Event Structures as a Foundation for Process Model Differencing, Part 1: Acyclic processes

Abel Armas-Cervantes, Luciano García-Bañuelos, and Marlon Dumas

Institute of Computer Science, University of Tartu, Estonia
{abel.armas,luciano.garcia,marlon.dumas}@ut.ee

Abstract. This paper considers the problem of comparing process models in terms of their behavior. Given two process models, the problem addressed is that of explaining their differences in terms of simple and intuitive statements. This model differencing operation is needed for example in the context of process consolidation, where analysts need to reconcile differences between process variants in order to produce consolidated process models. The paper presents an approach to acyclic process model differencing based on event structures. First the paper considers the use of prime event structures. It is found that the high level of node duplication inherent to prime event structures hinders on the usefulness of the difference diagnostics that can be extracted thereon. Accordingly, the paper defines a method for producing (asymmetric) event structures with reduced duplication.

1 Introduction

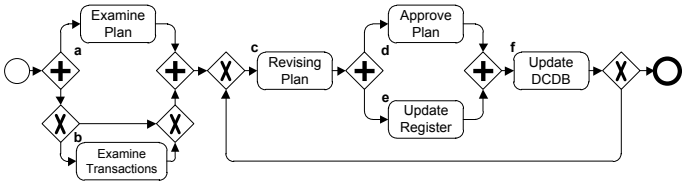
Large companies with mature business process practices often manage multiple versions of the same process. Such variants may stem from distinct products, different types of customers (e.g. corporate vs. private customers), different legislations across countries in which a company operates, or idiosyncratic choices made by multiple business units over time. In some scenarios, analysts need to find ways to consolidate multiple process variants into a single one for the purpose of improving efficiency and creating economies of scale. In this setting, analysts need to accurately understand the differences between multiple variants in order to determine how to reconcile them. In this paper, we define a process model differencing operator that provides intuitive guidance to analysts, allowing them to understand the differences between a pair of process variants.

Behavioral profiles (BPs) [1] are a promising approach to tackle problems pertaining to the management of process variants. The idea behind BPs is to encode a process in terms of behavioral relations between every pair of tasks. In BPs, a pair of tasks is related by one of three types of relations: *strict order*($\mapsto$), *exclusive order*($+$) or *interleaving*($|||$). BPs have been used for defining behavioral similarity metrics [2] and for process comparison and merging [3], among other applications. Nevertheless, BPs suffer from the following issues:

1. *BPs do not correspond to any known notion of equivalence.* Specifically, two processes may have the same BP while not being trace equivalent. Reciprocally, two processes may be trace equivalent, yet have different BPs.

2. *BPs mishandle the following patterns of behavior*: (a) task skipping, (b) behavior inside cycles – which is confused with interleaved parallelism – and (c) duplicate tasks.

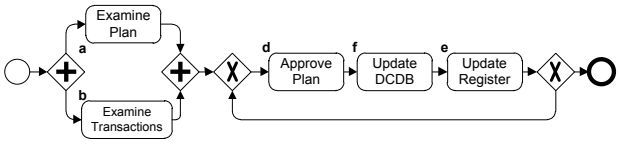
Consider for example the variants of a land development process depicted in Figure 1(a)-(c), and their BPs presented aside. Firstly, even if we abstract away from task "c" both variants are non-trace equivalent, yet their BPs are identical (Issue 1). Secondly, note that task "b" is not always executed in the first variant, meaning that it can be skipped (Figure 1(a)). This fact is not captured in the BPs (Issue 2a). Finally, consider tasks "d","e" and "f" present in both variants. The order in which these tasks are executed is captured in both BPs using the *interleaving* relation. However, the actual order clearly differs between the two variants. This issue stems from the fact that these tasks are contained in a cycle and BPs confuse cycles and interleaved parallelism (Issue 2b).



(a)

	a	b	c	d	e	f
a	+	‖	↦	↦	↦	↦
b	‖	+	↦	↦	↦	↦
c	↤	↤	‖	‖	‖	‖
d	↤	↤	‖	‖	‖	‖
e	↤	↤	‖	‖	‖	‖
f	↤	↤	‖	‖	‖	‖

(b)



(c)

	a	b	d	e	f
a	+	‖	↦	↦	↦
b	‖	+	↦	↦	↦
d	↤	↤	‖	‖	‖
e	↤	↤	‖	‖	‖
f	↤	↤	‖	‖	‖

(d)

Fig. 1. Variants of land development process

The paper makes a step forward by presenting an approach to process model differencing that does not suffer from issues 1 and 2a above – although it still suffers from not being able to handle cyclic models nor models with duplicate tasks. The presented approach is based on Event Structures (EVs), which allow us to ensure that two models are treated as equivalent iff they are fully concurrent bisimilar (cf. issue 1). First, the paper considers the use of Prime Event Structures (PES) for process model differencing. However, it is found that PESs inherently involve a significant amount of duplication, which degrades the usefulness of the diagnosis. To address this issue, the use of Acyclic Event Structures (AES) is considered. It is shown that AES can provide a more compact representation (less duplication), thus leading to a more compact diagnosis of differences.

In the proposed approach, differences between process models are captured by means of a (sparse) matrix wherein each cell represents a difference involving one or two tasks. This matrix can be directly translated into simple and intuitive statements. For instance, the difference between the process models depicted in

Figure 1 can be expressed in terms of statements of the following form:[1] "In model (a), task b **sometimes** precedes $\{d, e, f\}$, whereas in model (b) task b **always** precedes $\{d, e, f\}$", "Task c is present only in model (a)", "In model (a), task d and e are executed in any order, while d always precedes e in model (b)".

The discussion throughout the paper is developed assuming that the input process models are given as Petri nets. However, this is not a limitation given the existence of mappings from common process modeling notations to Petri nets.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces some notions and notation used in the rest of the paper. Section 4 presents the acyclic process model differencing approach based on PES and AES. Finally, Section 5 concludes and discusses directions for future work.

2 Related work

Approaches for process model comparison may be divided into those based on node label similarity, process structure similarity and behavior similarity [4, 5]. In this paper we focus on behavior similarity. Nevertheless, we acknowledge that node label similarity plays an important in the alignment of nodes (e.g., tasks) across the process models being compared. Here, we assume that such an alignment is given, i.e. for each node label in one model we are given the corresponding ("equivalent") node label in the other model. This is equivalent to assuming that the same node labels are used across both models being compared.

There exists a number of equivalence notions for concurrent systems [6]. Several methods for testing the equivalence of two systems according to these notions have been developed. However, most of these methods focus on determining whether or not two systems are equivalent. Relatively few previous research delves into the question of diagnosing all differences between two systems.

Perhaps one of the earliest work on diagnosing concurrent system differences is [7], which presents a technique to derive equations in a process algebra characterizing the differences between two *Labeled Transition Systems* (LTSs). This technique iteratively traverses the data structures used for testing bisimulation equivalence and generates a minimal equation characterizing differences for pairs of states in the input LTSs. However, the use of a process algebra makes the feedback difficult to grasp for end users (process analysts in our context). In [8], a method for assessing the dissimilarity of LTSs in terms of "edit" operations is presented. However, we contend that providing feedback in the form of a series of elementary edit operations (add or remove events) is not simple and intuitive, since it does not directly tell the analyst what relations exist in one model that do not exist in the other. [9] presents a method for diagnosing differences between pairs of process models using standard automata theory. Differences between a pair of models are captured by means of templates of the form "a node in a model has more dependencies than in the other one" or "a node in a model may be executed multiple times in one model but at most once in the other".

[1] Note that the cycle is not taken into account in this comparison.

The method of [9] suffers from two major limitations. First, the set of reported differences is not guaranteed to be complete. Second, the differences are given in a coarse-grained manner. With respect to the example in Figure 1, [9] gives as diagnosis that task "c" has additional dependencies in the first model viz. the second, without explaining the additional dependencies. We note that all three techniques above rely on LTSs and are not able to recognize concurrency relations. Thus, these techniques do not diagnose differences such as "in one model two tasks are done concurrently while in another they are done in sequence."

In this paper, we rely on event structures to capture the behavior of process models. Event structures represent concurrent systems in terms of behavioral relations between events. Other representations of process models in terms of behavioral relations have been proposed in the literature. *Causal footprints* [10] represent behavior as a set of tuples, each comprising a task, its preset, and its postset. These sets of tuples are then mapped to points in a vector space and euclidian distance is used to compute a metric of similarity between a given pair of footprints. This technique however is not intended to provide a diagnosis of differences between pairs of models. Alpha relations are another example of a representation of process models using behavioral relations [11]. Alpha relations include direct causality, conflict and concurrency, and are derived from execution logs for the purpose of process mining. Alpha causality is not transitive. This choice makes alpha relations unsuitable for process model comparison, because causality has a localized scope. Moreover, alpha relations cannot capture so-called "short loops" and hidden tasks (including scenarios where a task may be skipped). The ordering relations graph [12, 13] is another representation of (acyclic) process models in terms behavioral relations. However, it too has problems with hidden tasks. The class of asymmetric event structures characterized in the present paper is a refinement of the ordering relations graph.

3 Preliminaries

This section covers basic concepts on Petri nets and partial ordered sets. In particular, the notions of *branching processes*, *behavior relations*, and *fully concurrent bisimulation* are reviewed because they are cornerstones to our approach.

3.1 Petri nets

Definition 1 (Petri net, Net system). *A tuple* (P, T, F) *is a* Petri net*, where* P *is a set of* places*,* T *is a set of* transitions*, with* $P \cap T = \varnothing$*, and* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs. A* marking $M : P \to \mathbb{N}_0$ *is a function that associates each place* $p \in P$ *with a natural number (viz., place tokens). A* net system $N = (P, T, F, M_0)$ *is a Petri net* (P, T, F) *with an* initial marking M_0*.*

Places and transitions are conjointly referred to as *nodes*. We write $\bullet y = \{x \in P \cup T \mid (x, y) \in F\}$ and $y\bullet = \{z \in P \cup T \mid (y, z) \in F\}$ to denote the *preset* and *postset* of node y, respectively. F^+ and F^* denote the irreflexive and reflexive transitive closure of F, respectively.

The semantics of a net system can be defined in terms of markings. A marking M *enables* a transition t if $\forall p \in \bullet t : M(p) > 0$. Moreover, the occurrence of t leads to a new marking M', with $M'(p) = M(p) - 1$ if $p \in \bullet t \setminus t\bullet$, $M'(p) = M(p) + 1$ if $p \in t\bullet \setminus \bullet t$, and $M'(p) = M(p)$ otherwise. We use $M \xrightarrow{t} M'$ to denote the occurrence of t. The marking M_n is said to be reachable from M if there exists a sequence of transitions $\sigma = t_1 t_2 \dots t_n$ such that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$. A marking M of a net is *n-safe* if $M(p) \leq n$ for every place p. A net system N is said *n-safe* if all its reachable markings are *n-safe*. In the following we restrict ourselves to *1-safe* net systems. Hence, we identify the marking M with the set $\{p \in P \mid M(p) = 1\}$.

A labeled Petri net $N = (P, T, F, \lambda)$ is a net (P, T, F), where $\lambda : P \times T \to \Lambda \cup \{\tau\}$ is function that associates a node with a label. Given a node x, if $\lambda(x) \neq \tau$ then x is said to be *observable*, otherwise x is said to be *silent*. A labeled net system $N = (P, T, F, M_0, \lambda)$ is similarly defined. An example of labeled net system is shown in Figure 2. There, transitions display their corresponding label inside the rectangle. Note that t_2 is a silent transition, hence $\lambda(t_2) = \tau$. Herein, we consider each place to be labeled with its corresponding identifier, e.g., $\lambda(p_0) = p_0$.

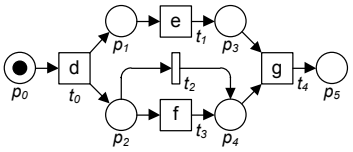


Fig. 2. A labeled net system

3.2 Deterministic and branching processes

The partial order semantics of a net system is commonly formulated in terms of runs or, more precisely, prefixes of runs that are referred to as *deterministic processes* [14]. A process itself can be represented as an acyclic net with no branching nor merging places, i.e., $\forall p \in P : |\bullet p| \leq 1 \wedge |p\bullet| \leq 1$. Alternatively, all runs can be accommodated in a single tree-like structure, called *branching process* [14], that may contain branching places and explicitly represents three behavior relations: *causality*, *concurrency* and *conflict* defined as follows.

Definition 2 (Behavior relations). *Let $N = (P, T, F)$ be a net and $x, y \in P \cup T$ two nodes in N.*

- *x and y are in* causal *relation, denoted as $x <_N y$, iff $(x, y) \in F^+$. The* inverse causality *relation is denoted $>_N$. We use $\leq_N$ to the denote the* reflexive causality *relation.*
- *x and y are in* conflict*, denoted as $x \#_N y$, iff there exist two transitions $t, t' \in T$ such that t and t' are distinct, $\bullet t \cap \bullet t' \neq \emptyset$, and $(t, x), (t', y) \in F^*$. If $x \#_N x$ then x is said to be in* self-conflict*.*
- *x and y are* concurrent*, denoted as $x \|_N y$, iff neither $x <_N y$, nor $y <_N x$, nor $x \#_N y$.*

The tuple $\mathcal{R} = (<_N, \#_N, \|_N)$ denotes the behavior relations of N. We can now provide a formal definition for branching process.

Definition 3 (Branching process). *Let $N = (P, T, F, M_0)$ be a net system. The* branching process *$\beta = (B, E, G, \rho)$ of N is the net (B, E, G) defined by the*

inductive rules in Figure 3. The rules also define the function $\rho : B \cup E \to P \cup T$ *that maps each node in* β *to a node in* N. $\varrho(B)$ *is a shorthand for* $\bigcup_{b \in B} \rho(b)$.

For sake of clarity, the elements of B and E in a branching process are called *conditions* and *events*, respectively. $Min(\beta)$ denotes the set of minimal elements of $B \cup E$ with respect to the transitive closure of G. Henceforth, $Min(\beta)$ corresponds to the set of places in the initial marking of N, i.e., $\varrho(Min(\beta)) = M_0$. Figure 4 presents the branching process of the net system from Figure 2. Note that every node in the branching process has multiple labels. The label outside of the node, say "$e_0(t_0)$", indicates that the underlying event is named "e_0" and that it corresponds to transition "t_0" in the net system. The events in the branching process also display the label in the original net system, e.g., "b". This label can be determined by a composition of functions λ and ρ, i.e., $\lambda_\beta =_{def} \lambda_N \circ \rho_\beta$.

$$\frac{p \in M_0}{b = \langle \varnothing, p \rangle \in B \qquad \rho(b) = p}$$

$$\frac{t \in T \quad B' \subseteq B \quad B'^2 \subseteq \|_\beta \quad \varrho(B') = {\bullet t}}{e = \langle B', t \rangle \in E \qquad \rho(e) = t}$$

$$\frac{e = \langle B', t \rangle \in E \qquad t\bullet = \{p_1, \ldots, p_n\}}{b_i = \langle t', p_i \rangle \in B \qquad \rho(b_i) = p_i}$$

Fig. 3. Branching process, inductive rules

One important characteristic of a branching process is that it does not contain merging conditions. To overcome this restriction, some nodes in the net system need to be represented more than once in the branching process. When comparing Figures 2 and Figure 4, we can notice that place p_4 is mapped to conditions b_4 and b_5, place p_5 is in turn mapped to b_6 and b_7, and that transition t_4 is mapped to events e_4 and e_5.

Note that G in a branching process is acyclic, and hence G^+ is a partial order. Moreover, no event $e \in E$ is in self-conflict. Engelfriet [14] showed that every Petri net has a unique (possibly infinite) maximal branching process up to isomorphism, a.k.a. *net unfolding*. The branching process of an acyclic net system is finite.

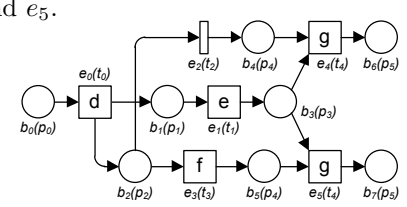


Fig. 4. Branching process of the net system in Figure 2

We continue by formally defining *deterministic processes*, which will be used for introducing the notion of equivalence we rely on, namely *fully concurrent bisimulation*. By the same token, we introduce the notion of *configuration*.

Definition 4 (Configuration, deterministic process). *Let* $N = (P, T, F, M_0)$ *be a net system and* $\beta = (B, E, G, \rho)$ *its corresponding branching process.*

- *A* configuration C *of* β *is a set of events,* $C \in E$*, such that:*
  - *i)* C *is causally closed, i.e.,* $e \in C \Rightarrow e' \leq_\beta e : e' \in C$*, and*
  - *ii)* C *is conflict free, i.e.,* $\forall e, e' \in C : \neg(e \#_\beta e')$.
- *A* deterministic process $\pi = (B_\pi, E_\pi, G_\pi, \rho)$ *is the net induced by a configuration* C*, where* $B_\pi = {\bullet C} \cup C\bullet$*,* $E_\pi = C$*, and* $G_\pi = G \cap (B_\pi \times E_\pi \cup E_\pi \times B_\pi)$.
- *The* initial deterministic process *of* N*, denoted* $\hat{\pi}$*, is the process induced by the empty configuration* $C = \varnothing$.

Let C and C' be configurations of β, such that $C \subset C'$, and π and π' be the processes induced by C and C', respectively. Moreover, if $X = C' \setminus C$, then we write $\pi' = \pi \oplus X$ and we say that π' is an *extension* of π. Given a process π we are interested in extensions to π appending exactly one observable event, whenever such extension exists, i.e., $\pi \oplus X$ such that $|\{x \mid x \in X \wedge \lambda_\beta(x) \neq \tau\}| = 1$. With abuse of notation, we shall write $\pi \oplus_\lambda X$ to denote a process extension with exactly one observable event. In a branching process, each event e can be associated with a unique set of events that causally precede e, denoted $\lceil e \rceil = \{e' \mid e' \leq e\}$, and referred to as the *local configuration* of e.

3.3 Fully Concurrent Bisimulation

The notion of equivalence that we adopt in this work is known as *fully concurrent bisimulation* [15]. This bisimulation can be informally stated as follows: given two net systems, any sequence of observable events (viz., run or process) that might be possibly performed by one net system must also be possibly performed by the other net system and vice-versa, as for other conventional bisimulations, but additionally causal dependencies between observable events must be preserved in the bisimulation. The latter is required to be in-line with the partial order semantics. Below, we provide the corresponding formal definition.

Definition 5 (Fully concurrent bisimulation). *Let $N_1 = (P_1, T_1, F_1, M_0^1, \lambda_1)$ and $N_2 = (P_2, T_2, F_2, M_0^2, \lambda_2)$ be labeled net systems, and $\beta_1 = (B_1, E_1, G_1, \rho_1)$ and $\beta_2 = (B_2, E_2, G_2, \rho_2)$ be their corresponding branching processes. The set of triples $\mathcal{B} \subseteq \Pi_1 \times \Gamma \times \Pi_2$ is a* fully concurrent bisimulation *for N_1 and N_2* iff*:*

- *Π_1 and Π_2 are the set of deterministic processes of N_1 and N_2, respectively, and $\Gamma : E_1 \to E_2$ is a relation from observable events in β_1 to observable events in β_2.*
- *If $\hat{\pi}_1$ and $\hat{\pi}_2$ are the initial deterministic processes of N_1 and N_2, respectively, then $(\hat{\pi}_1, \varnothing, \hat{\pi}_2) \in \mathcal{B}$.*
- *Γ is a bijection w.r.t. labeling, i.e., $\forall e \in Dom(\Gamma) : \lambda_{\beta_1}(e) = \lambda_{\beta_2}(\Gamma(e))$, preserving causality, i.e., $\forall e, e' \in Dom(\Gamma) : e <_{\pi_1} e' \Leftrightarrow \Gamma(e) <_{\pi_2} \Gamma(e')$.*
- *$\forall (\pi_1, \Gamma, \pi_2) \in \mathcal{B}$:*
  - *a) if $\pi_1' = \pi_1 \oplus_{\lambda_1} X_1$ is an extension of π_1 with exactly one observable event, then there exists a tuple $(\pi_1', \Gamma', \pi_2') \in \mathcal{B}$ with $\pi_2' = \pi_2 \oplus_{\lambda_2} X_2$ and $\Gamma \subseteq \Gamma'$.*
  - *b) if $\pi_2' = \pi_2 \oplus_{\lambda_2} X_2$ is an extension of π_2 with exactly one observable event, then there exists a tuple $(\pi_1', \Gamma', \pi_2') \in \mathcal{B}$ with $\pi_1' = \pi_1 \oplus_{\lambda_1} X_1$ and $\Gamma \subseteq \Gamma'$.*

Two net systems N_1 and N_2 are said fully concurrent bisimulation equivalent*, denoted $N^1 \approx_{fcb} N^2$, if there exists a fully concurrent bisimulation for them.*

Note that fully concurrent bisimulation is defined in terms of process extensions with exactly one observable event and, hence, there is implicitly an abstraction of invisible behavior. This abstraction is convenient for our purposes, such that we lift it to the the level of behavior relations. Let $N = (P, T, F, M_0, \lambda)$ be a labeled net system and $\beta = (B, E, G, \rho)$ be its branching process. Moreover, let E' be the set of observable events, i.e., $E' = \{e \mid e \in E \wedge \lambda_\beta(e) \neq \tau\}$. Then

$\mathcal{R}|_\lambda = (<_\beta \cap E'^2, \#_\beta \cap E'^2, <_\beta \cap E'^2)$ defines the *observable behavior relations*, that is the behavior relations of N restricted to observable behavior.

If there exists a mapping of transitions in two net systems such that this mapping preserves their underlying behavior relations, then we say their behavior relations are isomorphic. This is formally defined as follows.

Definition 6 (Isomorphism of observable behavior relations). *Let* $N_1 = (P_1, T_1, F_1, M_0^1, \lambda_1)$ *and* $N_2 = (P_2, T_2, F_2, M_0^2, \lambda_2)$ *be labeled net systems, and* $\beta_1 = (B_1, E_1, G_1, \rho_1)$ *and* $\beta_2 = (B_2, E_2, G_2, \rho_2)$ *be their corresponding branching processes. Moreover, let* $E_1' \subseteq E_1$ *and* $E_2' \subseteq E_2$ *be the set of observable events of the branching processes of* N_1 *and* N_2*, and* $\mathcal{R}|_{\lambda_1}$ *and* $\mathcal{R}|_{\lambda_2}$ *their corresponding observable behavior relations.* $\mathcal{R}|_{\lambda_1}$ *and* $\mathcal{R}|_{\lambda_2}$ *are said* isomorphic*, denoted* $\mathcal{R}|_{\lambda_1} \cong \mathcal{R}|_{\lambda_2}$, iff *there exists a bijection* $\varphi : E_1' \to E_2'$ *such that:*

- $\forall e \in E_1' : \lambda_{\beta_1}(e) = \lambda_{\beta_2}(\varphi(e))$*, and*
- $\forall e, e' \in E_1' : (e <_{\beta_1} e' \Leftrightarrow \varphi(e) <_{\beta_2} \varphi(e')) \vee (e \#_{\beta_1} e' \Leftrightarrow \varphi(e) \#_{\beta_2} \varphi(e')) \vee (e \|_{\beta_1} e' \Leftrightarrow \varphi(e) \|_{\beta_2} \varphi(e'))$.

The following Theorem establishes the relation between fully concurrent bisimulation equivalence and isomorphism of observable behavior relations for two net systems. The corresponding proof can be found in [12].

Theorem 1 ([12]). *Let* $N_1 = (P_1, T_1, F_1, M_0^1, \lambda_1)$ *and* $N_2 = (P_2, T_2, F_2, M_0^2, \lambda_2)$ *be labeled net systems. Moreover, let* $T_1' \subseteq T_1$ *and* $T_2' \subseteq T_2$ *be the set of observable transitions of* N_1 *and* N_2*, respectively. Assume there exists a bijection* $\psi : T_1' \to T_2'$ *such that* $\forall t \in T_1' : \lambda_1(t) = \lambda_2(\psi(t))$*, then the following holds:*

$$N_1 \approx_{fcb} N_2 \quad \Leftrightarrow \quad \mathcal{R}|_{\lambda_1} \cong \mathcal{R}|_{\lambda_2}$$

3.4 Partial Ordered Sets

We conclude this section by recalling some definitions from the theory of partial ordered sets (posets) [16]. Let $\langle D, \sqsubseteq \rangle$ be a poset. For a subset X of D, an element $y \in D$ is an upper (lower) bound of X, iff $x \sqsubseteq y$ ($x \sqsupseteq y$), for each element $x \in X$. An element $y \in D$ is a greatest (least) element, iff for each element $x \in D$ the property $x \sqsubseteq y$ ($x \sqsupseteq y$) holds. An element $y \in D$ is a maximal (minimal) element, iff there is no element $x \in D$ s.t. $y \sqsubset x$ ($x \sqsubset y$); D_{max} and D_{min} denote the sets of maximal and minimal elements of D, respectively. Two elements $x, y \in D$ are *consistent*, denoted $x \uparrow y$, iff they have a joint upper bound, i.e., $x \uparrow y \Leftrightarrow \exists\ z \in D : x \sqsubseteq z \wedge y \sqsubseteq z$; otherwise, they are said *inconsistent*. A subset X of D is *pairwise consistent*, written $X^{\Uparrow}$, iff every pair of elements in X is consistent in D, i.e., $X^{\Uparrow} \Leftrightarrow \forall x, y \in X : x \uparrow y$. A poset $\langle D, \sqsubseteq \rangle$ is *coherent*, iff each pairwise consistent subset X of D has a least upper bound (lub) $\sqcup X$. An element $x \in D$ is a *complete prime*, iff for each subset X of D, with lub $\sqcup X$, it holds $x \sqsubseteq \sqcup X \Rightarrow \exists\ y \in X : x \sqsubseteq y$. We shall write $\mathbb{P}_P$ to denote the set of complete primes of a poset P. A poset $P = \langle D, \sqsubseteq \rangle$ is *prime algebraic*, iff $\mathbb{P}_P$ is denumerable and every element in D is lub of the set of complete primes it dominates, i.e., $\forall\ x \in D : x = \sqcup\{y \mid y \in \mathbb{P}_P \ \wedge \ y \sqsubseteq x\}$. A set S is *denumerable*, iff it is empty or there exists an enumeration of S that is a surjective mapping from the set of positive integers onto S.

4 Comparison of acyclic process models

In this section, we explore the use of event structures (EVs) in the process model differencing. The presentation is organized in two parts. Subsection 4.1 introduces the more basic EV, namely *Prime Event Structures* (PESs). By the same token, we define an operator on event structures, that operationalizes the comparison of behavior relations. In spite of their faithful representation of behavior, PESs incur in a great amount of duplication. In Subsection 4.2, we consider *Asymmetric Event Structures* (AESs) as an alternative representation that reduces the amount of duplication observed in PESs.

4.1 Prime Event Structures

A *Prime Event Structure* (PES) is a model for computation introduced in [16]. Aligned with the concepts of in the previous section, we formally define PESs.

Definition 7 ((Labeled) Prime Event Structure). *Let $N = (P, T, F, M_0, \lambda)$ be a net system and $\beta = (B, E, G, \rho)$ be its corresponding branching process. The* prime event structure *of β is the tuple $\xi = (E, \leq_\xi, \#_\xi)$, where $\leq_\xi = \leq_\beta \cap E^2$ and $\#_\xi = \#_\beta \cap E^2$. A* labeled prime event structure *also considers a labeling function $\lambda_\xi = \lambda_N \circ \rho$, that associates each event $e \in E$ with the label of its corresponding transition $t \in T$, i.e., $\lambda_\xi(e) = \lambda_N(t) \;\Rightarrow\; \rho(e) = t$.*

The conflict relation $\#_\xi$ is said *hereditary* w.r.t. $\leq_\xi$, meaning that for all $e, e', e'' \in E$, if $e \#_\xi e'$ and $e' \leq_\xi e''$ then $e \#_\xi e''$. The behavior relations of a PES $\xi = (E, \leq_\xi, \#_\xi)$ are given by the tuple $\mathcal{R}_\xi = (<_\xi, \#_\xi, E^2 \setminus (<_\xi \cup >_\xi \cup \#_\xi))$ [16]. Clearly, PES behavior relations correspond to the behavior relations of the branching process of a net system as introduced in Definition 2, but restricted to the set of events. Armed with Theorem 1, we can use a finer restriction and focus only on observable behavior. A PES without invisible behavior shall be denoted $\bar{\xi}$.

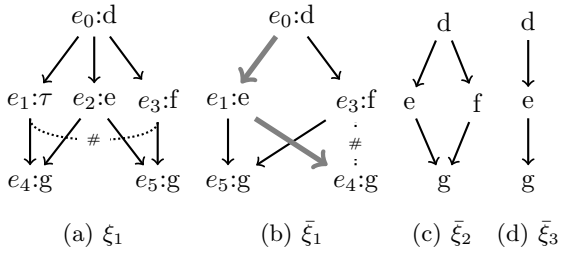


Fig. 5. Examples of prime event structures

Figures 5(a) and 5(b) present the PESs with and without invisible behavior for the net system in Figure 2, in the form of a labeled graph. There, nodes correspond to events, (solid) directed edges represent causality, e.g., e_0:d $\longrightarrow$ e_1:e in Figure 5(b), whereas (dotted, decorated) undirected edges represent conflict, e.g., e_3:f $\cdot\cdot\#\cdot\cdot$ e_4:g also in Figure 5(b). Both transitive causal and hereditary conflict relations are not shown to simplify the graph. When a pair of events is neither direct nor transitively connected, such events are considered to be concurrent. To further simplify the graphs, nodes in a PES will only display event labels and not their identifiers, when it is clear from the context, e.g., Figures 5(c)–(d).

	e_0:d	e_1:e	e_3:f	e_4:g	e_5:g
e_0:d	∥	<	<	<	<
e_1:e	>	∥	∥	<	<
e_3:f	>	∥	∥	#	<
e_4:g	>	>	#	∥	#
e_5:g	>	>	>	#	∥

(a) $\mathcal{R}_{\bar{\xi}_1}$

	d	e	f	g
d	∥	<	<	<
e	>	∥	∥	<
f	>	∥	∥	<
g	>	>	>	∥

(b) $\mathcal{R}_{\bar{\xi}_2}$

	d	e	g
d	∥	<	<
e	>	∥	<
g	>	>	∥

(c) $\mathcal{R}_{\bar{\xi}_3}$

Fig. 6. Matrix representation

The PES $\bar{\xi}_2$ is a variant of behavior for the net system in Figure 2, where the transition labeled f is not skipped (i.e., the silent transition t_2 is not present). The differences in behavior of $\bar{\xi}_1$ and $\bar{\xi}_2$ are evident. There is one run in $\bar{\xi}_1$ involving events $\{d, e, g\}$, cf., path highlighted with thick gray edges, and that is not present in $\bar{\xi}_2$. Note that the occurrence of event e_4:g precludes that of event e_3:f, because they are "in conflict", which corresponds with the intuition that the transition labeled f may be skipped.

Alternatively, the behavior relations of PESs can be represented with matrices, as illustrated in Figure 6. The subindexes of the relations were omitted for the sake of readability. As usual, we write $\mathcal{R}_{\xi}[e, e']$ to refer to the cell in the intersection of the row associated to event e and the column of e'. Therefore, $\mathcal{R}_{\bar{\xi}_2}[e, f] = \|$ asserts the fact that events e and f are concurrent in $\bar{\xi}_2$. Note that all behavior relations are represented in the matrix, including the inverse causal relation. Moreover, every event is self-concurrent, that by definition.

In order to characterize the differences on the behavior relations displayed by two PESs, we define a binary operator as follows.

Definition 8 (Symmetric difference of PES behavior relations). *Let $\xi_1 = (E_1, \leq_{\xi_1}, \#_{\xi_1}, \lambda_{\xi_1})$ and $\xi_2 = (E_2, \leq_{\xi_2}, \#_{\xi_2}, \lambda_{\xi_2})$ be labeled prime event structures, and let $\mathcal{R}_{\xi_1} = (<_{\xi_1}, \#_{\xi_1}, \|_{\xi_1})$ and $\mathcal{R}_{\xi_2} = (<_{\xi_2}, \#_{\xi_2}, \|_{\xi_2})$ be their corresponding behavior relations. The mappings of events w.r.t. labelings $\mu : E_1 \to E_2$ and $\breve{\mu} : E_1 \cup \{\omega\} \to E_2 \cup \{\omega\}$, where ω is a marker for identifying missing events, are defined as:*

- *μ maps pairs of events having the same labeling, i.e., $\forall (e_1, e_2) \in \mu : \lambda_{\xi_1}(e_1) = \lambda_{\xi_2}(e_2)$, and*
- *$\breve{\mu}$ extends μ by including event mismatches, such that if $(E_1 \setminus dom(\mu))$ and $(E_2 \setminus cod(\mu))$ are the set of events that could not be mapped in μ, then they are paired with ω, i.e., $\breve{\mu} = \mu \cup (E_1 \setminus dom(\mu)) \times \{\omega\} \cup \{\omega\} \times (E_2 \setminus cod(\mu))$.*

Let $(e_1, e_2), (e'_1, e'_2) \in \breve{\mu}$ be event mappings. The symmetric difference *of $\mathcal{R}_{\xi_1}$ and $\mathcal{R}_{\xi_2}$, denoted $\mathcal{R}_{\xi_1} \vartriangle \mathcal{R}_{\xi_2}$, is defined as follows:*

$$\mathcal{R}_{\xi_1} \vartriangle \mathcal{R}_{\xi_2}\,[(e_1, e_2), (e'_1, e'_2)] = \begin{cases} \cdot & \text{if } \mathcal{R}_{\xi_1}[e_1, e'_1] = \mathcal{R}_{\xi_2}[e_2, e'_2] \\ (\mathcal{R}_{\xi_1}[e_1, e'_1], \mathcal{R}_{\xi_2}[e_2, e'_2]) & \text{if } \mathcal{R}_{\xi_1}[e_1, e'_1] \neq \mathcal{R}_{\xi_2}[e_2, e'_2] \\ (\omega,\ \mathcal{R}_{\xi_2}[e_2, e'_2]) & \text{if } e_1 = \omega \text{ or } e'_1 = \omega \\ (\mathcal{R}_{\xi_1}[e_1, e'_1],\ \omega) & \text{if } e_2 = \omega \text{ or } e'_2 = \omega \end{cases}$$

where ω stands for unspecified behavior relation.

	(d,d)	(e,e)	(f,f)	(e_5:g,g)	(e_4:g,g)
(d,d)	·	·	·	·	·
(e,e)	·	·	·	·	·
(f,f)	·	·	·	·	$(\#, <)$
(e_5:g,g)	·	·	·	·	$(\#, \|\|)$
(e_4:g,g)	·	·	$(\#, >)$	$(\#, \|\|)$	·

(a) $\mathcal{R}_{\bar{\xi}_1} \triangle \mathcal{R}_{\bar{\xi}_2}$

	(d,d)	(e,e)	(f,ω)	(g,g)
(d,d)	·	·	$(<, \omega)$	·
(e,e)	·	·	$(\|\|, \omega)$	·
(f,ω)	$(>, \omega)$	$(\|\|, \omega)$	$(\|\|, \omega)$	$(<, \omega)$
(g,g)	·	·	$(>, \omega)$	·

(b) $\mathcal{R}_{\bar{\xi}_2} \triangle \mathcal{R}_{\bar{\xi}_3}$

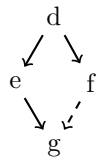
Fig. 7. Example symmetric difference of the behavior relations of PESs in Figure 5

Figures 7(a) presents the symmetric difference of the behavior relations of PES $\bar{\xi}_1$, $\bar{\xi}_2$ (cf., Figure 5). Every cell filled up with "·" stands for a perfect match on the behavior of the process models being compared. Interestingly, in this example a large portion of behavior is matched. Let us consider the last column in the matrix. Due to the symmetry of the relations, the analysis of the last row would lead to similar conclusions. On the one hand, $\mathcal{R}_{\bar{\xi}_1} \triangle \mathcal{R}_{\bar{\xi}_2}[(\text{f,f}), (e_4\text{:g,g})] = (\#, <)$ should be read as *"for the first process model, in some cases, task f is in conflict with task g whereas in the other process model task f always precedes task g"*. On the other hand, $\mathcal{R}_{\bar{\xi}_1} \triangle \mathcal{R}_{\bar{\xi}_2}[(e_5\text{:g,g}), (e_4\text{:g,g})] = (\#, \|)$ should be read as *"in the first process model task g is involved in two different computations (i.e., runs) that are in conflict"*. It should be recalled that, by definition, task g is self-concurrent. The overall conclusion would be that task f may be skipped in the first process model. The matrix in Figure 7(b) illustrates the case of missing tasks. The interpretation of this matrix is rather simple: $\mathcal{R}_{\bar{\xi}_2} \triangle \mathcal{R}_{\bar{\xi}_3}[(d,d), (f,\omega)] = (<, \omega)$ should be read as *"task f is only present in the first process model where, additionally, d precedes f"*, and so forth.

PESs provide a faithful representation of behavior but at the expense of duplication. This duplication stems from the fact that a new branch in the branching process is started at the point where each conflict place is found. Conversely, concurrency does not induce duplication. Hence, in the worst case scenario the number of events is exponential in size w.r.t. the average fan-out of the conflict places, i.e., $O(m^n)$ where n is the number of events and m is the average size of the poset of places. An additional side-effect is that the comparison of PESs requires a combinatorial number of matches among duplicate events.

4.2 Asymmetric Event Structures

To address the problems above, we consider an alternative to PESs known as *Asymmetric Event Structures* (AESs) [17]. In addition to the usual causality relation $\leq$, an AES introduces the *asymmetric conflict* relation $\nrightarrow$. Given two events $e, e' \in E$, we say that e is in asymmetric conflict with e', denoted $e \nrightarrow e'$, with two intuitive interpretations: (i) whenever both e and e' occur in a run, e is observed before e', and (ii) the occurrence of



(a) ϑ_1

	d	e	f	g
d	$\|\|$	<	<	<
e	>	$\|\|$	$\|\|$	<
f	>	$\|\|$	$\|\|$	$\nrightarrow$
g	>	>	$\nleftarrow$	$\|\|$

(b) $\mathcal{R}_{\vartheta_1}$

	(d,d)	(e,e)	(f,f)	(g,g)
(d,d)	·	·	·	·
(e,e)	·	·	·	·
(f,f)	·	·	·	$(<, \nrightarrow)$
(g,g)	·	·	$(>, \nleftarrow)$	·

(c) $\mathcal{R}_{\bar{\xi}_2} \triangle \mathcal{R}_{\vartheta_1}$

Fig. 8. Comparison with AESs

e' precludes that of e. We note that in the
original definition of AES, duplication of events is not a concern. Henceforth, we develop an approach to identifying the asymmetric conflict relation.

When comparing the AES ϑ_1 (cf., Figure 8(a)) with the PES $\bar{\xi}_1$ (cf., Figure 5(b)), both of them displaying the observed behavior of the net system in Figure 2, one can immediately note a more compact representation. While in $\bar{\xi}_1$, there are two events with the same label, namely $e4$:g and $e5$:g, in ϑ_1 only one event carries the label g. It turns out that the PES $\bar{\xi}_2$ (cf., Figure 5(c)) is also an AES, such that we can compare the observable behavior relations $\mathcal{R}_{\bar{\xi}_2}$ and $\mathcal{R}_{\vartheta_1}$ directly. The corresponding symmetric difference is presented in Figure 8(c).

In the following, we present a method to compute the asymmetric conflict relation. We start by computing an equivalence relation on the underlying branching process, relation that relies on node labeling and that respects the local environment of events[2]. Such an equivalence, referred to as *future equivalence*, has been introduced in [18] and can be formally stated as follows.

Definition 9 (Future equivalence). *Let $\beta = (B, E, G, \rho, \lambda_\beta)$ be a labeled branching process. The relation $\sim \subseteq B^2 \cup E^2$ on nodes of β is a* future equivalence *iff:*

- $\forall x, x' \in B \cup E : x \sim x' \Rightarrow \lambda_\beta(x) = \lambda_\beta(x') \ \wedge (\ x \#_\beta x' \vee \ x = x')$, *and*
- $\forall e, e' \in E : e \sim e' \Rightarrow \langle \bullet e \rangle_\sim = \langle \bullet e' \rangle_\sim \ \wedge \ \langle e \bullet \rangle_\sim = \langle e' \bullet \rangle_\sim$.

where $\langle x \rangle_\sim = \{x' \mid x' \sim x\}$ and $\langle X \rangle_\sim = \{\langle x \rangle_\sim \mid x \in X\}$.

Going back to the branching process in Figure 4, its corresponding future equivalence relation is $\{\{e_4{:}\mathrm{g}, e_5{:}\mathrm{g}\}, \{b_4{:}p_4, b_5{:}p_4\}, \{b_6{:}p_5, b_7{:}p_5\}\}$. Interestingly, the future equivalence relation can be used to *fold* a branching process into a Petri net that exhibits the same behavior (see [18, Theorem 8.7] for a formal proof). Indeed, after merging all future equivalent nodes of the branching process in Figure 4, we shall obtain the Petri net in Figure 2. An algorithm to compute future equivalences suitable for our setting is described in [19]. In contrast to previous work, we require future equivalent nodes to be in conflict: merging two concurrent events would eliminate one event from the run, merging two causal events would introduce a loop, both cases are undesirable in our setting.

Intuitively, an equivalence class $\langle e \rangle_\sim$ identifies a set of nodes in a branching process from which runs evolve isomorphically. Conversly, an equivalence class $\langle e \rangle_\sim$ can possibly have multiple different causes, collectively referred to as *history*. Such an intuition is formally defined as $\mathcal{H}(e) = \{\lceil e' \rceil \mid e' \in \langle e \rangle_\sim\}$. Now, let $e \in E$ be an event, then (i) the events in $\bigcap \mathcal{H}(e)$ are the *strict causes* of e, i.e., every event $e' \in \bigcup \mathcal{H}(e)$ is always observed before e, independently of the run, (ii) the events in $\bigcup \mathcal{H}(e) \setminus \bigcap \mathcal{H}(e)$ are the *weak causes* of e, i.e., every event $e' \in \bigcup \mathcal{H}(e) \setminus \bigcap \mathcal{H}(e)$ is observed before e in at least one run, but not in all runs. The notion of weak causes of an event is indeed the way we use for identifying the *asymmetric conflict* relation. The following definition formalizes the transformation of an AES.

Definition 10 (Branching process to AES). *Let $\beta = (B,\ E,\ G,\ \rho,\ \lambda_\beta)$ be a labeled branching process, and let $\sim$ be a future equivalence on β. The tuple*

[2] The environment of an event is to the set of conditions in its preset and postset

$\vartheta = (E_\vartheta,\ \leq^{\text{full}}_\vartheta,\ \#_\vartheta,\ \dashrightarrow_\vartheta,\ \lambda_\vartheta)$ *is the* asymmetric event structure *(AES) of* β *induced by* $\sim$*, where*

$$\begin{array}{lcl} E_\vartheta & = & \{\langle e\rangle_\sim \mid e \in E\},\\ \leq^{\text{full}}_\vartheta & = & \{(\langle e\rangle_\sim, \langle e'\rangle_\sim) \mid e, e' \in E \ \wedge\ e \leq_\beta e'\},\\ \#_\vartheta & = & \{(\langle e\rangle_\sim, \langle e'\rangle_\sim) \mid e, e' \in E \ \wedge\ (e, e') \in \#_\beta \setminus \sim\},\\ \dashrightarrow_\vartheta & = & \{(\langle e\rangle_\sim, \langle e'\rangle_\sim) \mid e, e' \in E \ \wedge\ e \in \bigcup \mathcal{H}(e') \setminus \bigcap \mathcal{H}(e')\},\ \textit{and}\\ \lambda_\vartheta(\langle e\rangle_\sim) & = & \lambda_\beta(e),\ \textit{for all } e \in E. \end{array}$$

Note that the relation $\leq^{\text{full}}_\vartheta$ is a partial order and embeds the asymmetric conflict relation. Henceforth, for AESs we decompose $\leq^{\text{full}}_\vartheta$ into the *strict cause* relation, i.e., $<_\vartheta = <^{\text{full}}_\vartheta \setminus \dashrightarrow_\vartheta$, and the *asymmetric conflict* relation, i.e., $\dashrightarrow_\vartheta$. The relation $\leq^{\text{full}}_\vartheta$ can then be trivially derived from $<_\vartheta$ and $\dashrightarrow_\vartheta$. Technically, $e \#_\vartheta e'$ can also be represented as $e \dashrightarrow_\vartheta e' \ \wedge\ e' \dashrightarrow_\vartheta e$, but we consider that the symmetric conflict is more appropriate as feedback to modelers. As for PES, the concurrency relation on AES is given by $\|_\vartheta = E^2_\vartheta \setminus (<^{\text{full}}_\vartheta \cup >^{\text{full}}_\vartheta \cup \#_\vartheta)$. The behavior relations of ϑ shall be denoted by the tuple $\mathcal{R}_\vartheta = (<_\vartheta,\ \#_\vartheta,\ \dashrightarrow_\vartheta,\ \|_\vartheta)$.

Definition 11 (AES Configuration). *Let* $\vartheta = (E_\vartheta,\ \leq^{\text{full}}_\vartheta,\ \#_\vartheta,\ \dashrightarrow_\vartheta,\ \lambda_\vartheta)$ *be an AES. An* AES configuration *of* ϑ *is a set of events* $C \subseteq E_\vartheta$ *such that*

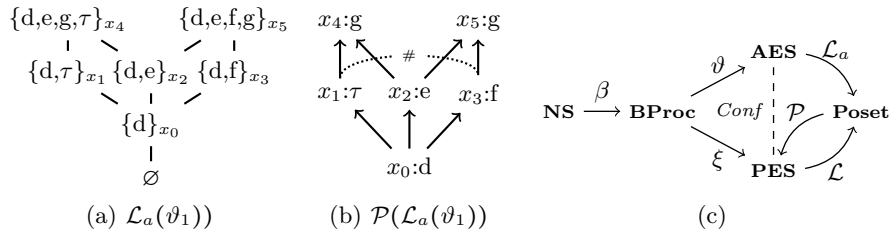
- $<^{\text{full}}_\vartheta \cap\, C^2$ *is* well-founded*, i.e., it forms a non-infinite chain,*
- *for all* $e \in C$*, if* $e' <^{\text{full}}_\vartheta e \ \wedge\ e' \notin C$*, then there exists* $e'' \in C$ *s.t.* $e' \#_\vartheta e''$*.*

As a consequence of the last condition, an AES configuration C is conflict-free, i.e., $\forall e, e' \in C : \neg(e \#_\vartheta e')$. The set of all AES configurations of an AES ϑ shall be denoted $Conf(\vartheta)$. In spite of the differences in their definitions, the set of configurations of a branching process β (cf., Definition 4) is the same as the set of AES configurations (cf., Definition 11) induced by a future equivalence $\sim$ on β. Intuitively, an equivalence class $\langle e\rangle_\sim$ corresponds to a set of events in the branching process from which computation evolves independently, but isomorphically in different branches. When $\langle e\rangle_\sim$ is mapped to a single event into an AES all the branches stemming from events in $\langle e\rangle_\sim$ are merged into a single branch. However, the original set of configurations for $\langle e\rangle_\sim$ can still be rebuilt by appending a copy of the merged branch to each configuration $\lceil e'\rceil$ in $\mathcal{H}(e) = \{\lceil e'\rceil \mid e' \in \langle e\rangle_\sim\}$. This intuition is formally confirmed in [19, Lemma 1]. Now, we define an order $\sqsubseteq$ on AES configurations, referred to as *AES configuration extension*, such that $C \sqsubseteq C'$ stands for "C can evolve into C'".

Definition 12 (AES Configuration Extension). *Let* $\vartheta = (E_\vartheta, \leq^{\text{full}}_\vartheta, \#_\vartheta, \dashrightarrow_\vartheta, \lambda_\vartheta)$ *be an AES and* $X, X' \subseteq E_\vartheta$ *sets of events. We say that* X' extends X*, denoted* $X \sqsubseteq X'$*, iff (i)* $X \subseteq X'$*, and (ii)* $\neg(e' <^{\text{full}}_\vartheta e)$ *for all* $e \in X, e' \in X' \setminus X$*.*

The relation above defines a partial order on the set of configurations of AES ϑ, denoted $\mathcal{L}_a(\vartheta) = \langle Conf(\vartheta),\ \sqsubseteq\rangle$. Following the approaches in [16] and [20], we now characterize $\mathcal{L}_a(\vartheta)$.

Theorem 2. *Let* $\vartheta = (E_\vartheta,\ \leq^{\text{full}}_\vartheta,\ \#_\vartheta,\ \dashrightarrow_\vartheta,\ \lambda_\vartheta)$*. Then,* $\mathcal{L}_a(\vartheta) = \langle Conf(\vartheta),\ \sqsubseteq\rangle$ *is a prime algebraic coherent partial order. Its complete primes are AES configurations of the form* $\lceil e\rceil_\mathcal{X} = \{e' \mid e, e' \in \mathcal{X} \ \wedge\ (e', e) \in \leq^{\text{full}}_\vartheta \cap\, \mathcal{X}^2\}$*, with* $\mathcal{X} \in Conf(\vartheta)$*.*



(a) $\mathcal{L}_a(\vartheta_1)$) (b) $\mathcal{P}(\mathcal{L}_a(\vartheta_1))$ (c)

Fig. 9. (a) Poset, and (b) PES induced from AES ϑ_1. (c) Overall transformations.

Proof. Let $\mathcal{X} \subseteq Conf(\vartheta)$ be a set of pairwise consistent AES configurations, i.e., $\mathcal{X}^{\Uparrow}$. If $e, e' \in \cup\{X \mid X \in \mathcal{X}\}$, then there exist $X, Y \in \mathcal{X}$ s.t. $e \in X$ and $e' \in Y$. Since $\mathcal{X}^{\Uparrow}$ there exists an AES configuration Z s.t. $X \sqsubseteq Z$ and $Y \sqsubseteq Z$. Therefore $\neg(e \#_\vartheta e')$ since Z is conflict-free, $\cup\{X \mid X \in \mathcal{X}\} \in Conf(\vartheta)$, and $\sqcup\mathcal{X} = \cup\mathcal{X}$ is lub in $Conf(\vartheta)$. Hence, $\mathcal{L}_a(\vartheta)$ is coherent.

For any AES configuration $X \in Conf(\vartheta)$, we have $X = \sqcup\{\lceil e \rceil_X \mid e \in X\}$. Hence, if X is complete prime, then there exists $e \in X$ s.t. $X = \lceil e \rceil_X$. This shows that the complete primes of $Conf(\vartheta)$ are AES configurations of the form $\lceil e \rceil_X$. Moreover, the formula $X = \sqcup\{\lceil e \rceil_X \mid e \in X\}$ alludes to the fact that any AES configuration is the lub of the complete primes it dominates. $\mathbb{P}_{\mathcal{L}_a(\vartheta)}$ is denumerable since it is a subset of the power set of a finite denumerable set, i.e., the set of events E_ϑ. Hence, $\mathcal{L}_a(\vartheta)$ is prime algebraic. □

Given the poset $\mathcal{L}_a(\vartheta) = \langle Conf(\vartheta), \sqsubseteq \rangle$, we know from [16] that $\mathcal{P}(\mathcal{L}_a(\vartheta)) = (\mathbb{P}_P, \leq, \#)$ is a prime event structure, where $\leq$ is $\sqsubseteq$ restricted to $\mathbb{P}_{\mathcal{L}_a(\vartheta)}$, and for all $\lceil e \rceil, \lceil e' \rceil \in \mathbb{P}_{\mathcal{L}_a(\vartheta)} : \lceil e \rceil \# \lceil e' \rceil$ iff $\lceil e \rceil$ and $\lceil e' \rceil$ are inconsistent in $\mathcal{L}_a(\vartheta)$. Moreover, a labeling function for such PES is given by $\lambda_{\mathcal{P}(\mathcal{L}_a(\vartheta))}(\lceil e \rceil) = \lambda_\vartheta(e)$ for all $\lceil e \rceil \in \mathbb{P}_{\mathcal{L}_a(\vartheta)}$. By applying the notions above, we can derive the poset shown in Figure 9(a) and then the PES in Figure 9(b). The isomorphism of Figures 9(b) and 5(a) is not a surprise, considering the canonicity of posets [16]. All transformations implied in the approach are summarized in Figure 9(c).

An AES built using Definition 10 considers all the equivalence classes in the branching process. However, the comparison of AESs may happen in two phases, first with observable behavior and then with a subset of τ-labeled events. We conjecture that only a subset of τ-labeled events needs to be kept to allow the construction of AES configurations and preserve fully concurrent bisimulation.

5 Conclusion

In this work, we defined an acyclic process model differencing operator based on prime event structures (PESs) and asymmetric event structures (AESs). The high level of duplication inherent to PES hinders on the usefulness of this representation for the purpose at hand. Hence, AESs were considered as an alternative to reduce duplication. A tailor-made method for computing AESs was described.

We foresee a number of avenues for future research. First, we aim at characterizing the minimal set of τ-labeled events required to preserve fully concurrent bisimulation. Naturally, we target to extend the current approach so as to cover

process models with cycles and duplicate tasks. A promising direction for dealing with cyclic process models includes techniques of net unfoldings for finding a finite representation of cyclic behavior. Finally, future research will include experiments to assess the readability of the feedback stemming from this approach and its applicability in real-world settings.

References

1. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. IEEE TOSEM **37**(3) (2011) 410–429
2. Weidlich, M., Mendling, J., Weske, M.: A Foundational Approach for Managing Process Variability. In: Proc. CAiSE 2011. LNCS 6741. Springer (2011) 267–282
3. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity - a proper metric. In: Proc. BPM 2011. LNCS 6896. Springer (2011) 166–181
4. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Sys. **36**(2) (2011) 498–516
5. Dumas, M., García-Bañuelos, L., Dijkman, R.: Similarity search of business process models. IEEE Data Eng. Bull. **32**(3) (2009) 23–28
6. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Inf. **37** (2001) 229–327
7. Cleaveland, R.: On automatically explaining bisimulation inequivalence. In: Proc. CAV 1991. LNCS 531. Springer (1991) 364–372
8. Sokolsky, O., Kannan, S., Lee, I.: Simulation-based graph similarity. In: Proc. TACAS 2006. LNCS 3920. Springer (2006) 426–440
9. Dijkman, R.: Diagnosing differences between business process models. In: Proc. BPM 2008. Volume 5240 of LNCS 5240. Springer (2008) 261–277
10. van Dongen, B., Dijkman, R., Mendling, J.: Measuring Similarity between Business Process Models. In: Proc. CAiSE 2008. Volume 5074. (2008) 450–464
11. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE TKDE **16**(9) (2004) 1128–1142
12. Polyvyany, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. Inf. Sys. **37**(6) (2012) 518–538
13. Polyvyanyy, A., García-Bañuelos, L., Fahland, D., Weske, M.: Maximal structuring of acyclic process models. CoRR **abs/1108.2384** (2011)
14. Engelfriet, J.: Branching processes of Petri nets. Acta Inf. **28** (1991) 575–591
15. Best, E., Devillers, R., Kiehn, A., Pomello, L.: Concurrent bisimulations in Petri nets. Acta Inf. **28** (1991) 231–264
16. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. Theoretical Computer Science **13** (1981) 85–108
17. Baldan, P., Corradini, A., Montanari, U.: Contextual petri nets, asymmetric event structures, and processes. Information and Computation **171**(1) (2001) 1–49
18. Fahland, D.: From Scenarios to Components. PhD thesis, Humboldt-Universität zu Berlin (2010)
19. Fahland, D., van der Aalst, W.: Simplifying mined process models: An approach based on unfoldings. In: Proc. BPM 2011. LNCS 6896. Springer (2011) 362–378
20. Boudol, G., Castellani, I.: Flow Models of Distributed Computations: Event Structures and Nets. Technical Report 1482, INRIA Sophia-Antipolis (1991)