informatica*, 37: 229 - 327, 2001.
7. U. Goltz, R. Gorrieri, and A. Rensink. On Syntactic and Semantic Action Refinement. *Lecture Notes in Computer Science*, 789: 385-404, 1994.
8. R. Gorrieri, A. Rensink. Action Refinement. *Handbook of Process Algebra*, Elsevier Science, 1047 - 1147, 2001.
9. J-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures*. PhD thesis, University of Twente, 1996.
10. J-P. Katoen, C. Baier, and D. Latella. Metric Semantics for True Concurrent Real Time. *Th. Comp. Sci.*, 254(1-2): 501 - 542, 2001.
11. J-P. Katoen, R. Langerak, E. Brinksma, D. Latella & T. Bolognesi. A Consistent Causality Based View on a Timed Process Algebra Including Urgent Interactions. *Formal Meth. in Sys. Design*, 12: 189 - 216, 1998.
12. J-P. Katoen, R. Langerak, D. Latella & E. Brinksma. On Specifying Real-Time Systems in a Causality-Based Setting. *Lecture Notes in Computer Science*, 1135: 385 - 405, 1996.
13. R. Langerak. Bundle Event Structures: A Non-Interleaving Semantics for LOTOS. In M. Diaz and R. Groz editors, *Formal Description Techniques V*, *IFIP Transactions*, C-10: 331 - 346, 1993.
14. R. Langerak, E. Brinksma, and J-P. Katoen. Causal Ambiguity and Partial Orders in Event Structures. *CONCUR'97*, Lecture Notes in Computer Science, 1243: 317 – 331, 1997.
15. M. Majster-Cederbaum, J. Wu. Action Refinement for True Concurrent Real Time. *Proc. ICECCS 2001*, IEEE Computer Society Press, 58 - 68, 2001.
16. M. Majster-Cederbaum, J. Wu. Towards Action Refinement for True Concurrent Real Time. *Acta Informatica*, 39(8): 531 - 577, 2003.
17. M. Majster-Cederbaum, J. Wu. Adding Action Refinement to Stochastic True Concurrency Models. *Lecture Notes in Computer Science*, 2885: 226 - 245, 2003.
18. D. Murphy. Time and Duration in Noninterleaving Concurrency. *Fundamenta Informaticae*, 19: 403 - 416, 1993.
19. A. Rabinovich, B. A. Trakhtenbrot. Behavior Structures and Nets. *Fund. Info.*, 11(4): 357 - 404, 1988.
20. M. van Sinderen, L. Ferreira Pires, C. A. Vissers, and J-P Katoen. A Design Model for Open Distributed Processing Systems. *Computer Networks and ISDN Systems*, 27: 1263 - 1285, 1995.
21. M. K. de Weger, H. Franken, and C. A. Vissers. A Development Model for Distributed Information Systems. In: *Proceedings of the 1st Int. Distributed Conference on High Performance Networking for Teleteaching (IDC'95)*, 1995.

22. J. Wu and H. Yue. Towards Action Refinement for Concurrent Systems with Causal Ambiguity. *Proc. SEFM 2004*, IEEE Computer Society Press, to appear, 2004.
23. A. Yakovlev, M. Kishinevsky, A. Kondratyev, and L. Lavagno. On the Models for Asynchronous Circuit Behaviour with OR causality. *Technical Report 463*, University of Newcastle upon Tyne, 1993.