

Adding Priority to Event Structures*

Youssef Arbach[†]

Kirstin Peters

Uwe Nestmann

Technische Universität Berlin, Germany

Event Structures (ESs) are mainly concerned with the representation of causal relationships between events, usually accompanied by other event relations capturing conflicts and disabling. Among the most prominent variants of ESs are *Prime* ESs, *Bundle* ESs, *Stable* ESs, and *Dual* ESs, which differ in their causality models and event relations. Yet, some application domains require further kinds of relations between events. Here, we add the possibility to express priority relationships among events.

We exemplify our approach on Prime, Bundle, Extended Bundle, and Dual ESs. Technically, we enhance these variants in the same way. For each variant, we then study the interference between priority and the other event relations. From this, we extract the redundant priority pairs—notably differing for the types of ESs—that enable us to provide a comparison between the extensions. We also exhibit that priority considerably complicates the definition of partial orders in ESs.

1 Introduction

Concurrency Model. Event Structures (ESs) are concerned with usually statically defined relationships that govern the possible occurrences of events, typically represented as *causality* (for precedence) and *conflict* (for choice). An event is a single occurrence or an instance of an action, and thus cannot be repeated. ESs were first used to give semantics to Petri nets in [15], then to give semantics to process calculi in [4, 9], and concurrent systems in general in [14]. The dynamics of an ES are usually provided either by the sets of traces compatible with the constraints, or by means of configurations based sets of events, possibly in their labeled partially-ordered variant (*lposets*).

Event Structures are *non-interleaving* models. In interleaving models, events take place linearly, one after the other. There, concurrency is expressed in terms of free choice or non-determinism, i.e. concurrent events can appear in any order. Event Structures show concurrency not as a linear order with free choice, but as independence between events, i.e. events are concurrent when they are related neither by conflict nor causally. This intuition manifests clearly in system runs, represented by the so-called configurations, or in terms of partial orders, where concurrent events are unordered.

Application Domain. We investigate the phenomenon of Dynamic Coalitions (DC). The term denotes a temporary collaboration between entities in order to achieve a common goal. Afterwards, such a coalition resolves itself, or is resolved. One example of a DC is the treatment of a stroke patient, taken from the medical sector, which inspires our work: *A patient gets a stroke which calls for the ambulance. In the meanwhile, the emergency room prepares to receive the patient. Then, the ambulance arrives and the patient is transferred to the emergency room. While the patient is in the emergency room, the latter communicates with the stroke unit to prepare for transferring the patient, and then the patient is sent to the stroke unit. Before the patient is discharged from the stroke unit, some therapists are invited by the stroke unit to join the patient treatment.* Such coalitions are called dynamic, as they evolve over

*Supported by the DFG Research Training Group SOAMED.

[†]Corresponding author: arbach@soamed.de

time, where new members can join, and others can leave, until the goal is achieved. We call this specific phenomenon the *formation* of a DC. Others call it the *membership dimension* of a DC [5].

Examining the application scenario, we observe that it can be naturally modeled by means of Event Structures. Firstly, it mentions events, e.g. a patient gets a stroke, the ambulance joins, and the stroke unit invites some therapists. Moreover, we are dealing with possible conflicts between the members, e.g. between the therapists. In addition, there is causality, for example the event, where the patient gets a stroke, causes the ambulance to join and the emergency room to prepare. Finally, there can be concurrency, i.e. multiple members of the coalition can work concurrently, e.g. the stroke unit prepares to receive the patient while the emergency room is still working on the patient.

Further Requirements. Applications may impose to limit the amount of concurrency in the specifications of some systems or, in interleaving models, to limit the non-determinism or free choice. For example, in our above-mentioned healthcare scenario, imagine that while the therapists are working on the patient outside the hospital, the patient gets another stroke, and then the ambulance again needs to involve and interrupt the work of the therapists. So, they cannot all work together at the same time. Besides, in this particular situation, the ambulance should have a higher *priority* to perform its events, such that only afterwards the therapists might continue their work. This is some kind of order, so there is a determinism here on who needs to go first, carried by the concept of priority. The precedence caused by priority is called “pre-emption” (cf. [12]). So, the event with higher priority pre-empts the event with lower priority: the higher-priority one must happen before any concurrently enabled event with lower priority, so a priority relation is (only) applied in a state of competition. For example, some processes compete to run on a processor. In the same way many members of a coalition compete for the patient. Some of them can work concurrently, due to the specifications of the system, and some cannot.

Overview. This paper is organized as follows. In §2, we start with the simplest form of ESs, the Prime ESs, add priority to it, discuss the overlapping between priority and the other relations of Prime ESs, and show how to reduce this overlapping. In §3, we introduce Bundle ESs, their traces, configurations, and lposets. Then we add priority to them and investigate the relation between priority and other event relations of BESs like enabling and precedence. In §4 and §5 we then study the two extensions Extended Bundle ESs and Dual ESs of Bundle ESs and how their different causality models modify the relationship of priority and the other event relations of the ESs. In §6, we summarize the work and conclude by comparing the results.

Related Work. Priorities are used as a mechanism to provide interrupts in systems concerned with processes. For example, in Process Algebra, Baeten et al. were the first to add priority in [1]. They defined it as a partial order relation $<$. Moreover, Camilleri, and Winskel integrated priority within the Summation operator in CCS [6]. Also, Lüttgen in [12] and Cleaveland et al. in [7] considered the semantics of Process Algebra with priority.

In Petri Nets, which are a non-interleaving model like Event Structures, Bause in [2, 3] added priority to Petri Nets in two different ways: static and dynamic. Dynamic means the priority relation evolves during the system run, while the static one means it is fixed since the beginning and will never change till the end. In that sense, static priority is what we define here.

In this paper, we add priority to different kinds of Event Structures. Then we analyze the overlapping between the priority relation and the other relations of the respective Event Structure in order to identify and remove redundant priority pairs. We observe that the possibility to identify and remove redundant priority pairs is strongly related to the causality model that is provided by the considered model of ESs.

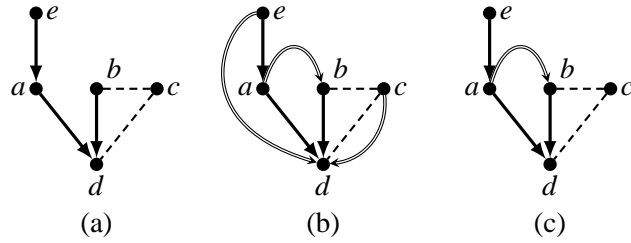


Figure 1: A Prime ES without priority in (a), with priority in (b), and after dropping redundant priority pairs in (c).

2 Priority in Prime Event Structures

Prime Event Structures (PESs), invented by Winskel [15], are the simplest and first version of ESs. Causality is expressed in terms of an enabling relation, i.e. a partial order between events. For an event to become enabled in PESs, all of its predecessors with respect to the enabling relation must take place; an enabled event may happen, but does not have to do so. There is also a conflict relation between events to provide choices, given as a binary symmetric relation, and a labeling function mapping events to actions.

Definition 2.1. A *Prime Event Structure (PES)* is a quadruple $\delta = (E, \#, \leq, l)$ where:

- E , a set of *events*
- $\# \subseteq E \times E$, an irreflexive symmetric relation (the *conflict* relation)
- $\leq \subseteq E \times E$, a partial order (the *enabling* relation)
- $l: E \rightarrow Act$, a *labeling* function

that additionally satisfies the following constraints:

1. **Conflict Heredity:** $\forall e, e', e'' \in E. e\#e' \wedge e' \leq e'' \implies e\#e''$
2. **Finite Causes:** $\forall e \in E. \{e' \in E \mid e' \leq e\}$ is finite

Figure 1 (a) shows an example of a Prime ES, where a single-line arrow represents enabling, directed from predecessors in \leq to successors. The dashed line represents a conflict. Note that this structure fulfills the two constraints of a PES.

A *configuration* is a representation of system state by means of the set of events that have occurred up to a certain point. In Prime ESs, a configuration is a conflict-free set of events $C \subseteq E$ that is left-closed under the enabling relation, i.e. no two events of C are in conflict and for all predecessors e with respect to \leq of an event $e' \in C$ it holds $e \in C$. Thus, given a Prime ES $\delta = (E, \#, \leq, l)$, a configuration C represents a *system run* of δ (or the *state* of δ after this run), where events not related by \leq occur concurrently.

A *trace* is a sequential version of a system run. It can be defined as a sequence of events which are conflict-free and where all the predecessors of an event in \leq precede that event in the trace. We will define it formally in another equivalent way, which we will rely on when we define priority later:

Let σ be a sequence of events $e_1 \cdots e_n$ such that $\{e_1, \dots, e_n\} \subseteq E$ in a PES $\delta = (E, \#, \leq, l)$. We refer to $\{e_1, \dots, e_n\}$ by $\bar{\sigma}$, and we call $\text{en}_\delta(\sigma)$ the set of events that are enabled by σ , where

$$\text{en}_\delta(\sigma) \triangleq \{e \in (E \setminus \bar{\sigma}) \mid (\forall e' \in E. e' \leq e \implies e' \in \bar{\sigma}) \wedge (\nexists e' \in \bar{\sigma}. e\#e')\}. \quad (1)$$

We use σ_i to denote the prefix $e_1 \cdots e_i$, for some $i < n$. Then, the sequence $\sigma = e_1 \cdots e_n$ is called a *trace* of δ iff

$$\forall i \leq n. e_i \in \text{en}_\delta(\sigma_{i-1}) \quad (2)$$

Accordingly, a trace is a linearization of a configuration respecting \leq . Usually many traces can be derived from one configuration. The differences between such traces of the same configuration result

from concurrent events that are independent, i.e. are related neither by enabling nor conflict. For example, in Figure 1 (a), the events c and a are independent and thus concurrent in a configuration like $\{e, a, c\}$. From $\{e, a, c\}$ the traces eac , eca , and cea can be derived for the structure in Figure 1 (a).

If we add priority to PESs, it should be a binary relation between events such that, whenever two concurrent events ordered in priority are enabled together, the one with the higher priority must preempt the other.¹ Thus we add a new acyclic relation $\prec \subseteq E \times E$, the *priority* relation, to Prime ESs and denote the pair (δ, \prec) as *prioritized Prime ES (PPES)*. Later on, we add priority in a similar way to other kinds of Event Structures. Sometimes, we expand a prioritized ES (δ, \prec) , where $\delta = (E, r_1, r_2, 1)$, to $(E, r_1, r_2, 1, \prec)$.

Figure 1 (b) illustrates a prioritized variant of Figure 1 (a), where the priority relation is represented by a double-lined arrow from the higher-priority event to the lower-priority one, showing the direction of precedence (pre-emption). Sometimes representing both an Event Structure and its associated priority relation in the same diagram becomes too confusing. In that case we visualize the priority relation in a separate diagram next to the structure.

Let us define the interpretation of \prec in a formal way: let $\sigma = e_1 \cdots e_n$ be a sequence of events in a PPES $\delta' = (\delta, \prec)$. We call σ a trace of δ' iff it is 1.) a trace of δ , and 2.) satisfies the following constraint:

$$\forall i < n. \forall e_j, e_h \in \bar{\sigma}. e_j \neq e_h \wedge e_j, e_h \in \text{en}_\delta(\sigma_i) \wedge e_h \prec e_j \implies j < h \quad (3)$$

For example, the sequence $ebad$ is a trace of Figure 1 (a) but not of Figure 1 (b) due to priority. Let us denote the set of traces of a structure as $T(\delta)$. By definition, the traces of a PPES $\delta' = (\delta, \prec)$ are a subset of the traces of the Prime ES δ .

Proposition 2.2. $T(\delta, \prec) \subseteq T(\delta)$.

If we analyze Figure 1 (a) and (b) we observe that, because of the conflict relation, no trace can contain both c and d . And even without the priority relation the enabling relation ensures that e always has to precede d . Since neither c and d nor e and d can be enabled together, i.e. do never compete, applying the priority relation between them is useless or trivial. Indeed we can always reduce the priority relation by dropping all pairs between events that are under \leq or $\#$ without affecting the set of traces.

Theorem 2.3. Let $(E, \#, \leq, 1, \prec)$ be an PPES, and let $\prec' \triangleq \prec \setminus \{(e, e') \mid e' \# e \vee e' \leq e \vee e \leq e'\}$. Then:

$$T(E, \#, \leq, 1, \prec) = T(E, \#, \leq, 1, \prec')$$

Proof. Straightforward from the Definitions of traces, (3), and (1). □

Figure 1 (c) shows the result of dropping the priority pairs that are redundant in Figure 1 (b). Note that after dropping all redundant pairs, there is no overlapping, neither between the priority and the enabling relation, nor between the priority and the conflict relation. The following theorem insures minimality of reduction.

Theorem 2.4. Let $(E, \#, \leq, 1, \prec)$ be an PPES, let $\prec' \triangleq \prec \setminus \{(e, e') \mid e' \# e \vee e' \leq e \vee e \leq e'\}$, and $\prec'' \subset \prec'$. Then $T(E, \#, \leq, 1, \prec) \neq T(E, \#, \leq, 1, \prec'')$.

Proof. Straightforward from the Definitions of traces, (3), and (1). □

This result is good for a modeler, since it implies unambiguity about whether a priority relation affects the behavior or not. In other words, after dropping all the redundant priority pairs, the remaining priority pairs always lead to pre-emption, limit concurrency and narrow down the possible traces. This is not the case for the following ESs, since they offer other causality models.

¹In fact we could define it as a partial order. However after dropping redundant priority pairs as explained later the priority relation is usually no longer transitive, i.e. no longer a partial order.

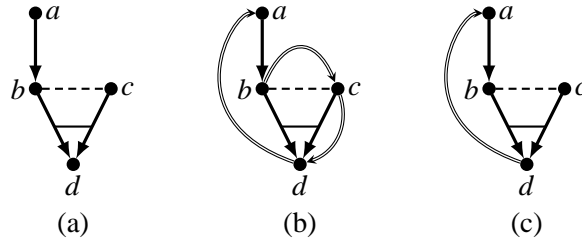


Figure 2: A Bundle ES without priority in (a), with priority in (b), and after dropping redundant priority pairs in (c).

3 Priority in Bundle Event Structures

Prime ESs are simple but also limited. Conflict heredity and the enabling relation of Prime ESs do not allow to describe some kind of optional or conditional enabling of events. Bundle ESs—among others—were designed to overcome these limitations [9]. Here an event can have different causes, i.e. they allow for disjunctive causality.

In Bundle ESs the conflict relation is as in Prime ESs an irreflexive and symmetric relation, but the enabling relation offers some optionality, based on bundles. A *bundle* (X, e) , also denoted by $X \mapsto e$, consists of a bundle set X and the event e it enables. A *bundle set* is a set of events that are pairwise in conflict. There can be several bundles $(X_1, e), \dots, (X_n, e)$ for the same event e . So—instead of a set of events as in Prime ESs—an event e in Bundle ESs is enabled by a set $\{X_1, \dots, X_n\}$ of bundle sets.

When one event of a set X_i takes place, then the bundle $X_i \mapsto e$ is said to be satisfied; and for e to be enabled all its bundles must be satisfied. In Bundle ESs (and also Extended Bundle ESs) no more than one event out of each set X_i can take place; this leads to causal unambiguity [10]. But since a bundle set can be satisfied by any of its members, this yields disjunctive causality and gives flexibility in enabling.

Definition 3.1. A *Bundle Event Structure (BES)* is a quadruple $\beta = (E, \#, \mapsto, l)$ where:

- E , a set of *events*
- $\# \subseteq E \times E$, an irreflexive symmetric relation (the *conflict* relation)
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *enabling* relation
- $l : E \rightarrow Act$, a *labeling* function

that additionally satisfies the following constraint:

$$\text{Stability: } \forall X \subseteq E. \forall e \in E. X \mapsto e \implies (\forall e_1, e_2 \in X. e_1 \neq e_2 \implies e_1 \# e_2) \quad (\text{SC})$$

Figure 2 (a) shows an example of a BES. The solid arrows denote causality, i.e. reflect the enabling relation, where the bar between the arrows shows that they belong to the same bundle and the dashed line denotes again a mutual conflict. Thus there are two bundles in this example, namely the singleton $\{a\} \mapsto b$ and $\{b, c\} \mapsto d$. As required by (SC) we have $b \# c$ and $c \# b$.

As in Prime ES, let $\sigma = e_1 \cdots e_n$ be a sequence of events and $\bar{\sigma} = \{e_1, \dots, e_n\}$ such that $\bar{\sigma} \subseteq E$. We use $\text{en}_\beta(\sigma)$ to refer to the set of events enabled by σ :

$$\text{en}_\beta(\sigma) \triangleq \{e \in (E \setminus \bar{\sigma}) \mid (\forall X \subseteq E. X \mapsto e \implies X \cap \bar{\sigma} \neq \emptyset) \wedge (\nexists e' \in \bar{\sigma}. e \# e')\} \quad (4)$$

Then the sequence $\sigma = e_1 \cdots e_n$ is called a *trace* of β iff

$$\forall i \leq n. e_i \in \text{en}_\beta(\sigma_{i-1}) \quad (5)$$

We denote the set of all valid traces in β as $T(\beta)$. A set of events $C \subseteq E$ is called a *configuration* of an BES β iff $\exists \sigma \in T(\beta). C = \bar{\sigma}$. Let $C(\beta)$ denote the set of configurations of β .

Definition (4) ensures that for an event e to be enabled, one event out of each pointing bundle set X with $X \mapsto e$ is necessary. In a trace, if there is one event of a bundle set, then we denote the corresponding bundle as *satisfied*. Remember that, because of the stability condition, no more than one event out of each bundle set can take place. In Figure 2, the sequence bd is not a trace since $\{a\} \mapsto b$ has never been satisfied. On the other hand, $abcd$ is not a trace either, since b conflicts with c . While $a, c, ab, ac, ca, cd, abd, acd, cad$, and cda are all traces.

The following lemma proves that whenever an event e is in a bundle set X pointing to e' , i.e. such that $X \mapsto e'$, then e and e' cannot be enabled together.

Lemma 3.2. *Let $\beta = (E, \#, \mapsto, 1)$ be an BES, and let $e, e' \in E$ such that $\exists X \subseteq E. e \in X \wedge X \mapsto e'$. Then:*

$$\forall \sigma = e_1 \cdots e_n \in T(\beta). \nexists i < n. e, e' \in \text{en}_\beta(\sigma_i)$$

Proof. Let $\sigma = e_1 \cdots e_n \in T(\beta)$ and $X \subseteq E$ such that $e \in X \wedge X \mapsto e'$. Assume $e \in \text{en}_\beta(\sigma_i)$ for some $i < n$. Then:

$$\begin{aligned} e \in \text{en}_\beta(\sigma_i) \wedge e \in X \wedge X \mapsto e' &\stackrel{\text{(SC)}}{\implies} e \in \text{en}_\beta(\sigma_i) \wedge e \in X \wedge X \mapsto e' \wedge (\forall e'' \in (X \setminus \{e\}). e \# e'') \\ &\stackrel{(4)}{\implies} X \mapsto e' \wedge (\forall e'' \in X. e'' \notin \bar{\sigma}_i) \stackrel{(4)}{\implies} e' \notin \text{en}_\beta(\sigma_i) \end{aligned}$$

Hence $\nexists i < n. e, e' \in \text{en}_\beta(\sigma_i)$. □

3.1 Labeled Partially Ordered Sets

Labeled partially ordered sets, abbreviated as lposets, are used as a semantical model for different kinds of ESs and other concurrency models (see e.g. [13]). In contrast to configurations, lposets do not only record the set of events that happened so far, but also reflect the precedence relations between these events. Here, we use them to describe the semantics of BESs (as well as of EBESs and DESs in the next sections). An lposet consists of a set, a partial order over this set, and a labeling function.

Definition 3.3. *A labeled partially ordered set (lposet) is a triple $\langle A, \leq, f \rangle$ where:*

- A , a finite set of *events*
- \leq , a *partial order* over A
- $f : A \rightarrow \text{Act}$, a *labeling* function

We use η to denote the empty lposet $\langle \emptyset, \emptyset, \emptyset \rangle$. A non-empty lposet $\langle A, \leq, f \rangle$ is visualized by a box containing all the events of A and where two events e_1 and e_2 are related by an arrow iff $e_1 \leq e_2$, where reflexive and transitive arrows are usually omitted. Figure 3 depicts several lposets, where e.g. the top right box visualizes the lposet $\langle \{a, b, d\}, \{(a, b), (b, d)\}, f \rangle$ for some (not visualized) labeling function f .

An lposet describes the semantics of a BES for a specific set of events. To describe the semantics of the entire BES, families of lposets are used. These families consist of several lposets that are related by a prefix relation on lposets [8]:

$$\langle A, \leq, f \rangle \text{ is a prefix of } \langle A', \leq', f' \rangle \iff A \subseteq A' \wedge \leq = (\leq' \cap (A' \times A)) \wedge f = f' \upharpoonright_A$$

Now a family \mathcal{P} of lposets is defined as a non-empty set of lposets that is downward closed under the lposet-prefix relation. It is shown in [8] that a family of lposets along with the prefix relation is itself a partially ordered set. As investigated by Rensink in [13], families of lposets (even posets) form a convenient underlying model for models of concurrency like BESs (or EBESs).

In order to define the lposets of a BES β , we build a partially ordered set (poset) over a configuration $C \in C(\beta)$. We define the partial order as a precedence relation $\prec_C \subseteq C \times C$ between events as follows:

$$e \prec_C e' \iff \exists X \subseteq E. e \in X \wedge X \mapsto e' \tag{6}$$

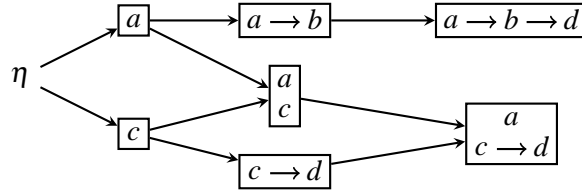


Figure 3: The family of lposets of the BES in Figure 2 (a).

and define \preceq_C as the reflexive and transitive closure of \prec_C . It is proved in [9] that \preceq_C is a partial order over C . Finally, by adding the labeling function $l|_C$, the triple $\langle C, \preceq_C, l|_C \rangle$ is an lposet. We call $L(\beta)$ the set of all lposets defined on $C(\beta)$. Figure 3 shows the largest family of lposets for the example in Figure 2 (a), where the arrows between lposets denote the prefix relation.

As proved in [9], each linearization (obeying the defined precedence relations) of a given lposet built from an BES (or EBES) yields an event trace of that structure.

3.2 Adding Priority to BESs

Again we add priority $\triangleleft \subseteq E \times E$ to EBESs as a binary acyclic relation between events such that, whenever two events are enabled together, the one with the higher priority pre-empts the other. We denote $\beta' = (\beta, \triangleleft) = (E, \#, \mapsto, 1, \triangleleft)$ as *prioritized Bundle ES (PBES)*. Figure 2 (b) illustrates a prioritized version of the BES in Figure 2 (a).

Also the semantics of \triangleleft is defined similarly to Prime ESs. A sequence of events $\sigma = e_1 \cdots e_n$ is a trace of (β, \triangleleft) iff 1.) $\sigma \in T(\beta)$ and 2.) σ satisfies the following constraint:

$$\forall i < n. \forall e_j, e_h \in \bar{\sigma}. e_j \neq e_h \wedge e_j, e_h \in \text{en}_\beta(\sigma_i) \wedge e_h \triangleleft e_j \implies j < h \quad (7)$$

Again the traces of a PBES (β, \triangleleft) are a subset of the traces of the corresponding BES β .

Proposition 3.4. $T(\beta, \triangleleft) \subseteq T(\beta)$.

For example the sequence cad is a trace of the BES in Figure 2 (a), but it is not a trace of the PBES in Figure 2 (b). Of course a larger priority relation filters more traces out than a smaller one.

Lemma 3.5. Let (β, \triangleleft) and (β, \triangleleft') be two PBES with $\triangleleft' \subseteq \triangleleft$. Then $T(\beta, \triangleleft) \subseteq T(\beta, \triangleleft')$.

Proof. Straightforward from the Definition of traces, (7), and $\triangleleft' \subseteq \triangleleft$. \square

We adapt the notion of configuration to prioritized BESs such that $\sigma \in C(\beta, \triangleleft)$ for a PBES (β, \triangleleft) iff $\exists \bar{\sigma} \in T(\beta, \triangleleft). C = \bar{\sigma}$. In Section 3.1 we define the semantics of BESs by families of lposets. Unfortunately doing the same for PBESs is not that simple. Consider the lposet $\boxed{a \ c \rightarrow d}$ of the BES β in Figure 2 (a). According to Figure 2 (b), d has a higher priority than a , i.e. $a \triangleleft d$. Hence $\boxed{a \ c \rightarrow d}$ does not describe the semantics of the PBES (β, \triangleleft) with respect to the Configuration $\{a, c, d\}$, because $cad \in T(\beta)$ but $cad \notin T(\beta, \triangleleft)$. In fact we cannot describe the semantics of PBESs by a family of lposets as depicted in Figure 3. Instead, to describe the semantics of (β, \triangleleft) with respect to $\{a, c, d\}$ we need the two different lposets $\boxed{a \rightarrow c \rightarrow d}$ and $\boxed{c \rightarrow d \rightarrow a}$.

The enabling relation defines precedence between events as used for \prec_C in (6), whereas priority rather defines some kind of conditional precedence. Priority affects the semantics only if the related events are enabled together. Thus the same problem with the definition of lposets appears for all kinds of Event Structures that are extended by priority. We leave the problem on how to fix the definition of lposets as future work.

3.3 Priority versus Enabling and Conflict

Again, as in Section 2, we can reduce the priority relation by removing redundant pairs, i.e. pairs that due to the enabling or conflict relation do not affect the semantics of the PBES. First we can—as already done in PPES—remove a priority pair $e \triangleleft e'$ or $e' \triangleleft e$ between an event e and its cause e' , because an event and its cause are never enabled together. Therefore e.g. the pair $d \triangleleft c$ in Figure 2 (b) is redundant because of $\{b, c\} \mapsto d$. Also a priority pair $e \triangleleft e'$ between two events that are in conflict is redundant, because these conflicting events never occur in the same trace. Consider for example the events b and c in Figure 2 (b). Because of $b\#c$ the pair $c \triangleleft b$ is redundant.

Theorem 3.6. *Let $(\beta, \triangleleft) = (E, \rightsquigarrow, \mapsto, 1, \triangleleft)$ be a PBES and*

$$\triangleleft' = \triangleleft \setminus \{ (e, e'), (e', e) \mid e\#e' \vee (\exists X \subseteq E. e \in X \wedge X \mapsto e') \}.$$

Then $T(\beta, \triangleleft) = T(\beta, \triangleleft')$.

Proof. $T(\beta, \triangleleft) \subseteq T(\beta, \triangleleft')$ follows from Lemma 3.5.

To show $T(\beta, \triangleleft') \subseteq T(\beta, \triangleleft)$, assume a trace $\sigma = e_1 \cdots e_n \in T(\beta, \triangleleft')$. We have to show that $\sigma \in T(\beta, \triangleleft)$, i.e. that $\sigma \in T(\beta)$ and that σ satisfies Condition (7). $\sigma \in T(\beta)$ follows from $\sigma \in T(\beta, \triangleleft')$ by the Definition of traces. σ satisfies Condition (7) when $\forall i < n. \forall e_j, e_h \in \bar{\sigma}. e_j \neq e_h \wedge e_j, e_h \in \text{en}_\beta(\sigma_i) \wedge e_h \triangleleft e_j \implies j < h$. Let us fix $i < n$ and $e_j, e_h \in \bar{\sigma}$. Assume $e_j \neq e_h, e_j, e_h \in \text{en}_\beta(\sigma_i)$, and $e_h \triangleleft e_j$. It remains to prove that $j < h$. Because of the Definition of \triangleleft' , there are three cases for $e_h \triangleleft e_j$:

Case $e_h \triangleleft' e_j$: Then $e_h \triangleleft' e_j \wedge e_j, e_h \in \bar{\sigma} \wedge \sigma \in T(\beta, \triangleleft') \xrightarrow{(7)} j < h$.

Case $e_j\#e_h \vee e_h\#e_j$: This case is not possible, because it is in contradiction to (5), (4), and $e_j, e_h \in \bar{\sigma}$.

Case $\exists X \subseteq E. (e_h \in X \wedge X \mapsto e_j) \vee (e_j \in X \wedge X \mapsto e_h)$: This case is not possible, because it is in contradiction to $e_j, e_h \in \text{en}_\beta(\sigma_i)$ and Lemma 3.2. \square

Note that priority is redundant for all pairs of events that are directly related by the bundle enabling relation or the conflict relation regardless of the direction of the priority pair. We say that this reduction is done at the structure level, since it is done w.r.t. the relations which are part of the Event Structure.

In PPESs enabling is a transitive relation and we can drop all priority pairs between events that are related by enabling. In the case of PBESs neither conflict nor enabling are transitive relations. For

example in the event structure $\begin{array}{ccc} e_1 & e_2 & e_3 \\ \bullet & \text{---} & \bullet & \text{---} & \bullet \end{array}$ (which can be both; a PES as well as a BES) we have $e_1\#e_2$ and $e_2\#e_3$ but not $e_1\#e_3$. Accordingly we cannot drop a priority pair $e_1 \triangleleft e_3$ because else the sequence e_1e_3 becomes a trace.

However in PPESs enabling is transitive, so whenever $e_1 \leq e_2$ and $e_2 \leq e_3$ there is $e_1 \leq e_3$ and we can also drop priority pairs relating e_1 and e_3 (compare e.g. with e, a , and d in Figure 1). In PBES the situation is different. For the PBES in Figure 2 we have $\{a\} \mapsto b$ and $\{b, c\} \mapsto d$ but d does not necessarily depend on a and thus we cannot drop the pair $a \triangleleft d$ since $cad \notin T(\beta, \triangleleft)$. Unfortunately this means that we do not necessarily drop the whole redundancy in priority if we reduce the priority relation as described in Theorem 3.6. For example $e_1 \triangleleft e_3$ is redundant in $(\{e_1, e_2, e_3\}, \emptyset, \{\{e_1\} \mapsto e_2, \{e_2\} \mapsto e_3\}, 1, \{e_1 \triangleleft e_3\})$, because in this special case e_1 is indeed a necessary cause for e_3 . Thus for PBESs \triangleleft' is not necessarily minimal, i.e. we cannot prove $\forall \triangleleft'' \subset \triangleleft'. T((E, \rightsquigarrow, \mapsto, 1, \triangleleft)) \neq T((E, \rightsquigarrow, \mapsto, 1, \triangleleft''))$ as we have done in Theorem 2.4 for PPESs.

For the PBES in Figure 2 the reduction described in Theorem 3.6 indeed suffices to remove all redundant priority pairs. The result is presented in Figure 2 (c).

3.4 Priority versus Precedence

In order to identify some more redundant priority pairs we consider configurations and lposets. If we analyze for example the configurations $\{a, b, c\}$ and $\{a, c, d\}$ of the PBES in Figure 2, we observe that, because of $\{a\} \mapsto b$ and $\{b, c\} \mapsto d$, the priority pair $a \triangleleft d$ is redundant in the first configuration while it is not in the second one. Thus, in some cases, i.e. with respect to some configurations (or lposets), we can also ignore priority pairs of events that are indirectly related by enabling. Since such a redundancy is relative to specific configurations and their traces, and since dropping priority pairs affects the whole set of traces obtained from a ES, we use the term “ignorance” rather than “dropping” for distinction, and we say that this ignorance is done at the configuration level. Priority ignorance is necessary while linearizing configurations and trying to obtain traces.

The cases in which priority pairs are redundant with respect to some configuration C are already well described by the precedence relation \preceq_C , i.e. we can identify redundant priority pairs easily from the lposets for C . Note that in BESs (and also EBESs) each configuration leads to exactly one lposet. The priority pair $a \triangleleft d$ is obviously redundant in the case of $\boxed{a \rightarrow b \rightarrow d}$ but not in the case of $\boxed{a \ c \rightarrow d}$.

To formalize this let $T(\beta, \triangleleft) \upharpoonright_C \triangleq \{\sigma \mid \sigma \in T(\beta, \triangleleft) \wedge \bar{\sigma} = C\}$ be the set of traces over the configuration $C \subseteq E$ for some BES $\beta = (E, \rightsquigarrow, \mapsto, 1)$. Thus $T(\beta, \triangleleft) \upharpoonright_C$ consists of all the traces of $T(\beta, \triangleleft)$ that are permutations of the events in C . Then for all configurations C all priority pairs $e \triangleleft e'$ such that $e' \preceq_C e$ or $e \preceq_C e'$ can be ignored.

Theorem 3.7. *Let (β, \triangleleft) be a PBES, $\langle C, \preceq_C, 1 \rangle \in L(\beta)$, and $\triangleleft' \triangleq \triangleleft \setminus \{(e, e') \mid e' \preceq_C e \vee e \preceq_C e'\}$. Then:*

$$T(\beta, \triangleleft) \upharpoonright_C = T(\beta, \triangleleft') \upharpoonright_C$$

Proof. Note that by induction on \preceq and Lemma 3.2, $e_j \preceq_C e_h$ as well as $e_h \preceq_C e_j$ imply that e_j and e_h cannot be enabled together in a trace of $T(\beta) \upharpoonright_C$. With this argument the proof is straightforward from the definitions of traces, \triangleleft' , traces over a configuration, Lemma 3.5, and (7). \square

Consider once more the PBES (β, \triangleleft) of Figure 2 with respect to the configuration $\{a, b, d\}$. We have $\{a\} \mapsto b$, $\{b, c\} \mapsto d$, and $a \triangleleft d$. As explained before we cannot drop the priority pair $a \triangleleft d$, because of the sequence $cad \notin T(\beta, \triangleleft)$. However with Theorem 3.7 we can ignore $a \triangleleft d$ for the semantics of (β, \triangleleft) if we limit our attention to $\{a, b, d\}$, because $T(\beta, \triangleleft) \upharpoonright_{\{a, b, d\}} = \{abd\} = T(\beta) \upharpoonright_{\{a, b, d\}}$.

For PBESs ignorance ensures that \triangleleft' is minimal with respect a configuration C .

Theorem 3.8. *Let (β, \triangleleft) be a PBES, $\langle C, \preceq_C, 1 \rangle \in L(\beta)$ for some configuration $C \in C(\beta, \triangleleft)$, $\triangleleft' \triangleq \triangleleft \setminus \{(e, e') \mid e' \preceq_C e \vee e \preceq_C e'\}$, and $\triangleleft'' \subset \triangleleft'$. Then $T(\beta, \triangleleft) \upharpoonright_C \neq T(\beta, \triangleleft'') \upharpoonright_C$.*

Proof. Because of $\triangleleft'' \subset \triangleleft'$, there are some $e, e' \in E$ such that $e \triangleleft e'$ but $e \not\triangleleft' e'$, $e' \not\triangleleft_C e$, and $e \not\triangleleft_C e'$. Note that each linearization of a given lposet that respects the precedence relation is a trace [9]. Thus $e' \not\triangleleft_C e$ and $e \not\triangleleft_C e'$ imply that $T(\beta, \triangleleft'') \upharpoonright_C$ contains a trace such that e and e' are enabled together and e precedes e' . Because of $e \triangleleft e'$ such a trace cannot be contained in $T(\beta, \triangleleft) \upharpoonright_C$. So $T(\beta, \triangleleft) \upharpoonright_C \neq T(\beta, \triangleleft'') \upharpoonright_C$. \square

In the following two sections we consider two extensions of Bundle ESs.

4 Priority in Extended Bundle Event Structures

The first extension of Bundle ESs we consider are *Extended Bundle Event Structures (EBESs)*. Bundle ESs were developed to give semantics to LOTOS in [9], but since the conflict relation was symmetric, they could not give semantics to the disable operator of LOTOS. Thus Extended Bundle ESs were introduced in the same reference.

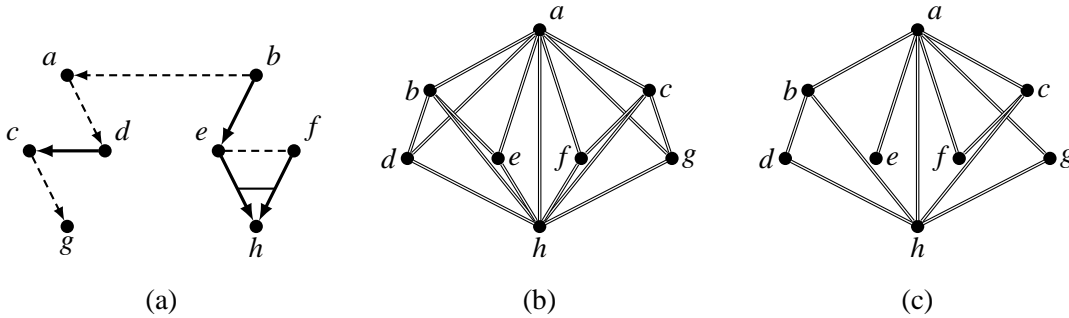


Figure 4: A PEBES $(\varepsilon, \triangleleft)$ with ε in (a) and \triangleleft in (b) as a Hasse diagram with transitivity exposed. (c) shows \triangleleft after dropping redundant priority pairs.

Extended Bundle ESs are similar to Bundle ESs except that the conflict relation is replaced by the so-called *asymmetric conflict* relation or *disabling* relation. If an event e_1 disables another event e_2 , denoted by $e_2 \rightsquigarrow e_1$, then once e_1 takes place e_2 cannot take place anymore, i.e. e_1 can never precede e_2 . Accordingly, disabling can be considered as an exclusion relation. Note that the asymmetric conflict or disabling relation is an irreflexive relation $\rightsquigarrow \subseteq E \times E$ but is not necessarily asymmetric as the name suggests, i.e. $e_1 \rightsquigarrow e_2 \implies e_2 \not\rightsquigarrow e_1$ does not necessarily hold for all events e_1 and e_2 . Therefore Extended Bundle ESs are a generalization of Bundle ESs, and thus are more expressive [9].

Formally an Extended Bundle ES is a quadruple $\varepsilon = (E, \rightsquigarrow, \mapsto, 1)$, where the stability condition is adapted as follows:

$$\text{Stability: } \forall X \subseteq E. \forall e \in E. X \mapsto e \implies (\forall e_1, e_2 \in X. e_1 \neq e_2 \implies e_1 \rightsquigarrow e_2) \quad (\text{SC}')$$

Note that stability again ensures that two distinct events of a bundle set are in mutual conflict. We adapt the Definitions of $\text{en}_\varepsilon(\sigma)$, traces, $\text{T}(\varepsilon)$, configurations, and $\text{C}(\varepsilon)$ accordingly.

Figure 4 (a) shows an example of an EBES. The solid arrows denote causality, i.e. reflect the enabling relation, where the bar between the arrows shows that they belong to the same bundle and the dashed line denotes again a mutual conflict as required by the stability condition (SC'). The dashed arrow denotes disabling, e.g. $b \rightsquigarrow a$ and $ba \in \text{T}(\varepsilon)$ but $ab \notin \text{T}(\varepsilon)$ in this example.

In order to define lposets of EBESs, we have to adapt the precedence relation $\prec_C \subseteq C \times C$ such that it also covers disabling:

$$e \prec_C e' \iff (\exists X \subseteq E. X \mapsto e' \wedge e \in X) \vee e \rightsquigarrow e' \quad (8)$$

Again \preceq_C denotes the reflexive and transitive closure of \prec_C . The Definitions of lposets and $\text{L}(\varepsilon)$ are then adapted accordingly. [9] proves that \preceq_C is a partial order over C and that each linearization (obeying the defined precedence relations) of a given lposet built from an EBES yields an event trace of that structure. Furthermore, it is proved in [9, 8] that given two EBESs $\varepsilon, \varepsilon'$ their lposets are equal iff their traces are equal, i.e. $\text{L}(\varepsilon) = \text{L}(\varepsilon') \iff \text{T}(\varepsilon) = \text{T}(\varepsilon')$.

$(\varepsilon, \triangleleft) = (E, \rightsquigarrow, \mapsto, 1, \triangleleft)$ is a *prioritized Extended Bundle ES (PEBES)*, where $\varepsilon = (E, \rightsquigarrow, \mapsto, 1)$ is an EBES and $\triangleleft \subseteq (E \times E)$ is the acyclic priority relation. Figure 4 illustrates an example of a PEBES with the EBES in Figure 4 (a) and the priority relation in Figure 4 (b). A sequence of events $\sigma = e_1 \cdots e_n$ is a trace of $(\varepsilon, \triangleleft)$ iff 1.) $\sigma \in \text{T}(\varepsilon)$ and 2.) σ satisfies the following constraint:

$$\forall i < n. \forall e_j, e_h \in \bar{\sigma}. e_j \neq e_h \wedge e_j, e_h \in \text{en}_\varepsilon(\sigma_i) \wedge e_h \triangleleft e_j \implies j < h \quad (9)$$

$C \in \text{C}(\varepsilon, \triangleleft)$ iff $\exists \sigma \in \text{T}(\varepsilon, \triangleleft). \bar{\sigma} = C$. Again $\text{T}(\varepsilon, \triangleleft) \subseteq \text{T}(\varepsilon)$ and $\triangleleft' \subseteq \triangleleft$ implies $\text{T}(\varepsilon, \triangleleft) \subseteq \text{T}(\varepsilon, \triangleleft')$.

Lemma 4.1. *Let $(\varepsilon, \triangleleft)$ and $(\varepsilon, \triangleleft')$ be two PEBES with $\triangleleft' \subseteq \triangleleft$. Then $\text{T}(\varepsilon, \triangleleft) \subseteq \text{T}(\varepsilon, \triangleleft')$.*

Proof. Straightforward from the Definition of traces, (9), and $\triangleleft' \subseteq \triangleleft$. □

Similar to PBESs, we can remove a priority pair $e \triangleleft e'$ or $e' \triangleleft e$ between an event e and its cause e' , because an event and its cause are never enabled together. Therefore e.g. the pair $e \triangleleft b$ in Figure 4 is redundant because of $\{b\} \mapsto e$. Also a priority pair $e \triangleleft e'$ between an event e' and its disabler e , i.e. for $e' \rightsquigarrow e$, does not affect the semantics, since e must follow e' anyway. Consider for example the events a and d in Figure 4. Because of $a \rightsquigarrow d$, a always pre-empts d and thus $d \triangleleft a$ is redundant.

Theorem 4.2. *Let $(\varepsilon, \triangleleft)$ be a PEBES and*

$$\triangleleft' \triangleq \triangleleft \setminus \{ (e, e') \mid e' \rightsquigarrow e \vee (\exists X \subseteq E. (e \in X \wedge X \mapsto e') \vee (e' \in X \wedge X \mapsto e)) \}.$$

Then $T(\varepsilon, \triangleleft) = T(\varepsilon, \triangleleft')$.

Proof. Similar to the proof of Theorem 3.6, where the second case is replaced by:

Case $e_j \rightsquigarrow e_h$: Because of the Definition of traces, $\sigma = e_1 \cdots e_n \in T(\varepsilon)$ and $e_j, e_h \in \bar{\sigma}$ imply that $e_j \in \text{en}_\varepsilon(\sigma_{j-1})$. Then $e_j \rightsquigarrow e_h \wedge e_j \in \text{en}_\varepsilon(\sigma_{j-1}) \wedge e_h \in \bar{\sigma} \implies j < h$. \square

Note that priority for events that are directly related by the bundle enabling relation is always redundant, regardless whether the cause has the higher priority or the effect does. On the other hand we can reduce pairs of events that are related by disabling only if the event has a higher priority than its disabler. Consider for example the PEBES $(\{e, e'\}, \{e \rightsquigarrow e'\}, \emptyset, 1, \{e \triangleleft e'\})$. The only traces of this PEBES are e and e' , but if we remove the priority pair $e \triangleleft e'$ we have the additional trace ee' . Similarly we cannot remove the $e \rightsquigarrow e'$ here, because this yields to the additional trace $e'e$.

The result of dropping redundant priority pairs for the PEBES in Figure 4 as described by Theorem 4.2 is presented in Figure 4 (c). Note that after dropping redundant pairs the priority relation is not a partial order anymore.

Again limiting our attention to a specific configuration allows us to ignore some more priority pairs. In contrast to PBESs we can sometimes also ignore priority pairs that overlap with disabling. Consider for example $b \rightsquigarrow a$ and $a \rightsquigarrow d$ of the PEBES in Figure 4. The priority pair $d \triangleleft b$ is redundant with respect to the configuration $\{a, b, d\}$ but not with respect to the configuration $\{b, d\}$. Note that again the direction of the priority pair is important in the case of indirect disabling but not in the case of indirect enabling. If we for instance replace $d \triangleleft b$ in Figure 4 by $b \triangleleft d$, then $\{a, b, d\}$ is not a configuration anymore and $b \triangleleft d$ is not redundant in all remaining configurations containing b and d .

The cases in which priority pairs are redundant with respect to some configuration C are again well described by the precedence relation \preceq_C , i.e. we can identify redundant priority pairs easily from the lposet of C . The priority pair $h \triangleleft b$ is obviously redundant in the case of $\boxed{b \rightarrow e \rightarrow h}$ but not in the case of $\boxed{b \rightarrow f \rightarrow h}$ and $d \triangleleft b$ is obviously redundant in the case of $\boxed{b \rightarrow a \rightarrow d}$ but not in the case of $\boxed{b \ d}$. Let $T(\varepsilon, \triangleleft) \upharpoonright_C \triangleq \{\sigma \mid \sigma \in T(\varepsilon, \triangleleft) \wedge \bar{\sigma} = C\}$ be the set of traces over C . Then for all configurations C we can ignore all priority pairs $e \triangleleft e'$ such that $e' \preceq_C e$.

Theorem 4.3. *Let $(\varepsilon, \triangleleft)$ be a PEBES, $\langle C, \preceq_C, 1 \rangle \in L(\varepsilon)$, and $\triangleleft' \triangleq \triangleleft \setminus \{ (e, e') \in C \times C \mid e' \preceq_C e \}$. Then:*

$$T(\varepsilon, \triangleleft) \upharpoonright_C = T(\varepsilon, \triangleleft') \upharpoonright_C$$

Proof. Note that $T(\varepsilon, \triangleleft) \upharpoonright_C \subseteq T(\varepsilon, \triangleleft)$ and $T(\varepsilon, \triangleleft') \upharpoonright_C \subseteq T(\varepsilon, \triangleleft')$.

By Lemma 4.1, $T(\varepsilon, \triangleleft) \subseteq T(\varepsilon, \triangleleft')$ and thus also $T(\varepsilon, \triangleleft) \upharpoonright_C \subseteq T(\varepsilon, \triangleleft') \upharpoonright_C$.

To show $T(\varepsilon, \triangleleft') \upharpoonright_C \subseteq T(\varepsilon, \triangleleft) \upharpoonright_C$, assume a trace $\sigma = e_1 \cdots e_n \in T(\varepsilon, \triangleleft') \upharpoonright_C$. We have to show that $\sigma \in T(\varepsilon, \triangleleft) \upharpoonright_C$, i.e. that $\forall e \in \bar{\sigma}. e \in C$, $\sigma \in T(\varepsilon)$, and that σ satisfies Condition (9). $\forall e \in \bar{\sigma}. e \in C$ and $\sigma \in T(\varepsilon)$ follows from $\sigma \in T(\varepsilon, \triangleleft') \upharpoonright_C$ by the Definition of traces of PEBESs. σ satisfies Condition (9) if $\forall i < n. \forall e_j, e_h \in \bar{\sigma}. e_j \neq e_h \wedge e_j, e_h \in \text{en}_\varepsilon(\sigma_i) \wedge e_h \triangleleft e_j \implies j < h$. Let us fix $i < n$ and $e_j, e_h \in \bar{\sigma}$. Assume $e_j \neq e_h$, $e_j, e_h \in \text{en}_\varepsilon(\sigma_i)$, and $e_h \triangleleft e_j$. It remains to prove that $j < h$.

Because of the Definition of \prec' , assumption $e_h \prec e_j$ implies that $e_h \prec' e_j$ or $e_j \preceq_C e_h$. In the first case $j < h$ follows, because of the Definition of traces and (9), from $e_h \prec' e_j$, $e_j, e_h \in \bar{\sigma}$, and $\sigma \in T(\mathcal{E}, \prec') \upharpoonright_C$. The other case, i.e. that $e_j \preceq_C e_h$ and $e_j \neq e_h$ implies $j < h$, was already proved in [9]. \square

Consider once more the PEBES (\mathcal{E}, \prec) of Figure 4 with respect to the configuration $\{b, e, h\}$. We have $\{b\} \mapsto e$, $\{e, f\} \mapsto h$, and $h \prec b$. As explained before we cannot drop the priority pair $h \prec b$, because of the trace $fhb \notin T(\mathcal{E}, \prec)$. However with Theorem 4.3 we can ignore $h \prec b$ —and also $h \prec e$ and $e \prec b$ —for the semantics of (\mathcal{E}, \prec) if we limit our attention to $\{b, e, h\}$, because $T(\mathcal{E}, \prec) \upharpoonright_{\{b, e, h\}} = T(\mathcal{E}, (\prec \setminus \{h \prec b, h \prec e, e \prec b\})) \upharpoonright_{\{b, e, h\}}$.

Similarly we can ignore $c \prec a$ if we limit our attention to the configuration $C = \{a, c, d\}$, since $T(\mathcal{E}, \prec) \upharpoonright_C = T(\mathcal{E}, (\prec \setminus \{c \prec a\})) \upharpoonright_C$. Note that here the precedence pair $a \preceq_C c$ that allows us to ignore $c \prec a$ results from the correlation between a disabling pair $a \rightsquigarrow d$ and an enabling pair $\{d\} \mapsto c$. Thus with Theorem 4.3 we can ignore even priority pairs that are redundant in specific situations because of combining enabling and disabling.

This combination prohibits us on the other hand from ignoring priority of the opposite direction, the direction which is compatible with the precedence direction. That is possible only with precedence resulted from enabling purely as it is the case in Theorem 3.7 for PBESs. For instance, suppose that $b \prec h$ for the structure in Figure 4 then we can ignore this priority pair in a configuration $\{b, e, h\}$. That is not formulated in the Theorem 4.3 above, since \preceq_C abstracts from the relation between events. While in contrast to EBESs, the conflict relation is symmetric in Bundle ESs, and precedence results only from enabling. Thus, in contrast to PBESs, we do not have minimality of priority ignorance in PEBESs.

5 Priority in Dual Event Structures

Dual ESs are the second extension of BES examined here. The stability constraint in BESs and EBESs prohibits two events from the same bundle to take place in the same system run. Thus e.g. $\{b, e, f, h\}$ is not a configuration of the EBES in Figure 4. It provides some kind of stability to the causality in the structure. More precisely due to stability in every trace or lposet for every event the necessary causes can be determined. Without the conflict between e and f in the example the trace $befh$ is possible. But then it is impossible to determine whether h is caused in $befh$ by e or f . The stability constraint prohibits such ambiguity. On the other hand such a constraint limits the expressiveness of the structure, and forces an XOR condition between the elements of a bundle set. In some system specifications a more relaxed definition may be useful. Dual ESs provide such a relaxed definition.

The definition of Dual ESs varies between [8] and [10], but both show the causal ambiguity explained above. In [8] Dual ESs are based on Extended Bundle ESs, while in [10] they are based on Bundle ESs. Since we have studied the relation between disabling and priority, and want to focus here on the effect of causal ambiguity on priority, we analyze the version of [10]. A *Dual Event Structure (DES)* is a quadruple $\Delta = (E, \#, \mapsto, 1)$ similar to Definition 3.1 but without the stability condition.

The Definitions of $\text{en}_\Delta(\sigma)$, traces, and $T(\Delta)$ are similar to Section 3 for BESs. Of course the deletion of the stability condition leads to additional traces, e.g. for structure in Figure 2 (a) we obtain the additional traces $abcd, abdc, acbd, acdb, cabd, cadb$, and $cdab$. Figure 5 shows a DES taken from [10].

Causal ambiguity affects the way lposets are built, since the causal order is not clear². In [10] Langerak et al. tried to solve this problem. They illustrated that there are different causality interpre-

²Since there is no disable relation here, causality is the only source of order in lposets.

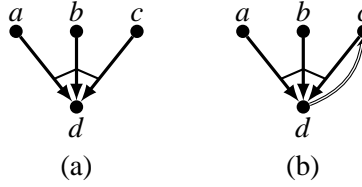


Figure 5: A Dual ES without priority in (a) and with priority in (b).

tations possible for causal ambiguity. They defined five different intensional posets: liberal, bundle-satisfaction, minimal, early and late posets. Intensional means posets are defined depending on the causality relations in the structure, while observational on the other hand means posets are obtained out of event traces (where no structure of the system is available, but only behavior). We examine these different kinds and their relations informally for brevity.

Langerak et al. illustrated that in order to detect the cause of an event like d in the trace $abcd$ of Figure 5, one should consider the prefix of d (i.e. abc) and then can have the following interpretations:

- Liberal Causality: means that any cause out of the prefix is accepted as long as bundles are satisfied: abc, ab, b, ac, bc , etc. are all accepted as a cause for d . Then all events in a cause precede d in

$$\boxed{\begin{array}{c} a \\ b \\ c \end{array} \twoheadrightarrow d}, \boxed{\begin{array}{c} a \\ b \\ c \end{array} \rightarrow d}, \boxed{\begin{array}{c} a \\ b \\ c \end{array} \rightarrow d}, \boxed{\begin{array}{c} a \\ b \\ c \end{array} \twoheadrightarrow d}, \boxed{\begin{array}{c} a \\ b \\ c \end{array} \twoheadrightarrow d}$$

the built poset, e.g. the posets for the last causes are

- Bundle-Satisfaction Causality: bundles are satisfied by exactly one event: b, ab, ac are accepted causes but not abc .
- Minimal Causality: bundles are satisfied so that no subset of a cause is accepted. So b, ac are accepted but not ab or bc .
- Early Causality: the earliest bundle-satisfaction cause is accepted: b is accepted, but not ac as c happened later than b .³
- Late Causality: (cf. [10], it will be skipped here).

[10] shows that equivalence in one kind of posets between two structures implies equivalence in some other specific kinds of posets, but equivalence in any of the kinds implies equivalence in traces.

We add priority to DESs in the same way as before. A *prioritized Dual ES (PDES)* is the tuple (Δ, \prec) , where Δ is a DES and \prec is the acyclic priority relation. Also the definitions of traces, $T(\Delta, \prec)$, configurations, and $C(\Delta, \prec)$ are adapted similar as in the section before.

Since the conflict relation provided here is the same as in Bundle ESs, we can remove redundant priority pairs that overlap with the conflict relation as described in Theorem 3.6, i.e. whenever there is $e\#e'$ or $e'\#e$ then $e \prec e'$ is redundant and can be removed. The situation for enabling is different because of the missing stability condition. The priority pair $c \prec d$ in the PDES in Figure 5 is not redundant, because it removes some traces. The reason is that c is not anymore a necessary cause for d , since d can be enabled by b even if c occurs in the same trace. So at the structure level of PDESs we cannot in general remove priority pairs because of overlappings with the enabling relation.

In the case of PBESs and PEBESs partial orders help to identify redundant priority pairs at the configuration level. Unfortunately, we cannot do the same here. Let us consider the configuration $\{a, b, c, d\}$,

and consider liberal causality. Indeed applying (3.7) on the poset $\boxed{\begin{array}{c} a \\ b \end{array} \twoheadrightarrow d \leftarrow c}$ and the priority $c \prec d$ yields the sequence $abcd$ which is not a trace. On the other hand, considering bundle-satisfaction causal-

³In [11] a procedure is defined to detect how early is a cause depending on binary numbers.

ity, the poset $\boxed{a \begin{matrix} b \\ c \end{matrix} \rightrightarrows d}$ with the same priority yields the same sequence $abcd$ again. The same will be for minimal causality and a poset like $\boxed{a \begin{matrix} \rightrightarrows d \\ c \end{matrix} b}$.

In fact none of the mentioned kinds of posets can be used alone without the priority, and thus ignorance is not possible with causal ambiguity w.r.t. a single poset for a configuration. Even when $c \prec d$ seems to yield pre-emption in the poset $\boxed{a \ b \rightarrow d \ c}$, one can have the linearization $acbd$ which is a trace, and priority turns out to be redundant in this very trace (but not in the whole poset). The reason is partial orders here do not necessarily represent causes.

6 Conclusions

We have added priority to different Event Structures: Prime ESs as a simple model with conjunctive causality, Bundle ESs with disjunctive causality, Extended Bundle ESs with asymmetric conflict, and Dual ESs with causal ambiguity. In all cases, priority leads to trace filtering and limits concurrency and non-determinism. We then analyze the relationship between the new priority relation and the other relations of the ESs. Since priority has an effect only if the related events are enabled together, overlappings between the priority relation and the other relations of the ESs sometimes lead to redundant priority pairs.

In PPESs, PBESs, and PEBESs priority is redundant between events that are related directly by causality. Moreover in all considered ESs priority is redundant between events that are related directly by the conflict relation. But in the case of PEBESs the conflict relation implements asymmetric conflicts. Hence in contrast to the other ESs we have to take the direction of the disabling relation into account.

The main difference between redundancy of priority in PPESs and the other three models is due to events that are indirectly related by causality. In PPESs causality is a transitive relation, i.e. all pairs which are indirectly related by causality are directly related by causality as well. The enabling relation of the other models is not transitive. Thus priority pairs between events that are only indirectly related by enabling are not necessarily redundant. Unfortunately and unlike PPESs, this means that we cannot ensure after removing the redundant priority pairs that the remaining priority pairs necessarily lead to pre-emption. So the other models hold more ambiguity to a modeler.

Instead we show that if we limit our attention to a specific configuration C a priority pair $e \prec e'$ is redundant if $e' \preceq_C e \vee e \preceq_C e'$ for PBESs or if $e' \preceq_C e$ for PEBESs. This allows us to ignore, for the semantics with respect to specific configurations, additionally priority pairs between events indirectly related by enabling for PBESs and by enabling, by disabling, or even by combinations of enabling and disabling for PEBESs. In the case of PBESs we obtain a minimality result this way.

Unfortunately in PDESs even priority pairs between events that are directly related by causality are not necessarily redundant. So from a modelers perspective, priority in DESs hold the biggest ambiguity among all the studied ESs. In other words, one cannot figure out the role priority plays at design time or structure level, and whether this priority yields pre-emption or not. Even at the configuration level, that is not possible in general due to causal ambiguity.

Thus our main contributions are: 1.) We add priority as a binary acyclic relation on events to ESs. 2.) We show that the relation between priority and other event relations of an ES can lead to redundant priority pairs, i.e. to priority pairs that do never (or at least for some configurations not) affect the behavior of the ES. 3.) Then we show how to completely remove such pairs in PPESs and that this is in general not possible in ESs with a more complex causality model like PBESs, PEBESs, or PDESs. 4.) Instead we show how to identify all priority pairs that are redundant with respect to configurations in PBESs and that the situation in PEBESs and DESs is different. 5.) We show how to identify (some of the) redundant

priority pairs at the level of configurations in PEBESs and 6.) that again this is in general not possible in the same way for PDESs.

After dropping or ignoring redundant priority pairs as described above, the minimum potential for overlapping between priority and causality can be found in PPESs, while the maximum is in PDESs. In PPESs all remaining priority pairs indeed affect the semantics, i.e. exclude traces. In PBESs the same holds with respect to specific configurations. In PEBESs after dropping the redundant priority pairs the disabling relation has no overlapping with only priority directed in the opposite direction.

In Section 3.2 we show that adding priority complicates the definition of families of lposets to capture the semantics of prioritized ESs. We observe that because of priority a single configuration may require several lposets to describe its semantics. The same already applies for DESs because of the causal ambiguity. However note that priority does not lead to causal ambiguity. Thus, we can define the semantics of prioritized ESs by families of lposets if we do not insist on the requirement that there is exactly one lposet for each configuration. We leave the definition of such families of lposets for future work. Such families of lposets for prioritized ESs may also help to identify and ignore redundant priority pairs in the case of PEBESs and PDESs. Another interesting topic for further research is to analyze how priority influences the expressive power of ESs.

References

- [1] J.C.M. Baeten, J.A. Bergstra & J.W. Klop (1986): *Syntax and defining equations for an interrupt mechanism in process algebra*. *Fundamenta Informaticae* 9, pp. 127–167.
- [2] F. Bause (1996): *On the analysis of Petri nets with static priorities*. *Acta Informatica* 33(7), pp. 669–685, doi:10.1007/s002360050065.
- [3] F. Bause (1997): *Analysis of Petri nets with a dynamic priority method*. In: *Proceedings of Application and Theory of Petri Nets, LNCS 1248*, Springer, pp. 215–234, doi:10.1007/3-540-63139-9_38.
- [4] G. Boudol & I. Castellani (1991): *Flow models of distributed computations: event structures and nets*. Technical Report, INRIA.
- [5] J.W. Bryans, J.S. Fitzgerald, C.B. Jones & I. Mozolevsky (2006): *Dimensions of Dynamic Coalitions*. Technical Report, Newcastle upon Tyne.
- [6] J. Camilleri & G. Winskel (1995): *CCS with Priority Choice*. *Information and Computation* 116(1), pp. 26–37, doi:10.1006/inco.1995.1003.
- [7] R. Cleaveland, G. Lüttgen & V. Natarajan (1999): *Priority in Process Algebra*. ICASE report, NASA.
- [8] J.P. Katoen (1996): *Quantitative and Qualitative Extensions of Event Structures*. Ph.D. thesis, Twente.
- [9] R. Langerak (1992): *Transformations and Semantics for LOTOS*. Ph.D. thesis, Twente.
- [10] R. Langerak, E. Brinksma & J.P. Katoen (1997): *Causal ambiguity and partial orders in event structures*. In: *Proceedings of CONCUR, LNCS, Springer*, pp. 317–331, doi:10.1007/3-540-63141-0_22.
- [11] R. Langerak, R. Brinksma & J.P. Katoen (1997): *Causal ambiguity and partial orders in event structures*. Technical Report, Twente.
- [12] G. Lüttgen (1998): *Pre-emptive Modeling of Concurrent and Distributed Systems*. Ph.D. thesis, Passau.
- [13] A. Rensink (1992): *Posets for Configurations!* In: *Proceedings of CONCUR, LNCS 630*, Springer, pp. 269–285, doi:10.1007/BFb0084797.
- [14] R. van Glabbeek & U. Goltz (2001): *Refinement of actions and equivalence notions for concurrent systems*. *Acta Informatica* 37, pp. 229–327, doi:10.1007/s002360000041.
- [15] G. Winskel (1980): *Events in Computation*. Ph.D. thesis, Edinburgh.