

# Action Refinement Applied to Late Decisions

Harald Fecher

Christian Albrecht Universität zu Kiel  
Institut für Informatik  
24098 Kiel, Germany

Mila Majster-Cederbaum

Universität Mannheim  
Fakultät für Mathematik und Informatik,  
68131 Mannheim, Germany

**Abstract.** In modular approaches to specify concurrent systems a system is built up from components using various operators as e.g. the sequential, the parallel, or the choice (+) operator. Usually the choice between two components, i.e.  $P_1 + P_2$ , is taken in favor of that component that is first to *start* an action. We follow here the alternative view that a choice is taken in favor of that component that is the first to *terminate* an action (end-based choice). This alternative has various applications and interesting implications. In particular the different points of view lead to different action refinement operators. The contribution of this paper is to present here an action refinement operator for the end-based view in a suitable true concurrency setting and establish two equivalences that are the coarsest congruences for this refinement operator with respect to trace (respectively bisimulation) equivalence.

**Keywords:** action refinement, event structure, true concurrency, equivalence, coarsest congruence

## 1. Introduction

Concurrent systems can be modelled, for example, by action based process algebras [Hoa85, Mil89], by action based true concurrent or interleaving semantic models. In these approaches actions are usually considered to be instantaneous, i.e. durationless. Modular approaches to model concurrent (reactive) systems provide operators to compose a system from components such as the sequential (;), the parallel (||), or the choice (+) operator. In the system  $P_1 + P_2$ , if  $P_1$  is first to *execute* an instantaneous action, then the choice is taken in favor of  $P_1$ .

---

*Correspondence and offprint requests to:* Mila Majster-Cederbaum  
Universität Mannheim  
Fakultät für Mathematik und Informatik,  
68131 Mannheim, Germany, e-mail: mcb@pi2.informatik.uni-mannheim.de

If however real time aspects of systems have to be modelled and/or action refinement operators [GR01] are employed, we have to take into account that actions consume time<sup>1</sup>. If durational actions are considered, we have to state what it means to ‘execute’ an action. Does it mean to start execution or to end execution? The answer to this question has consequences for the interpretation of the choice operator as it is illustrated by the following example. Consider a process that consists of a choice between actions  $a$  and  $b$ . The duration of  $a$  is 3 and the duration of  $b$  is 1. Action  $a$  starts at time 0 and action  $b$  starts at time 1. If executing an action means starting it then the system performs  $a$ . If executing an action means terminating it then the system performs  $b$ , as  $b$  terminates first. In other words, the choice is either triggered by the start of actions or by the end of actions.

In most approaches in the literature, a choice is triggered by the start of an action, for example in timed systems [AM96, GRS95, MCW01, Mur93, Vog95] and in the context of action refinement [AH94, DD93, GG01, GR01, MCW01, Vog91]. But it is reasonable to consider approaches where choices are determined by the ending of actions:

- In stochastic approaches, it is common to consider a *race policy* approach [AMCB84, BG98, HR94], i.e. the fastest action triggers the choice. Consequently, a choice has to be triggered at the end of the action’s duration, since it is usually not known a priori which action is the fastest.
- The end-based point of view is useful for hierarchical system development, where complex activities are specified by single actions in the first system design steps. There the end-based choice is always of interest if abstract actions (those which will be refined) are involved in a choice, i.e. if they can trigger a choice. This is illustrated by the following generic example.

**Example 1.1.** Consider a factory in which a component broke down. This component can be replaced by two alternative machines. Therefore, the factory will order both machines from the depot and will use the machine that will be delivered first. Ordering both machines reduces the waiting time. On an abstract level this situation of the factory can be modelled by

$$P = order; ((deliver_1; run_1) + (deliver_2; run_2))$$

where *order* denotes the ordering of both machines,  $deliver_i$  denotes the receiving of the delivery of the  $i^{\text{th}}$  machine and  $run_i$  denotes the production with the  $i^{\text{th}}$  machine. The choice in  $P$  is taken when a machine is delivered, i.e. the choice is either triggered by  $deliver_1$  or by  $deliver_2$ , as usual.

In practice, the delivery of the  $i^{\text{th}}$  machine will not take place at once, for example if the machines consist of a number of components, which will be delivered from different places. Consequently the components will arrive at different times, i.e. action  $deliver_i$  is time consuming. In this case the choice in  $P$  has to be considered to be end-based, since the factory will choose that machine for which all components are completely delivered first (and not the machine of which one component is delivered first).

This is easily understood when we consider the next system design phase, where actions  $deliver_i$  are specified in more detail, i.e. they are refined by a process  $W_i$  (for example  $W_1 = comp_1 || comp_2 || comp_3$ ). Then the choice is triggered when either  $W_1$  or  $W_2$  terminates and not when the first action is executed by  $W_1$  or  $W_2$ . In particular, the actions of  $W_2$  that are executed before the termination of  $W_1$  remain visible, i.e. they are not made undone after the termination of  $W_1$ , and vice versa. This makes clear that the end-based choice can be viewed as some kind of parallelism, where  $W_1$  and  $W_2$  run in parallel until one of them terminates.

- The possibility of late decisions is also motivated and examined in Z [SCW98].

In this paper, we concentrate on the hierarchical system design. More precisely, we develop an action refinement operator for untimed event structures<sup>2</sup> with respect to the end-based point of view.

It turns out that the action refinement operator in an end-based setting is not compatible with conventional equivalence notions. In most equivalences the processes  $P_1 = a$  and  $P_2 = a + a$  are identified. Refining  $a$  in an end-based setting leads usually to processes that are no longer equivalent under these equivalences. Therefore, new equivalences that are congruences with respect to this refinement operator have to be established. Two are presented here: the ICT-equivalence and the FIC-equivalences. Furthermore, we show that the ICT-equivalence is the coarsest equivalence for end-based action refinement with respect to trace equivalence and the FIC-equivalence is the coarsest equivalence for end-based action refinement with respect to bisimulation equivalence. The presented paper is an extension of [FMC02].

The paper is organized as follows. A new class of event structures is introduced in Section 2. The action refinement

<sup>1</sup> Action refinement operators can split an action into a start- and an end-action, hence the action’s duration can be modelled in some sense.

<sup>2</sup> Event structures are a true concurrent model where action refinement can be easily defined.

operator with respect to the end-based view is established in Section 3. The two new equivalences are introduced in Section 4, which also contains the (coarsest) congruence statements. A conclusion is given in Section 5.

## 2. Extended Termination Bundle Event Structure (ETBES)

Event structures are partial order based models used for describing concurrent systems. They usually consist of events representing action occurrences, a causality relation between events, and a conflict relation between events describing a disabling mechanism. Two constraints have to be imposed on event structures in order to give a reasonable definition of a refinement operator with respect to the end-based point of view:

1. Each event in an event structure represents a unique occurrence of an action. This is necessary, since otherwise the occurrence of an action could be started more than once. In *prime event structures* [NPW81] such a unique representation can not be guaranteed, as pointed out, for example, in [GG01, Section 2.3].
2. The class of event structures must allow to model disruption, since a disrupt operation can result from the end-based refinement operator, which will be discussed in more detail in Remark 3.2. This is for example not the case for *prime event structures*, *flow event structures* [BC89, BC94], *stable event structures* [Win89] and *bundle event structures* [Lan93].

In order to adequately model sequential composition of event structures, they must have a termination mechanism: the second event structure may only start to be executed if the first one terminates. There exist different kinds of termination philosophies for process systems:

- A process terminates by executing an additional  $\surd$ -action, which is, for example, the case in LOTOS [BB87].
- A process terminates by executing its ‘final’ action [BFP01]. For example the process that consists of the parallel execution of action  $a$  and  $b$  ( $a||b$ ) terminates by executing  $a$  if  $b$  was executed before or it terminates by executing  $b$  if  $a$  was executed before.

In our first paper [FMC02], we followed the  $\surd$ -termination philosophy. More precisely, we define an end-based refinement operator in the class of *extended bundle event structures* [Lan92], since these event structures can model disruption with respect to the  $\surd$ -termination philosophy. Unfortunately, our newly introduced equivalences (the ICT- and the FIC-equivalence) fail to yield the coarsest equivalences for the end-based action refinement in extended bundle event structures [FMC02].

In this paper, we follow the ‘final’ action termination philosophy. Therefore, we introduce a new class of event structures, called *extended termination bundle event structures*, where disruption can be modelled with respect to the ‘final’ action termination philosophy. This class of event structures is used in Section 3 in order to define an action refinement operator with respect to the end-based view and with respect to the ‘final’ action termination philosophy. Extended termination bundle event structures are a generalization of (extended) bundle event structures [Lan92, Lan93]. In bundle event structures the causality relation is modelled by sets of events (called bundles) pointing to events: an event is enabled if for every bundle pointing to it there is an event in the bundle that already occurred in the previous execution. Disabling is modeled by a binary conflict relation. Extended termination bundle event structures allow a more liberal disabling mechanism, in particular they model non-disabling rather than disabling. In extended bundle event structures termination is modelled by pointing explicitly to a termination event. In contrast to this we model termination by collecting certain subsets of events into a termination set. A run of an extended termination bundle event structures terminates if for every termination set  $X$  there is an event in  $X$  that occurred in the run.

Let  $\text{Obs}$  be an uncountable set of all *observable actions* and let  $\tau$  be the *internal action*. Furthermore,  $\text{Act} = \text{Obs} \cup \{\tau\}$  denotes the set of all *actions*. We use the following notations:  $\mathcal{P}(M)$  denotes the powerset of  $M$ ,  $\mathcal{P}_{fin}(M)$  denotes the set of all finite subsets of  $M$  and  $M_1 \rightarrow M_2$  denotes the set of all partial functions from  $M_1$  to  $M_2$ . Furthermore, the domain of a partial function  $f$  is the set  $\{m \mid f(m) \text{ is defined}\}$ , and is denoted by  $\text{dom}(f)$ . The function  $f \upharpoonright M'_1$ , where  $M'_1 \subseteq M_1$ , denotes the restriction of function  $f$  to the domain  $M'_1$ . Furthermore,  $f(M'_1)$  denotes the set  $\{f(m_1) \mid m_1 \in M'_1\}$ . For any binary relation  $\odot$  we write  $p \odot q$  if and only if  $(p, q) \in \odot$ . The set  $M \setminus M'$  denotes the set where all elements of  $M'$  are removed from  $M$ . We assume a fixed countable set of events  $\mathcal{U}$  such that  $\forall e, e' \in \mathcal{U} : (e, e') \in \mathcal{U}$  and  $\bullet \in \mathcal{U}$ .<sup>3</sup> The constraints on set  $\mathcal{U}$  result from technical reasons, i.e. they

<sup>3</sup> Such a set  $\mathcal{U}$  can be achieved as follows: Let  $\mathcal{U}_0$  be an at most countable set containing  $\bullet$ . We put  $\mathcal{U}_{i+1} = \mathcal{U}_i \cup (\mathcal{U}_i \times \mathcal{U}_i)$ , for  $i \geq 0$ , and  $\mathcal{U} = \bigcup_{i \in \mathbb{N}} \mathcal{U}_i$ .

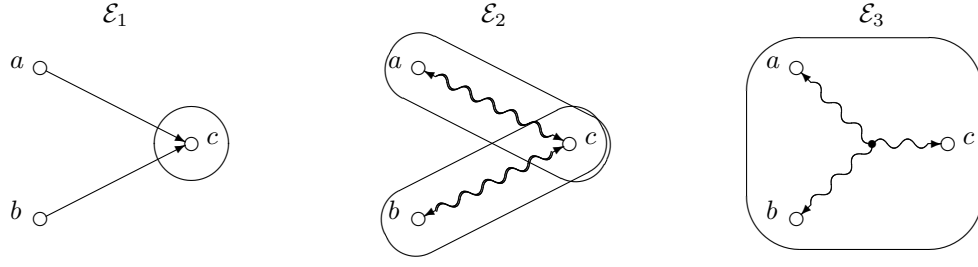


Fig. 1. Some Extended Termination Bundle Event Structures

guarantee the well-definedness of the refinement operator, which will be presented in Section 3. We introduce the following kind of event structures. Their intuitive meaning is explained below.

**Definition 2.1 (Extended Termination Bundle Event Structure).** An *extended termination bundle event structure* (*eTbes*)  $\mathcal{E} = (E, \succ, \mapsto, T, l)$  is an element of  $\mathcal{P}(\mathcal{U}) \times \mathcal{P}(\mathcal{P}(\mathcal{U}) \times \mathcal{U}) \times \mathcal{P}(\mathcal{P}(\mathcal{U}) \times \mathcal{U}) \times \mathcal{P}(\mathcal{P}(\mathcal{U})) \times (\mathcal{U} \rightarrow Act)$  such that

- $\succ \subseteq \mathcal{P}(E) \times E$  and  $\forall e \in E : \exists Z : Z \succ e$  and  $\forall (Z, e) \in \succ : e \in Z$
- $\mapsto \subseteq \mathcal{P}(E) \times E$
- $T \subseteq \mathcal{P}(E)$  and  $T \neq \emptyset$
- $\text{dom}(l) = E$

Let **ETBES** denote the set of all extended termination bundle event structures.

We call  $E$  the *set of events*,  $\succ$  the *witness relation*,  $\mapsto$  the *causality relation*,  $T$  the *termination set* and  $l$  the *action-labelling function*. Set  $X$  is called *bundle* (with respect to  $e$ ) if  $X \in T$  or  $X \succ e$  or  $X \mapsto e$ . The attribute *extended* in the name of the class of event structures defined above is used to emphasize (as it is done for extended bundle event structures) that these event structures can model disruption.

The intuitive meaning of a witness-bundle  $Z$  of  $e$  ( $Z \succ e$ ) is that event  $e$  is still possible after a system run  $r$  if no event occurring in  $r$  is contained in  $Z$ . In other words, a system run disables an event  $e$  if all witness-bundles of  $e$  contain an element of the system run. The constraints imposed on the witness relation are: Firstly, every event  $e$  must have a witness-bundle, since otherwise  $e$  would be considered to be disabled and hence, could be omitted. Secondly, every witness-bundle of  $e$  has to contain  $e$ , since the execution of an event disables itself, i.e. every event can be executed only once.

The meaning of the causality relation  $X \mapsto e$  is: before  $e$  may be executed after a system run  $r$ , an event of  $X$  has to have occurred in  $r$ . A system run of an eTbes is terminated if all bundles in the termination set are satisfied, i.e. every element of  $T$  contains an event of the system run. The constraint  $T \neq \emptyset$  on the termination set ensures that an eTbes is not able to terminate immediately, i.e. it can only terminate by executing an action. However,  $T$  might consist of the empty set only, which indicates that no termination may occur, i.e., there are only infinite executions or deadlocking executions possible. The labelling function indicates which action is observable when an event is executed.

**Remark 2.2.** ETBES ordered by the standard order [Lan92] fails to yield a complete partial order<sup>4</sup>. The standard order can be used to obtain a complete partial order if only those eTbes are considered where the witness relation, the causality relation and the termination set satisfy an additional constraint [FMCW02]. Since it is not essential in this paper to obtain a complete partial order, we omit this constraint in order to enhance the readability (the additional constraint does not harm our theory).

**Example 2.3.** Some eTbes are shown in Figure 1. Here, the events are depicted as dots and their corresponding actions appear next to the dots (we do not name the events explicitly and we identify them with the actions if no confusion arises). The witness relation is illustrated by wavy lines<sup>5</sup>. More precisely, a witness-bundle  $Z \succ e$  is depicted by a

<sup>4</sup> The complete partial order theory is used to give meanings to recursive processes.

<sup>5</sup> Note that we use wavy lines rather than dashed lines in order to avoid confusion, since dashed lines are already used in event structures to model disablement whereas we model non-disablement.

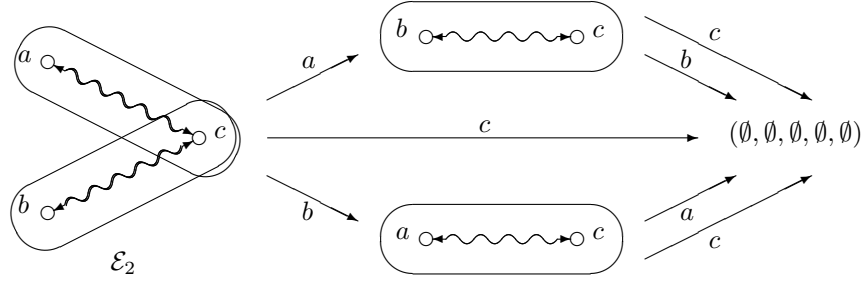


Fig. 2. Event Execution Derived from ETBES

wavy arrow from the elements of  $Z \setminus \{e\}$  to  $e$ . Furthermore, witness-bundle  $\{e\} \succ e$  is omitted in the illustrations, if it is the only witness-bundle for  $e$ . For example, the witnesses  $\{a\} \succ a$ ,  $\{b\} \succ b$  and  $\{c\} \succ c$  exist in  $\mathcal{E}_1$ . Sometimes, the same wavy lines are used in different witness-bundles, for example the witness-bundles in  $\mathcal{E}_3$  are  $\{a, b, c\} \succ a$ ,  $\{a, b, c\} \succ b$  and  $\{a, b, c\} \succ c$ . The causality relation is depicted by straight lines. A bundle  $X \in T$  is displayed by surrounding its events by a closed line.

$\mathcal{E}_1$  describe a process where  $a$  and  $b$  can occur ‘in parallel’ and when both have happened  $c$  may happen, which leads to termination of that process.  $\mathcal{E}_2$  describe a process where  $a$  and  $b$  may occur ‘in parallel’ but may be disrupted by  $c$ . Termination happens if either both actions  $a$  and  $b$  occurred or  $c$  occurred.  $\mathcal{E}_3$  describes a process where exactly one of the action  $a$ ,  $b$ , or  $c$  may occur, which also leads to termination.

Hereafter, we consider  $\mathcal{E}$  to be  $(E, \succ, \mapsto, T, l)$ ,  $\mathcal{E}_i$  to be  $(E_i, \succ_i, \mapsto_i, T_i, l_i)$  and in general  $\mathcal{E}$  to be  $(E_{\mathcal{E}}, \succ_{\mathcal{E}}, \mapsto_{\mathcal{E}}, T_{\mathcal{E}}, l_{\mathcal{E}})$ . Furthermore,  $\text{init}(\mathcal{E})$  denotes the set of events which are ready to execute and  $\Upsilon(T, e)$  holds if and only if  $e$  is a *termination event* with respect to  $T$ , i.e.  $\mathcal{E}$  terminates by executing  $e$ . Formally:

**Definition 2.4.** Let  $\mathcal{E}$  be an eTbes. Then the set of *initial events* of  $\mathcal{E}$ , denoted by  $\text{init}(\mathcal{E})$ , and the *termination predicate*  $\Upsilon \subseteq \mathcal{P}(\mathcal{P}(\mathcal{U})) \times \mathcal{U}$  are defined by

$$\text{init}(\mathcal{E}) = \{e \in E \mid \neg(\exists X : X \mapsto e)\} \quad \Upsilon(T, e) \iff \forall X \in T : e \in X.$$

## 2.1. Event Execution in an eTbes.

The remainder of an eTbes with respect to event  $e$  describes the eTbes after the execution of  $e$ . Therefore, we remove all events which are disabled by  $e$ , i.e. we only keep those events that have a witness-bundle which does not contain  $e$ . Please remember that an event has to be an element of all its witness-bundles. Hence, it disables itself. Furthermore, all bundles (from causality, witness or termination) that contain  $e$  are removed, since the execution of  $e$  fulfills the requirements specified by those bundles, i.e. these bundles contain an element of the system run. Formally:

**Definition 2.5 (Remainder of an eTbes).** Let  $\mathcal{E} \in \text{ETBES}$  and  $e \in \text{init}(\mathcal{E})$ . Then the *remainder*  $\mathcal{E}_{[e]}$  of  $\mathcal{E}$  is given by  $(E', \succ', \mapsto', T', l')$  where

$$\begin{aligned} E' &= \{e' \in E \mid \exists Z : Z \succ e' \wedge e \notin Z\} \\ \succ' &= \{(Z \cap E', e') \mid e' \in E' \wedge Z \succ e' \wedge e \notin Z\} \\ \mapsto' &= \{(X \cap E', e') \mid e' \in E' \wedge X \mapsto e' \wedge e \notin X\} \\ T' &= \{X \cap E' \mid X \in T \wedge e \notin X\} \\ l' &= l \upharpoonright E' \end{aligned}$$

**Remark 2.6.** Note that the remainder  $\mathcal{E}_{[e]}$  of an eTbes  $\mathcal{E}$  is again an eTbes if and only if  $e$  is not a termination event, i.e.  $\neg\Upsilon(T, e)$  (the termination set becomes the empty set if  $e$  is a termination event). Hence, after termination no further event execution is possible.

The event execution of an eTbes is later used to define equivalence notions on eTbes. The event execution obtained from  $\mathcal{E}_2$  of Figure 1 is presented in Figure 2.

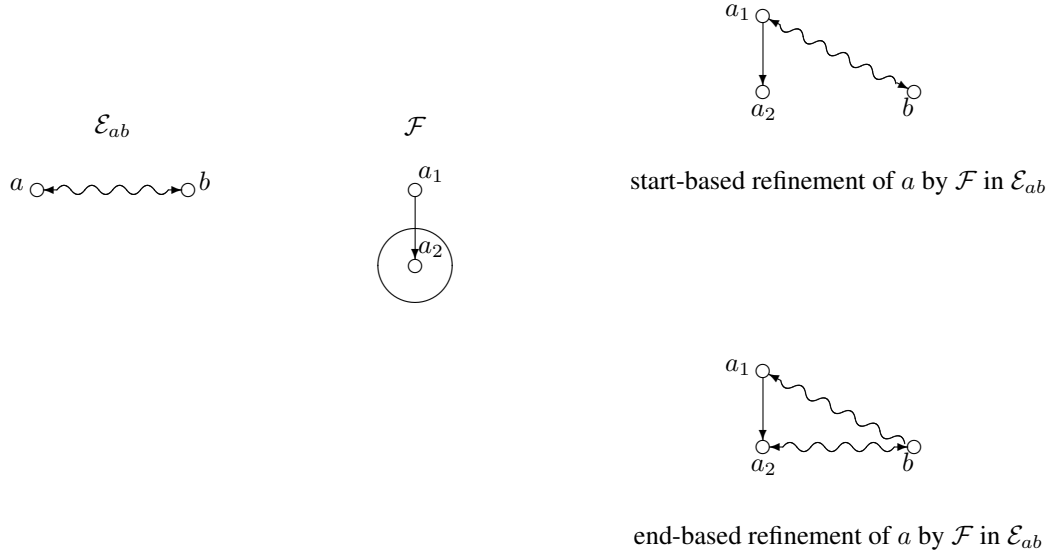


Fig. 3. Start-Based versus End-Based Refinement

### 3. An End-Based Refinement Operator on ETBES

Termination bundle event structures are appropriate to define an end-based refinement operator. This refinement operator differs from the classical definition with respect to the conflict relation: Only the termination of the refining processes is used in our approach to define the disabling whereas every event (or every initial event) is used in the standard (start-based) approach. More precisely, in the classical definition we have: If  $e$  is in conflict with  $e'$  and  $e$  ( $e'$  respectively) is refined to  $\mathcal{E}_e$  ( $\mathcal{E}_{e'}$  respectively), then every (initial) event of  $\mathcal{E}_e$  is placed in conflict with every (initial) event of  $\mathcal{E}_{e'}$  and vice versa. In our definition, we use every termination bundle of  $\mathcal{E}_e$  as a witness bundle for every event of  $\mathcal{E}_{e'}$ , i.e. the events of  $\mathcal{E}_{e'}$  may only be executed if they are executed before the termination of  $\mathcal{E}_e$ . And, of course, we use every termination bundle of  $\mathcal{E}_{e'}$  as a witness bundle for every event of  $\mathcal{E}_e$ . By this approach, we guarantee that a choice is triggered by the termination of the refining process.

The difference between the start-based and the end-based refinement operator is illustrated in Figure 3, where we depict extended termination bundle event structures after the refinement in the start-based and in the end-based approaches.  $\mathcal{E}_{ab}$  models the alternative executions of  $a$  or  $b$  without terminating the process.  $\mathcal{F}$  models the sequential executions of  $a_1$  followed by  $a_2$  with termination. It can be seen that  $a_1$  is in conflict with  $b$  in the start-based approach after the refinement, whereas it is possible that  $a_1$  precedes  $b$  in the end-based approach. Thus the sequence  $a_1, b$  is only a trace of the event structure corresponding to the end-based view.

First, we present an end-based refinement operator where every event is refined by a (possible different) event structure. This definition is later used to define a refinement operator where actions rather than events are refined. The witness relation of the end-based refinement operator is determined by taking a witness  $Z$  of  $e$  from the process that is refined. Then for every element of  $Z$  different from  $e$  we take a termination set of the corresponding refining process and for  $e$  we take a witness of the corresponding refining process. Furthermore, also a termination set of the corresponding refining process of  $e$  is taken in order to avoid that a refining event remains possible after the termination of the refining process. A witness of the refined process is obtained as the union of the chosen sets. A causality-bundle of the refined system is either (i) a causality-bundle of the corresponding refinement or (ii) it is a union of termination sets of the corresponding refinements obtained from a causality-bundle of the process that is refined. Furthermore, it is sufficient to restrict to the corresponding initial events in the latter case. A termination set of the refined system is the union of termination sets of the corresponding refinements obtained from a termination-bundle of the process that is refined. Formally:

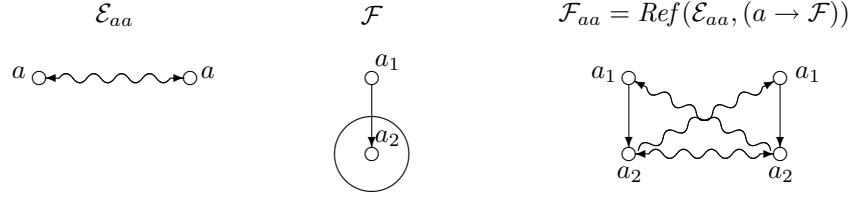


Fig. 4. End-Based Refinement in ETBES

**Definition 3.1.** The event refinement operator  $\underline{Ref} : \mathbf{ETBES} \times (\mathcal{U} \rightarrow \mathbf{ETBES}) \rightarrow \mathbf{ETBES}$  is given by

$\underline{Ref}(\mathcal{E}, \vartheta) = (\tilde{E}, \tilde{\succ}, \tilde{\mapsto}, \tilde{T}, \tilde{l})$  where

$$\begin{aligned} \tilde{E} &= \{(e, \hat{e}) \mid e \in E \wedge \hat{e} \in E_{\vartheta(e)}\} \\ \tilde{\succ} &= \{(\tilde{Z}, (e, \hat{e})) \mid \exists Z : Z \succ e \wedge \exists f : Z \rightarrow \mathcal{P}(\mathcal{U}) : \tilde{Z} = \{(e', \hat{e}') \in \tilde{E} \mid e' \in Z \wedge \hat{e}' \in f(e')\} \wedge \\ &\quad \forall e' \in Z : ((e' \neq e \wedge f(e') \in T_{\vartheta(e')}) \vee (e' = e \wedge \exists \hat{X} \in T_{\vartheta(e')}, \tilde{Z} : \tilde{Z} \succ_{\vartheta(e')} \hat{e} \wedge f(e') = \hat{Z} \cup \hat{X})\}\} \\ \tilde{\mapsto} &= \{(\{e\} \times X', (e, \hat{e})) \mid X' \mapsto_{\vartheta(e)} \hat{e}\} \cup \{(\tilde{X}, (e, \hat{e})) \mid \hat{e} \in \text{init}(\vartheta(e)) \wedge \exists X : X \mapsto e \wedge \exists f : X \rightarrow \mathcal{P}(\mathcal{U}) : \\ &\quad \tilde{X} = \{(e', \hat{e}') \in \tilde{E} \mid e' \in X \wedge \hat{e}' \in f(e')\} \wedge \forall e' \in X : f(e') \in T_{\vartheta(e')}\} \\ \tilde{T} &= \{\tilde{X} \mid \exists X \in T \wedge \exists f : X \rightarrow \mathcal{P}(\mathcal{U}) : \tilde{X} = \{(e, \hat{e}) \in \tilde{E} \mid e \in X \wedge \hat{e} \in f(e)\} \wedge \forall e \in X : f(e) \in T_{\vartheta(e)}\} \\ \tilde{l}(e, \hat{e}) &= l_{\vartheta(e)}(\hat{e}) \end{aligned}$$

It is easily checked that the event refinement operator is well defined, i.e. it really yields an element of **ETBES**.

Suppose  $A \subseteq \mathcal{Act}$ . The action refinement operator  $Ref_A : \mathbf{ETBES} \times (A \rightarrow \mathbf{ETBES}) \rightarrow \mathbf{ETBES}$  is derived from the event refinement operator by

$$Ref_A(\mathcal{E}, \theta) = \underline{Ref}(\mathcal{E}, \vartheta) \text{ where } \vartheta(e) = \begin{cases} \theta(l(e)) & \text{if } e \in E \wedge l(e) \in A \\ (\{\bullet\}, \{(\{\bullet\}, \bullet)\}, \emptyset, \{\bullet\}, \{(\bullet, l(e))\}) & \text{if } e \in E \wedge l(e) \notin A \\ (\emptyset, \emptyset, \emptyset, \{\emptyset\}, \emptyset) & \text{otherwise} \end{cases}$$

We omit index  $A$  in  $Ref_A$ , if  $A$  is clear from the context. An example that illustrates how the refinement operator  $Ref$  behaves is given in Figure 4. There  $\mathcal{E}_{aa}$  denotes the alternative between two events that have the same action label  $a$ . Note that in the refined system  $\mathcal{F}_{aa}$  both events labelled with  $a_1$  may occur in a single system run. Furthermore, if an event  $e'$  is a necessary causality of  $e$ , i.e.  $e$  can only be executed after the execution of  $e'$ , we sometimes omit  $e$  in the witness bundles of  $e'$ , since it has no consequence for the behavior. For example, in  $\mathcal{F}_{aa}$  of Figure 4, the witness bundles to events labelled with  $a_1$  have to contain both events labelled with  $a_2$ .

**Remark 3.2.** Our refinement operator allows the modelling of a disrupt mechanism (if a choice mechanism exists) as it is used, for example, in LOTOS [BB87], which is illustrated by the following example: Suppose we want to model a process  $G$  where process  $\mathcal{F}$  from Figure 3 can be interrupted by the process  $\mathcal{E}_b$  consisting of the single event labelled with action  $b$ . This process is obtained by taking a choice between  $\mathcal{E}_b$  and the process consisting of the single event labelled with the fresh action name  $a$  and having  $\{a\}$  as the only termination bundle. This yields  $\mathcal{E}_{ab}$  from Figure 3 with the additional termination set  $\{a\}$ . Refining in this eTbes action  $a$  by  $\mathcal{F}$  with the end-based refinement operator yields the eTbes that describes process  $G$ , where  $\mathcal{F}$  can be interrupted by  $\mathcal{E}_b$ .

## 4. Equivalences for ETBES

First, we give a short overview over some standard equivalence notions and argue that they are not reasonable for an end-based action refinement operator, since they are not preserved by the end-based refinement operator. Then we introduce two new equivalences that are preserved by the end-based refinement operator. Moreover, each of them is the coarsest equivalence with respect to a standard equivalence notion (trace and bisimulation equivalence). These new equivalences are also examined with respect to their discriminating power. Before we continue, we present a formal definition of coarsest congruence for an operator with respect to an equivalence.

**Definition 4.1.** Suppose  $M$  is a set and  $\sim_{equiv}$  is an equivalence relation on  $M$ . Furthermore, suppose  $Op$  is an operator from a (possible infinite) tuple of elements from  $M$  into  $M$ . Then an equivalence relation  $\sim_{co}$  is a *congruence* for  $Op$  if  $\sim_{co}$  is preserved by  $Op$ , i.e., for all tuples  $(m_0, m_1, \dots), (m'_0, m'_1, \dots)$  such that  $m_i \sim_{co} m'_i$  for  $i \in \mathbb{N}$ , we

have  $\text{Op}(m_0, m_1, \dots) \sim_{\text{coco}} \text{Op}(m'_0, m'_1, \dots)$ . An equivalence relation  $\sim_{\text{coco}}$  is a *coarsest congruence for Op with respect to  $\sim_{\text{equiv}}$*  if

- $\sim_{\text{coco}}$  is an equivalence relation on  $M$  contained in  $\sim_{\text{equiv}}$ ,
- $\sim_{\text{coco}}$  is a congruence for Op, and
- every congruence for Op that is contained in  $\sim_{\text{equiv}}$  is also contained in  $\sim_{\text{coco}}$ .

#### 4.1. Standard Equivalence Notions

*Trace equivalence* for **ETBES** is defined as follows, where we distinguish between non-terminating and terminating traces. We take as traces the action label sequence obtained from an event sequence of a possible execution based on the remainder definition.

**Definition 4.2 (Trace Equivalence).** Let  $\mathcal{E} \in \mathbf{ETBES}$ . Then the *action traces* of  $\mathcal{E}$  are defined by

$$\text{Tr}(\mathcal{E}) = \{(l(e_i))_{i \leq n} \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \dots, \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1} \wedge \neg \Upsilon(\mathcal{E}_i, e_i)\} \cup \{(l(e_i))_{i \leq n} \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \dots, \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1} \wedge \Upsilon(\mathcal{E}_n, e_n)\}.$$

Two  $\mathcal{E}, \mathcal{E}' \in \mathbf{ETBES}$  are *trace equivalent*, denoted by  $\mathcal{E} \sim_t \mathcal{E}'$ , if  $\text{Tr}(\mathcal{E}) = \text{Tr}(\mathcal{E}')$ .

For *strong bisimulation equivalences* each possible execution of each side has to be matched by the other process leading again to bisimilar processes. Furthermore, a non-terminating execution has to be matched by a non-termination one and a terminating one by a terminating one.

**Definition 4.3 (Strong Bisimulation Equivalence).** A *bisimulation*  $\mathcal{R}$  is a subset of  $\mathbf{ETBES} \times \mathbf{ETBES}$  such that whenever  $(\mathcal{E}_1, \mathcal{E}_2) \in \mathcal{R}$ , then

- $e_1 \in \text{init}(\mathcal{E}_1)$  implies that there is  $e_2$  such that  $l_1(e_1) = l_2(e_2)$  and  $\Upsilon(T_1, e_1) \Leftrightarrow \Upsilon(T_2, e_2)$  and  $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}) \in \mathcal{R}$
- $e_2 \in \text{init}(\mathcal{E}_2)$  implies that there is  $e_1$  such that  $l_1(e_1) = l_2(e_2)$  and  $\Upsilon(T_1, e_1) \Leftrightarrow \Upsilon(T_2, e_2)$  and  $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}) \in \mathcal{R}$

We say that  $\mathcal{E}_1, \mathcal{E}_2$  are *strong bisimilar* (or *strong bisimulation equivalent*), denoted by  $\mathcal{E}_1 \sim_b \mathcal{E}_2$ , if and only if there is a bisimulation  $\mathcal{R}$  such that  $(\mathcal{E}_1, \mathcal{E}_2) \in \mathcal{R}$ .

Equivalences that lie between the trace and the bisimulation equivalence are given in [Gla01, VD98].

ST semantics, originally defined in [GV87], turns out to be the coarsest congruence for action refinement in the start-based setting [AH94, Gla90, GL95, Vog93]. In the ST-approach, actions are not considered to be atomic, as in standard interleaving semantics. Instead, the execution of an action is split into the two distinguished events of the start and the ending (termination) of an action, where the ending is uniquely related to its start.

Further true concurrency equivalences are:

**Step equivalence:** Here, a finite multiset of actions, i.e. a finite set of events, may be executed in one step, as opposed to the interleaving approach, where only single actions may be executed. In [Pom86] trace and bisimulation versions of this equivalence have been proposed.

**Pomset equivalence:** Here a finite, labelled and partially ordered set of events, more precisely a pomset [Pra86], may be executed. Pomsets are equivalence classes with respect to the action labelling and the order, i.e. two labelled and partial ordered sets are equivalent if there exists a label and order preserving bijection between the sets. The order of a pomset corresponds to the causality order of the events. In [BC87] trace and bisimulation versions of this equivalence have been proposed.

**History preserving equivalence:** Here, the causal order in which events have been executed is additionally taken into account. There are different versions depending on to what degree the past information is taken into account. There exists *weak*, *normal* and *hereditary history preserving bisimulations* [Bed91, DNM87, GG01].

[GG01] examines which equivalence notions are preserved by a start-based action refinement operator in a configuration structure setting. There it is shown that pomset trace equivalence, history preserving bisimulation and hereditary history preserving bisimulation are preserved by a start-based action refinement operator. As mentioned before, the ST-equivalence is also preserved by a start-based action refinement. All other equivalences of this subsection are not preserved by a start-based action refinement.

The action refinement operator in an end-based setting is not compatible with the equivalence notions mentioned above. This can be seen as follows: In the case of the end-based refinement operator (*Ref*) any equivalence that implies trace equivalence (Definition 4.2) and that identifies  $\mathcal{E}_a$  from Figure 5 and  $\mathcal{E}_{aa}$  from Figure 4 (like all the equivalences mentioned do) is not preserved. This is the case, because  $\mathcal{F}_a$  from Figure 5, which refines  $a$  in  $\mathcal{E}_a$  by  $\mathcal{F}$ , and  $\mathcal{F}_{aa}$  from



Fig. 5. End-Based Refinement in ETBES (2)

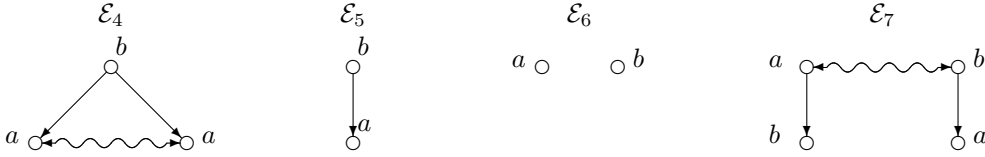


Fig. 6. Some Further Extended Termination Bundle Event Structures

Figure 4, which refines  $a$  in  $\mathcal{E}_{aa}$  by  $\mathcal{F}$ , are not trace equivalent. Resource bisimulation [CDL99, CNL99], which is only defined on process algebra terms, is the only equivalence known to us that does not identify the process algebra terms corresponding to  $\mathcal{E}_a$  and  $\mathcal{E}_{aa}$ , which are  $a$ , respectively  $a + a$ , by using the notation from the introduction.

In the following subsections, we present new equivalences which are indeed congruences for our refinement operator. For simplicity, we introduce the following definition, which determines the initial events of an event structure with respect to their labels.

**Definition 4.4.** Define  $\text{init}_A(\mathcal{E}) = \{e \in \text{init}(\mathcal{E}) \mid l(e) \in A\}$ , where  $A \subseteq \text{Act}$ . Furthermore, we write  $\text{init}_a(\mathcal{E})$  as a short hand for  $\text{init}_{\{a\}}(\mathcal{E})$ .

## 4.2. ICT-Equivalence on ETBES

The first considered equivalence notion is derived from trace equivalence. An equivalence notion which is a congruence for the end-based refinement operator has to distinguish between  $\mathcal{E}_{aa}$  from Figure 4 and  $\mathcal{E}_a$  from Figure 5. One way to achieve this is to guarantee that the number of the initial events which are labelled by the same action have to be equal, i.e.  $\mathcal{E}$  and  $\mathcal{E}'$  can only be equivalent if  $|\text{init}_a(\mathcal{E})| = |\text{init}_a(\mathcal{E}')|$  for all  $a \in \text{Obs}$ . Moreover, we also have to guarantee a relationship between the numbers of the initial events with the same label of the corresponding remainders of the event structures. For example, consider  $\mathcal{E}_4$  and  $\mathcal{E}_5$  from Figure 6, where two alternative  $a$  actions may be executed after  $b$  in  $\mathcal{E}_4$  and no alternative exists after the execution of  $b$  in  $\mathcal{E}_5$ . Then  $(b, a_1, a_1)$  is a possible trace of  $\text{Ref}(\mathcal{E}_4, (a \rightarrow \mathcal{F}))$  but not of  $\text{Ref}(\mathcal{E}_5, (a \rightarrow \mathcal{F}))$ . Hence,  $\text{Ref}(\mathcal{E}_4, (a \rightarrow \mathcal{F}))$  and  $\text{Ref}(\mathcal{E}_5, (a \rightarrow \mathcal{F}))$  are not trace equivalent.

Further difficulties become evident by a closer look at  $\mathcal{E}_6$  and  $\mathcal{E}_7$  from Figure 6, where  $\mathcal{E}_6$  models the parallel execution of  $a$  and  $b$  and  $\mathcal{E}_7$  models the alternative between ( $a$  followed by  $b$ ) and ( $b$  followed by  $a$ ):  $\mathcal{E}_6$  and  $\mathcal{E}_7$  satisfy our above criterion, but not of  $\text{Ref}(\mathcal{E}_7, (a \rightarrow \mathcal{F}))$  and  $(a_1, b, a_1)$  is not a possible trace of  $\text{Ref}(\mathcal{E}_6, (a \rightarrow \mathcal{F}))$ , i.e. our tentative relation is not a congruence for the refinement.

Therefore, we introduce the *initial event traces* of an eTbes. They consist of an event execution sequence and of a finite subset of the initial events for every execution step (these events can be considered as those events that are partly executed during the step). Two eTbes,  $\mathcal{E}_1$  and  $\mathcal{E}_2$ , are considered to be equivalent if every initial event trace of  $\mathcal{E}_1$  can be mapped by an injective function  $f$  to an initial event trace of  $\mathcal{E}_2$  and vice versa. Furthermore, this function has to be *label preserving*, i.e.  $\forall e_1 \in E_1 : l_1(e_1) = l_2(f(e_1))$ . The equivalence is precisely given by the following definition, where  $\mathcal{E}_{[e]}$  is defined in Definition 2.5.

**Definition 4.5 (ICT-equivalence).** Let  $\mathcal{E} \in \text{ETBES}$ . Then the *initial event traces* of  $\mathcal{E}$  are defined by  $\text{Tr}^{ic}(\mathcal{E}) =$

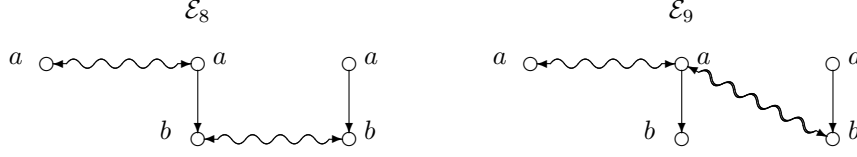


Fig. 7. Non ICT-Equivalent eTbes



Fig. 8. ICT-Equivalent eTbes

$\{(e_i, \gamma_i)_{i \leq n} \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \dots, \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1} \wedge \gamma_i \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\mathcal{E}_i)) \wedge \neg \Upsilon(\mathcal{E}_n, e_n)\} \cup$   
 $\{(e_i, \gamma_i)_{i \leq n} \vee \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \dots, \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1} \wedge \gamma_i \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\mathcal{E}_i)) \wedge \Upsilon(\mathcal{E}_n, e_n)\}.$

Two  $\mathcal{E}, \mathcal{E}' \in \mathbf{ETBES}$  are *initial corresponding trace equivalent* (ICT-equivalent), denoted by  $\mathcal{E} \sim_{ICT} \mathcal{E}'$ , if

- for every  $(e_i, \gamma_i)_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E})$  there is an injective, label preserving function  $f : (\bigcup_{i \leq n} (\gamma_i \cup \{e_i\})) \rightarrow E'$  such that  $(f(e_i), f(\gamma_i))_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E}')$  (analogously for  $(e_i, \gamma_i)_{i \leq n} \vee \in \text{Tr}^{ic}(\mathcal{E})$ ) and
- for every  $(e'_i, \gamma'_i)_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E}')$  there is an injective, label preserving function  $f' : (\bigcup_{i \leq n} (\gamma'_i \cup \{e'_i\})) \rightarrow E$  such that  $(f'(e'_i), f'(\gamma'_i))_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E})$  (analogously for  $(e'_i, \gamma'_i)_{i \leq n} \vee \in \text{Tr}^{ic}(\mathcal{E}')$ )

$\mathcal{E}_4$  and  $\mathcal{E}_5$ , and also  $\mathcal{E}_6$  and  $\mathcal{E}_7$  from Figure 6 are not ICT-equivalent. In addition, the event structures from Figure 7 are not ICT-equivalent either. This holds, since in  $\mathcal{E}_8$  it is possible that both events labelled by  $b$  become enabled, which is not the case for  $\mathcal{E}_9$ . Two ICT-equivalent event structures are shown in Figure 8, where the left one models the alternative of a single  $a$  execution and an  $a$  execution followed by  $b$  and where the right one models the alternative of the two identical processes where  $a$  is followed by  $b$ .

**Proposition 4.6.** Two ICT-equivalent eTbes are also trace equivalent, i.e.  $\sim_{ICT} \subseteq \sim_t$ .

*Proof.* Obvious.  $\square$

**Theorem 4.7.** ICT-equivalence is a congruence for the refinement operator  $Ref$ , i.e. for any  $A \subseteq \text{Obs}$  and  $\theta : A \rightarrow \mathbf{ETBES}$  we have  $\mathcal{E} \sim_{ICT} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{ICT} \theta'(a)$  implies  $Ref(\mathcal{E}, \theta) \sim_{ICT} Ref(\mathcal{E}', \theta')$ .

*Proof.* The proof is given in Appendix A.1.  $\square$

**Remark 4.8.** The ICT-equivalence is also a congruence for the standard operators on event structures, like the choice, the sequential and the parallel operator.

**Theorem 4.9.** ICT-equivalence is the coarsest congruence for  $Ref$  with respect to trace equivalence, i.e.  $\sim_{ICT} \subseteq \sim_t$  and  $\sim_{ICT}$  is a congruence for  $Ref$  and for every equivalence  $\equiv \subseteq \sim_t$  that is a congruence for  $Ref$  we have  $\equiv \subseteq \sim_{ICT}$ . Moreover, if  $\forall A \subseteq \text{Obs}, \theta : A \rightarrow \mathbf{ETBES} : Ref(\mathcal{E}, \theta) \sim_t Ref(\mathcal{E}', \theta)$  then  $\mathcal{E} \sim_{ICT} \mathcal{E}'$ .

*Proof.* The proof is given in Appendix A.1.  $\square$

We are also interested in a congruence which implies strong bisimilarity (Definition 4.3). ICT-equivalence does not yield such an equivalence.

**Lemma 4.10.** Strong bisimilarity does not follow from ICT-equivalence.



Fig. 9. FIC-Equivalent eTbes (1)

*Proof.* The eTbes from Figure 8 are not bisimilar but ICT-equivalent.  $\square$

### 4.3. FIC-Equivalence on ETBES

An equivalence that is derived from bisimulation equivalence and that is a congruence for the end-based refinement operator has to relate the initial events, as it is done by the ICT-equivalence. Therefore, we extend the definition of a bisimulation relation by a third component which denotes a (label preserving) bijection between the initial events. This bijection has to be maintained (on the remaining initial events) after event executions. More precisely, it is sufficient that the bijection is maintained for any finite subset of the initial events, since only a finite number of events can be active (started and not finished). This is formalized by the following definition.

**Definition 4.11 (FIC-equivalence).** A *finite-initial corresponding bisimulation* (FIC-bisimulation)<sup>6</sup>  $\mathcal{R}$  is a subset of  $\text{ETBES} \times \text{ETBES} \times (\mathcal{U} \rightarrow \mathcal{U})$  such that whenever  $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$ , then

- $\text{dom}(f) = \text{init}_{\text{Obs}}(\mathcal{E}_1)$ ,
- $f$  is a labelling preserving isomorphism between  $\text{init}_{\text{Obs}}(\mathcal{E}_1)$  and  $\text{init}_{\text{Obs}}(\mathcal{E}_2)$
- $e_1 \in \text{init}(\mathcal{E}_1) \wedge I \in \mathcal{P}_{\text{fin}}(\text{init}_{\text{Obs}}(\mathcal{E}_1))$  implies that there exist  $e_2$  and  $f'$  such that  $l_1(e_1) = l_2(e_2)$  and  $\Upsilon(T_1, e_1) \Leftrightarrow \Upsilon(T_2, e_2)$  and  $l_1(e_1) \in \text{Obs} \Rightarrow e_2 = f(e_1)$  and  $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}, f') \in \mathcal{R}$  and  $f \upharpoonright (I \cap \text{init}_{\text{Obs}}(\mathcal{E}_{1[e_1]})) = f' \upharpoonright I$  and  $f^{-1} \upharpoonright (f(I) \cap \text{init}_{\text{Obs}}(\mathcal{E}_{2[e_2]})) = f'^{-1} \upharpoonright f(I)$
- $e_2 \in \text{init}(\mathcal{E}_2) \wedge I \in \mathcal{P}_{\text{fin}}(\text{init}_{\text{Obs}}(\mathcal{E}_2))$  implies that there exist  $e_1$  and  $f'$  such that  $l_1(e_1) = l_2(e_2)$  and  $\Upsilon(T_1, e_1) \Leftrightarrow \Upsilon(T_2, e_2)$  and  $l_1(e_1) \in \text{Obs} \Rightarrow e_2 = f(e_1)$  and  $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}, f') \in \mathcal{R}$  and  $f \upharpoonright (I \cap \text{init}_{\text{Obs}}(\mathcal{E}_{1[e_1]})) = f' \upharpoonright I$  and  $f^{-1} \upharpoonright (f(I) \cap \text{init}_{\text{Obs}}(\mathcal{E}_{2[e_2]})) = f'^{-1} \upharpoonright f(I)$

We say that  $\mathcal{E}_1, \mathcal{E}_2$  are *FIC-bisimilar* (or *FIC-equivalent*), denoted by  $\mathcal{E}_1 \sim_{\text{FIC}} \mathcal{E}_2$ , if and only if there is a FIC-bisimulation  $\mathcal{R}$  and an  $f : \mathcal{U} \rightarrow \mathcal{U}$  such that  $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$ .

The eTbes from Figure 8 are not FIC-equivalent, whereas the eTbes from Figure 9 are FIC-equivalent. The left eTbes from Figure 9 models the alternative between  $a$  and  $c$  and whenever one of them is completely executed a unique event labelled with  $b$  may be executed. The left eTbes from Figure 9 is similar to the right one except that two different events labelled with  $b$  may be executed depending if  $a$  or  $c$  is completely executed. The eTbes from Figure 10 are also FIC-equivalent. The eTbes depicted there models the possible execution of arbitrary  $a$  actions in parallel and they only differ in the case that one  $a$  event is dependent on the complete execution of another one in the eTbes depicted on the right hand side.

**Proposition 4.12.** Two FIC-equivalent eTbes are also strong bisimulation equivalent, i.e.  $\sim_{\text{FIC}} \subset \sim_b$ .

*Proof.* Follows from the fact that ignoring the third component of a FIC-bisimulation yields a bisimulation.  $\square$

**Theorem 4.13.** FIC-equivalence is a congruence for the refinement operator  $\text{Ref}$ , i.e. for any  $A \subseteq \text{Obs}$  and  $\theta : A \rightarrow \text{ETBES}$  we have  $\mathcal{E} \sim_{\text{FIC}} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{\text{FIC}} \theta'(a)$  implies  $\text{Ref}(\mathcal{E}, \theta) \sim_{\text{FIC}} \text{Ref}(\mathcal{E}', \theta')$ .

*Proof.* The proof is given in Appendix A.2.  $\square$

<sup>6</sup> The finite-initial corresponding (FIC) bisimulation is called finite unique initial (FUI) bisimulation in [FMC02].



Fig. 10. FIC-Equivalent eTbes (2)

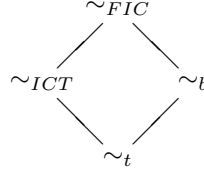


Fig. 11. Relations Between the Equivalences

**Remark 4.14.** The FIC-equivalence is also a congruence for the standard operator on event structure, like the choice, the sequential and the parallel operator.

**Theorem 4.15.** FIC-equivalence is the coarsest congruence for  $Ref$  with respect to bisimulation equivalence, i.e.  $\sim_{FIC} \subseteq \sim_b$  and  $\sim_{FIC}$  is a congruence for  $Ref$  and for every equivalence  $\equiv \subseteq \sim_b$  that is a congruence for  $Ref$  we have  $\equiv \subseteq \sim_{FIC}$ .

Moreover, if  $\forall A \subseteq \text{Obs}, \theta : A \rightarrow \mathbf{ETBES} : Ref(\mathcal{E}, \theta) \sim_b Ref(\mathcal{E}', \theta)$  then  $\mathcal{E} \sim_{FIC} \mathcal{E}'$ .

*Proof.* The proof is given in Appendix A.2.  $\square$

**Proposition 4.16.** Two FIC-equivalent eTbes are also ICT-equivalent, i.e.  $\sim_{FIC} \subseteq \sim_{ICT}$ .

*Proof.* Suppose  $\mathcal{E} \sim_{FIC} \mathcal{E}'$ , then by Theorem 4.13 we have  $\forall \theta : Ref(\mathcal{E}, \theta) \sim_{FIC} Ref(\mathcal{E}', \theta)$ . From the inclusions  $\sim_{FIC} \subseteq \sim_b \subseteq \sim_t$  we obtain  $\forall \theta : Ref(\mathcal{E}, \theta) \sim_t Ref(\mathcal{E}', \theta)$ . Thus by Theorem 4.9 it follows that  $\mathcal{E} \sim_{ICT} \mathcal{E}'$ .  $\square$

## 5. Conclusion

We considered the view that a choice is triggered by the ending of actions. In particular, we investigated an action refinement operator with respect to the end-based choice view. This operator is defined on a new class of event structures called extended termination bundle event structures. Furthermore, we investigated two new equivalences, called ICT- and FIC-equivalence, since non of the standard equivalences is preserved by the end-based refinement operator. In particular, a process consisting of a single action  $a$  is distinguished from the process consisting of the alternative of two events labelled with  $a$  (expressed as process algebra terms:  $a$  and  $a + a$  are distinguished). We showed that ICT-equivalence is the coarsest congruence for the end-based refinement operator with respect to trace equivalence and FIC-equivalence is the coarsest congruence for the end-based refinement operator with respect to bisimulation equivalence. The relation between these equivalences is summarized in Figure 11: If two equivalences are connected via a line, then the lower one identifies strictly more elements than the upper one.

The comparison of ICT- and FIC-equivalence with resource bisimulation [CDL99, CNL99], which also distinguishes  $a$  and  $a + a$ , is not straightforward, since resource bisimulation is only defined on process algebra terms and the adaption to event structures is not straightforward. In particular, the process algebra expressions on which the resource bisimulation is defined do not contain parallel composition and it is unclear how resource bisimulation can be extended to handle parallel composition adequately. Nevertheless, if our equivalence is interpreted on such simple process algebra terms we obtain that FIC-equivalence is contained in resource bisimulation but not vice versa. The latter point follows from the fact that the process  $a$  followed by a deadlock will be identified with the deadlock-process via the resource bisimulation but not via ICT- and FIC-equivalence. The inclusion of FIC-equivalence in the resource

bisimulation equivalence follows from the fact that the equations that completely characterize FIC-equivalence on such a kind of simple process algebra are also valid for the resource bisimulation equivalence. ICT-equivalence is incomparable with resource bisimulation equivalence, since (i) the process  $a$  followed by a deadlock will be identified with the deadlock process via resource bisimulation and (ii) the process modelling the alternative between a single  $a$  execution and an  $a$  execution followed by  $b$  (denoted as process algebra term by  $a + a; b$ ) is distinguished by resource bisimulation from the process modelling the alternative of the two identical processes where  $a$  is followed by  $b$  (denoted as process algebra term by  $a; b + a; b$ ).

## References

- [AH94] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Information and Computation*, 115:179–247, 1994.
- [AM96] Luca Aceto and David Murphy. Timing and causality in process algebra. *Acta Informatica*, 33:317–350, 1996.
- [AMCB84] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.
- [BB87] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.
- [BC87] Gérard Boudol and Ilaria Castellani. On the semantics of concurrency: Partial orders and transition systems. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *TAPSOFT (Volume 1)*, volume 249 of *LNCS*, pages 123–137. Springer-Verlag, 1987.
- [BC89] Gérard Boudol and Ilaria Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In de Bakker et al. [dBdRR89], pages 411–427.
- [BC94] Gérard Boudol and Ilaria Castellani. Flow models of distributed computations: Three equivalent semantics for CCS. *Information and Computation*, 114:247–314, 1994.
- [Bed91] Marek A. Bednarczyk. Hereditary history preserving bisimulation or what is the power of the future perfect in program logics. Technical report, Institute of Computer Science, Polish Academy of Science, 1991.
- [BFP01] Jan A. Bergstra, Wan Fokkink, and Alban Ponse. Process algebra with recursive operations. In Bergstra et al. [BPS01], pages 333–389.
- [BG98] Marco Bernardo and Roberto Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.
- [BPS01] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, 2001.
- [CDL99] Flavio Corradini, Rocco De Nicola, and Anna Labella. Graded modalities and resource bisimulation. In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *FSTTCS*, volume 1738 of *LNCS*, pages 381–393. Springer-Verlag, 1999.
- [CNL99] Flavio Corradini, Rocco De Nicola, and Anna Labella. Models of nondeterministic regular expressions. *Journal of Computer and System Sciences*, 59:412–449, 1999.
- [dBdRR89] J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors. *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*. Springer-Verlag, 1989.
- [DD93] Philippe Darondeau and Pierpaolo Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 118:21–48, 1993.
- [DNM87] Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. Observational equivalences for concurrency models. In M. Wirsing, editor, *Formal Description of Programming Concepts – III, Proceedings of the 3<sup>th</sup> IFIP WG 2.2 working conference*, Ebberup 1986, pages 105–129. North-Holland, 1987.
- [FMC02] Harald Fecher and Mila Majster-Cederbaum. Taking decisions late: End-based choice combined with action refinement. In John Derrick, Eerke Boiten, Jim Woodcock, and Joakim von Wright, editors, *REFINE 2002*, volume 70 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.
- [FMCW02] Harald Fecher, Mila Majster-Cederbaum, and Jinzhao Wu. Bundle event structures: A revised cpo approach. *Information Processing Letters*, 83:7–12, 2002.
- [GG01] Rob van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37:229–327, 2001.
- [GL95] Roberto Gorrieri and Cosimo Laneve. Split and ST bisimulation semantics. *Information and Computation*, 118:272–288, 1995.
- [Gla90] Rob van Glabbeek. The refinement theorem for ST-bisimulation semantics. In M. Broy and C.B. Jones, editors, *Proceedings IFIP TC2 Working Conference on Programming Concepts and Methods*, Sea of Gallilee, Israel, April 1990, pages 27–52. North Holland, 1990.
- [Gla01] Rob van Glabbeek. The linear time–branching time spectrum I. The semantics of concrete, sequential processes. In Bergstra et al. [BPS01], pages 3–99.
- [GR01] Roberto Gorrieri and Arend Rensink. Action refinement. In Bergstra et al. [BPS01], pages 1047–1147.
- [GRS95] Roberto Gorrieri, Marco Roccetti, and Enrico Stancampiano. A theory of processes with durational actions. *Theoretical Computer Science*, 140:73–94, 1995.
- [GV87] Rob van Glabbeek and Frits Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *PARLE, Parallel Architectures and Languages Europe (Volume II)*, volume 259 of *LNCS*, pages 224–242. Springer-Verlag, 1987.
- [Hoa85] C. A. R. Hoare. *Communications Sequential Processes*. International Series in Computer Science. Prentice Hall, 1985.
- [HR94] Holger Hermanns and Michael Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *PAPM*, 1994.
- [Lan92] Rom Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, Department of Computer Science, University of Twente, 1992.

- [Lan93] Rom Langerak. Bundle event structures: A non-interleaving semantics for LOTOS. In M. Diaz and R. Groz, editors, *Formal Description Techniques, V*, pages 331–346. Elsevier, 1993.
- [MCW01] Mila Majster-Cederbaum and Jinzhao Wu. Action refinement for true concurrent real-time. In *Proc. 7th IEEE int. Conf. on Engineering of Complex Computer Systems*, pages 58–68. IEEE Computer Society Press, 2001.
- [Mil89] Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.
- [Mur93] David Murphy. Time and duration in noninterleaving concurrency. *Fundamenta Informaticae*, 19:403–416, 1993.
- [NPW81] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13:85–108, 1981.
- [Pom86] Lucia Pomello. Some Equivalence Notions for Concurrent Systems. An Overview. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *LNCS*, pages 381–400. Springer-Verlag, 1986.
- [Pra86] Vaughan Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15:33–71, 1986.
- [SCW98] Susan Stepney, David Cooper, and Jim Woodcock. More powerful data refinement in Z: pushing the state of the art in industrial refinement. In J.P. Bowen, A. Fett, and M.G. Hinchey, editors, *ZUM*, volume 1493 of *LNCS*, pages 284–307. Springer-Verlag, 1998.
- [VD98] Simone Vegliani and Rocco De Nicola. Possible worlds for process algebras. In D. Sangiorgi and R. de Simone, editors, *CONCUR*, volume 1466 of *LNCS*, pages 179–193. Springer-Verlag, 1998.
- [Vog91] Walter Vogler. Failures semantics based on interval semiwords is a congruence for refinement. *Distributed Computing*, 4:139–162, 1991.
- [Vog93] Walter Vogler. Bisimulation and action refinement. *Theoretical Computer Science*, 114:173–200, 1993.
- [Vog95] Walter Vogler. Timed testing of concurrent systems. *Information and Computation*, 121:149–171, 1995.
- [Win89] Glynn Winskel. An introduction to event structures. In de Bakker et al. [dBdRR89], pages 364–397.

## A. Proofs

In the following, let function  $\pi_i$  denote the projection to the  $i^{\text{th}}$  component. Suppose  $f, f' : M_1 \rightarrow M_2$ , then we write  $f(m_1) \simeq f'(m'_1)$  to denote that  $f(m_1)$  is defined  $\Leftrightarrow f'(m'_1)$  is defined and  $f(m_1)$  is defined  $\Rightarrow f(m_1) = f'(m'_1)$ .

The following lemma states how the event execution of  $\underline{\text{Ref}}(\mathcal{E}, \vartheta)$  can be reduced to event execution of  $\mathcal{E}$  and  $\vartheta$ .

**Lemma A.1.** Suppose  $\mathcal{E} \in \text{ETBES}$ ,  $\vartheta : \mathcal{U} \rightarrow \text{ETBES}$ . Then

$$\underline{\text{Ref}}(\mathcal{E}, \vartheta)_{[(e, \hat{e})]} \simeq \begin{cases} \underline{\text{Ref}}(\mathcal{E}_{[e]}, \vartheta) & \text{if } \Upsilon(T_{\vartheta(e)}, \hat{e}) \\ \underline{\text{Ref}}(\mathcal{E}, \vartheta[e \rightarrow \vartheta(e)_{[e]}]) & \text{if } \neg \Upsilon(T_{\vartheta(e)}, \hat{e}) \end{cases} .$$

Furthermore,  $\forall e \in \mathcal{U} \setminus E : \underline{\text{Ref}}(\mathcal{E}, \vartheta[e \rightarrow \mathcal{E}']) \simeq \underline{\text{Ref}}(\mathcal{E}, \vartheta)$  holds for any  $\mathcal{E}' \in \text{ETBES}$ .

Moreover,  $\Upsilon(T_{\underline{\text{Ref}}(\mathcal{E}, \vartheta)}, (e, \hat{e})) \Leftrightarrow (\Upsilon(T, e) \wedge \Upsilon(T_{\vartheta(e)}, \hat{e}))$

*Proof.* Straightforward.  $\square$

### A.1. Proofs of the ICT-statements

To simplify the proof we introduce a variant of the initial corresponding traces for which the equivalence notion coincides.

**Definition A.2.** Let  $\mathcal{E}, \mathcal{E}' \in \text{ETBES}$ . Define

$$\tilde{\text{Tr}}^{ic}(\mathcal{E}) = \{((e_i, \gamma_i)_{i < n}, \gamma_n) \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \dots, \mathcal{E}_n : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n-1 : \mathcal{E}_{i+1} \wedge \gamma_j \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\mathcal{E}_j)) \wedge ((\neg \Upsilon(\mathcal{E}_n, e_n) \vee n = 0) \Rightarrow \gamma_n \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\mathcal{E}_n))) \wedge (\Upsilon(\mathcal{E}_n, e_n) \Rightarrow \gamma_n = \{\sqrt{\}\})\}.$$

Furthermore, define  $\mathcal{E} \approx_{ICT} \mathcal{E}'$  if

- for every  $((e_i, \gamma_i)_{i < n}, \gamma_n) \in \tilde{\text{Tr}}^{ic}(\mathcal{E})$  there is an injective, labelling (and  $\sqrt{\}$ ) preserving function  $f : (\gamma_n \cup \bigcup_{i < n} (\gamma_i \cup \{e_i\})) \rightarrow (E' \cup \{\sqrt{\}\})$  such that  $((f(e_i), f(\gamma_i))_{i < n}, f(\gamma_n)) \in \tilde{\text{Tr}}^{ic}(\mathcal{E}')$ ;
- and symmetrical as in Definition 4.5.

**Lemma A.3.** The initial corresponding trace equivalence coincides with the equivalence defined in Definition A.2, i.e.  $\sim_{ICT} = \approx_{ICT}$ .

*Proof.* Suppose  $(e_i, \gamma_i)_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E})$  and  $\mathcal{E} \approx_{ICT} \mathcal{E}'$ . Then  $((e_i, \gamma_i)_{i \leq n}, \emptyset) \in \tilde{\text{Tr}}^{ic}(\mathcal{E})$  and so there is  $f : (\bigcup_{i \leq n} (\gamma_i \cup \{e_i\})) \rightarrow (E' \cup \{\sqrt{\}\})$  such that  $((f(e_i), f(\gamma_i))_{i \leq n}, \emptyset) \in \tilde{\text{Tr}}^{ic}(\mathcal{E}')$ . This implies  $(f(e_i), f(\gamma_i))_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E}')$  as required. The case if  $(e_i, \gamma_i)_{i \leq n} \sqrt{\} \in \text{Tr}^{ic}(\mathcal{E})$  and  $\mathcal{E} \approx_{ICT} \mathcal{E}'$  can be analogously shown.

Now suppose  $((e_i, \gamma_i)_{i < n}, \gamma_n) \in \tilde{\text{Tr}}^{ic}(\mathcal{E})$  and  $\mathcal{E} \sim_{ICT} \mathcal{E}'$ . We proceed by making a case analysis:

$\gamma_n = \emptyset \vee \gamma_n = \{\sqrt{\cdot}\}$ : Similar to the above reasoning.

$\gamma_n \neq \emptyset$ : Let  $e_n \in \gamma_n$ . Then  $(e_i, \gamma_i)_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E})$  or  $(e_i, \gamma_i)_{i \leq n} \vee \in \text{Tr}^{ic}(\mathcal{E})$ . So there is a function  $f : (\bigcup_{i \leq n} (\gamma_i \cup \{e_i\})) \rightarrow E'$  such that  $(f(e_i), f(\gamma_i))_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E}')$  (respectively  $(f(e_i), f(\gamma_i))_{i \leq n} \vee \in \text{Tr}^{ic}(\mathcal{E}')$ ). This implies  $((f(e_i), f(\gamma_i))_{i < n}, f(\gamma_n)) \in \tilde{\text{Tr}}^{ic}(\mathcal{E}')$  which completes the proof, since  $\bigcup_{i \leq n} (\gamma_i \cup \{e_i\}) = \gamma_n \cup \bigcup_{i < n} (\gamma_i \cup \{e_i\})$ .  $\square$

**Theorem (4.7).** ICT-equivalence is a congruence for the refinement operator  $\text{Ref}$ , i.e. for any  $A \subseteq \text{Obs}$  and  $\theta : A \rightarrow \text{ETBES}$  we have  $\mathcal{E} \sim_{ICT} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{ICT} \theta'(a)$  implies  $\text{Ref}(\mathcal{E}, \theta) \sim_{ICT} \text{Ref}(\mathcal{E}', \theta')$ .

*Proof.* For simplicity, we write  $\tilde{\mathcal{E}}$  for  $\text{Ref}_A(\mathcal{E}, \theta) = \underline{\text{Ref}}(\mathcal{E}, \vartheta)$  and  $\tilde{\mathcal{E}}'$  for  $\text{Ref}_A(\mathcal{E}', \theta') = \underline{\text{Ref}}(\mathcal{E}', \vartheta')$ , where  $\vartheta$  and  $\vartheta'$  are defined as in Section 3.

Suppose  $((e_i, \hat{e}_i), \tilde{\gamma}_i)_{i \leq n} \in \text{Tr}^{ic}(\tilde{\mathcal{E}})$ . The case when an element is taken from  $\text{Tr}^{ic}(\tilde{\mathcal{E}}')$  follows by symmetrical arguments. As an immediate consequence of Lemma A.1 we get the existence of  $\mathcal{E}_0, \dots, \mathcal{E}_{n+1}$  and of  $\vartheta_0, \dots, \vartheta_{n+1}$  such that for  $i \leq n$  we have

$$\mathcal{E}_0 = \mathcal{E} \wedge \vartheta_0 = \vartheta \wedge e_i \in \text{init}(\mathcal{E}_i) \wedge \hat{e}_i \in \text{init}(\vartheta_i(e_i)) \wedge (\Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i) \Rightarrow (\mathcal{E}_{i+1} = \mathcal{E}_{i[e_i]} \wedge \vartheta_{i+1} = \vartheta_i)) \wedge (\neg \Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i) \Rightarrow (\mathcal{E}_{i+1} = \mathcal{E}_i \wedge \vartheta_{i+1} = \vartheta_i[e_i \rightarrow \vartheta_i(e_i)_{[\hat{e}_i]}])) \wedge \neg(\Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i) \wedge \Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i))$$

Define  $I = \{i \in \{0, \dots, n\} \mid \Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i)\}$ . Assume  $\{k_0, \dots, k_{|I|-1}\} = I$  and  $k_i < k_{i+1}$ . Then we have  $\mathcal{E}_{k_0} = \mathcal{E}$  and  $\mathcal{E}_{k_j[e_{k_j}]} = \mathcal{E}_{k_{j+1}}$ . Furthermore, define  $\gamma_{k_j} = \left( \bigcup_{i > k_{j-1}}^{\min\{k_j, n\}} (\pi_1(\tilde{\gamma}_i) \cup \{e_i\}) \right) \cap \text{init}_{\text{Obs}}(\mathcal{E}_{k_j})$  for  $j \leq |I|$  where  $k_{-1} = -1$  and  $k_{|I|} = n + 1$ . Hence,  $((e_{k_i}, \gamma_{k_i})_{j < |I|}, \gamma_{n+1}) \in \tilde{\text{Tr}}^{ic}(\mathcal{E})$ .

From  $\mathcal{E} \sim_{ICT} \mathcal{E}'$  and Lemma A.3 we get the existence of an injective, labelling preserving  $f : (\bigcup_{j < |I|} (\gamma_{k_j} \cup \{e_{k_j}\}) \cup \gamma_{n+1}) \rightarrow E'$  such that  $((f(e_{k_i}), f(\gamma_{k_i}))_{j < |I|}, f(\gamma_{n+1})) \in \tilde{\text{Tr}}^{ic}(\mathcal{E}')$ .

Now define  $I^e = \{j \in \{0, \dots, n\} \mid e = e_j\}$  for any  $e \in E$ . Assume  $I^e = \{k_0^e, \dots, k_{|I^e|-1}^e\}$  and  $k_i^e < k_{i+1}^e$ .

Define  $\gamma_{k_j^e}^e = \bigcup_{i > k_{j-1}^e}^{\min\{k_j^e, n\}} \{\hat{e} \mid (e, \hat{e}) \in \tilde{\gamma}_i\}$  for  $j \leq |I^e|$  where  $k_{-1}^e = -1$  and  $k_{|I^e|}^e = n + 1$ . Furthermore, we have  $\vartheta_{k_0^e}(e) = \theta(l(e))$  and  $\vartheta_{k_j^e}(e)_{[\hat{e}_{k_j^e}]} = \vartheta_{k_{j+1}^e}(e)$ . Hence,  $e \notin \{e_i \mid i \in I\} \Rightarrow ((\hat{e}_{k_j^e}, \gamma_{k_j^e}^e)_{j < |I^e|}, \gamma_{n+1}^e) \in \tilde{\text{Tr}}^{ic}(\theta(l(e)))$  and  $e \in \{e_i \mid i \in I\} \Rightarrow ((\hat{e}_{k_j^e}, \gamma_{k_j^e}^e)_{j < |I^e|}, \{\sqrt{\cdot}\}) \in \tilde{\text{Tr}}^{ic}(\theta(l(e))) \wedge \gamma_{n+1}^e = \emptyset$ .

From the fact that  $\theta(l(e)) \sim_{ICT} \theta'(l(e))$  and from Lemma A.3 we know that there exists an injective, labelling preserving function  $f^e : (\gamma_{n+1}^e \cup \bigcup_{j < |I^e|} (\gamma_{k_j^e}^e \cup \{\hat{e}_{k_j^e}\})) \rightarrow E_{\theta'(l(e))}$  such that  $((f^e(\hat{e}_{k_j^e}), f^e(\gamma_{k_j^e}^e))_{j < |I^e|}, f^e(\gamma_{n+1}^e)) \in \tilde{\text{Tr}}^{ic}(\theta'(l(e)))$ , respectively  $((f^e(\hat{e}_{k_j^e}), f^e(\gamma_{k_j^e}^e))_{j < |I^e|}, \{\sqrt{\cdot}\}) \in \tilde{\text{Tr}}^{ic}(\theta'(l(e)))$ .

Define  $\tilde{f} : (\bigcup_{i \leq n} (\tilde{\gamma}_i \cup \{(e_i, \hat{e}_i)\})) \rightarrow \tilde{E}'$  as  $\tilde{f}(e, \hat{e}) = (f(e), f^e(\hat{e}))$ .

Then it is easily seen that  $\tilde{f}$  is an injective and labelling preserving function. Furthermore, define  $\mathcal{E}'_0 = \mathcal{E}'$ ,  $\vartheta'_0 = \vartheta'$  and

$$\mathcal{E}'_{i+1} = \begin{cases} \mathcal{E}'_{i[f(e_i)]} & \text{if } \Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i) \\ \mathcal{E}'_i & \text{otherwise} \end{cases} \quad \vartheta'_{i+1} = \begin{cases} \vartheta'_i & \text{if } \Upsilon(T_{\vartheta_i(e_i)}, \hat{e}_i) \\ \vartheta'_i[f(e_i) \rightarrow \vartheta'_i(f(e_i))_{[f^e(\hat{e}_i)}]] & \text{otherwise} \end{cases}$$

From  $((f(e_{k_i}), f(\gamma_{k_i}))_{j < |I|}, f(\gamma_{n+1})) \in \tilde{\text{Tr}}^{ic}(\mathcal{E}')$  and  $((f^e(\hat{e}_{k_j^e}), f^e(\gamma_{k_j^e}^e))_{j < |I^e|}, f^e(\gamma_{n+1}^e)) \in \tilde{\text{Tr}}^{ic}(\theta'(l(e)))$ , respectively  $((f^e(\hat{e}_{k_j^e}), f^e(\gamma_{k_j^e}^e))_{j < |I^e|}, \{\sqrt{\cdot}\}) \in \tilde{\text{Tr}}^{ic}(\theta'(l(e)))$ , it is easily checked that  $\mathcal{E}'_j$  and  $\vartheta'_j$  are well defined. More precisely, for any  $j$  and any  $e \in E$  we have  $\mathcal{E}'_{k_{j+1}} = \mathcal{E}'_{k_j[f(e_{k_j})]}$  and  $\vartheta'_{k_{j+1}}(f(e)) = \vartheta'_{k_j}(f(e))_{[f^e(\hat{e}_{k_j^e})]}$ .

Furthermore, define  $\tilde{\mathcal{E}}'_i = \underline{\text{Ref}}(\mathcal{E}'_i, \vartheta'_i)$  for  $i \leq n + 1$ . Then  $\tilde{\mathcal{E}}'_0 = \tilde{\mathcal{E}}'$  and  $\tilde{\mathcal{E}}'_{i+1} = \tilde{\mathcal{E}}'_{i[\tilde{f}(e_i, \hat{e}_i)]}$  by Lemma A.1.

Suppose  $(e, \hat{e}) \in \tilde{\gamma}_i$ . Let  $m \in \{0, \dots, |I|\}$  such that  $k_{m-1} < i \leq k_m$ . Hence,  $e \in \gamma_{k_m}$ , which implies  $f(e) \in f(\gamma_{k_m}) \subseteq \text{init}(\mathcal{E}'_{k_m})$ . And so from the definition of  $\mathcal{E}'_j$  we get  $f(e) \in \text{init}(\mathcal{E}'_i)$ , since  $k_{m-1} < i \leq k_m$ . Furthermore, take  $\hat{m} \leq |I^e|$  such that  $k_{\hat{m}-1}^e < i \leq k_{\hat{m}}^e$ . Hence,  $\hat{e} \in \gamma_{k_{\hat{m}}^e}^e$ , which implies  $f^e(\hat{e}) \in f^e(\gamma_{k_{\hat{m}}^e}^e) \subseteq \text{init}(\vartheta'_{k_{\hat{m}}^e}(f(e)))$ . And so from the definition of  $\vartheta'_j$  we get  $f^e(\hat{e}) \in \text{init}(\vartheta'_i(f(e)))$ . Therefore,  $\tilde{f}(e, \hat{e}) \in \text{init}(\tilde{\mathcal{E}}'_i)$ .

Additionally, we have  $\neg \Upsilon(\underline{\text{Ref}}(\mathcal{E}'_n, \vartheta'_n), \tilde{f}(e_n, \hat{e}_n))$ . Thus, we showed  $(\tilde{f}(e_i, \hat{e}_i), \tilde{f}(\tilde{\gamma}_i))_{i \leq n} \in \text{Tr}^{ic}(\tilde{\mathcal{E}}')$ .

The case  $((e_i, \hat{e}_i), \tilde{\gamma}_i)_{i \leq n} \vee \in \text{Tr}^{ic}(\tilde{\mathcal{E}})$  follows by similar arguments.  $\square$

**Theorem (4.9).** ICT-equivalence is the coarsest congruence for  $Ref$  with respect to trace equivalence, i.e.  $\sim_{ICT} \subseteq \sim_t$  and  $\sim_{ICT}$  is a congruence for  $Ref$  and for every equivalence  $\equiv \subseteq \sim_t$  that is a congruence for  $Ref$  we have  $\equiv \subseteq \sim_{ICT}$ . Moreover, if  $\forall A \subseteq \text{Obs}, \theta : A \rightarrow \mathbf{ETBES} : Ref(\mathcal{E}, \theta) \sim_t Ref(\mathcal{E}', \theta)$  then  $\mathcal{E} \sim_{ICT} \mathcal{E}'$ .

*Proof.* Suppose  $(e_i, \gamma_i)_{i \leq n} \vee \in \text{Tr}^{ic}(\mathcal{E})$ . We define a refinement  $\theta'$  which is used to construct a corresponding trace. Therefore, let  $\kappa : \mathcal{U} \rightarrow \mathbb{N}$  be an isomorphism. Furthermore, define  $\hat{E} = \bigcup_{i \leq n} (\gamma_i \cup \{e_i\})$  and  $\hat{E}_a = \{e \in \hat{E} \mid l(e) = a\}$ . Additionally, define  $\delta : E \rightarrow \mathbb{N}$  by  $\delta(e) = 1 + |\{i \mid e \in \gamma_i\}|$ . Moreover, let  $A \subseteq \mathcal{A}ct$  be the set of all action-names occurring in  $\mathcal{E}$  or in  $\mathcal{E}'$ , i.e.  $A = \{l(e) \mid e \in E\} \cup \{l'(e') \mid e' \in E'\}$ . And let  $\mu : E \times \mathbb{N} \rightarrow \text{Obs} \setminus A$  be an injective function. Such a function exists, since  $\text{Obs}$  is uncountable.

Our idea of  $\theta'$  is that we replace any event  $e$  of  $\hat{E}$  by the sequential composition of actions  $\mu(e, 1), \dots, \mu(e, \delta(e) + 1)$ . Since  $\theta'$  only maps action-names instead of events, we take the sum of all the corresponding events, i.e.  $\theta'(a) = \mathcal{E}'_a$ , where

$$\begin{aligned} \mathcal{E}'_a = & \left( \{ \star_2^{\kappa(e)} \star_1^j \bullet \mid e \in \hat{E}_a \wedge 1 \leq j \leq \delta(e) \}, \right. \\ & \{ (\{ \star_2^k \star_1 \bullet \mid k \neq \kappa(e) \} \cup \{ \star_2^{\kappa(e)} \star_1^j \bullet, \star_2^{\kappa(e)} \star_1^j \bullet \} \mid e \in \hat{E}_a \wedge 1 \leq j \leq \delta(e) \}, \\ & \{ (\{ \star_2^{\kappa(e)} \star_1^j \bullet, \star_2^{\kappa(e)} \star_1^{j+1} \bullet \} \mid e \in \hat{E}_a \wedge 1 \leq j < \delta(e) \}, \\ & \{ \{ \star_2^{\kappa(e)} \star_1^{\delta(e)} \bullet \mid e \in \hat{E}_a \} \}, \\ & \left. \{ (\star_2^{\kappa(e)} \star_1^j \bullet, \mu(e, j)) \mid e \in \hat{E}_a \wedge 1 \leq j \leq \delta(e) \} \right). \end{aligned}$$

and  $\star_1, \star_2$  be two distinct elements of  $\mathcal{U}$ . Furthermore, the sequence  $\star_2^k \star_1^j \bullet$  is considered to be right bracketed (i.e.  $\star_2^k \star_1^j \bullet \hat{=} (\star_2(\dots(\star_2(\star_1(\dots(\star_1, \bullet)\dots)))\dots))$ ) and therefore is an element of  $\mathcal{U}$ .

Let  $m = n + \sum_{i=0}^n |\gamma_i|$  and define  $I_{-1} = \gamma_0$  and  $s_{-1} = 0$  and for  $i \in \{0, \dots, m-1\}$ :

- if  $I_{i-1} \neq \emptyset$ , then  $s_i = s_{i-1}$ ,  $I_i = I_{i-1} \setminus \{e\}$  and  $\tilde{e}_i = (e, \star_2^{\kappa(e)} \star_1^{|\{j \mid j \leq s_{i-1} \wedge e \in \gamma_j\}|} \bullet)$ , where  $e \in I_{i-1}$
- if  $I_{i-1} = \emptyset$ , then  $s_i = s_{i-1} + 1$ ,  $I_i = \gamma_{s_i}$  and  $\tilde{e}_i = \begin{cases} (e_{s_{i-1}}, \bullet) & \text{if } l(e_{s_{i-1}}) = \tau \\ (e_{s_{i-1}}, \star_2^{\kappa(e_{s_{i-1}})} \star_1^{\delta(e_{s_{i-1}})} \bullet) & \text{otherwise} \end{cases}$

$$\text{and } \tilde{e}_m = \begin{cases} (e_n, \bullet) & \text{if } l(e_n) = \tau \\ (e_n, \star_2^{\kappa(e_n)} \star_1^{\delta(e_n)} \bullet) & \text{otherwise} \end{cases}.$$

It is easily seen that  $\{\pi_1(\tilde{e}_i) \mid i \in \{0, \dots, m\}\} = \hat{E}$ . Define  $\tilde{\mathcal{E}}_0 = Ref_A(\mathcal{E}, \theta')$  and  $\tilde{\mathcal{E}}_{i+1} = \tilde{\mathcal{E}}_{i[\tilde{e}_i]}$  for  $i \leq m$ . The  $\tilde{\mathcal{E}}_i$  are well defined which, can be seen by induction, as follows. Obviously for  $i = 0$ . Suppose  $\tilde{e}_i = (e, \star_2^{\kappa(e)} \star_1^j \bullet)$ . By Lemma A.1 we obtain that  $\tilde{\mathcal{E}}_i = Ref_{[e_0] \dots [e_{s_{i-1}-1}]}(\mathcal{E}, \theta')$ , where  $\theta'_i(e) = \theta'(l(e))_{[\star_2^{\kappa(e)} \star_1^j \bullet], \dots, [\star_2^{\kappa(e)} \star_1^q \bullet]}$  for  $l(e) \neq \tau$  with  $q = |\{j \mid e = \pi_1(\tilde{e}_j) \wedge j < i\}|$ . It is easily seen that  $(e, \star_2^{\kappa(e)} \star_1^j \bullet), \dots, (e, \star_2^{\kappa(e)} \star_1^{j-1} \bullet)$  appears in the sequence before  $\tilde{e}_i$ . Thus  $\star_2^{\kappa(e)} \star_1^j \bullet \in \text{init}(\theta'_i(e))$ . Furthermore,  $e_{s_{i-1}} \in \text{init}(\mathcal{E}_{[e_0] \dots [e_{s_{i-1}-1}]})$ , since  $(e_j, \gamma_j)_{j \leq n} \in \text{Tr}^{ic}(\mathcal{E})$ . Hence,  $\tilde{e}_i \in \tilde{\mathcal{E}}_i$ . Furthermore, by Lemma A.1 we obtain  $\Upsilon(\tilde{\mathcal{E}}_n, \tilde{e}_n)$ .

From the definition of  $\tilde{\mathcal{E}}_i$  it follows that  $(\alpha_i)_{(i \leq m)} \in \text{Tr}(Ref_A(\mathcal{E}, \theta'))$ , where  $\alpha_i = l_{Ref_A(\mathcal{E}, \theta')}(\tilde{e}_i)$ . Therefore, we get  $(\alpha_i)_{(i \leq m)} \in \text{Tr}(Ref_A(\mathcal{E}', \theta'))$ , since  $Ref_A(\mathcal{E}, \theta') \sim_t Ref_A(\mathcal{E}', \theta')$ . Hence, there exists  $(\tilde{e}'_i)_{(i \leq m)}$  such that  $\tilde{\mathcal{E}}'_0 = Ref_A(\mathcal{E}', \theta')$  and  $\tilde{\mathcal{E}}'_{i+1} = \tilde{\mathcal{E}}'_{i[\tilde{e}'_i]}$  are well defined and  $\alpha_i = l_{Ref_A(\mathcal{E}', \theta')}(\tilde{e}'_i)$ .

From the injectivity of  $\mu$  we get  $\forall i \leq m : \pi_1(\tilde{e}_i) \neq \tau \Rightarrow \pi_2(\tilde{e}_i) = \pi_2(\tilde{e}'_i)$ . Define  $e'_j = \pi_1(\tilde{e}'_i)$  where  $i$  is chosen such that  $(\tilde{e}_i = (e_j, \star_2^{\kappa(e_j)} \star_1^j \bullet)) \vee (l(e_j) = \tau \wedge \tilde{e}_i = (e_j, \bullet))$ . Now, we verify by induction that

$$\tilde{\mathcal{E}}'_i = Ref_{[e'_0] \dots [e'_{s_{i-1}-1}]}(\mathcal{E}', \theta'), \text{ where} \tag{1}$$

$$\theta''_i(e') = \begin{cases} \theta'_i(\pi_1(\tilde{e}_j)) & \text{if } (e', \star_2^{\kappa(e)} \star_1^j \bullet) = \tilde{e}'_j \\ \theta'(l'(e')) & \text{if } l'(e') \in A \wedge \forall j : (e', \star_2^{\kappa(e)} \star_1^j \bullet) \neq \tilde{e}'_j \\ (\{\bullet\}, \{(\{\bullet\}, \bullet)\}, \emptyset, \{\bullet\}, \{(\bullet, l(e))\}) & \text{if } e \in E \wedge l(e) \notin A \\ (\emptyset, \emptyset, \emptyset, \{\emptyset\}, \emptyset) & \text{otherwise} \end{cases}.$$

Obviously for  $i = 0$ . We proceed by making a case analysis:

$l(\pi_1(\tilde{e}_i)) = \tau$ : By induction and Lemma A.1 we get  $\tilde{\mathcal{E}}'_{i+1} = \underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}[\pi_1(\tilde{e}'_i)], \vartheta''_i)$ , which is equal to  $\underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}, \vartheta''_{i+1})$ .

$\tilde{e}_i = (e_j, \star_2^{\kappa(e_j)} \star_1^{\delta(e_j)} \bullet)$ : Then  $\Upsilon(\vartheta'_i, \pi_2(\tilde{e}'_i))$ , since  $\tilde{e}_i$  and  $\tilde{e}'_i$  have the same label and  $\mu$  is injective. Thus by induction and Lemma A.1 we get  $\tilde{\mathcal{E}}'_{i+1} = \underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}[\pi_1(\tilde{e}'_i)], \vartheta''_i)$ .

Furthermore, there is  $k < i$  such that  $\tilde{e}'_k = (\pi_1(\tilde{e}'_i), \star_2^{\kappa(e)} \star_1^{\bullet})$ , since otherwise  $\tilde{e}'_i \notin \text{init}(\tilde{\mathcal{E}}'_i)$ . From the injectivity of  $\mu$  we obtain that  $\pi_1(\tilde{e}_i) = \pi_1(\tilde{e}_k)$ , since  $\tilde{e}_k$  and  $\tilde{e}'_k$  have the same label. Hence,  $\tilde{\mathcal{E}}'_{i+1} = \underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}, \vartheta''_{i+1})$ , as required.

Otherwise: Then  $\neg \Upsilon(\vartheta'_i, \pi_2(\tilde{e}'_i))$ . Hence,  $\tilde{\mathcal{E}}'_{i+1} = \underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}, \vartheta''_i[\pi_1(\tilde{e}'_i) \rightarrow \vartheta''_i(\pi_1(\tilde{e}'_i))_{[\pi_2(\tilde{e}'_i)}]])$  by induction and Lemma A.1. Furthermore, there is  $k \leq i$  such that  $\tilde{e}'_k = (\pi_1(\tilde{e}'_i), \star_2^{\kappa(e)} \star_1^{\bullet})$ , since otherwise  $\tilde{e}'_i \notin \text{init}(\tilde{\mathcal{E}}'_i)$ . From the injectivity of  $\mu$  we obtain that  $\pi_1(\tilde{e}_i) = \pi_1(\tilde{e}_k)$ . Hence,  $\tilde{\mathcal{E}}'_{i+1} = \underline{Ref}(\mathcal{E}'_{[e'_0]..[e'_{s_{i-1}-1}]}, \vartheta''_{i+1})$ , as required.

From (1) we obtain that  $\pi_1(\tilde{e}_i) = \pi_1(\tilde{e}_j) \Leftrightarrow \pi_1(\tilde{e}'_i) = \pi_1(\tilde{e}'_j)$ . Hence, the function  $f : \hat{E} \rightarrow E'$  with  $f(e) = \pi_1(\tilde{e}'_i)$  whenever  $e = \pi_1(\tilde{e}_i)$  is well defined, labelling preserving and injective.

Additionally, (1) becomes  $\tilde{\mathcal{E}}'_i = \underline{Ref}(\mathcal{E}'_{[f(e_0)]..[f(e_{s_{i-1}-1})]}], \vartheta''_i)$ , where  $\vartheta''_i$  is defined by

$$\vartheta''_i(e') = \begin{cases} \vartheta'_i(f^{-1}(e')) & \text{if } f^{-1}(e') \text{ is defined} \\ \theta'(l'(e')) & \text{if } l'(e') \in A \wedge f^{-1}(e') \text{ is undefined} \\ (\{\bullet\}, \{(\{\bullet\}, \bullet)\}, \emptyset, \{\bullet\}, \{(\bullet, l(e))\}) & \text{if } e' \in E' \wedge l(e') \notin A \\ (\emptyset, \emptyset, \emptyset, \{\emptyset\}, \emptyset) & \text{otherwise} \end{cases}.$$

Furthermore, by Lemma A.1 and from  $\Upsilon(\tilde{\mathcal{E}}_n, \tilde{e}_n)$  we obtain  $\Upsilon(\mathcal{E}'_{[f(e_0)]..[f(e_{s_{n-1}-1})]}, f(e_{s_{n-1}}))$ . From this and (1) it follows that  $(f(e_i), f(\gamma_i))_{i \leq n} \sqrt{\in} \text{Tr}^{ic}(\mathcal{E})$ .

The case when  $(e_i, \gamma_i)_{i \leq n} \in \text{Tr}^{ic}(\mathcal{E})$  follows by similar arguments, in particular chose  $\delta(e_n) = 2 + |\{i \mid e_n \in \gamma_i\}|$ .

The cases where a trace is taken from  $\text{Tr}^{ic}(\mathcal{E}')$  follows by symmetrical arguments.  $\square$

## A.2. Proofs of the FIC-statements

**Theorem (4.13).** FIC-equivalence is a congruence for the refinement operator  $Ref$ , i.e. for any  $A \subseteq \text{Obs}$  and  $\theta : A \rightarrow \text{ETBES}$  we have  $\mathcal{E} \sim_{FIC} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{FIC} \theta'(a)$  implies  $Ref(\mathcal{E}, \theta) \sim_{FIC} Ref(\mathcal{E}', \theta')$ .

*Proof.* Let  $\mathcal{R}$  and  $\mathcal{R}_a$  be finite-initial corresponding bisimulations such that  $(\mathcal{E}, \mathcal{E}', g) \in \mathcal{R}$  and  $(\theta(a), \theta'(a), g_a) \in \mathcal{R}_a$ . Let  $\text{cons} : \mathcal{U} \rightarrow \mathcal{U}$  be the function that maps  $\bullet$  to  $\bullet$  and is undefined elsewhere. Define

$$\begin{aligned} \mathcal{R}_{Ref} = & \{ (\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'), f) \mid \exists \tilde{f} : \exists \tilde{I} \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}})) : \exists F : \mathcal{U} \rightarrow (\mathcal{U} \rightarrow \mathcal{U}) : \\ & (\forall \tilde{e} \in \tilde{E} \setminus \tilde{I} : (\tilde{l}(\tilde{e}) \in A \Rightarrow \tilde{\vartheta}(\tilde{e}) = \theta(\tilde{l}(\tilde{e})) \wedge (\tilde{e} \in \text{init}(\tilde{\mathcal{E}}) \Rightarrow F(\tilde{e}) = g_{\tilde{l}(\tilde{e})})) \wedge \\ & (\forall \tilde{e}' \in \tilde{E}' \setminus \tilde{f}(\tilde{I}) : (\tilde{l}'(\tilde{e}') \in A \Rightarrow \tilde{\vartheta}'(\tilde{e}') = \theta'(\tilde{l}'(\tilde{e}')))) \wedge \\ & (\forall \tilde{e} \in \tilde{I} : (\tilde{\vartheta}(\tilde{e}), \tilde{\vartheta}'(\tilde{f}(\tilde{e})), F(\tilde{e})) \in \mathcal{R}_{\tilde{l}(\tilde{e})}) \wedge (\forall \tilde{e} \in \text{init}(\tilde{\mathcal{E}}) : (\tilde{l}(\tilde{e}) \notin A \Rightarrow F(\tilde{e}) = \text{cons}) \wedge \\ & (\tilde{\mathcal{E}}, \tilde{\mathcal{E}}', \tilde{f}) \in \mathcal{R} \wedge f(e, \hat{e}) \simeq \begin{cases} (\tilde{f}(e), F(e)(\hat{e})) & \text{if } (e, \hat{e}) \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})) \\ \text{undefined} & \text{otherwise} \end{cases} \} \end{aligned}$$

For all  $(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'), f) \in \mathcal{R}_{Ref}$  we have:  $f$  is always an isomorphism between  $\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}))$  and  $\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'))$ ,  $f$  is labelling preserving and

$$f^{-1}(e', \hat{e}') = (\tilde{f}^{-1}(e'), F(\tilde{f}^{-1}(e'))^{-1}(\hat{e}')) \quad (2)$$

where  $\tilde{f}, F$  are its corresponding function. This holds, since  $f(\tilde{f}^{-1}(e'), F(\tilde{f}^{-1}(e'))^{-1}(\hat{e}')) = (e', \hat{e}')$ .

Suppose  $(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'), f) \in \mathcal{R}_{Ref}$ . Let  $\tilde{I}$  be the corresponding set and let  $\tilde{f}, F$  be the corresponding functions.

In the following we will show that  $\mathcal{R}_{Ref}$  is a FIC-bisimulation. Therefore, suppose  $(e, \hat{e}) \in \text{init}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})) \wedge I \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})))$ . We proceed by making a case analysis:

$\tilde{l}(e) \notin A$ : Then  $\hat{e} = \bullet$  and from Lemma A.1 we get

$$\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]} = \underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta}) \text{ and } \Upsilon(T_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}, (e, \hat{e})) \Leftrightarrow \Upsilon(\tilde{T}, e). \quad (3)$$

Define  $\tilde{I}' = \tilde{I} \cup \pi_1(I)$ . Since  $(\tilde{\mathcal{E}}, \tilde{\mathcal{E}}', \tilde{f}) \in \mathcal{R} \wedge \tilde{I}' \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}))$  we have

$$\begin{aligned} \exists e', \tilde{f}' : \tilde{l}(e) = \tilde{l}'(e') \wedge \Upsilon(\tilde{T}, e) \Leftrightarrow \Upsilon(\tilde{T}', e') \wedge (\tilde{\mathcal{E}}_{[e]}, \tilde{\mathcal{E}}'_{[e']}, \tilde{f}') \in \mathcal{R} \wedge \tilde{l}(e) \in \text{Obs} \Rightarrow \tilde{f}(e) = e' \wedge \\ \tilde{f} \upharpoonright (\tilde{I}' \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[e]})) = \tilde{f}' \upharpoonright \tilde{I}' \wedge \tilde{f}^{-1} \upharpoonright (f(\tilde{I}') \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']})) = \tilde{f}'^{-1} \upharpoonright f(\tilde{I}'). \end{aligned} \quad (4)$$

We have  $\tilde{l}(e) \notin A$  implies  $\tilde{l}'(e') \notin A$ . And so by Lemma A.1 we obtain  $\underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}') = \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \bullet)]}$  and  $\Upsilon(\tilde{T}', e') \Leftrightarrow \Upsilon(T_{\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')}_{[(e', \bullet)]}, (e', \bullet))$ . Furthermore, the labels of  $(e', \bullet)$  and  $(e, \bullet)$  coincide and  $l_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}(e, \bullet) \in \text{Obs} \Rightarrow f(e, \bullet) = (\tilde{f}(e), \bullet) = (e', \bullet)$ .

Now define  $F'$  and  $f'$  by

$$F'(e'') = \begin{cases} \text{cons} & \text{if } l(e'') \notin A \wedge e'' \in \text{init}(\tilde{\mathcal{E}}) \\ F(e) & \text{otherwise} \end{cases} \quad (5)$$

$$f'(e'', \hat{e}'') \simeq \begin{cases} (\tilde{f}'(e''), F'(e'')(\hat{e}'')) & \text{if } (e'', \hat{e}'') \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta})) \\ \text{undefined} & \text{otherwise} \end{cases}. \quad (6)$$

Then  $(\underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}'), f') \in \mathcal{R}_{Ref}$  where  $\tilde{I}' \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[e]})$  is its corresponding set and  $\tilde{f}', F'$  are its correspondent functions.

Furthermore, from  $\tilde{f} \upharpoonright (\tilde{I}' \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[e]})) = \tilde{f}' \upharpoonright \tilde{I}'$  we get that  $f'$  and  $f$  coincide on  $I \cap \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]})$ .

Suppose  $(e''', \hat{e}''') \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \bullet)]}) \cap f(I)$ . Then from (2) and from the fact that  $\tilde{f}^{-1} \upharpoonright (f(\tilde{I}') \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']})) = \tilde{f}'^{-1} \upharpoonright f(\tilde{I}')$  we obtain

$$f'^{-1}(e''', \hat{e}''') = (\tilde{f}'^{-1}(e'''), F'(\tilde{f}'^{-1}(e'''))^{-1}(\hat{e}''')) = (\tilde{f}^{-1}(e'''), F(\tilde{f}^{-1}(e'''))^{-1}(\hat{e}''')) = f^{-1}(e''', \hat{e}'''),$$

which completes this case.

$\tilde{l}(e) \in A \wedge \Upsilon(T_{\tilde{\vartheta}(e)}, \hat{e})$ : We get that (3) holds. Define  $\tilde{I}' = \tilde{I} \cup \pi_1(I)$ . From  $(\tilde{\mathcal{E}}, \tilde{\mathcal{E}}', \tilde{f}) \in \mathcal{R} \wedge \tilde{I}' \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}))$  we obtain (4).

Furthermore,  $(\tilde{\vartheta}(e), \tilde{\vartheta}'(e'), F(e)) \in \mathcal{R}_{\tilde{l}(e)}$  and  $\hat{e} \in \text{init}(\tilde{\vartheta}(e))$ , since  $e' = \tilde{f}(e)$ . Hence, there exists  $\hat{e}' \in \text{init}(\tilde{\vartheta}')$  such that  $l_{\tilde{\vartheta}(e)}(\hat{e}) = l_{\tilde{\vartheta}'(e')}(\hat{e}')$  and  $\Upsilon(T_{\tilde{\vartheta}(e)}, \hat{e}) \Leftrightarrow \Upsilon(T_{\tilde{\vartheta}'(e')}, \hat{e}')$  and  $l_{\tilde{\vartheta}(e)}(\hat{e}) \in \text{Obs} \Rightarrow F(e)(\hat{e}) = \hat{e}'$ .

By Lemma A.1,  $\underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}') = \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \hat{e}')]}$  and  $(\Upsilon(\tilde{T}', e') \wedge \Upsilon(T_{\tilde{\vartheta}'(e')}, \hat{e}')) \Leftrightarrow \Upsilon(T_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}, (e', \hat{e}'))$ . The labels of  $(e', \hat{e}')$  and  $(e, \hat{e})$  coincide and  $l_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}(e, \hat{e}) \in \text{Obs}$  implies that  $f(e, \hat{e}) = (\tilde{f}(e), F(e)(\hat{e})) = (e', \hat{e}')$ .

Define  $F'$  and  $f'$  as in (5) and (6). From the definition of  $\mathcal{R}_{Ref}$  we obtain

$$(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]}, \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \hat{e}')]}, f') = (\underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}'), f') \in \mathcal{R}_{Ref}$$

where  $\tilde{I}' \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[e]})$  is its corresponding set and  $\tilde{f}', F'$  are its corresponding functions (please note that  $e \notin E_{\tilde{\mathcal{E}}_{[e]}}$ ).

Similar to the above case we conclude that  $f'$  and  $f$  coincide on  $\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]}) \cap I$  and that  $f'^{-1}$  and  $f^{-1}$  coincide on  $\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \hat{e}')]}) \cap f(I)$

$\tilde{l}(e) \in A \wedge \neg \Upsilon(T_{\tilde{\vartheta}(e)}, \hat{e})$ : By Lemma A.1,  $\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]} = \underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}[e \rightarrow \tilde{\vartheta}(e)_{[\hat{e}]})]$  and  $\neg \Upsilon(T_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}, (e, \hat{e}))$ .

Define  $I^e = \{\hat{e}'' \mid (e, \hat{e}'') \in I\}$ . Since  $I^e \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\vartheta}(e)))$  and  $(\tilde{\vartheta}(e), \tilde{\vartheta}'(f(e)), F(e)) \in \mathcal{R}_{\tilde{l}(e)}$  there exists  $\hat{e}' \in \text{init}(\tilde{\vartheta}'(\tilde{f}(e)))$  and  $\hat{f}$  such that  $(\tilde{\vartheta}(e)_{[\hat{e}]}, \tilde{\vartheta}'(\tilde{f}(e))_{[\hat{e}']}, \hat{f}) \in \mathcal{R}_{\tilde{l}(e)}$  and  $l_{\tilde{\vartheta}(e)}(\hat{e}) = l_{\tilde{\vartheta}'(\tilde{f}(e))}(\hat{e}')$  and  $l_{\tilde{\vartheta}(e)}(\hat{e}) \in \text{Obs} \Rightarrow F(e)(\hat{e}) = \hat{e}'$  and  $\Upsilon(T_{\tilde{\vartheta}(e)}, \hat{e}) \Leftrightarrow \Upsilon(T_{\tilde{\vartheta}'(\tilde{f}(e))}, \hat{e}')$  and  $F(e) \upharpoonright (I^e \cap \text{init}_{\text{Obs}}(\tilde{\vartheta}(e)_{[\hat{e}]}) = \hat{f} \upharpoonright I^e$  and  $F(e)^{-1} \upharpoonright (F(e)(I^e) \cap \text{init}_{\text{Obs}}(\tilde{\vartheta}'(\tilde{f}(e))_{[\hat{e}']})) = \hat{f}^{-1} \upharpoonright F(e)(I^e)$ .

By Lemma A.1,  $\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(\tilde{f}(e), \hat{e}')] = \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'[\tilde{f}(e) \rightarrow \tilde{\vartheta}'(\tilde{f}(e))_{[\hat{e}']}]}$  and  $\neg \Upsilon(T_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}, (\tilde{f}(e), \hat{e}'))$ . The labels of  $(\tilde{f}(e), \hat{e}')$  and  $(e, \hat{e})$  coincide and  $l_{\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})}(e, \hat{e}) \in \text{Obs}$  implies  $f(e, \hat{e}) = (\tilde{f}(e), F(e)(\hat{e})) = (\tilde{f}(e), \hat{e}')$ .

Now define  $f'$  by

$$f'(e'', \hat{e}'') \simeq \begin{cases} (\tilde{f}(e), \hat{f}(\hat{e}'')) & \text{if } (e'', \hat{e}'') \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta})) \wedge e'' = e \\ (\tilde{f}(e''), F(e'')(\hat{e}'')) & \text{if } (e'', \hat{e}'') \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}_{[e]}, \tilde{\vartheta})) \wedge e'' \neq e \\ \text{undefined} & \text{otherwise} \end{cases}.$$

Then  $(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}[e \rightarrow \tilde{\vartheta}(e)_{[e]}]), (\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}'[\tilde{f}(e) \rightarrow \tilde{\vartheta}'(\tilde{f}(e))_{[e']}])), f') \in \mathcal{R}_{Ref}$  where  $\tilde{I} \cup \{e\}$  is its corresponding set and  $\tilde{f}, F' = F[e \rightarrow \tilde{f}]$  are its corresponding functions.

From  $F(e) \upharpoonright (I^e \cap \text{init}_{\text{Obs}}(\tilde{\vartheta}(e)_{[e]})) = \tilde{f} \upharpoonright I^e$  we get that  $f'$  and  $f$  coincide on  $\text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta})_{[(e, \hat{e})]}) \cap I$ .

Suppose  $(e''', \hat{e}''') \in \text{init}_{\text{Obs}}(\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[(e', \hat{e}')]}) \cap f(I)$ . Then from (2) and from the fact that  $\tilde{f}^{-1} \upharpoonright F(e)(I^e) = F(e)^{-1} \upharpoonright (F(e)(I^e) \cap \text{init}_{\text{Obs}}(\tilde{\vartheta}'(\tilde{f}(e))_{[e']}))$  we obtain

$$f'^{-1}(e''', \hat{e}''') = (\tilde{f}^{-1}(e'''), F'(\tilde{f}^{-1}(e'''))^{-1}(e''')) = (\tilde{f}^{-1}(e'''), F(\tilde{f}^{-1}(e'''))^{-1}(e''')) = f^{-1}(e''', \hat{e}''').$$

The last condition of the FIC-bisimulation can be derived by symmetrical arguments. Thus, we proved that  $\mathcal{R}_{Ref}$  is a FIC-bisimulation.

By definition  $Ref_A(\mathcal{E}, \theta) = \underline{Ref}(\mathcal{E}, \vartheta)$  and  $Ref_A(\mathcal{E}', \theta') = \underline{Ref}(\mathcal{E}', \vartheta')$  for the corresponding  $\vartheta, \vartheta'$ . Define

$$F(e) = \begin{cases} g_l(e) & \text{if } l(e) \in A \wedge e'' \in \text{init}(\tilde{\mathcal{E}}) \\ \text{cons} & \text{otherwise} \end{cases} \quad \text{and} \\ f(e, \hat{e}) \simeq \begin{cases} (g(e), F(e)(\hat{e})) & \text{if } (e, \hat{e}) \in \text{init}_{\text{Obs}}(\underline{Ref}(\mathcal{E}, \vartheta)) \\ \text{undefined} & \text{otherwise} \end{cases}.$$

Then it is easy to check that  $(Ref_A(\mathcal{E}, \theta), Ref_A(\mathcal{E}', \theta'), f) \in \mathcal{R}_{Ref}$ , where  $\emptyset$  is its corresponding set and  $g, F$  are its corresponding functions.  $\square$

**Theorem (4.15).** FIC-equivalence is the coarsest congruence for  $Ref$  with respect to bisimulation equivalence, i.e.  $\sim_{FIC} \subseteq \sim_b$  and  $\sim_{FIC}$  is a congruence for  $Ref$  and for every equivalence  $\equiv \subseteq \sim_b$  that is a congruence for  $Ref$  we have  $\equiv \subseteq \sim_{FIC}$ .

Moreover, if  $\forall A \subseteq \text{Obs}, \theta : A \rightarrow \mathbf{ETBES} : Ref(\mathcal{E}, \theta) \sim_b Ref(\mathcal{E}', \theta)$  then  $\mathcal{E} \sim_{FIC} \mathcal{E}'$ .

*Proof.* Define  $A \subseteq \text{Act}$  to be the set of all action-names occurring in  $\mathcal{E}$  or in  $\mathcal{E}'$ , i.e.  $A = \{l(e) | e \in E\} \cup \{l'(e') | e' \in E'\}$ . Let  $\mu : \{1, 2\} \times A \times \mathbb{N} \rightarrow \text{Obs} \setminus A$  be an injective function. Such a function exists, since  $\text{Obs}$  is uncountable. We define for all  $a \in A$  an eTbes  $\mathcal{E}_a$ , which corresponds to the process algebra term  $X = \mu(1, a, 0); \mu(2, a, 0) + X[l]$ , where  $\ell(\mu(i, a, n)) = \mu(i, a, n+1)$ . Let  $\star_1, \star_2$  be two distinct elements of  $\mathcal{U}$ . In the definition the sequences  $\star_2^n \star_1 \star_i \bullet$  are considered to be right bracketed and therefore to be elements of  $\mathcal{U}$  (compare with the proof of Theorem 4.9).

$$\begin{aligned} \mathcal{E}_a = & \left( \{ \star_2^n \star_1 \star_i \bullet \mid n \in \mathbb{N} \wedge i \in \{1, 2\} \}, \right. \\ & \left. \{ (\{ \star_2^n \star_1 \star_1 \bullet \mid n \in \mathbb{N} \setminus \{j\} \}) \cup \{ \star_2^j \star_1 \star_i \bullet \}, \star_2^j \star_1 \star_i \bullet \mid j \in \mathbb{N} \wedge i \in \{1, 2\} \}, \right. \\ & \left. \{ (\{ \star_2^n \star_1 \star_1 \bullet \}, \star_2^n \star_1 \star_2 \bullet) \mid n \in \mathbb{N} \}, \right. \\ & \left. \{ \{ \star_2^n \star_1 \star_2 \bullet \mid n \in \mathbb{N} \} \}, \right. \\ & \left. \{ (\star_2^n \star_1 \star_i \bullet, \mu(i, a, n)) \mid n \in \mathbb{N} \wedge i \in \{1, 2\} \} \right) \end{aligned}$$

Define  $\theta' : A \rightarrow \mathbf{ETBES}$  by  $\theta'(a) = \mathcal{E}_a$ . Furthermore, define  $\mathcal{E}^{(a, n)}$  by

$$\mathcal{E}^{(a, n)} = (\{ \star_2^n \star_1 \star_2 \bullet \}, \{ (\{ \star_2^n \star_1 \star_2 \bullet \}, \star_2^n \star_1 \star_2 \bullet) \}, \emptyset, \{ \{ \star_2^n \star_1 \star_2 \bullet \} \}, \{ (\star_2^n \star_1 \star_2 \bullet, \mu(2, a, n)) \}).$$

Let  $\mathcal{R}_b$  be a strong bisimulation such that  $(Ref_A(\mathcal{E}, \theta), Ref_A(\mathcal{E}', \theta')) \in \mathcal{R}_b$ . Without loss of generality,  $\mathcal{R}_b$  contains only elements which can be derived from  $(Ref_A(\mathcal{E}, \theta), Ref_A(\mathcal{E}', \theta'))$ . Furthermore, let  $\kappa : \mathcal{U} \rightarrow \mathbb{N}$  be an isomorphism. We define the relation  $Ref_{FIC}$  by

$$\begin{aligned} \mathcal{R}_{FIC} = & \{ (\tilde{\mathcal{E}}, \tilde{\mathcal{E}}', \tilde{f}) \mid \tilde{f} : \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}) \rightarrow \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}') \text{ is a labeling preserving isomorphism} \wedge \\ & \forall \tilde{I} \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}})) : \exists \tilde{J} \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}')) : \tilde{I} \subseteq \tilde{J} \wedge \exists \tilde{\vartheta}, \tilde{\vartheta}' : \\ & \left( \forall \tilde{e} \in \tilde{E} : \tilde{\vartheta}(\tilde{e}) = \begin{cases} \mathcal{E}_{\tilde{l}(\tilde{e})} & \text{if } \tilde{e} \in \tilde{E} \setminus \tilde{J} \\ \mathcal{E}^{(\tilde{l}(\tilde{e}), \kappa(\tilde{f}(\tilde{e})))} & \text{if } \tilde{e} \in \tilde{J} \end{cases} \right) \wedge \\ & \left( \forall \tilde{e}' \in \tilde{E}' : \tilde{\vartheta}'(\tilde{e}') = \begin{cases} \mathcal{E}_{\tilde{l}'(\tilde{e}')} & \text{if } \tilde{e}' \in \tilde{E}' \setminus \tilde{f}(\tilde{J}) \\ \mathcal{E}^{(\tilde{l}'(\tilde{e}'), \kappa(\tilde{f}'(\tilde{e}')))} & \text{if } \tilde{e}' \in \tilde{f}(\tilde{J}) \end{cases} \right) \wedge \\ & (\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')) \in \mathcal{R}_b \} \end{aligned}$$

In the following we show that  $Ref_{FIC}$  is a FIC-bisimulation. Therefore, suppose  $(\tilde{\mathcal{E}}, \tilde{\mathcal{E}}', \tilde{f}) \in Ref_{FIC}$ .

Then  $\tilde{f}$  is such a required isomorphism by definition. Now suppose  $\tilde{e} \in \text{init}(\tilde{\mathcal{E}})$  and  $\tilde{I} \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}))$ . Then there is  $\tilde{J}$  such that  $\tilde{I} \cup \{\tilde{e}\} \subseteq \tilde{J}$ , and  $(\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')) \in \mathcal{R}_b$ , where  $\tilde{\vartheta}, \tilde{\vartheta}'$  are the corresponding functions. We proceed by making a case analysis:

$\tilde{l}(\tilde{e}) \in \text{Obs} \wedge \neg \Upsilon(\tilde{\mathcal{E}}, \tilde{e})$ : Then  $\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}) \xrightarrow{\mu(2, \tilde{l}(\tilde{e}), \kappa(\tilde{f}(\tilde{e})))} \underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta})$  by Lemma A.1, since  $\tilde{l}(\tilde{e}) \in A$ . Furthermore, because of the fact that  $\mathcal{R}_b$  is a strong bisimulation, there exists  $e'$  such that  $(\underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']}) \in \mathcal{R}_b$  and  $l_{\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']}}(e') = \mu(2, \tilde{l}(\tilde{e}), \kappa(\tilde{f}(e)))$ . Thus,  $\pi_1(e') = f(\tilde{e})$  and  $\tilde{l}'(\pi_1(e')) = \tilde{l}(\tilde{e})$  by the injectivity of  $\kappa$  and  $\mu$ . Moreover, we have  $\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']} = \underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}')$ , where  $e' = \pi_1(e')$ .

$\tilde{l}(\tilde{e}) = \tau \wedge \neg \Upsilon(\tilde{\mathcal{E}}, \tilde{e})$ : Then  $\underline{Ref}(\tilde{\mathcal{E}}, \tilde{\vartheta}) \xrightarrow{\tau} \underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta})$  by Lemma A.1. From the fact that  $\mathcal{R}_b$  is a strong bisimulation, there exists  $e'$  such that  $(\underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta}), \underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']}) \in \mathcal{R}_b$  and  $l_{\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']}}(e') = \tau$ . Thus,  $\tilde{l}'(e') = \tau$  and  $\underline{Ref}(\tilde{\mathcal{E}}', \tilde{\vartheta}')_{[e']} = \underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}')$ , where  $e' = \pi_1(e')$ .

The cases when  $\Upsilon(\tilde{\mathcal{E}}, \tilde{e})$  holds are carried out analogously. Furthermore, we have  $\Upsilon(\tilde{\mathcal{E}}, \tilde{e}) \Leftrightarrow \Upsilon(\tilde{\mathcal{E}}', \tilde{e}')$  by Lemma A.1. So it remains to find a function  $\hat{f} : \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}) = \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']})$  that satisfies the necessary constraints. Therefore, define functions  $\hat{f}_i$  and eTbes  $\tilde{\mathcal{E}}_i, \tilde{\mathcal{E}}'_i$  with  $i \in \mathbb{N}$  as follows:  $\hat{f}_0 = \tilde{f} \cap (\tilde{J} \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}))$ ,  $\tilde{\mathcal{E}}_0 = \underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta})$  and  $\tilde{\mathcal{E}}'_0 = \underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}')$ .

for  $2n+1$ : If  $\kappa^{-1}(n) \notin \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}) \vee \hat{f}_{2n}(\kappa^{-1}(n))$  is defined, then  $\hat{f}_{2n+1} = \hat{f}_{2n}$ ,  $\tilde{\mathcal{E}}_{2n+1} = \tilde{\mathcal{E}}_{2n}$  and  $\tilde{\mathcal{E}}'_{2n+1} = \tilde{\mathcal{E}}'_{2n}$ .

If  $\kappa^{-1}(n) \in \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}) \wedge \hat{f}_{2n}(\kappa^{-1}(n))$  is undefined, then  $(\kappa^{-1}(n), \star_2^n \star_1 \star_1 \bullet) \in \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{2n})$ . Hence, there is  $\tilde{e}'_{2n}$  with the same label such that  $(\tilde{\mathcal{E}}_{2n[\tilde{e}'_{2n}]}, \tilde{\mathcal{E}}'_{2n[\tilde{e}'_{2n}]}) \in \mathcal{R}_b$ , where  $\tilde{e}'_{2n} = (\kappa^{-1}(n), \star_2^n \star_1 \star_1 \bullet)$ . Define  $\hat{f}_{2n+1} = \hat{f}_{2n} \cup \{(\kappa^{-1}(n), \pi_1(\tilde{e}'_{2n}))\}$ ,  $\tilde{\mathcal{E}}_{2n+1} = \tilde{\mathcal{E}}_{2n[\tilde{e}'_{2n}]}$  and  $\tilde{\mathcal{E}}'_{2n+1} = \tilde{\mathcal{E}}'_{2n[\tilde{e}'_{2n}]}$ .

for  $2n+2$ : If  $\kappa^{-1}(n) \notin \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']}) \vee \hat{f}_{2n}^{-1}(\kappa^{-1}(n))$  is defined, then  $\hat{f}_{2n+1} = \hat{f}_{2n}$ ,  $\tilde{\mathcal{E}}_{2n+1} = \tilde{\mathcal{E}}_{2n}$  and  $\tilde{\mathcal{E}}'_{2n+1} = \tilde{\mathcal{E}}'_{2n}$ .

If  $\kappa^{-1}(n) \in \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']}) \wedge \hat{f}_{2n}^{-1}(\kappa^{-1}(n))$  is undefined, then  $(\kappa^{-1}(n), \star_2^n \star_1 \star_1 \bullet) \in \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{2n})$ . Hence, there is  $\tilde{e}'_{2n}$  with the same label such that  $(\tilde{\mathcal{E}}_{2n[\tilde{e}'_{2n}]}, \tilde{\mathcal{E}}'_{2n[\tilde{e}'_{2n}]}) \in \mathcal{R}_b$ , where  $\tilde{e}'_{2n} = (\kappa^{-1}(n), \star_2^n \star_1 \star_1 \bullet)$ . Define  $\hat{f}_{2n+1} = \hat{f}_{2n} \cup \{(\pi_1(\tilde{e}'_{2n}), \kappa^{-1}(n))\}$ ,  $\tilde{\mathcal{E}}_{2n+1} = \tilde{\mathcal{E}}_{2n[\tilde{e}'_{2n}]}$  and  $\tilde{\mathcal{E}}'_{2n+1} = \tilde{\mathcal{E}}'_{2n[\tilde{e}'_{2n}]}$ .

$\hat{f}$  is defined by  $\hat{f} = \bigcup_{n \in \mathbb{N}} \hat{f}_n$ .

$\hat{f}$  is a partial function, since  $\hat{f}_i(\kappa^{-1}(n))$  is defined implies that its corresponding refinement  $(\mathcal{E}_i)$  does not possess events labelled by  $\mu(1, \tilde{l}(\kappa^{-1}(n)), n)$ . Hence, it is not defined twice. Moreover,  $\hat{f}$  is a labelling preserving isomorphism between  $\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}) \rightarrow \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}'_{[e']})$ , since every event of both sides will be considered in the definition.

By definition, the restriction of  $\hat{f}$  to  $\tilde{J} \cap \text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]})$  is equal to  $\tilde{f}$  if it is restricted to this set. This restriction constraint also holds for  $\hat{f}^{-1}$ , since otherwise  $\underline{Ref}(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\vartheta})$  and  $\underline{Ref}(\tilde{\mathcal{E}}'_{[e']}, \tilde{\vartheta}')$  can not be bisimilar.

It remains to prove that  $(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\mathcal{E}}'_{[e']}, \hat{f}) \in \mathcal{R}_{FIC}$ . Therefore, let  $\hat{I} \in \mathcal{P}_{fin}(\text{init}_{\text{Obs}}(\tilde{\mathcal{E}}_{[\tilde{e}]}))$ . Define  $m = \max(\{2n+1 \in \mathbb{N} \mid \kappa^{-1}(n) \in \hat{I}\})$  and  $\hat{J} = \text{dom}(\hat{f}_m)$ . Furthermore,  $(\tilde{\mathcal{E}}_m, \tilde{\mathcal{E}}'_m) \in \mathcal{R}_b$  and  $\tilde{\mathcal{E}}_m, \tilde{\mathcal{E}}'_m$  satisfies the requirements. Hence,  $(\tilde{\mathcal{E}}_{[\tilde{e}]}, \tilde{\mathcal{E}}'_{[e']}, \hat{f}) \in \mathcal{R}_{FIC}$ .

The third requirement of the FIC-bisimulation follows by symmetrical arguments. Thus we have proved that  $Ref_{FIC}$  is a FIC-bisimulation.

The construction of a function  $f$  such that  $(\mathcal{E}, \mathcal{E}', f) \in Ref_{FIC}$  is analogous to the construction of  $\hat{f}$ . Hence,  $\mathcal{E} \sim_{FIC} \mathcal{E}'$ .  $\square$