

## **Event Structures for Arbitrary Disruption**

### **Harald Fecher**

*Christian Albrecht Universität zu Kiel*  
*Technische Fakultät, Institut für Informatik,*  
*Hermann-Rodewaldstr. 3, 24118 Kiel, Germany*  
*hf@informatik.uni-kiel.de*

### **Mila Majster-Cederbaum**

*Universität Mannheim*  
*Fakultät für Mathematik und Informatik,*  
*D7, 27, 68131 Mannheim, Germany*  
*mcb@pi2.informatik.uni-mannheim.de*

---

**Abstract.** In process algebras that allow for some form of disruption, it is important to state when a process terminates. One option is to include a termination action  $\surd$ . Another approach is that the ‘final’ executed action of a process terminates the process. The semantics of the former approach has been investigated in the literature in detail, e.g. by providing consistent true-concurrency and operational descriptions. The ‘final’ executed action termination approach, which is more adequate for modelling, still lacks the existence of a detailed true concurrent description. In order to give a true concurrency model of such a termination view, we introduce a new class of event structures. This type of event structures models disabling by indicating sets of precursor events. We show that the introduced class of event structures has more expressive power with respect to event traces than the common event structures. We also give a classification of the expressive power in terms of sets of event traces. A consistency result of an operational and a denotational semantics is shown.

**Keywords:** event structures, disruption, termination, expressive power, process algebra

---

Address for correspondence: Harald Fecher, Christian Albrecht Universität zu Kiel,  
Technische Fakultät, Institut für Informatik,  
Hermann-Rodewaldstr. 3, 24118 Kiel, Germany  
phone: 0049-431-8803733, fax: 0049-431-8807617,  
hf@informatik.uni-kiel.de

## 1. Introduction

Disrupt mechanisms are important to model many realistic systems and have hence found their way into various process algebras [2, 3, 14, 15]. The disrupt operator of LOTOS [11], called disabling operator, is denoted by  $B_1 [ > B_2$ . Here, any action executed by  $B_2$  disables  $B_1$  as long as  $B_1$  has not terminated. It is important to understand disruption in order to handle timeout adequately. Timeouts are an important concept in many applications.

For the description of the disrupt operator, in the definition of an operational semantics it is necessary to specify when a process terminates. This can be achieved in two ways:

- By introducing an additional internal action  $\surd$ , which will be executed in order to terminate the process. An additional syntactical expression  $\mathbf{1}$  is also provided to indicate the process that may terminate immediately. In particular the process consisting of action  $a$  results in  $\mathbf{1}$  by executing  $a$  and thereafter action  $\surd$  can be executed. This approach is for example taken in the case of LOTOS [11].
- An alternative approach to deal with termination is to specify that a process terminates when it executes its ‘final’ action [8, 2]. For example the process  $a \parallel b$ , which denotes the parallel composition of actions  $a$  and  $b$ , terminates by executing  $a$  if  $b$  was executed before or it terminates by executing  $b$  if  $a$  was executed before. In this approach, which is called *fa-approach* in the following, there is no need to extend the syntactical expressions by further expressions, as  $\mathbf{1}$ , to handle termination.

The  $\surd$ -approach can be easily imbedded in the fa-approach by using an additional action  $\surd$ . For example, a process  $B$  of the  $\surd$ -approach corresponds to the process  $B; \surd$ , where  $;$  denotes the sequential composition. On the other hand, the fa-approach allows us to specify a broader class of models as can be seen in the following example:

**Example 1.1.** We want to model a nuclear power plant. Let  $P_{sd}$  be a process that controls the shut down of the reactor and let  $P_{nr}$  be the process that describes the normal running behavior of the reactor. The process  $P_{nr}$  is controlled automatically and only informs the environment about its activity. In other words, in our model the actions of process  $P_{nr}$  are observable but not accessible for the environment and in particular they must not be used for synchronization outside  $P_{nr}$ . In process  $P_{nr}$  an action *stop* indicates that the normal running is terminated and the control is given immediately to the process that controls the shut down of the reactor. Then a simple specification of a nuclear power plant is given by

$$P_{nr}; P_{sd}$$

where  $;$  denotes the sequential composition. A more realistic specification of a nuclear power plant may invoke the shut down process  $P_{sd}$  for various other reasons, e.g. if the temperature reaches a critical point. In addition, the shutting down may be combined with other activities as, e.g., setting an alarm off.

Let us consider a nuclear power plant with the action *stop* and a temperature triggered shut down that also invokes an alarm. Then in any system run either a normal termination of  $P_{nr}$  by *stop* or a disruption of  $P_{nr}$  by an external critical temperature message may happen, but it should be prevented that both occur. In particular, once the stop action is executed no alarm should be set off. Let action  $t$  denote that the temperature of the reactor reaches a critical point and let action  $a$  denote that an alarm is set off. The natural representation of the reactor control in LOTOS is given by

$$((P_{nr} [ > t); P_{sd}) \parallel_{\{t\}} (t; a)$$

where  $B_1 \parallel_A B_2$  denotes the parallel execution of  $B_1$  and  $B_2$  with synchronization on the actions in  $A$ . However, according to the semantics of LOTOS,  $t$  can happen after *stop*. This originates in the fact that the  $\surd$  action and not the *stop* action terminates  $P_{nr}$ . Hence, an alarm with expensive consequences may be unnecessarily set off.

Furthermore, the fa-approach is more appropriate to model timeout expressions in time extensions. For example, we want to model a timeout operator where a timeout at time  $t$  shall occur during the execution of process  $B$  as long as  $B$  is not terminated. In the  $\surd$ -approach the environment has no means to prevent the timeout, since the  $\surd$ -action is an internal action and therefore the system can delay the termination until the timeout takes place. The only possibility to circumvent this situation is to use an urgency approach, i.e. time may not proceed if for example a  $\surd$ -action is enabled. But urgency increases the complexity and should be avoided if possible. On the other hand, it is easily seen that the fa-approach is suitable to model the mentioned timeout operator.

Event structures are a suitable model for the denotational semantics of, e.g., process algebras. In particular they are useful to give a deeper insight to the causality of the event-execution of processes. For example, this was helpful to establish an operational semantics of process algebras with respect to action refinement, e.g. [13] and the citation within. For the  $\surd$ -action termination approach, [22] provided an event based denotational semantics that is consistent with the operational one. But it is not clear how to obtain an event based denotational semantics for a process algebra that contains a disrupt operator and that follows the fa-philosophy: *Prime event structures* [26], *flow event structures* [12] and *stable event structures* [29] require a symmetrical conflict relation which makes it hard to model disruption.

*Extended bundle event structures* [22], which are used to give a denotational semantics to LOTOS, *dual event structures* [21] and *asymmetric event structures* [7] allow the modelling of disruption, since the symmetry condition for the conflict relation is dropped. *Inhibitor event structures* [6] can also model disruption. If (and how) these types of event structures could be used to define a denotational semantics that also incorporates the fa-philosophy is highly questionable. E.g. it is not possible to model the process  $(a \parallel b) [> c]$  with only three events.

Also *configurations* [17] do not provide a smooth way to model disruption. Consider for example the process  $\tilde{B}$ , which consists of the disruption of  $a; b$  ( $a$  sequentially followed by  $b$ ) by the action  $c$  (i.e.  $\tilde{B} = (a; b) [> c]$ ). An intuitive approach is to assume that  $\tilde{B}$  has three events denoted by  $a, b, c$ . Then the sets  $\emptyset, \{a\}, \{c\}, \{a, b\}$  can be considered as configurations, but what about  $\{a, c\}$ ? Assuming it is not a configuration contradicts the existing execution  $\xrightarrow{a} \xrightarrow{c}$ . On the other hand, assuming that  $\{a, c\}$  is also a configuration leads to the interpretation that the execution  $\xrightarrow{c} \xrightarrow{a}$  is legal<sup>1</sup>, which contradicts the branching structure of  $\tilde{B}$ , since after the disruption (by  $c$ ) no further actions from the left process may be executed. Event automata [28], which consist of a set of configuration with an explicit event execution relation between the configurations, and *local event structures* [20], which are event automata where the execution relation also considers step execution, can model disruption with respect to the fa-approach. But they do not describe the dependency between the events intensionally, i.e. they do not describe causalities or conflicts between events by using relations on events. An intentional representation has usually the advantage of making the extension with time aspects less expensive.

In [19] a logical approach to causalities of events is presented, where each event is assigned to a formula, which indicates when this event is enabled. In this approach, it is not easy to model disabling.

---

<sup>1</sup>by the definition of [17]

Another drawback is that it is hard to find a suitable graphical notation for this approach. On the other hand most classes of event structures have a suitable graphical notation, which increases readability.

In this paper, we present a new class of event structures that is suitable to model the fa-philosophy, introduce a graphical representation for these event structures and study their properties. This new class of event structures, called *precursor event structures*, is a generalization of Winskel's event structures [29] in the sense that a relation  $\succ$  between sets of events and events, i.e.  $\succ \subseteq \mathcal{P}(E) \times E$ , is used to model disabling. The meaning of this relation is analogous to the meaning of the causality relation of Winskel's event structures [29], i.e. a condition  $(Z \succ e)$  indicates that event  $e$  is disabled in a system run if every event of  $Z$  appears in the system run.

The result of this paper are:

- We present an fa-based denotational semantics of a process algebra that contains disruption in terms of precursor event structures and we show its consistency with an operational semantics.
- We show that the class of precursor event structures has more expressive power with respect to event traces than the class of prime [26], flow [12], stable [29], bundle [23], extended bundle [22] and dual event structures [21].
- We give a classification of the expressive power of the class of precursor event structures in terms of sets of event traces.

Parts of the results of this paper appeared in [16].

The paper is organized as follows. Section 2 presents the syntax and Section 3 presents the operational semantics of our process algebra. The denotational semantics is given in Section 4, where also the class of precursor event structures is introduced. This section also examines the expressive power of this class of event structures and contains the consistency result that the transition system derived from the denotational semantics is bisimilar to the operational semantics.

## 2. Syntax of the Process Algebra

Let  $\tau$  denote the *internal action*. Furthermore, let  $\text{Obs}$  be a set such that  $\tau \notin \text{Obs}$ . We call  $\text{Obs}$  the set of *observable actions*. The *set of all actions*  $\mathcal{Act}$  is defined by  $\mathcal{Act} = \{\tau\} \cup \text{Obs}$ . A relabelling function  $f$  is a function from  $\mathcal{Act}$  to  $\mathcal{Act}$  such that  $f(\tau) = \tau$ . We denote the set of all labelling functions by  $\mathcal{F}_L$ . Furthermore, assume a fixed countable set of *process variables*  $\text{Var}$  which is disjoint from  $\mathcal{Act}$ .

The process algebra expressions  $\text{EXP}$  are defined by the following BNF-grammar.

$$B ::= \mathbf{0} \mid a \mid B + B \mid B; B \mid B [ > B \mid B \parallel_A B \mid B[f] \mid B \setminus A \mid x$$

where  $f \in \mathcal{F}_L$ ,  $x \in \text{Var}$ ,  $a \in \mathcal{Act}$  and  $A \subseteq \text{Obs}$ . A *process with respect to*  $\text{EXP}$  is a pair  $\langle \text{decl}, B \rangle$  consisting of a declaration  $\text{decl} : \text{Var} \rightarrow \text{EXP}$  and an expression  $B \in \text{EXP}$ . Let  $\text{PA}$  denote the set of all processes. We sometimes call an expression  $B \in \text{EXP}$  a process if  $\text{decl}$  is clear from the context.

The expressions have the following intuitive meaning:  $\mathbf{0}$  is the inactive process, i.e. it cannot execute any action.  $a$  is the process that executes  $a$  and terminates.  $B_1 + B_2$  is a choice between the behaviors described by  $B_1$  and  $B_2$ .  $B_1; B_2$  is the sequential composition, i.e.  $B_1$  proceeds until it terminates, after which  $B_2$  takes over.  $B_1 [ > B_2$  is the disruption of  $B_1$  by  $B_2$ , i.e. any action from  $B_2$  disables  $B_1$  as long

as  $B_1$  has not terminated.  $B_1 \parallel_A B_2$  describes the parallel execution of  $B_1$  and  $B_2$  where both processes have to synchronize on actions from  $A$ . The process terminates if both sides terminate in the case of synchronization or if one terminates and the other one has already terminated. The relabelling process  $B[f]$  executes action  $f(a)$  if  $B$  executes action  $a$ . The restriction process  $B \setminus A$  executes action  $a$  if  $B$  executes action  $a$  provided  $a$  is not contained in  $A$ . The behavior of  $x$  is given by the declaration.

Process algebras, like [18, 11], that are based on the presented synchronization and contain an expression for a termination process (denoted by  $\mathbf{1}$ ) can model a restriction operator in terms of the parallel operator ( $B \parallel_A \mathbf{1}$ ). Since we do not introduce an expression for a termination process, we include the restriction operator, as in [9, 25]. As it turns out, the restriction operator plays also a crucial role for the operational definition of the parallel operator in our setting.

### 3. Operational Semantics for PA

The operational semantics of a process is given by a transition system.

#### Definition 3.1. (Transition System)

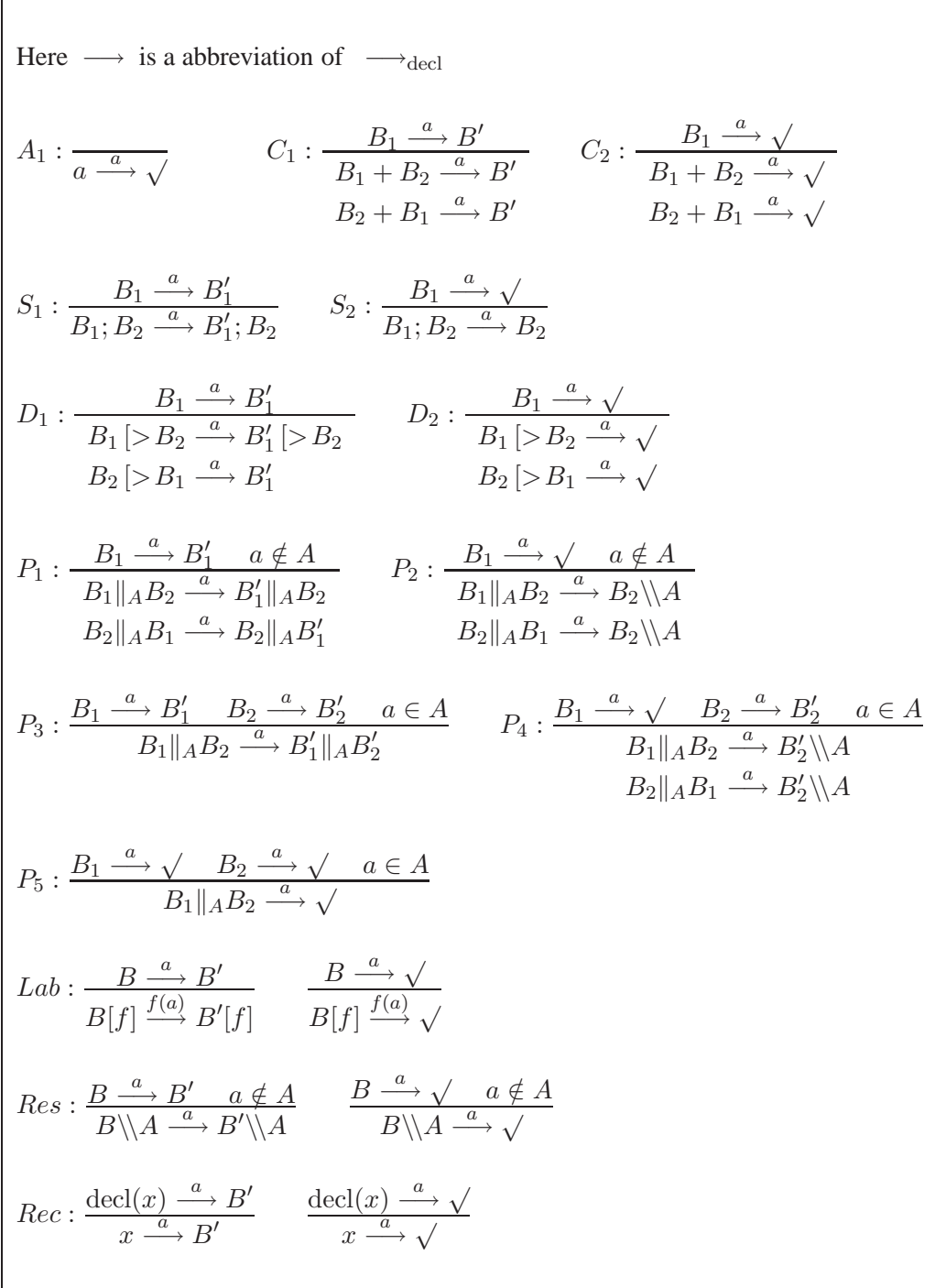
A *labelled transition system* (with predicates) is a tuple  $(S, L, \longrightarrow, \{P_i \mid i \in \mathcal{I}\}, \bar{s})$  with

- $S$ , a non-empty set of states
- $L$ , a set of labels
- $\longrightarrow \subseteq S \times L \times S$ , a transition relation
- $\forall i \in \mathcal{I} : P_i \subseteq S$ , a collection of predicates over the predicate index set  $\mathcal{I}$
- $\bar{s} \in S$ , the initial state.

We will write  $p \xrightarrow{a} q$  rather than  $(p, a, q) \in \longrightarrow$ .

As stated in the introduction we adopt the philosophy that the ‘final’ action terminates the process. Therefore, we have to distinguish between ‘final’ actions and ‘non-final’ actions. In transition systems, the fa-philosophy is usually modelled by using a predicate  $\xrightarrow{a} \surd$  with respect to action  $a$  [8, 2], where  $B \xrightarrow{a} \surd$  indicates that  $B$  terminates by executing  $a$ . Instead of using a transition system of the form  $(S, L, \longrightarrow, \{\xrightarrow{a} \surd \mid a \in \text{Act}\}, \bar{s})$ , we encode the predicate directly in the transition relation by  $\longrightarrow \subseteq S \times L \times (S \cup \{\surd\})$ , where  $B \xrightarrow{a} \surd$  means  $B \in \xrightarrow{a} \surd$ . In the following, a transition system will be a quadruple  $(S, L, \longrightarrow, \bar{s})$  where  $\longrightarrow \subseteq S \times L \times (S \cup \{\surd\})$ . The transition rules of  $\longrightarrow_{\text{decl}} \subseteq \text{EXP} \times \text{Act} \times (\text{EXP} \cup \{\surd\})$  with respect to  $\text{decl} : \text{Var} \rightarrow \text{EXP}$  are presented in Figure 1.

In the following, we explain the rules which deviate from the standard ones: The process  $a$  can execute  $a$  and terminates by executing this action. The process that can only execute action  $a$  and evolves into a non-terminated process that is unable to proceed can be modelled, for example, by  $a; \mathbf{0}$ . The transition rule for the choice operator is the standard CCS-rule [25]. If the first process of the sequential composition terminates by executing  $a$  (rule  $S_2$ ), then  $B_1; B_2$  evolves to  $B_2$  by executing  $a$ . The rules for the disrupt operator are as expected, in particular if  $B_1$  terminates by executing  $a$  then so does  $B_1 [ > B_2$ .

Figure 1. Transition Rules for  $\longrightarrow_{\text{decl}}$

In the case of the parallel operator, we distinguish the cases whether the subprocesses terminate by executing the actions or not. The second rule states that if a subprocess terminates by executing a non-synchronizing action, then this process has to be removed and all actions in the synchronization set have to be forbidden for the remaining process. In the case that both processes terminate by executing an action from the synchronization set, the whole process terminates by executing this action.

The rules for the relabelling operator and the restriction operator only depend on the action name and preserve termination, as expected.

## 4. Denotational Semantics for PA

Classical event structures are not appropriate to define an fa-based denotational semantics for our process algebra. To make this clear, we consider *dual event structures*<sup>2</sup> [21], which are strictly more expressive than prime, flow, stable and extended bundle event structures [21]. Before we present the definition of dual event structures we introduce the following notations:  $\mathcal{P}(M)$  denotes the powerset of  $M$ ,  $\mathcal{P}_f(M)$  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  $M'_1$ . 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  $\Rightarrow f(m_1) = f'(m'_1)$ . Function  $\pi_i$  denotes the projection to the  $i^{\text{th}}$  component. 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  $\forall e, e' \in \mathcal{U} : (e, e'), (\star, e), (e, \star) \in \mathcal{U}$  and  $\bullet \in \mathcal{U}$  and  $\star \notin \mathcal{U}$ . The constraints on set  $\mathcal{U}$  result from technical reasons, i.e. they guarantee the well-definedness of the presented operators (Subsection 4.2).

### Definition 4.1. (Dual event structure)

A dual event structure  $\mathcal{E}^d = (E^d, \rightsquigarrow^d, \mapsto^d, l^d)$  is an element of  $\mathcal{P}(\mathcal{U}) \times \mathcal{P}(\mathcal{U} \times \mathcal{U}) \times \mathcal{P}(\mathcal{P}(\mathcal{U}) \times \mathcal{U}) \times (\mathcal{U} \rightarrow \text{Act})$  such that

- $\rightsquigarrow^d \subseteq (E^d \times E^d)$  and  $\rightsquigarrow^d$  is irreflexive
- $\mapsto^d \subseteq \mathcal{P}(E^d) \times E^d$
- $\text{dom}(l^d) = E^d$

A sequence of distinct events  $(e_1, \dots, e_n)$  is an *event trace* of  $\mathcal{E}^d$  if and only if

- every event is well caused, i.e.  $\forall i < n : \forall X_i : X_i \mapsto^d e_{i+1} \Rightarrow X_i \cap \{e_1, \dots, e_i\} \neq \emptyset$  and
- every event is not disabled at its execution position, i.e.  $\forall i < n : \forall j < i : \neg(e_{i+1} \rightsquigarrow^d e_{j+1})$ .

The intuitive meaning of a dual event structure is the following: If  $e$  is in conflict to  $e'$ , i.e.  $e \rightsquigarrow^d e'$ , then  $e'$  disables  $e$  forever, but not vice versa.  $X \mapsto^d e$  means that before  $e$  may be executed an event from  $X$  has to be executed. The labelling function indicates which action is observable when the event is executed. In dual event structures, it is implicitly assumed that an event is disabled by its execution.

<sup>2</sup>Dual event structures are obtained from extended bundle event structures [22] by dropping the bundle stability constraint.

Usually, termination is modelled in dual event structures by introducing events labelled with  $\surd$ , which corresponds to the operational semantics using an additional internal action  $\surd$ . In the case of the fa-philosophy, it is more natural to encode termination information in event structures by predicates on events in order to stay close to the operational semantics.

However, event structures including information about termination are not powerful enough to handle disruption in an fa-setting, which is illustrated in the following: The process  $((a \parallel_{\emptyset} b) [ > c ]; d)$  can be modelled by dual event structures if we base our modelling on the operational semantics of LOTOS<sup>3</sup>. This is done by using an event  $e'$  labelled by  $\tau$  and by requesting  $e \rightsquigarrow e'$ , where  $e$  is the event corresponding to (labelled by)  $c$ . Dual event structures are not appropriate to model the above expression in an fa-setting<sup>4</sup>: If we put  $c$  in conflict with  $a$ , then  $c$  can be disabled before  $b$  happens and by symmetry the same argument holds for  $b$ . But  $c$  has to be in conflict with some action since otherwise  $c$  remains enabled after the execution of  $a$  and  $b$ .

Therefore, a conflict relation that is based on a binary relation on events is not appropriate to model this kind of disrupt operator in the context of the fa-philosophy. In the following, we introduce a class of event structures, which allows for the modelling of  $((a \parallel_{\emptyset} b) [ > c ]; d)$  in an fa-setting by using a relation between sets of events and events.

#### 4.1. Precursor event structures

*Precursor event structures* are a generalization of Winskel's event structures [29]: They contain an additional set of event sets ( $T$ ) to model termination. Furthermore, sets of events (rather than single events) disable other events.

##### Definition 4.2. (prees)

A *precursor event structure* (prees)  $\mathcal{E} = (E, \succ, \mapsto, T, l)$  is an element of  $\mathcal{P}(U) \times \mathcal{P}(\mathcal{P}(U) \times U) \times \mathcal{P}(\mathcal{P}(U) \times U) \times \mathcal{P}(\mathcal{P}(U)) \times (U \rightarrow \mathcal{A}ct)$  such that

- $\succ \subseteq \mathcal{P}_f(E) \times E$  and  $\forall e \in E : \neg(\emptyset \succ e)$  and  $\forall e \in E : \{e\} \succ e$ <sup>5</sup>
- $\mapsto \subseteq \mathcal{P}_f(E) \times E$
- $T \subseteq \mathcal{P}_f(E)$  and  $\emptyset \notin T$
- $\text{dom}(l) = E$

A sequence of distinct events  $(e_1, \dots, e_n)$  is an *event trace* of  $\mathcal{E}$  if and only if

- every event is well caused, i.e.  $\forall i < n : \exists X_i \subseteq \{e_1, \dots, e_i\} : X_i \mapsto e_{i+1}$ ,
- every event is not disabled at its execution position, i.e.  $\forall i < n : \forall X_i \subseteq \{e_1, \dots, e_i\} : \neg(X_i \succ e_{i+1})$  and
- it is not terminated during its execution, i.e.  $\forall X \subseteq \{e_1, \dots, e_{n-1}\} : X \notin T$ .

<sup>3</sup>After the execution of  $a$  and  $b$  in  $((a \parallel_{\emptyset} b) [ > c ]; d)$ , a  $\tau$ -action can be executed, which disables  $c$ .

<sup>4</sup>Action  $c$  becomes immediately disabled in  $((a \parallel_{\emptyset} b) [ > c ]; d)$  after actions  $a$  and  $b$  have been executed.

<sup>5</sup>Please note that we reversed the order of the conflict relation used in dual event structures.

An event trace  $(e_1, \dots, e_n)$  of  $\mathcal{E}$  is *terminated* if and only if  $\exists X \in T : X \subseteq \{e_1, \dots, e_n\}$ .

Let  $\text{Tr}^e(\mathcal{E})$  denote the set of all event traces of  $\mathcal{E}$ ,  $\text{Tr}_\vee^e(\mathcal{E})$  denote the set of all terminated event traces of  $\mathcal{E}$  and let **PREES** denote the set of all precursor event structures.

We call  $E$  the *set of events*,  $\succ$  the *conflict* relation,  $\mapsto$  the *causality* relation,  $T$  the *set of termination sets* and  $l$  the *action-labelling* function. Set  $X$  is called precursor (with respect to  $e$ ) if  $X \in T$  or  $X \succ e$  or  $X \mapsto e$ .

The intuitive meaning of the conflict relation is that event  $e$  is disabled in a system run (event trace) if there is a conflict precursor  $Z$  of  $e$  ( $Z \succ e$ ) such that the system run contains all elements from  $Z$ . The intuitive meaning of the causality relation  $\mapsto$  and of the set of termination sets  $T$  is similar to  $\succ$ . For example, a system run of a prees is terminated if there is an element  $X$  of  $T$  where every element of  $X$  appears in the system run.

The constraints imposed on the conflict relation are: no event is immediately disabled ( $\neg(\emptyset \succ e)$ ), since otherwise the event can be omitted. Furthermore, the execution of an event disables itself ( $\{e\} \succ e$ ), i.e. every event can happen only once. The constraint  $\emptyset \notin T$  on the set of termination sets ensures that a precursor event structure may not terminate immediately, i.e. it can only terminate by executing an action. It is also reasonable to consider only finite sets of events, since a system run can only contain finite sets.

Note that causality, disabling and termination are orthogonal concepts. In particular, termination cannot be modelled through disabling, since then no difference between deadlock and termination, which is essential for the sequential operator, can be made.

**Example 4.1.** Some precursor event structures are shown in Figure 2. Here, the events are depicted as dots, where their corresponding action names are shown close to the dots (we do not name the events explicitly and identify them with the action names if no confusion arises). The conflict relation is illustrated by wavy lines. More precisely, a conflict  $Z \succ e$  is depicted by a wavy arrow from the elements of  $Z$  to  $e$ . Furthermore, we do not draw the conflict precursors of the form  $Z \succ e$  where  $e \in Z$ . Sometimes, the same (wavy) lines are used in different precursors, for example the wavy line in  $\mathcal{E}_3$  corresponds to  $\{a\} \succ b$  and  $\{b\} \succ a$ . The causality relation is depicted similarly to the conflict relation, except that straight lines are used instead of wavy lines. A termination set  $X$  is displayed by surrounding its events by a closed line. Furthermore, we omit supersets, e.g. in  $\mathcal{E}_1$  we do not draw the causality  $\{a\} \mapsto b$ , since the constraint specified by this causality can be derived from the constraint specified by the causality  $\emptyset \mapsto b$ .

The set of event traces of  $\mathcal{E}_4$  is  $\text{Tr}^e(\mathcal{E}_4) = \{(a), (b), (c), (a, b), (b, a), (a, c), (b, c)\}$ .

Hereafter,  $\mathcal{E}$  is considered to be  $(E, \succ, \mapsto, T, l)$  and  $\mathcal{E}_i$  to be  $(E_i, \succ_i, \mapsto_i, T_i, l_i)$ .

#### 4.1.1. Expressive Power

We say that a set  $\mathbf{V}$  of finite sequences of events (set of event traces) is described by a class of event structures if there is an event structure of this class which has exactly the event traces of  $\mathbf{V}$  as its event traces.

The expressive power of classes of event structures can be measured by comparing the set of event traces described by them. This is a more discriminating measure than that of event automata [27], which are based on configurations. For simplicity, we neglect the termination information when we compare

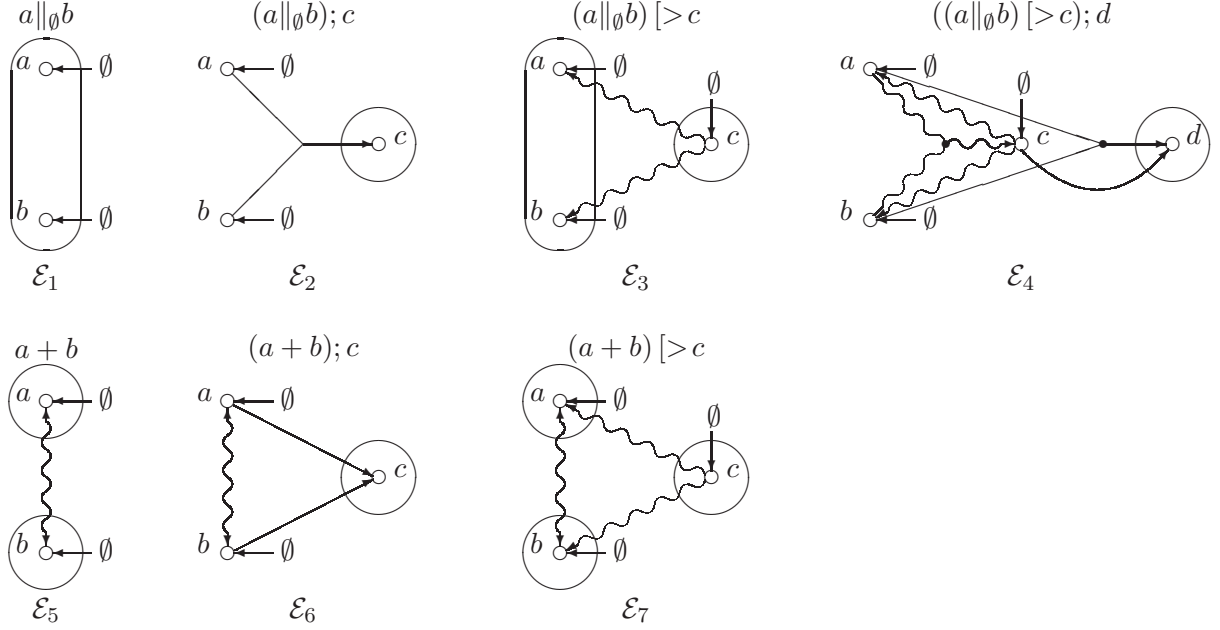


Figure 2. Some Precursor Event Structures

the expressive power. The termination information can be easily included by using a predicate indicating which of the event traces are terminated.

**Theorem 4.1.** Every set of event traces that is described by a prime [26], flow [12], stable [29], bundle [23], extended bundle [22] or dual event structure (Definition 4.1) is also described by an precursor event structures, but not vice versa.

**Proof:**

The proof is given in Appendix A. □

We also give a classification of the set of event traces that corresponds to a prees. In order to have a termination sensitive classification, we first introduce termination sensitive and labelled sets of event traces.

**Definition 4.3.** A *termination sensitive, labelled set of event traces* is a triple  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  such that

- $\mathbf{V}$  is a set of event traces, i.e. a set of finite sequences of distinct elements of  $\mathcal{U}$ ,
- $\overline{\mathbf{V}}$  is the set of the terminated event traces with  $\overline{\mathbf{V}} \subset \mathbf{V} \setminus \{\epsilon\}$  and all elements in  $\overline{\mathbf{V}}$  are maximal (cannot be a proper prefix of an event traces of  $\mathbf{V}$ ) and
- $l : \mathcal{U} \rightarrow \mathcal{Act}$  where every event appears in  $\mathbf{V}$  have to be in the domain of  $l$ .

The set of all termination sensitive, labelled set of event traces are denoted by  $\mathcal{V}$ .

The termination sensitive, labelled set of event traces obtained from a prees  $\mathcal{E}$  is denoted by  $\text{Tr}(\mathcal{E})$ , i.e.  $\text{Tr}(\mathcal{E}) = (\text{Tr}^e(\mathcal{E}), \text{Tr}_{\surd}^e(\mathcal{E}), l)$ .

**Definition 4.4.** A termination sensitive, labelled set of event traces  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  is

- *non-empty* if  $\mathbf{V} \neq \emptyset$ .
- is *prefix closed* if  $\forall (e_1, \dots, e_n) \in \mathbf{V} : (e_1, \dots, e_{n-1}) \in \mathbf{V}$ .
- is *history-order independent* if  $\forall (e_1, \dots, e_{n+1}), (e'_1, \dots, e'_n) \in \mathbf{V} : \{e_1, \dots, e_n\} = \{e'_1, \dots, e'_n\} \Rightarrow (e'_1, \dots, e'_n, e_{n+1}) \in \mathbf{V}$ .
- is *termination-order independent* if  $\forall (e_1, \dots, e_q), (e'_1, \dots, e'_n) \in \mathbf{V} : \{e_1, \dots, e_q\} \subseteq \{e'_1, \dots, e'_n\} \wedge (e_1, \dots, e_q) \in \overline{\mathbf{V}} \Rightarrow (e'_1, \dots, e'_n) \in \overline{\mathbf{V}}$ .
- is *interrupt free* if  $\forall (e_1, \dots, e_{n+1}), (e'_1, \dots, e'_m, e) \in \mathbf{V}, e''_1, \dots, e''_q \in \mathcal{U} : ((e_1, \dots, e_{n+1}, e) \notin \mathbf{V} \wedge \{e'_1, \dots, e'_m\} \subseteq \{e_1, \dots, e_n\} \wedge \{e_1, \dots, e_{n+1}\} \subseteq \{e''_1, \dots, e''_q\}) \Rightarrow (e''_1, \dots, e''_q, e) \notin \mathbf{V}$ .

The prefix closedness means that each event has to be executed as a singleton, i.e. there is no step execution enforced (this would be for example the case if  $e$  can only execute together with event  $e'$ ). The non-emptiness together with the prefix closedness guarantees that the empty sequence is contained. The history-order independence means that the future behavior only depends on the executed events and not on the order of the executed events. The termination-order independence states that termination is determined by sets of events, i.e. if a set of events indicate termination then every execution sequence that contains the events of the set has to be a terminated execution sequence. The interrupt-freeness states that an event that was enabled and that becomes disabled has to stay disabled forever. Similarly to termination the enabling and the disabling is determined here by sets of events.

**Theorem 4.2.** Let  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  be a termination sensitive, labelled set of event traces. Then  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  is non-empty, prefix closed, history-order independent, termination-order independent and interrupt free if and only if  $\exists \mathcal{E} \in \text{PREES} : \mathbf{V} = \text{Tr}^e(\mathcal{E}) \wedge \overline{\mathbf{V}} = \text{Tr}_{\surd}^e(\mathcal{E})$ .

**Proof:**

The proof is given in Appendix A. □

#### 4.1.2. Transition Systems from PREES

Here, we describe how to obtain an action-labelled transition system from a prees. This construction will be used to establish a consistency result for the denotational and the operational semantics.

First, we define the initial events of a prees, i.e. those events which are ready to execute and we define a termination predicate to indicate when a prees terminates by executing an event  $e$ :

**Definition 4.5.** Let  $\mathcal{E}$  be a prees. 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

$$\text{init}(\mathcal{E}) = \{e \in E \mid \emptyset \mapsto e\} \qquad \Upsilon(T, e) \iff \{e\} \in T.$$

In order to obtain a transition system from a prees, we define the remainder [4, 23, 24] of a prees with respect to an initial event. The remainder with respect to event  $e$  describes the system after the execution of  $e$ . All events which are disabled by  $e$  are removed. Please remember that  $\{e\} \succ e$ , hence  $e$  disables itself. After the execution of  $e$ , we remove also those precursors that contain a disabled event, since these precursors cannot be contained in further system runs and hence have no impact on the behavior.

**Definition 4.6. (Remainder of a prees)**

Let  $\mathcal{E} \in \mathbf{PREES}$  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

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

Please note that the remainder  $\mathcal{E}_{[e]}$  is a prees whenever  $\neg\Upsilon(T, e)$ . In the other case, the empty set is an element of the set of termination sets and therefore the remainder is not a prees. This is useful for our theory, since we forbid further event execution after termination. The definition of the remainder coincides with the definition of event traces in the following sense:

**Proposition 4.1.** Suppose  $\mathcal{E} \in \mathbf{PREES}$ . Then for all  $e_1, \dots, e_n \in \mathcal{U}$  we have

$$(e_1, \dots, e_n) \in \text{Tr}^e(\mathcal{E}) \iff \mathcal{E}_{[e_1] \dots [e_n]} \text{ is defined.}$$

**Proof:**

The proof is given in Appendix A. □

The remainders are used in the following definition to obtain an action based interleaving semantics for **PREES**.

**Definition 4.7.** The transition relation  $\hookrightarrow \subseteq \mathbf{PREES} \times \text{Act} \times (\mathbf{PREES} \cup \{\checkmark\})$  is defined by  $\hookrightarrow = \{(\mathcal{E}, l(e), \mathcal{E}_{[e]}) \mid e \in \text{init}(\mathcal{E}) \wedge \neg\Upsilon(T, e)\} \cup \{(\mathcal{E}, l(e), \checkmark) \mid e \in \text{init}(\mathcal{E}) \wedge \Upsilon(T, e)\}$ .

The transition system obtained from  $\mathcal{E}_4$  of Figure 2 is presented in Figure 3.

**4.1.3. Complete Partial Order ( $\trianglelefteq$ ) on PREES**

In order to give a denotational semantics to PA we turn **PREES** into an  $\omega$ -complete partial order. The order on **PREES** is given by:

**Definition 4.8.** Let  $\mathcal{E} \in \mathbf{PREES}$  and  $E' \in \mathcal{P}_f(\mathcal{U})$  such that  $E' \subseteq E$ . Then the restriction of  $\mathcal{E}$  to  $E'$  is

$$\mathcal{E} \upharpoonright E' = (E', \succ \cap (\mathcal{P}_f(E') \times E'), \mapsto \cap (\mathcal{P}_f(E') \times E'), T \cap \mathcal{P}_f(E'), l \upharpoonright E').$$

A prees  $\mathcal{E}_1$  is smaller than a prees  $\mathcal{E}_2$ , written  $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ , if  $E_1 \subseteq E_2$  and  $\mathcal{E}_1 = \mathcal{E}_2 \upharpoonright E_1$ .

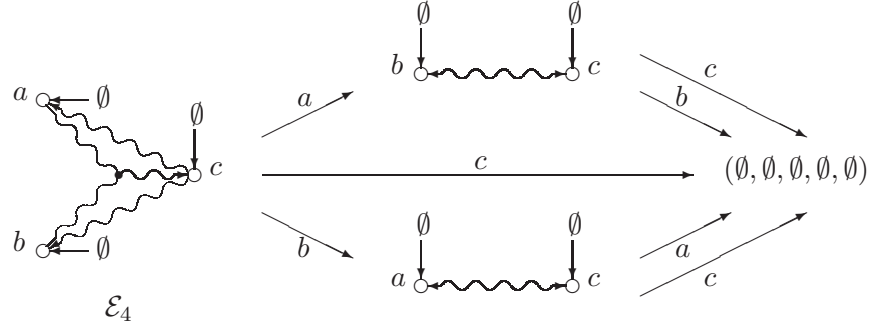


Figure 3. Transition System Derived from PREES

Please note, that the above definition is an adaption of the standard order on Winskel's event structures [29]. In particular,  $\succ_1 = \succ_2 \cap (\mathcal{P}_f(E_1) \times E_1)$  holds whenever  $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ . It is easily seen that the restriction of a prees is again a prees.

**Theorem 4.3.** The set of all prees ordered by  $\trianglelefteq$  is an  $\omega$ -complete partial order, where the least upper bound of an  $\omega$ -chain  $(\mathcal{E}_i)_{i \in \mathbb{N}}$  is given by  $\bigsqcup_i \mathcal{E}_i = (\bigcup_i E_i, \bigcup_i \succ_i, \bigcup_i \mapsto_i, \bigcup_i T_i, \bigcup_i l_i)$ .

**Proof:**

The proof is given in Appendix A. □

## 4.2. Operators on PREES

Here, we present the operators on PREES that are later used to define the denotational semantics. For simplicity, it is not explicitly defined in these operators that a termination set disables every other event, since in the meaning of event structures this is implicitly the case.

### Definition 4.9. (Operators on PREES)

Let  $A \subseteq \text{Obs}$ . Then define

$+$  : PREES  $\times$  PREES  $\rightarrow$  PREES with  $\mathcal{E}_1 + \mathcal{E}_2 \simeq (E_1 \cup E_2, \tilde{\succ}, \mapsto_1 \cup \mapsto_2, T_1 \cup T_2, l_1 \cup l_2)$  where

$$\tilde{\succ} = \succ_1 \cup \succ_2 \cup \{(\{e_i\}, e_j) \mid e_i \in E_i \wedge e_j \in E_j \wedge \{i, j\} = \{1, 2\}\}$$

$;$  : PREES  $\times$  PREES  $\rightarrow$  PREES with  $\mathcal{E}_1 ; \mathcal{E}_2 \simeq (E_1 \cup E_2, \tilde{\succ}, \mapsto, T_2, l_1 \cup l_2)$  where

$$\tilde{\succ} = \succ_1 \cup \succ_2 \cup (T_1 \times E_1)$$

$$\mapsto = \mapsto_1 \cup \{(X_2, e_2) \mid X_2 \mapsto_2 e_2 \wedge e_2 \notin \text{init}(\mathcal{E}_2)\} \cup (T_1 \times \text{init}(\mathcal{E}_2))$$

$[>$  : PREES  $\times$  PREES  $\rightarrow$  PREES with  $\mathcal{E}_1 [> \mathcal{E}_2 \simeq (E_1 \cup E_2, \tilde{\succ}, \mapsto_1 \cup \mapsto_2, T_1 \cup T_2, l_1 \cup l_2)$  where

$$\tilde{\succ} = \succ_1 \cup \succ_2 \cup \{(\{e_2\}, e_1) \mid e_1 \in E_1 \wedge e_2 \in E_2\}$$

$\parallel_A : \mathbf{PREES} \times \mathbf{PREES} \rightarrow \mathbf{PREES}$  with  $\mathcal{E}_1 \parallel_A \mathcal{E}_2 = (\tilde{E}, \tilde{\succ}, \tilde{\mapsto}, \tilde{T}, \tilde{l})$  where

$$\begin{aligned} \tilde{E} &= (E_1^f \times \{\star\}) \cup (\{\star\} \times E_2^f) \cup E^s \\ E_i^f &= \{e \in E_i \mid l_i(e) \notin A\} \\ E^s &= \{(e_1, e_2) \in E_1 \times E_2 \mid l_1(e_1) = l_2(e_2) \in A\} \\ \tilde{\succ} &= \{(\tilde{Z}, (e_1, e_2)) \in \mathcal{P}_f(\tilde{E}) \times \tilde{E} \mid \exists i : \pi_i(\tilde{Z}) \succ_i e_i\} \cup \\ &\quad \{(\tilde{Z}, (e_1, e_2)) \in \mathcal{P}_f(\tilde{E}) \times \tilde{E} \mid \exists i : \pi_i(\tilde{Z}) \in T_i \wedge e_i \neq \star\} \\ \tilde{\mapsto} &= \{(\tilde{X}, (e_1, e_2)) \in \mathcal{P}_f(\tilde{E}) \times \tilde{E} \mid \forall i : e_i = \star \vee \exists X_i \subseteq \pi_i(\tilde{X}) : X_i \mapsto_i e_i\} \\ \tilde{T} &= \{\tilde{X} \in \mathcal{P}_f(\tilde{E}) \mid \forall i : \exists X_i \subseteq \pi_i(\tilde{X}) : X_i \in T_i\} \\ \tilde{l}((e_1, e_2)) &= \begin{cases} l_1(e_1) & \text{if } e_2 = \star \\ l_2(e_2) & \text{otherwise} \end{cases} \end{aligned}$$

$\text{Lab} : (\mathbf{PREES} \times \mathcal{F}_L) \rightarrow \mathbf{PREES}$  with  $\text{Lab}(\mathcal{E}, f) = (E, \succ, \mapsto, T, f \circ l)$ .

$\backslash\backslash_A : \mathbf{PREES} \rightarrow \mathbf{PREES}$  with  $\mathcal{E} \backslash\backslash_A = \mathcal{E} \upharpoonright \{e \in E \mid l(e) \notin A\}$

To simplify our presentation we introduce two auxiliary operators:

$\text{Shift}_1 : \mathbf{PREES} \rightarrow \mathbf{PREES}$  with  $\text{Shift}_1(\mathcal{E}) = (\tilde{E}, \tilde{\succ}, \tilde{\mapsto}, \tilde{T}, \tilde{l})$  where

$$\begin{aligned} \tilde{E} &= E \times \{\star\} \\ \tilde{\succ} &= \{(Z \times \{\star\}, (e, \star)) \mid Z \succ e\} \\ \tilde{\mapsto} &= \{(X \times \{\star\}, (e, \star)) \mid X \mapsto e\} \\ \tilde{T} &= \{X \times \{\star\} \mid X \in T\} \\ \tilde{l}(e, \star) &= l(e) \end{aligned}$$

$\text{Shift}_2 : \mathbf{PREES} \rightarrow \mathbf{PREES}$  with  $\text{Shift}_2(\mathcal{E}) = (\tilde{E}, \tilde{\succ}, \tilde{\mapsto}, \tilde{T}, \tilde{l})$  where

$$\begin{aligned} \tilde{E} &= \{\star\} \times E \\ \tilde{\succ} &= \{(\{\star\} \times Z, (\star, e)) \mid Z \succ e\} \\ \tilde{\mapsto} &= \{(\{\star\} \times X, (\star, e)) \mid X \mapsto e\} . \\ \tilde{T} &= \{\{\star\} \times X \mid X \in T\} \\ \tilde{l}(\star, e) &= l(e) \end{aligned}$$

Please note that  $+$ ,  $;$  and  $[>$  are defined if the involved  $\mathcal{E}_1$  and  $\mathcal{E}_2$  have disjoint set of events, i.e.  $E_1 \cap E_2 = \emptyset$ . We are only interested in such cases. The operator symbols are overloaded, which does not cause any problems, since they can be uniquely determined from the context.

We will give some comments on the definition of these operators: For the sequential composition, we have to ensure that all events from the first event structure are disabled in the sequential composition (which is done by  $T_1 \times E_1 \subseteq \tilde{\succ}$ ), since otherwise there exist traces that contain events of the first event structure after the execution of a terminating set of events of the first event structures. For example,  $(a, b, c, d)$  would become a trace of  $((a \parallel_{\emptyset} b) [> c]; d$ , which contradicts the operational semantics. Please note, that the sequential operator is the only operator that generates sets in the left hand side of  $\succ$ , since for the other operators the disabling character of the termination is sufficient. Furthermore, in the sequential composition the initial events of the second process can only be executed if the first process has terminated.

The events of the parallel operator are those which result from synchronization ( $E^s$ ) together with the events that cannot synchronize  $(E_1^f \times \{\star\}) \cup (\{\star\} \times E_2^f)$ . An event  $e = (e_1, e_2)$  of  $\mathcal{E}_1 \parallel_A \mathcal{E}_2$  is disabled if one of its component is disabled. Consequently, no event of a component can synchronize with more than one component, since it disables itself. Furthermore, if one component terminates, then all its events and the events that results from synchronization are disabled. Such a constraint is necessary, since otherwise the denotation of  $(a \triangleright b) \parallel_{\emptyset} c$  would be able to perform action  $b$  after the execution of  $a$ . An event obtained from synchronization is only enabled if its two components are enabled, hence each projection of the precursor to a component has to be a causality for this component. A process is terminated if and only if both sides are terminated, which is defined analogously to the causality relation.

The restriction operator removes all forbidden events, i.e. those labelled with elements from  $A$ . Furthermore, only those precursors are considered that do not contain events that are labelled with elements from  $A$ , since these precursors can never be contained in a system run.

**Lemma 4.1.** All operators of Definition 4.9 are well defined, i.e. they really yield elements of **PREES** (if they are defined), and they are continuous with respect to  $\preceq$ .

**Proof:**

Straightforward, where the *continuity on events* [29] technique can be used to verify continuity.  $\square$

In order to argue that the above operators match the intuition, we additionally introduce operators on termination sensitive, labelled set of event traces and show that they corresponds to the operators on **PREES**. We write  $w_1 w_2$  to denote the concatenation of two finite sequences. The concatenation of two sets of finite sequences  $\mathbf{V}_1$  and  $\mathbf{V}_2$  is defined by  $\mathbf{V}_1 \cdot \mathbf{V}_2 = \{w_1 w_2 \mid w_1 \in \mathbf{V}_1 \wedge w_2 \in \mathbf{V}_2\}$ . Furthermore, the empty sequence is denoted by  $\epsilon$ .

**Definition 4.10. (Operators on  $\mathcal{V}$ )**

Let  $A \subseteq \text{Obs}$ . Then define

$$+ : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V} \text{ with } (\mathbf{V}_1, \overline{\mathbf{V}}_1, l_1) + (\mathbf{V}_2, \overline{\mathbf{V}}_2, l_2) \simeq (\mathbf{V}_1 \cup \mathbf{V}_2, \overline{\mathbf{V}}_1 \cup \overline{\mathbf{V}}_2, l_1 \cup l_2).$$

$$; : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V} \text{ with } (\mathbf{V}_1, \overline{\mathbf{V}}_1, l_1) ; (\mathbf{V}_2, \overline{\mathbf{V}}_2, l_2) \simeq (\mathbf{V}_1 \cup (\overline{\mathbf{V}}_1 \cdot \mathbf{V}_2), \overline{\mathbf{V}}_1 \cdot \overline{\mathbf{V}}_2, l_1 \cup l_2).$$

$$[\triangleright] : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V} \text{ with } (\mathbf{V}_1, \overline{\mathbf{V}}_1, l_1) [\triangleright] (\mathbf{V}_2, \overline{\mathbf{V}}_2, l_2) \simeq (\mathbf{V}_1 \cup ((\mathbf{V}_1 \setminus \overline{\mathbf{V}}_1) \cdot \mathbf{V}_2), \overline{\mathbf{V}}_1 \cup ((\mathbf{V}_1 \setminus \overline{\mathbf{V}}_1) \cdot \overline{\mathbf{V}}_2), l_1 \cup l_2).$$

$$\parallel_A : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V} \text{ with } (\mathbf{V}_1, \overline{\mathbf{V}}_1, l_1) \parallel_A (\mathbf{V}_2, \overline{\mathbf{V}}_2, l_2) \simeq (\hat{\mathbf{V}}, \hat{\overline{\mathbf{V}}}, \hat{l}) \text{ where}$$

$$\begin{aligned} \hat{\mathbf{V}} &= (\{\hat{w}_1^{(0)} \hat{w}_2^{(0)} \hat{w}_1^{(1)} \hat{w}_2^{(1)} \hat{w}_1^{(2)} \hat{w}_2^{(2)} \dots \hat{w}_1^{(n)} \hat{w}_2^{(n)} \hat{w}_1^{(n)} \mid \exists w_1^{(0)} w_3^{(0)} \dots w_1^{(n)} w_3^{(n)} \in \mathbf{V}_1, : \\ &\quad \exists w_2^{(0)} w_4^{(0)} \dots w_2^{(n)} w_4^{(n)} \in \mathbf{V}_2 : \forall j \leq n : \\ &\quad ((w_1^{(j)} = \epsilon \wedge \hat{w}_1^{(j)} = \epsilon) \vee (w_1^{(j)} \in \mathcal{U} \wedge \hat{w}_1^{(j)} = (w_1^{(j)}, \star) \wedge l_1(w_1^{(j)}) \notin A)) \wedge \\ &\quad ((w_2^{(j)} = \epsilon \wedge \hat{w}_2^{(j)} = \epsilon) \vee (w_2^{(j)} \in \mathcal{U} \wedge \hat{w}_2^{(j)} = (\star, w_2^{(j)}) \wedge l_2(w_2^{(j)}) \notin A)) \wedge \\ &\quad ((w_3^{(j)} = \epsilon \wedge w_4^{(j)} = \epsilon \wedge \hat{w}_3^{(j)} = \epsilon) \vee \\ &\quad (w_3^{(j)} \in \mathcal{U} \wedge w_4^{(j)} \in \mathcal{U} \wedge \hat{w}_3^{(j)} = (w_3^{(j)}, w_4^{(j)}) \wedge l_1(w_3^{(j)}) = l_2(w_4^{(j)}) \in A)) \} \end{aligned}$$

$\hat{\overline{\mathbf{V}}}$  is analogously defined as  $\hat{\mathbf{V}}$  except that  $\overline{\mathbf{V}}_i$  is used instead of  $\mathbf{V}_i$

$$\tilde{l}((e_1, e_2)) \simeq \begin{cases} l_1(e_1) & \text{if } e_2 = \star \\ l_2(e_2) & \text{otherwise} \end{cases}$$

$\text{Lab} : \mathcal{V} \times \mathcal{F}_L \rightarrow \mathcal{V}$  with  $\text{Lab}((\mathbf{V}, \overline{\mathbf{V}}, l), f) = (\mathbf{V}, \overline{\mathbf{V}}, f \circ l)$ .

$\backslash\backslash A : \mathcal{V} \rightarrow \mathcal{V}$  with  
 $(\mathbf{V}, \overline{\mathbf{V}}, l) \backslash\backslash A = (\{e_0 \cdots e_n \in \mathbf{V} \mid \forall j \leq n : l(e_j) \notin A\}, \{e_0 \cdots e_n \in \overline{\mathbf{V}} \mid \forall j \leq n : l(e_j) \notin A\}, l)$ .

We give some comments on the definition of these operators. The choice operator combines the possibilities of both arguments. An event trace of the sequential operator is either an event trace of the first process or a concatenation of a terminated event trace of the first process with an event trace of the second process, where the resulting event trace is terminated if the event trace from the second process is terminated. Event traces of the disrupt operator are obtained by taking all event traces of the first process together with the concatenation of event traces from the first and the second process, where the first one must not be terminated (termination disables the disruption). The parallel operator takes those event traces that are obtained by the interleaving of event traces from the first and the second process, where the synchronization condition has to be satisfied. The parallel operator also renames the events in order to be consistent with respect to the event denotation of the  $\mathcal{E}$ -based parallel operator. The relabelling operator just relabels the events and the restriction operator just removes all event traces that contain an event that is labelled with an element from  $A$ .

**Proposition 4.2.** Suppose  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E} \in \mathbf{PREES}$  such that  $E_1 \cap E_2 = \emptyset$  then

$$\begin{aligned} \text{Tr}(\mathcal{E}_1 + \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) + \text{Tr}(\mathcal{E}_2) & \text{Tr}(\mathcal{E}_1 ; \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) ; \text{Tr}(\mathcal{E}_2) & \text{Tr}(\mathcal{E}_1 [> \mathcal{E}_2]) &= \text{Tr}(\mathcal{E}_1) [> \text{Tr}(\mathcal{E}_2)] \\ \text{Tr}(\mathcal{E}_1 \parallel_A \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) \parallel_A \text{Tr}(\mathcal{E}_2) & \text{Tr}(\text{Lab}(\mathcal{E}, f)) &= \text{Lab}(\text{Tr}(\mathcal{E}), f) & \text{Tr}(\mathcal{E} \backslash\backslash A) &= \text{Tr}(\mathcal{E}_1) \backslash\backslash A \end{aligned}$$

**Proof:**

The proof is given in Appendix A. □

### 4.3. Denotational Meaning

First, we define the denotational semantics of expressions (EXP) with respect to variable assignments. Then variable assignments are derived from declarations, which are used to define the denotational semantics of processes (PA).

**Definition 4.11.** Let  $\llbracket \_ \rrbracket_\rho : \text{EXP} \times (\text{Var} \rightarrow \mathbf{PREES}) \rightarrow \mathbf{PREES}$  be defined as follows (where  $\rho : \text{Var} \rightarrow \mathbf{PREES}$ )

$$\begin{aligned} \llbracket a \rrbracket_\rho &= (\{\bullet\}, \{(\{\bullet\}, \bullet)\}, \{(\emptyset, \bullet)\}, \{\{\bullet\}\}, \{(\bullet, a)\}) & \llbracket \mathbf{0} \rrbracket_\rho &= (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset) \\ \llbracket B_1 + B_2 \rrbracket_\rho &= \text{Shift}_1(\llbracket B_1 \rrbracket_\rho) + \text{Shift}_2(\llbracket B_2 \rrbracket_\rho) & \llbracket B_1 ; B_2 \rrbracket_\rho &= \text{Shift}_1(\llbracket B_1 \rrbracket_\rho) ; \text{Shift}_2(\llbracket B_2 \rrbracket_\rho) \\ \llbracket B_1 [> B_2] \rrbracket_\rho &= \text{Shift}_1(\llbracket B_1 \rrbracket_\rho) [> \text{Shift}_2(\llbracket B_2 \rrbracket_\rho)] & \llbracket B_1 \parallel_A B_2 \rrbracket_\rho &= \llbracket B_1 \rrbracket_\rho \parallel_A \llbracket B_2 \rrbracket_\rho \\ \llbracket B[f] \rrbracket_\rho &= \text{Lab}(\llbracket B \rrbracket_\rho, f) & \llbracket B \backslash\backslash A \rrbracket_\rho &= \llbracket B \rrbracket_\rho \backslash\backslash A \\ \llbracket x \rrbracket_\rho &= \rho(x) \end{aligned}$$

**Remark 4.1.**  $\llbracket B \rrbracket_\rho$  is continuous with respect to  $\leq$  for every  $B \in \text{EXP}$ .

Assume  $\text{decl} : \text{Var} \rightarrow \text{EXP}$ . Then define  $\mathcal{F}_{\text{decl}} : (\text{Var} \rightarrow \mathbf{PREES}) \rightarrow (\text{Var} \rightarrow \mathbf{PREES})$  with  $\mathcal{F}_{\text{decl}}(\rho)(x) = \llbracket \text{decl}(x) \rrbracket_\rho$ . From Remark 4.1 it follows that  $\mathcal{F}_{\text{decl}}$  is continuous. Therefore, from the

complete partial order theory [1] we get  $\llbracket \_ \rrbracket : (\text{Var} \rightarrow \text{EXP}) \rightarrow (\text{Var} \rightarrow \text{PREES})$  with  $\llbracket \text{decl} \rrbracket = \text{fix}(\mathcal{F}_{\text{decl}}) = \bigsqcup_n \mathcal{F}_{\text{decl}}^n(\perp)$  is well defined.

**Definition 4.12. (Denotational Semantics)**

Define  $\llbracket \_ \rrbracket : \text{PA} \rightarrow \text{PREES}$  by  $\llbracket \langle \text{decl}, B \rangle \rrbracket = \llbracket B \rrbracket_{\llbracket \text{decl} \rrbracket}$ .

**Example 4.2.** The denotational semantics of some processes is illustrated in Figure 2.

#### 4.4. Consistency of both Semantics

Here, we show that the transition system derived from the denotational semantics and the operational semantics yield bisimilar transition systems.

**Definition 4.13. (Bisimilarity)**

Two transition systems  $(S, L, \longrightarrow, \bar{s})$  and  $(S', L, \longrightarrow', \bar{s}')$  over the same set of labels are *bisimilar* if there is a *bisimulation*, i.e. a relation  $\mathcal{R} \subseteq S \times S'$  such that  $(\bar{s}, \bar{s}') \in \mathcal{R}$  and for which for all  $(s_1, s'_1) \in \mathcal{R}$  we have:

- if  $s_1 \xrightarrow{a} s_2$  then there is  $s'_2$  such that  $(s_2, s'_2) \in \mathcal{R}$  and  $s'_1 \xrightarrow{a'} s'_2$  and  $s_2 = \checkmark \Leftrightarrow s'_2 = \checkmark$
- if  $s'_1 \xrightarrow{a'} s'_2$  then there is  $s_2$  such that  $(s_2, s'_2) \in \mathcal{R}$  and  $s_1 \xrightarrow{a} s_2$  and  $s_2 = \checkmark \Leftrightarrow s'_2 = \checkmark$ .

**Theorem 4.4. (Consistency)**

Suppose  $\langle \text{decl}, B \rangle \in \text{PA}$ . Then the transition systems  $(\text{EXP}, \text{Act}, \longrightarrow_{\text{decl}}, B)$  and  $(\text{PREES}, \text{Act}, \hookrightarrow, \llbracket \langle \text{decl}, B \rangle \rrbracket)$  are bisimilar.

**Proof:**

The proof is given in Appendix B. It uses a new proof technique that can also handle unguarded recursion, which is not the case for example in the consistency result of [4].  $\square$

The two transition systems of Theorem 4.4 are not isomorphic in general. Consider for example  $\text{decl}$  with  $\text{decl}(x) = a; x$ . Then the expression  $x$  yields a finite transition system with respect to  $\longrightarrow_{\text{decl}}$ , whereas  $\llbracket \langle \text{decl}, x \rangle \rrbracket$  yields an infinite transition system with respect to  $\hookrightarrow$ .

## A. Proofs

**Theorem 4.1.** Every set of event traces that is described by a prime [26], flow [12], stable [29], bundle [23], extended bundle [22] or dual event structure (Definition 4.1) is also described by an precursor event structures, but not vice versa.

**Proof:**

From [21] we know that every set of event traces described by an event structure of a cited class is also described by a dual event structure. That a set of event traces described by a dual event structure can also be described by an precursor event structure is shown by mapping the dual event structure  $\mathcal{E}^d = (E^d, \rightsquigarrow^d, \mapsto^d, l^d)$  to  $\Omega(\mathcal{E}^d) = (E^d, \succ, \mapsto, \emptyset, l^d)$  where  $\succ = \{(\{e'\}, e) \mid e \rightsquigarrow^d e' \vee e = e'\}$  and

$\mapsto = \{(X, e) \in \mathcal{P}_f(E^d) \times E^d \mid \forall X^d \in \mathcal{P}(E^d) : X^d \mapsto^d e \Rightarrow X \cap X^d \neq \emptyset\}$ .

It is easily checked that  $\Omega(\mathcal{E}^d) \in \mathbf{PREES}$ .

In the following, we show that  $\mathcal{E}^d$  and  $\Omega(\mathcal{E}^d)$  have the same set of event traces.

Suppose  $(e_1, \dots, e_n)$  is an event trace of  $\mathcal{E}^d$ . Then  $\forall i < n : \{e_1, \dots, e_i\} \mapsto e_{i+1}$ , since  $\forall i < n : \forall X_i : X_i \mapsto^d e_{i+1} \Rightarrow X_i \cap \{e_1, \dots, e_i\} \neq \emptyset$  have to hold. Furthermore,  $\forall i < n : \forall X_i \subseteq \{e_1, \dots, e_i\} : \neg(X_i \succ e_{i+1})$ , since  $\forall i < n : \forall j < i : \neg(e_{i+1} \rightsquigarrow^d e_{j+1})$  have to hold. Hence,  $(e_1, \dots, e_n) \in \text{Tr}^e(\Omega(\mathcal{E}^d))$ .

Suppose  $(e_1, \dots, e_n) \in \text{Tr}^e(\Omega(\mathcal{E}^d))$  then for all  $i$  we have the existence of  $X_i \subseteq \{e_1, \dots, e_i\}$  such that  $X_i \mapsto e_{i+1}$ . Thus,  $\forall X^d \in \mathcal{P}(E^d) : X^d \mapsto^d e_{i+1} \Rightarrow X_i \cap X^d \neq \emptyset$ . Hence,  $\forall X_d : X_d \mapsto^d e_{i+1} \Rightarrow X_d \cap \{e_1, \dots, e_i\} \neq \emptyset$ , which shows the causality constraint. Furthermore,  $\forall X'_i \subseteq \{e_1, \dots, e_i\} : \neg(X'_i \succ e_{i+1})$  holds. Hence  $\forall j < i : \neg(e_{i+1} \rightsquigarrow^d e_{j+1})$ . Thus,  $(e_1, \dots, e_n)$  is an event trace of  $\mathcal{E}^d$ .

We showed that **PREES** can describe all sets of event traces that are describable by dual event structures. On the other hand, the set of event traces obtained from  $\mathcal{E}_4$  of Figure 2 cannot be described by a dual event structure.  $\square$

**Theorem 4.2.** Let  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  be a termination sensitive, labelled set of event traces. Then  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  is non-empty, prefix closed, history-order independent, termination-order independent and interrupt free if and only if  $\exists \mathcal{E} \in \mathbf{PREES} : \mathbf{V} = \text{Tr}^e(\mathcal{E}) \wedge \overline{\mathbf{V}} = \text{Tr}^e_{\setminus}(\mathcal{E})$ .

**Proof:**

We verify every direction separately:

$\Leftarrow$ : Let  $\mathcal{E} \in \mathbf{PREES}$ . The non-emptiness, prefix closedness history-order independence and the termination order independency is easily seen.

Now assume

$$(e_1, \dots, e_{n+1}) \in \text{Tr}^e(\mathcal{E}) \tag{1}$$

$$(e'_1, \dots, e'_m, e) \in \text{Tr}^e(\mathcal{E}) \tag{2}$$

$$(e_1, \dots, e_{n+1}, e) \notin \text{Tr}^e(\mathcal{E}) \tag{3}$$

$$\{e'_1, \dots, e'_m\} \subseteq \{e_1, \dots, e_n\} \wedge \{e_1, \dots, e_n\} \subseteq \{e''_1, \dots, e''_q\}$$

From the definition of event traces of  $\mathcal{E}$  and (3) we obtain

$$(\exists i < n + 1 : \exists X_i \subseteq \{e_1, \dots, e_i\} : \neg(X_i \mapsto e_{i+1})) \vee \tag{4}$$

$$(\exists X \subseteq \{e_1, \dots, e_{n+1}\} : \neg(X \mapsto e)) \vee \tag{5}$$

$$(\exists i < n + 1 : \exists X_i \subseteq \{e_1, \dots, e_i\} : X_i \succ e_{i+1}) \vee \tag{6}$$

$$(\exists X \subseteq \{e_1, \dots, e_{n+1}\} : X \succ e) \tag{7}$$

From (1) we obtain that (4) and (6) is not possible. Furthermore, (5) cannot be valid, since (2) holds. Thus (7) has to hold, i.e. there is  $X \subseteq \{e_1, \dots, e_{n+1}\}$  such that  $X \succ e$ . But this  $X$  shows that  $(e''_1, \dots, e''_q, e) \notin \text{Tr}^e(\mathcal{E})$ , hence the interrupt freeness is proved.

The maximality of the terminated event traces is an immediate consequence of the definition of  $\text{Tr}^e(\mathcal{E})$ .

$\Rightarrow$ : Let  $(\mathbf{V}, \overline{\mathbf{V}}, l)$  have the stated properties. Define  $\mathcal{E} = (\mathcal{U}, \succ, \mapsto, T, l)$  by

$$\begin{aligned}
 \succ &= \{(\{e_1, \dots, e_n\}, e) \mid \exists e'_1, \dots, e'_i : \{e'_1, \dots, e'_i\} \subseteq \{e_1, \dots, e_n\} \wedge (e'_1, \dots, e'_i, e) \in \mathbf{V} \wedge \\
 &\quad (e_1, \dots, e_n) \in \mathbf{V} \wedge (e_1, \dots, e_n, e) \notin \mathbf{V}\} \cup \{(\{e\}, e) \mid e \in \mathcal{U}\} \\
 \mapsto &= \{(\{e_1, \dots, e_n\}, e) \mid (e_1, \dots, e_n, e) \in \mathbf{V}\} \\
 T &= \{\{e_1, \dots, e_n\} \mid (e_1, \dots, e_n) \in \overline{\mathbf{V}}\}.
 \end{aligned}$$

It is easily seen that  $\mathcal{E} \in \mathbf{PREES}$ .

In the following, we show that for all  $e_0, \dots, e_n$  we have  $((e_0, \dots, e_n) \in \mathbf{V} \iff (e_0, \dots, e_n) \in \text{Tr}^e(\mathcal{E})) \wedge ((e_0, \dots, e_n) \in \overline{\mathbf{V}} \iff (e_0, \dots, e_n) \in \text{Tr}_{\checkmark}^e(\mathcal{E}))$ . This is done by induction on  $n$ .

$(e_0, \dots, e_n) \in \mathbf{V} \Rightarrow (e_0, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ : The case if  $n \in \{0, 1\}$  is easily seen. Now suppose  $(e_0, \dots, e_{n+2}) \in \mathbf{V}$ . Then  $(e_0, \dots, e_{n+1}) \in \mathbf{V}$ , since  $\mathbf{V}$  is prefixed closed. And so by induction  $\forall i < n + 1 : \forall X_i \subseteq \{e_0, \dots, e_i\} : \neg(X_i \succ e_{i+1})$ . Now let  $X \subseteq \{e_0, \dots, e_{n+1}\}$ .

Assume that  $X \succ e_{n+2}$ . Then there exists  $e'_0, \dots, e'_j$  and  $e''_0, \dots, e''_k$  such that  $X = \{e''_0, \dots, e''_k\} \wedge \{e'_0, \dots, e'_j\} \subseteq \{e''_0, \dots, e''_k\} \wedge (e'_0, \dots, e'_j, e_{n+2}) \in \mathbf{V} \wedge (e''_0, \dots, e''_k) \in \mathbf{V} \wedge (e''_0, \dots, e''_k, e_{n+2}) \notin \mathbf{V}$ . From the interrupt freeness of  $\mathbf{V}$  we obtain  $(e_0, \dots, e_{n+2}) \notin \mathbf{V}$ , which is a contradiction. Hence,  $\forall X \subseteq \{e_0, \dots, e_{n+1}\} : \neg(X \succ e_{n+1})$ .

The causality constraints are an immediate consequence of the prefix closedness of  $\mathbf{V}$ .

By the maximality constraint on  $\mathbf{V}$  we get  $(e_0, \dots, e_{n+1}) \notin \overline{\mathbf{V}}$ . Hence,  $(e_0, \dots, e_{n+1}) \notin \text{Tr}_{\checkmark}^e(\mathcal{E})$  by induction. Therefore, for all  $X \subseteq \{e_0, \dots, e_{n+1}\} : X \notin T$ , which establish the termination constraint. Hence  $(e_0, \dots, e_{n+2}) \in \mathbf{V}$ .

$(e_0, \dots, e_n) \in \mathbf{V} \Leftarrow (e_0, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ : The case  $n = 0$  is easily seen. Suppose  $(e_0, \dots, e_{n+1}) \in \text{Tr}^e(\mathcal{E})$ . Then we have  $(e_0, \dots, e_n) \in \mathbf{V}$  by induction, since  $(e_0, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ . From the causality constraint of an event trace, we obtain that  $\exists e'_0, \dots, e'_i : \{e'_0, \dots, e'_i\} \subseteq \{e_0, \dots, e_n\} \wedge (e'_0, \dots, e'_i, e_{n+1}) \in \mathbf{V}$ .

Assume that  $(e_0, \dots, e_{n+1}) \notin \mathbf{V}$ . Then by the definition of  $\mathcal{E}$  we have  $\{e_0, \dots, e_n\} \succ e_{n+1}$ . Hence  $(e_0, \dots, e_{n+1}) \notin \text{Tr}^e(\mathcal{E})$ , which is a contradiction. Thus,  $(e_0, \dots, e_{n+1}) \in \mathbf{V}$ .

$(e_0, \dots, e_n) \in \overline{\mathbf{V}} \Rightarrow (e_0, \dots, e_n) \in \text{Tr}_{\checkmark}^e(\mathcal{E})$ : Let  $(e_0, \dots, e_n) \in \overline{\mathbf{V}}$  then we have already shown that  $(e_0, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ . Thus,  $(e_0, \dots, e_n) \in \text{Tr}_{\checkmark}^e(\mathcal{E})$  by the definition of  $T$ .

$(e_0, \dots, e_n) \in \overline{\mathbf{V}} \Leftarrow (e_0, \dots, e_n) \in \text{Tr}_{\checkmark}^e(\mathcal{E})$ : Let  $(e_0, \dots, e_n) \in \text{Tr}_{\checkmark}^e(\mathcal{E})$ . Then by definition there is  $X \subseteq \{e_0, \dots, e_{n-1}\} : (X \cup \{e_n\}) \in T$ . Hence, there exists  $e'_0, \dots, e'_q$  such that  $(e'_0, \dots, e'_q, e_n) \in \overline{\mathbf{V}} \wedge \{e'_0, \dots, e'_q\} = X$ . From the termination order independency we obtain  $(e_0, \dots, e_n) \in \overline{\mathbf{V}}$ .  $\square$

**Theorem 4.3.** Suppose  $\mathcal{E} \in \mathbf{PREES}$ . Then for all  $e_1, \dots, e_n \in \mathcal{U}$  we have

$$(e_1, \dots, e_n) \in \text{Tr}^e(\mathcal{E}) \iff \mathcal{E}_{[e_1] \dots [e_n]} \text{ is defined.}$$

**Proof:**

First we state that  $\mathcal{E}_{[e_1] \dots [e_n]} = (E', \succ', \mapsto', T', l')$  with

$$\begin{aligned}
 E' &= \{e' \in E \mid \forall Z \subseteq \{e_1, \dots, e_n\} : \neg(X \succ e')\} \\
 \succ' &= \{(Z', e') \mid e' \in E' \wedge Z' \subseteq E' \wedge \exists Z : Z \succ e' \wedge Z' = Z \setminus \{e_1, \dots, e_n\}\} \\
 \mapsto' &= \{(X', e') \mid e' \in E' \wedge X' \subseteq E' \wedge \exists X : X \mapsto e' \wedge X' = X \setminus \{e_1, \dots, e_n\}\} \\
 T' &= \{X' \mid X' \subseteq E' \wedge \exists X \in T : X' = X \setminus \{e_1, \dots, e_n\}\} \\
 l' &= l \upharpoonright E'
 \end{aligned}$$

if it is defined. This can be shown by induction on  $n$ , which is omitted here.

The main statement is shown by induction on  $n$ , where the base case  $n = 0$  is easily seen.

$\Rightarrow$ : If  $(e_1, \dots, e_{n+1}) \in \text{Tr}^e(\mathcal{E})$  then  $(e_1, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ . Hence, by induction  $\mathcal{E}_{[e_1] \dots [e_n]}$  is defined.

From the non disabling condition of event traces of  $\mathcal{E}$  we obtain  $e_{n+1} \in E'$ , from the well causality constraint we get  $\emptyset \mapsto' e_{n+1}$  and from the termination constraint we obtain that  $\mathcal{E}_{[e_1] \dots [e_n]} \in \mathbf{PREES}$ .

Thus  $\mathcal{E}_{[e_1] \dots [e_{n+1}]}$  is defined.

$\Leftarrow$ : If  $\mathcal{E}_{[e_1] \dots [e_{n+1}]}$  is defined then  $\mathcal{E}_{[e_1] \dots [e_n]}$  is defined and so by induction  $(e_1, \dots, e_n) \in \text{Tr}^e(\mathcal{E})$ . Therefore, it is only left to show that

- $\forall X \subseteq \{e_1, \dots, e_n\} : \neg(X \succ e_{n+1})$ , which has to be the case, since otherwise  $e_{n+1} \notin E'$ ,
- $\exists X \subseteq \{e_1, \dots, e_n\} : X \mapsto e_{n+1}$ , which has to be the case, since otherwise  $e_{n+1}$  is not enabled in  $\mathcal{E}_{[e_1] \dots [e_n]}$  and
- $\forall X \subseteq \{e_1, \dots, e_n\} : X \notin T$ , which has to be the case, since otherwise  $\mathcal{E}_{[e_1] \dots [e_n]} \notin \mathbf{PREES}$ . □

**Theorem 4.4.** The set of all prees ordered by  $\trianglelefteq$  is an  $\omega$ -complete partial order, where the least upper bound of an  $\omega$ -chain  $(\mathcal{E}_i)_{i \in \mathbb{N}}$  is given by  $\bigsqcup_i \mathcal{E}_i = (\bigcup_i E_i, \bigcup_i \succ_i, \bigcup_i \mapsto_i, \bigcup_i T_i, \bigcup_i l_i)$ .

**Proof:**

It is easily seen that  $\trianglelefteq$  is a partial order with  $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$  as its least element. Furthermore,  $\mathcal{E} = \bigsqcup_i \mathcal{E}_i$  is a prees. In the following we only consider  $T$ . The cases  $\succ$  and  $\mapsto$  follow analogously.

upper bound: Obviously,  $T_j \subseteq \bigcup_i T_i$ .

Let  $X \subseteq \bigcup_i E_i$  such that  $X \subseteq E_j$  and  $\exists i : X \in T_i$ . Thus  $X \subseteq E_i$  and from  $\mathcal{E}_j \trianglelefteq \mathcal{E}_i$  or  $\mathcal{E}_i \trianglelefteq \mathcal{E}_j$  we get  $X \in T_j$ , as required.

least upper bound: Let  $\mathcal{E}'$  be a prees such that  $\mathcal{E}_i \trianglelefteq \mathcal{E}'$  for all  $i \in \mathbb{N}$ . Then  $\bigcup_i E_i \subseteq E'$ .

Let  $X \in T$ . Then  $\exists j : X \in T_j$ . Hence,  $X \in T'$ , since  $\mathcal{E}_j \trianglelefteq \mathcal{E}'$ .

Let  $X \in T'$  such that  $X \subseteq \bigcup_i E_i$ . Since  $X$  is finite, there exists  $j \in \mathbb{N}$  such that  $X \subseteq E_j$ . Hence,  $X \in T_j$ . Then by definition  $X' \in T$ . □

**Proposition 4.2.** Suppose  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E} \in \mathbf{PREES}$  such that  $E_1 \cap E_2 = \emptyset$  then

$$\begin{aligned} \text{Tr}(\mathcal{E}_1 + \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) + \text{Tr}(\mathcal{E}_2) & \text{Tr}(\mathcal{E}_1 ; \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) ; \text{Tr}(\mathcal{E}_2) & \text{Tr}(\mathcal{E}_1 [ > \mathcal{E}_2 ] ) &= \text{Tr}(\mathcal{E}_1) [ > \text{Tr}(\mathcal{E}_2) ] \\ \text{Tr}(\mathcal{E}_1 \parallel_A \mathcal{E}_2) &= \text{Tr}(\mathcal{E}_1) \parallel_A \text{Tr}(\mathcal{E}_2) & \text{Tr}(\text{Lab}(\mathcal{E}, f)) &= \text{Lab}(\text{Tr}(\mathcal{E}), f) & \text{Tr}(\mathcal{E} \setminus A) &= \text{Tr}(\mathcal{E}_1) \setminus A \end{aligned}$$

**Proof:**

We present the proof of  $\text{Tr}(\mathcal{E}_1 \parallel_A \mathcal{E}_2) = \text{Tr}(\mathcal{E}_1) \parallel_A \text{Tr}(\mathcal{E}_2)$ . The other cases are simpler and hence are omitted.

Suppose  $(\tilde{e}_0, \dots, \tilde{e}_n) \in \text{Tr}^e(\mathcal{E}_1 \parallel_A \mathcal{E}_2)$ . Define for  $i \leq n$ :

$$w_1^{(i)} = \begin{cases} e & \text{if } \tilde{e}_i = (e, \star) \\ \epsilon & \text{otherwise} \end{cases} \quad w_3^{(i)} = \begin{cases} e & \text{if } \tilde{e}_i = (e, e') \\ \epsilon & \text{otherwise} \end{cases}$$

Now, we will argue that  $w_1^{(0)}w_3^{(0)} \dots w_1^{(n)}w_3^{(n)} \in \text{Tr}^e(\mathcal{E}_1)$ : the causality and the conflict constraint is easily seen. Suppose the termination constraint is violated, then by the definition of  $\|_A$  all events corresponds to  $E_1$  are disabled, hence  $(\tilde{e}_0, \dots, \tilde{e}_n) \notin \text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$ . Contradiction.

$w_2^{(i)}$  and  $w_4^{(i)}$  can be analogously defined. Hence,  $(\tilde{e}_0, \dots, \tilde{e}_n) \in \text{Tr}(\mathcal{E}_1) \|_A \text{Tr}(\mathcal{E}_2)$  is an immediately consequence of the definitions of  $w_j^{(i)}$ .

It is also straightforwardly checked that every termination trace of  $\text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$  is also a termination trace of  $\text{Tr}(\mathcal{E}_1) \|_A \text{Tr}(\mathcal{E}_2)$ .

Suppose  $\hat{w}_1^{(0)}\hat{w}_2^{(0)}\hat{w}^{(0)} \dots \hat{w}_1^{(n)}\hat{w}_2^{(n)}\hat{w}^{(n)} \in \text{Tr}(\mathcal{E}_1) \|_A \text{Tr}(\mathcal{E}_2)$ . We use induction on  $n$ .

If  $\hat{w}_1^{(n)} = \epsilon$  then  $\hat{w}_1^{(0)}\hat{w}_2^{(0)}\hat{w}^{(0)} \dots \hat{w}_1^{(n)} \in \text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$  by induction. If  $\hat{w}_1^{(n)} = (e, \star)$  then the index of the causality set obtained in  $\mathcal{E}_1$  can be used to obtain a causality set in  $\text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$ . Suppose  $\hat{w}_1^{(0)}\hat{w}_2^{(0)}\hat{w}^{(0)} \dots \hat{w}_1^{(n)} \notin \text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$  because of conflicts. But this cannot be the case, because otherwise  $w_1^{(0)}w_3^{(0)} \dots w_1^{(n)} \in \text{Tr}^e(\mathcal{E}_1)$  would be contradicted by the disabling or by the termination constraint. The termination constraint of  $\hat{w}_1^{(0)}\hat{w}_2^{(0)}\hat{w}^{(0)} \dots \hat{w}_1^{(n)}$  can be easily seen. Hence,  $\hat{w}_1^{(0)}\hat{w}_2^{(0)}\hat{w}^{(0)} \dots \hat{w}_1^{(n)} \in \text{Tr}^e(\mathcal{E}_1 \|_A \mathcal{E}_2)$ .

The other cases,  $\hat{w}_2^{(n)}$  and  $\hat{w}_3^{(n)}$ , are shown analogously.  $\square$

## B. Proof of Theorem 4.4

The main problem in verifying Theorem 4.4 is to handle recursion, since it will be hard to appropriately relate an action execution in the SOS-transition system to an action execution in the corresponding **PREES**-denotation. The technique used in [4] cannot be applied, since there recursion is handled via metric spaces, and not via complete partial orders. This has the consequence that it is not even possible to give a denotation to unguarded processes, since the distance function will not be contractive in these cases. The technique used in [18] has the disadvantage that event identifiers are introduced at the language level and therefore finite systems can never be obtained if recursion is present.

In order to solve the problem of relating action execution to event execution, we first introduce an event-based SOS-transition system, where every action execution is annotated by the event-name of its **PREES**-denotation. Then we show that the event-based transition system of process  $\langle \text{decl}, B \rangle$  is bisimilar to  $(\text{EXP}, \text{Act}, \longrightarrow_{\text{decl}}, B)$  and also to  $(\text{PREES}, \text{Act}, \hookrightarrow, \llbracket \langle \text{decl}, B \rangle \rrbracket)$ . Hence, Theorem 4.4 follows by the transitivity of the bisimilarity [25].

The advantage of our approach is that we can handle unguarded processes.

### B.1. Event Based Transition System.

Let  $\text{EXP}^e$  be the set that contains exactly the elements generated by

$$C ::= B \mid C; B \mid C \text{>} B \mid C \|_A C \mid C[f] \mid C \setminus A \mid [C]_i$$

where  $B \in \text{EXP}$ ,  $f \in \mathcal{F}_L$ ,  $i \in \{1, 2\}$  and  $A \subseteq \text{Obs}$ . Let  $\text{PA}^e = (\text{Var} \rightarrow \text{EXP}) \times \text{EXP}^e$ . Please note that variables are still assigned to the standard expressions. In Figure 4 the event-based transition rules  $\longrightarrow'_{\text{decl}} \subseteq \text{EXP}^e \times (\text{Act} \times \mathcal{U}) \times (\text{EXP}^e \cup \{\sqrt{\phantom{x}}\})$  are presented. In the following  $\bar{C}$  denotes an element of  $\text{EXP}^e \cup \{\sqrt{\phantom{x}}\}$  and  $\bar{B}$  denotes an element of  $\text{EXP} \cup \{\sqrt{\phantom{x}}\}$ .

Here  $\longrightarrow$  is an abbreviation of  $\longrightarrow'_{\text{decl}}$

$$A'_1 : \frac{}{a \xrightarrow{a} \checkmark} \quad C'_1 : \frac{B_1 \xrightarrow{a} C'}{B_1 + B_2 \xrightarrow{(e,*)} [C']_1} \quad C'_2 : \frac{B_1 \xrightarrow{a} \checkmark}{B_1 + B_2 \xrightarrow{(e,*)} \checkmark}$$

$$B_2 + B_1 \xrightarrow{(*)} [C']_2 \quad B_2 + B_1 \xrightarrow{(*)} \checkmark$$

$$S'_1 : \frac{C \xrightarrow{a} C'}{C; B \xrightarrow{(e,*)} C'; B} \quad S'_2 : \frac{C \xrightarrow{a} \checkmark}{C; B \xrightarrow{(e,*)} [B]_2}$$

$$D'_1 : \frac{C \xrightarrow{a} C'}{C [ > B \xrightarrow{(e,*)} C' [ > B} \quad \frac{B \xrightarrow{a} C'}{C [ > B \xrightarrow{(*)} [C']_2} \quad D'_2 : \frac{C \xrightarrow{a} \checkmark}{C [ > B \xrightarrow{(e,*)} \checkmark} \quad \frac{B \xrightarrow{a} \checkmark}{C [ > B \xrightarrow{(*)} \checkmark}$$

$$P'_1 : \frac{C_1 \xrightarrow{a} C'_1 \quad a \notin A}{C_1 \|_A C_2 \xrightarrow{(e,*)} C'_1 \|_A C_2} \quad P'_2 : \frac{C_1 \xrightarrow{a} \checkmark \quad a \notin A}{C_1 \|_A C_2 \xrightarrow{(e,*)} ([C'_2]_2) \setminus A}$$

$$C_2 \|_A C_1 \xrightarrow{(*)} C_2 \|_A C'_1 \quad C_2 \|_A C_1 \xrightarrow{(*)} ([C'_2]_1) \setminus A$$

$$P'_3 : \frac{C_1 \xrightarrow{e_1} C'_1 \quad C_2 \xrightarrow{e_2} C'_2 \quad a \in A}{C_1 \|_A C_2 \xrightarrow{(e_1, e_2)} C'_1 \|_A C'_2} \quad P'_4 : \frac{C_1 \xrightarrow{e_1} \checkmark \quad C_2 \xrightarrow{e_2} C'_2 \quad a \in A}{C_1 \|_A C_2 \xrightarrow{(e_1, e_2)} ([C'_2]_2) \setminus A}$$

$$C_2 \|_A C_1 \xrightarrow{(e_1, e_2)} ([C'_2]_1) \setminus A$$

$$P'_5 : \frac{C_1 \xrightarrow{e_1} \checkmark \quad C_2 \xrightarrow{e_2} \checkmark \quad a \in A}{C_1 \|_A C_2 \xrightarrow{(e_1, e_2)} \checkmark} \quad Lab' : \frac{C \xrightarrow{a} C'}{C[f] \xrightarrow{f(a)} C'[f]} \quad \frac{C \xrightarrow{a} \checkmark}{C[f] \xrightarrow{f(a)} \checkmark}$$

$$Res' : \frac{C \xrightarrow{a} C' \quad a \notin A}{C \setminus A \xrightarrow{a} C' \setminus A} \quad \frac{C \xrightarrow{a} \checkmark \quad a \notin A}{C \setminus A \xrightarrow{a} \checkmark}$$

$$Rec' : \frac{\text{decl}(x) \xrightarrow{a} C'}{x \xrightarrow{a} C'} \quad \frac{\text{decl}(x) \xrightarrow{a} \checkmark}{x \xrightarrow{a} \checkmark}$$

$$N' : \frac{C \xrightarrow{a} C'}{[C]_1 \xrightarrow{(e,*)} [C']_1} \quad \frac{C \xrightarrow{a} \checkmark}{[C]_1 \xrightarrow{(e,*)} \checkmark}$$

$$[C]_2 \xrightarrow{(*)} [C']_2 \quad [C]_2 \xrightarrow{(*)} \checkmark$$

Figure 4. Event Based Transition Rules with respect to  $\longrightarrow'_{\text{decl}}$

## B.2. The First Bisimilarity Result

In order to verify the bisimilarity, we introduce the following relation between  $\text{EXP}^e$  and  $\text{EXP}$ . An expression  $C$  from  $\text{EXP}^e$  and an expression  $B$  from  $\text{EXP}$  are related if  $B$  results from  $C$  by removing all  $[\_]$  expressions in  $C$ . This is formalized by the following function, where we also count the  $[\_]$  symbols in  $C$ , which is done in order to allow induction verification with this relation. The relation is more precisely described as follows:  $B$  combined with a natural number  $n$  maps to the set of all  $C$  that contain  $n$   $[\_]$ -expressions and the removing of these expressions in  $C$  yields  $B$ .

**Definition B.1.**  $\Xi : \mathbb{N} \times \text{EXP} \rightarrow \mathcal{P}(\text{EXP}^e)$  is defined as follows, where  $I = \{1, 2\}$ .

$$\begin{aligned} \Xi(0, B) &= \{B\} \\ \Xi(n+1, B) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B)\} \quad \text{if } B \in \{0, a, B_1 + B_2, x\} \\ \Xi(n+1, B_1; B_2) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B_1; B_2)\} \cup \{C_1; B_2 \mid C_1 \in \Xi(n+1, B_1)\} \\ \Xi(n+1, B_1 [ > B_2) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B_1 [ > B_2)\} \cup \{C_1 [ > B_2 \mid C_1 \in \Xi(n+1, B_1)\} \\ \Xi(n+1, B_1 \|_A B_2) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B_1 \|_A B_2)\} \cup \\ &\quad \{C_1 \|_A C_2 \mid \exists m \in \mathbb{N} : m \leq n+1 \wedge C_1 \in \Xi(m, B_1) \wedge C_2 \in \Xi(n+1-m, B_2)\} \\ \Xi(n+1, B[f]) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B[f])\} \cup \{C[f] \mid C \in \Xi(n+1, B)\} \\ \Xi(n+1, B \setminus A) &= \{[\tilde{C}]_i \mid i \in I \wedge \tilde{C} \in \Xi(n, B \setminus A)\} \cup \{C \setminus A \mid C \in \Xi(n+1, B)\} \end{aligned}$$

The well-definedness of  $\Xi$  is easily seen. Now we are ready to verify the first bisimulation result.

**Lemma B.1.** Suppose  $B \in \text{EXP}$ . Then  $(\text{EXP}, \text{Act}, \xrightarrow{\text{decl}}, B)$  and  $(\text{EXP}^e, \text{Act}, \xrightarrow{\text{decl}'}, B)$  are bisimilar, where  $C \xrightarrow{\text{decl}''} \bar{C} \Leftrightarrow \exists e \in \mathcal{U} : C \xrightarrow{e'} \bar{C}$ .

**Proof:**

Define  $\mathcal{R} = \{(B, C) \in \text{EXP} \times \text{EXP}^e \mid \exists n : C \in \Xi(n, B)\}$ . To verify that  $\mathcal{R}$  is a bisimulation, we show

$$(B \xrightarrow{\text{decl}} \bar{B} \wedge C \in \Xi(n, B)) \Rightarrow \exists e, \bar{C}, m : C \xrightarrow{e'} \bar{C} \wedge (\bar{C} \in \Xi(m, \bar{B}) \vee \bar{C} = \bar{B} = \surd) \quad (8)$$

The proof of (8) works by induction on the depth of inference of  $B \xrightarrow{\text{decl}} \bar{B}$  combined with the value of  $n$ . Then (8) can be easily checked with the following procedure:

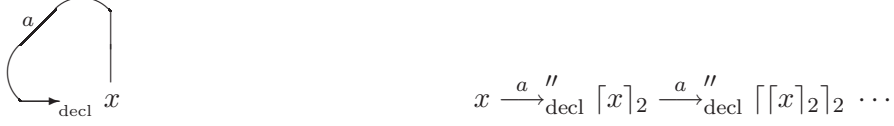
- applying rule  $N'$  whenever  $C = [\tilde{C}]_i$ . In these cases  $n$  is reduced by one and  $B \xrightarrow{\text{decl}} \bar{B}$  remains unaffected and therefore the hypotheses yields the result.
- applying the corresponding rules of  $B \xrightarrow{\text{decl}} \bar{B}$  whenever  $C$  is different from  $[\tilde{C}]_i$ . In these cases the depth of inference is reduced and  $n$  does not get increased and therefore the hypotheses yield the result.

Another fact is

$$(C \xrightarrow{e'} \bar{C} \wedge C \in \Xi(n, B)) \Rightarrow \exists \bar{B}, m : B \xrightarrow{\text{decl}} \bar{B} \wedge (C' \in \Xi(m, \bar{B}) \vee \bar{C} = \bar{B} = \surd) \quad (9)$$

This equation can be proved by induction on the depth of inference of  $C \xrightarrow{e'} \bar{C}$ .

Now we are ready to verify that  $\mathcal{R}$  is a bisimulation:

Figure 5. Transition System of  $\text{decl}(x) = a; x$ 

- It is clear that  $(B, B) \in \mathcal{R}$ .
- Suppose  $(B_1, C_1) \in \mathcal{R}$  and  $B_1 \xrightarrow{a}_{\text{decl}} B_2$ . Then  $\exists e, C_2, m : C_1 \xrightarrow{e'}_{\text{decl}} C_2 \wedge C_2 \in \Xi(m, B_2)$  by (8). Thus  $C_1 \xrightarrow{a''}_{\text{decl}} C_2$  and  $(B_2, C_2) \in \mathcal{R}$ , as required. And analogously for  $B_1 \xrightarrow{a}_{\text{decl}} \checkmark$ .
- Suppose  $(B_1, C_1) \in \mathcal{R}$  and  $C_1 \xrightarrow{a''}_{\text{decl}} C_2$ . Then  $C_1 \xrightarrow{e'}_{\text{decl}} C_2$  for some  $e$ . Hence,  $\exists B_2, m : B_1 \xrightarrow{a}_{\text{decl}} B_2 \wedge C_2 \in \Xi(m, B_2)$  by (9). And analogously for  $C_1 \xrightarrow{a''}_{\text{decl}} \checkmark$ .

□

**Remark B.1.** The transition systems mentioned in Lemma B.1 are not isomorphic. Consider for example  $\text{decl}$  with  $\text{decl}(x) = a; x$ . Then the expression  $x$  yields a finite transition system with respect to  $\longrightarrow_{\text{decl}}$ , whereas an infinite one is derived with respect to  $\longrightarrow''_{\text{decl}}$ . This is illustrated in Figure 5.

### B.3. The Second Bisimilarity Result.

We extend the denotational semantics to  $\text{PA}^e$ .

#### Definition B.2. (Denotational semantics of $\text{PA}^e$ )

Define  $\llbracket \_ \rrbracket' : \text{PA}^e \rightarrow \text{PREES}$  by

$$\begin{aligned}
\llbracket \langle \text{decl}, B \rangle \rrbracket' &= \llbracket \langle \text{decl}, B \rangle \rrbracket \\
\llbracket \langle \text{decl}, C; B \rangle \rrbracket' &= \text{Shift}_1(\llbracket \langle \text{decl}, C \rangle \rrbracket') ; \text{Shift}_2(\llbracket \langle \text{decl}, B \rangle \rrbracket') \\
\llbracket \langle \text{decl}, C \triangleright B \rangle \rrbracket' &= \text{Shift}_1(\llbracket \langle \text{decl}, C \rangle \rrbracket') \triangleright \text{Shift}_2(\llbracket \langle \text{decl}, B \rangle \rrbracket') \\
\llbracket \langle \text{decl}, C[f] \rangle \rrbracket' &= \text{Lab}(\llbracket \langle \text{decl}, C \rangle \rrbracket', f) \quad \llbracket \langle \text{decl}, C_1 \parallel_A C_2 \rangle \rrbracket' = \llbracket \langle \text{decl}, C_1 \rangle \rrbracket' \parallel_A \llbracket \langle \text{decl}, C_2 \rangle \rrbracket' \\
\llbracket \langle \text{decl}, C \setminus A \rangle \rrbracket' &= \llbracket \langle \text{decl}, C_2 \rangle \rrbracket' \setminus A \quad \llbracket \langle \text{decl}, [C]_i \rangle \rrbracket' = \text{Shift}_i(\llbracket \langle \text{decl}, C \rangle \rrbracket')
\end{aligned}$$

It is easily checked that  $\llbracket \_ \rrbracket'$  is well defined. The next lemma illustrates how the different operators behave on event execution. This lemma is essential to relate the event-based transition system with the event execution in **PREES**.

**Lemma B.2.** Suppose  $\mathcal{E}, \mathcal{E}_1, \mathcal{E}_2 \in \mathbf{PREES}$  such that  $E_1 \cap E_2 = \emptyset$ . Then

$$\begin{aligned}
(\mathcal{E}_1 + \mathcal{E}_2)_{[e]} &\simeq \begin{cases} \mathcal{E}_{1[e]} & \text{if } e \in E_1 \\ \mathcal{E}_{2[e]} & \text{otherwise} \end{cases} \\
(\mathcal{E}_1 ; \mathcal{E}_2)_{[e]} &\simeq \begin{cases} \mathcal{E}_2 & \text{if } e \in \text{init}(\mathcal{E}_1) \wedge \Upsilon(T_1, e) \\ \mathcal{E}_{1[e]} ; \mathcal{E}_2 & \text{otherwise} \end{cases} \\
(\mathcal{E}_1 [ > \mathcal{E}_2 ]_{[e]} &\simeq \begin{cases} \mathcal{E}_{2[e]} & \text{if } e \in E_2 \\ \mathcal{E}_{1[e]} [ > \mathcal{E}_2 & \text{otherwise} \end{cases} \\
(\mathcal{E}_1 \parallel_A \mathcal{E}_2)_{[(e_1, \star)]} &\simeq \begin{cases} \mathcal{E}_{1[e_1]} \parallel_A \mathcal{E}_2 & \text{if } \neg \Upsilon(T_1, e_1) \wedge l_1(e_1) \notin A \\ \text{Shift}_2(\mathcal{E}_2) \setminus\!\!\setminus A & \text{if } e_1 \in \text{init}(\mathcal{E}_1) \wedge \Upsilon(T_1, e_1) \wedge l_1(e_1) \notin A \end{cases} \\
(\mathcal{E}_1 \parallel_A \mathcal{E}_2)_{[(\star, e_2)]} &\simeq \begin{cases} \mathcal{E}_1 \parallel_A \mathcal{E}_{2[e_2]} & \text{if } \neg \Upsilon(T_2, e_2) \wedge l_2(e_2) \notin A \\ \text{Shift}_1(\mathcal{E}_1) \setminus\!\!\setminus A & \text{if } e_2 \in \text{init}(\mathcal{E}_2) \wedge \Upsilon(T_2, e_2) \wedge l_2(e_2) \notin A \end{cases} \\
(\mathcal{E}_1 \parallel_A \mathcal{E}_2)_{[(e_1, e_2)]} &\simeq \begin{cases} \mathcal{E}_{1[e_1]} \parallel_A \mathcal{E}_{2[e_2]} & \text{if } \neg \Upsilon(T_1, e_1) \wedge \neg \Upsilon(T_2, e_2) \\ \text{Shift}_2(\mathcal{E}_{2[e_2]}) \setminus\!\!\setminus A & \text{if } e_1 \in \text{init}(\mathcal{E}_1) \wedge \Upsilon(T_1, e_1) \wedge \neg \Upsilon(T_2, e_2) \\ \text{Shift}_1(\mathcal{E}_{1[e_1]}) \setminus\!\!\setminus A & \text{if } e_2 \in \text{init}(\mathcal{E}_2) \wedge \Upsilon(T_2, e_2) \wedge \neg \Upsilon(T_1, e_1) \\ (\emptyset, \emptyset, \emptyset, \{\emptyset\}, \emptyset) & \text{if } \forall i \in \{1, 2\} : e_i \in \text{init}(\mathcal{E}_i) \wedge \Upsilon(T_i, e_i) \end{cases} \\
&\text{whenever } l_1(e_1) = l_2(e_2) \in A \\
\text{Lab}(\mathcal{E}, f)_{[e]} &\simeq \text{Lab}(\mathcal{E}_{[e]}, f) \\
(\mathcal{E} \setminus\!\!\setminus A)_{[e]} &\simeq \begin{cases} \mathcal{E}_{[e]} \setminus\!\!\setminus A & \text{if } l(e) \notin A \\ \text{undefined} & \text{otherwise} \end{cases} \\
\text{Shift}_1(\mathcal{E})_{[(e, \star)]} &\simeq \text{Shift}_1(\mathcal{E}_{[e]}) \\
\text{Shift}_2(\mathcal{E})_{[(\star, e)]} &\simeq \text{Shift}_2(\mathcal{E}_{[e]})
\end{aligned}$$

**Proof:**

Straightforward. □

Now we show that a transition labelled with event  $e$  of the event-based transition system can be matched in the **PREES**-denotation by executing this  $e$ :

**Lemma B.3.** Suppose  $\langle \text{decl}, C \rangle \in \text{PA}^e$  and  $C \xrightarrow[e]{a}^l_{\text{decl}} \bar{C}$ . Then

$$e \in \text{init}(\mathcal{E}) \wedge l(e) = a \wedge (\bar{C} \neq \surd \Rightarrow \bar{\mathcal{E}} = \mathcal{E}_{[e]}) \wedge (\bar{C} = \surd \Leftrightarrow \Upsilon(T, e))$$

with  $\mathcal{E} = \llbracket \langle \text{decl}, C \rangle \rrbracket'$  and  $\bar{\mathcal{E}} = \llbracket \langle \text{decl}, \bar{C} \rangle \rrbracket'$ .

**Proof:**

We use induction on the depth of inference of  $C \xrightarrow[e]{a}^l_{\text{decl}} \bar{C}$ . Then the equation can be verified by case analysis on the derivation rules, where Lemma B.2 is used. In the case of *Rec'* we make use of the fact that  $\llbracket \langle \text{decl}, x \rangle \rrbracket = \llbracket \langle \text{decl}, \text{decl}(x) \rangle \rrbracket$  for any  $x \in \text{Var}$ . □

The converse of Lemma B.3 also holds, i.e. the execution of event  $e$  in the **PREES**-denotation of  $C$  can be matched by a transition from  $C$  that is labelled with  $e$ .

**Lemma B.4.** Let  $\langle \text{decl}, C \rangle \in \text{PA}^e$ ,  $\mathcal{E} = \llbracket \langle \text{decl}, C \rangle \rrbracket'$  and  $e \in \text{init}(\mathcal{E})$ . Then  $\exists \bar{C} : C \xrightarrow[e]{l(e)'}_{\text{decl}} \bar{C}$ .

**Proof:**

First we show if  $n \in \mathbb{N}$ ,  $B \in \text{EXP}$  and  $\mathcal{E}_n = \llbracket B \rrbracket_{\mathcal{F}_{\text{decl}}^n(\perp)}$  then

$$e \in \text{init}(\mathcal{E}_n) \Rightarrow \exists \bar{C} \in \text{EXP}^e : B \xrightarrow[e]{l_n(e)'}_{\text{decl}} \bar{C} \quad (10)$$

This is done by induction on  $n$  combined with the structure of  $B$  where the lexicographical order is used. Furthermore, a case analysis on the structure of  $B$  is used. We only present here the case  $B = x$ : Suppose  $e \in \text{init}(\llbracket x \rrbracket_{\mathcal{F}_{\text{decl}}^n(\perp)})$  then  $n > 0$ . Therefore,  $\llbracket x \rrbracket_{\mathcal{F}_{\text{decl}}^n(\perp)} = \mathcal{F}_{\text{decl}}^n(\perp)(x) = \llbracket \text{decl}(x) \rrbracket_{\mathcal{F}_{\text{decl}}^{n-1}(\perp)}$ . The rest follows by induction, since  $n$  is reduced. Thus (10) is established.

The main statement follows now by structural induction on  $C$ . We only present the case  $C = B \in \text{EXP}$ . By Remark 4.1 we get  $\llbracket \langle \text{decl}, B \rangle \rrbracket' = \bigsqcup_n \llbracket B \rrbracket_{\mathcal{F}_{\text{decl}}^n(\perp)}$ . Then it is easily seen that there is  $m$  such that  $e \in \text{init}(\llbracket B \rrbracket_{\mathcal{F}_{\text{decl}}^m(\perp)})$  and the labels on  $e$  coincide. And so the result follows by (10).  $\square$

Now we are ready to verify the second bisimulation result, which concludes the proof.

**Lemma B.5.** Suppose  $\langle \text{decl}, C \rangle \in \text{PA}^e$ . Then the transition systems  $(\mathbf{PREES}, \text{Act}, \hookrightarrow, \llbracket \langle \text{decl}, C \rangle \rrbracket')$  and  $(\text{EXP}^e, \text{Act}, \xrightarrow[\text{decl}]'' , C)$  are bisimilar, where  $\xrightarrow[\text{decl}]''$  is defined as in Lemma B.1.

**Proof:**

Define  $\mathcal{R} = \{(C', \llbracket \langle \text{decl}, C' \rangle \rrbracket') \mid C' \in \text{EXP}^e\}$ . Then  $(C, \llbracket \langle \text{decl}, C \rangle \rrbracket') \in \mathcal{R}$  by definition.

Suppose  $C_1 \in \text{EXP}^e$  and  $C_1 \xrightarrow[\text{decl}]'' a \bar{C}$ . Then  $C_1 \xrightarrow[e]{a'}_{\text{decl}} \bar{C}$  for some  $e$ . By Lemma B.3, we get  $\bar{C} \in \text{EXP}^e \Rightarrow \llbracket \langle \text{decl}, C_1 \rangle \rrbracket' \xrightarrow{a} \llbracket \langle \text{decl}, \bar{C} \rangle \rrbracket'$  and  $\bar{C} = \surd \Leftrightarrow \Upsilon(\pi_4(\llbracket \langle \text{decl}, C_1 \rangle \rrbracket'), e)$ , as required.

Suppose  $C_1 \in \text{EXP}^e$  and  $\llbracket \langle \text{decl}, C_1 \rangle \rrbracket' \xrightarrow{a} \bar{\mathcal{E}}$ . Then there is  $e \in \text{init}(\llbracket \langle \text{decl}, C_1 \rangle \rrbracket')$  such that  $\bar{\mathcal{E}} = \llbracket \langle \text{decl}, C_1 \rangle \rrbracket'_{[e]}$  and  $\pi_5(\llbracket \langle \text{decl}, C_1 \rangle \rrbracket')(e) = a$ . By Lemma B.4 there is  $\bar{C}$  such that  $C_1 \xrightarrow[e]{a'}_{\text{decl}} \bar{C}$ . Moreover,  $\bar{C} \in \text{EXP}^e \Rightarrow \llbracket \langle \text{decl}, C_1 \rangle \rrbracket'_{[e]} = \llbracket \langle \text{decl}, \bar{C} \rangle \rrbracket'$  and  $\bar{C} = \surd \Leftrightarrow \Upsilon(\pi_4(\llbracket \langle \text{decl}, C_1 \rangle \rrbracket'), e)$  by Lemma B.3, which concludes the proof.  $\square$

## References

- [1] Abramsky, S., Jung, A.: Domain Theory, in: *Handbook of Logic in Computer Science* (S. Abramsky, D. M. Gabbay, T. S. E. Maibaum, Eds.), vol. 3, Clarendon Press, 1994, 1–168.
- [2] Baeten, J. C. M., Bergstra, J. A.: *Mode Transfer in Process Algebra*, Report CSR 00-01, Vakgroep Informatica, Technische Universiteit Eindhoven, 2000.
- [3] Baeten, J. C. M., Bergstra, J. A., Klop, J. W.: Syntax and defining equations for an interrupt mechanism in process algebra, *Fundamenta Informaticae*, **9**, 1986, 127–168.
- [4] Baier, C., Majster-Cederbaum, M. E.: The Connection between an Event Structure Semantics and an Operational Semantics for TCSP, *Acta Informatica*, **31**, 1994, 81–104.

- [5] de Bakker, J., de Roever, W.-P., Rozenberg, G., Eds.: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, vol. 354 of LNCS, Springer-Verlag, 1989.
- [6] Baldan, P., Busi, N., Corradini, A., Pinna, G. M.: Functional Concurrent Semantics for Petri Nets with Read and Inhibitor Arcs, *CONCUR 2000 – Concurrency Theory* (C. Palamidessi, Ed.), LNCS, vol. 1877 of LNCS, Springer-Verlag, 2000, 442–457.
- [7] Baldan, P., Corradini, A., Montanari, U.: Contextual Petri Nets, Asymmetric Event Structures, and Processes, *Information and Computation*, **171**, 2001, 1–49.
- [8] Bergstra, J. A., Fokkink, W., Ponse, A.: Process Algebra with Recursive Operations, in: Bergstra et al. [10], 333–389.
- [9] Bergstra, J. A., Klop, J. W.: Process Algebra for Synchronous Communication, *Information and Control*, **60**, 1984, 109–137.
- [10] Bergstra, J. A., Ponse, A., Smolka, S. A., Eds.: *Handbook of Process Algebra*, North-Holland, 2001.
- [11] Bolognesi, T., Brinksma, E.: Introduction to the ISO Specification Language LOTOS, *Computer Networks and ISDN Systems*, **14**, 1987, 25–59.
- [12] Boudol, G., Castellani, I.: Permutation of transitions: an event structure semantics for CCS and SCCS, in: de Bakker et al. [5], 411–427, 411–427.
- [13] Bravetti, M., Gorrieri, R.: Deciding and Axiomatizing Weak ST Bisimulation for a Process Algebra with Recursion and Action Refinement, *ACMTCL: ACM Transactions on Computational Logic*, **3**, 2002.
- [14] Dierkens, B.: *New Features in PSF I – Interrupts, Disrupts, and Priorities*, Report P9417, Programming Research Group - University of Amsterdam, 1994.
- [15] Engels, A., Cobben, T.: Interrupt and Disrupt in MSC: Possibilities and Problems, *Proceedings fo the 1st Workshop of the SDL Forum Society on SDL and MSC* (Y. Lahav, A. Wolisz, J. Fischer, E. Holz, Eds.), Informatikberichte, number 104 in Informatikberichte, Humboldt-Universitt zu Berlin, 1998.
- [16] Fecher, H.: *Action Refinement in End-Based Choice Settings*, Ph.D. Thesis, Universität Mannheim, 2003.
- [17] Glabbeek, R. v., Plotkin, G. D.: Configuration Structures, *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1995, 199–209.
- [18] Gorrieri, R., Rensink, A.: Action Refinement, in: Bergstra et al. [10], 1047–1147.
- [19] Gunawardena, J.: A Generalized Event Structure for the Muller Unfolding of a Safe Net, *CONCUR '93* (E. Best, Ed.), LNCS, vol. 715 of LNCS, Springer-Verlag, 1993, 278–292.
- [20] Hoogers, P. W., Kleijn, H. C. M., Thiagarajan, P. S.: An event structure semantics for general Petri nets, *Theoretical Computer Science*, **153**, 1996, 129–170.
- [21] Katoen, J.-P.: *Quantitative and Qualitative Extension of Event Structures*, Ph.D. Thesis, Enschede: Centre for Telematics and Information Technology, P.O. Box 217 - 7500 AE Enschede - The Netherlands, 1996.
- [22] Langerak, R.: *Transformations and Semantics for LOTOS*, Ph.D. Thesis, Department of Computer Science, University of Twente, 1992.
- [23] Langerak, R.: Bundle Event Structures: A Non-Interleaving Semantics for LOTOS, *Formal Description Techniques, V* (M. Diaz, R. Groz, Eds.), Elsevier, 1993, 331–346.
- [24] Majster-Cederbaum, M., Roggenbach, M.: Transition systems from event structures revisited, *Information Processing Letters*, **67**, 1998, 119–124.
- [25] Milner, R.: *Communication and Concurrency*, International Series in Computer Science, Prentice Hall, 1989.

- [26] Nielsen, M., Plotkin, G., Winskel, G.: Petri Nets, Event Structures and Domains, Part I, *Theoretical Computer Science*, **13**, 1981, 85–108.
- [27] Pinna, G. M., Poigne, A.: Event Automata as a Generic Model of Reactive Systems, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software* (M. Broy, S. Jähnichen, Eds.), LNCS, vol. 1009 of LNCS, Springer-Verlag, 1995, 74–91.
- [28] Pinna, G. M., Poigné, A.: On the nature of events: another perspective in concurrency, *Theoretical Computer Science*, **138**(2), 1995, 425–454.
- [29] Winskel, G.: An Introduction to Event Structures, in: de Bakker et al. [5], 364–397, 364–397.