Behavioral Comparison of Process Models Based on Canonically Reduced Event Structures

Abel Armas-Cervantes[1], Paolo Baldan[2], Marlon Dumas[1], and Luciano García-Bañuelos[1]

[1] Institute of Computer Science, University of Tartu, Estonia
{abel.armas,marlon.dumas,luciano.garcia}@ut.ee
[2] Department of Mathematics, University of Padova, Italy.
baldan@math.unipd.it

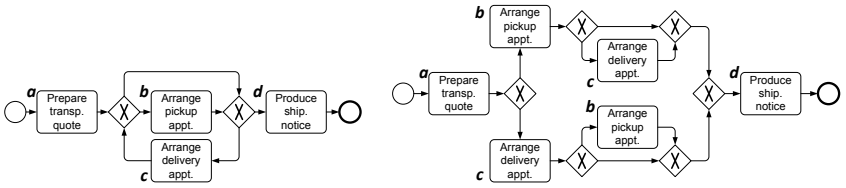
Abstract. We address the problem of diagnosing behavioral differences between pairs of business process models. Specifically, given two process models, we seek to determine if they are behaviorally equivalent, and if not, we seek to describe their differences in terms of behavioral relations captured in one model but not in the other. The proposed solution is based on a translation from process models to Asymmetric Event Structures (AES). A naïve version of this translation suffers from two limitations. First, it produces redundant difference diagnostic statements because an AES may contain unnecessary event duplication. Second, it is not applicable to process models with cycles. To tackle the first limitation, we propose a technique to reduce event duplication in an AES while preserving canonicity. For the second limitation, we propose a notion of unfolding that captures all possible causes of each event in a cycle. From there we derive an AES where repeated events are distinguished from non-repeated ones and that allows us to diagnose differences in terms of repetition and causal relations in one model but not in the other.

1 Introduction

Comparing models of business process variants is a basic operation when managing collections of process models [1]. In some cases, syntactic matching of nodes or edges are sufficient to understand differences between two variants. However, two variants may be syntactically different and still be behaviorally equivalent or they may be very similar syntactically but quite different behaviorally, as changes in a few gateways or edges may entail significant behavioral differences.

This paper presents a technique to compare business processes in terms of behavioral relations between tasks. The technique diagnoses differences in the form of binary behavioral relations (e.g., causality and conflict) that hold in one model but not in the other. For example, given the models in Fig. 1[3] we seek to describe their differences via statements of the form: *"In model M_1, after* Prepare transportation quote *it is possible to execute either* Arrange delivery appointment

[3] Based on an order fulfillment process presented in [2].



(a) M_1 (b) M_2

Fig. 1. Variants of business process models

and Produce shipment notice*, or only* Produce shipment notice*; whereas in* M_2 *after* Prepare transportation quote*,* Arrange delivery appointment *is followed by* Produce shipment notice*".* The diagnosis also considers cyclic behavior, e.g. *"In model* M_1*, activity* Arrange delivery appointment *can be executed many times in a run; whereas in* M_2 *it is executed only once".*

The key idea of the proposal is to compare abstract representations of the process models based on binary behavioral relations. If two process models have isomorphic abstract representations, then they are behaviorally equivalent and otherwise we can use error-correcting graph matching to diagnose the differences. To this end, we adopt a well-known model of concurrency known as *event structures* [3], where computations are represented via events (activity occurrences) and behavioral relations between events. There are different types of event structures comprising different relations, such as *Prime Event Structures* [3] (PESs) and *Asymmetric Event Structures* [4] (AESs). For the purpose of comparison, more compact representations are desirable as they lead to more concise diagnosis of relations existing in one process and not in the other. In this respect, AESs are more compact than PESs and in prior work [5], we proposed a behavior-preserving folding of AESs. However, the work in [5] shows that in some cases multiple non-isomorphic AESs exist that represent the same behavior.

The contributions of the paper are threefold: (i) we extend our work in [5], by proposing a deterministic order on the folding of AESs that produces a canonical represention of the behavior of a given process model, (ii) we extend our approach to support the differencing of process models with cycles, and (iii) we propose an approach to verbalize differences. For the sake of presentation, we assume that the input process are represented as Petri nets. Transformations from other process modeling notations (e.g. BPMN) to Petri nets are given elsewhere [6].

The paper is structured as follows, Section 2 discusses related work. Section 3 provides definitions of notions used in the rest of the paper. The proposed techniques are presented in Section 4. Finally Section 5 summarizes the contributions and discusses future work.

2 Related work

Approaches for process model comparison can be divided into those based on node label similarity, process structure similarity and behavior similarity [1]. In

this paper we focus on behavioral similarity. Nevertheless, we acknowledge that node label similarity plays an important role in the alignment of nodes (e.g., tasks) across the process models being compared. In our work, 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.

There are many equivalence notions for concurrent systems [7], ranging from trace equivalence (processes are equivalent if they have the same set of traces) to bisimulation equivalence, to finer equivalences which preserve some concurrency features of computations (two models are equivalent if they have same sets of runs taking into account concurrency between events). Few methods have been proposed to diagnose differences between processes based on various notions of equivalence. The paper [8] presents a technique to derive equations in a process algebra characterizing the differences between two *Labeled Transition Systems* (LTSs). The use of a process algebra makes the feedback difficult to grasp for end users (process analysts in our context) and the technique relies on a notion of equivalence that does not take into account the concurrent structure in the process (a process model with concurrency and its sequential simulation are equivalent). In [9], a method for assessing dissimilarity of LTSs in terms of "edit" operations is presented. However, such feedback on LTSs does not tell the analyst what relations exist in one model that do not exist in the other. Also, it is based on a notion of equivalence that does take concurrency into account. The same remarks apply to [10], which presents a method for diagnosing differences between pairs of process models using standard automata theory. In addition, in [10] the set of reported differences is not guaranteed to be complete.

Behavioral Profiles (BP) [11] and *Causal Behavior Profiles* [12] are two approaches to represent processes using binary relations. They abstract a process using a $n \times n$ matrix, where n is the number of tasks in the process. Each cell contains one out of three relations: *strict order*, *exclusive order* or *interleaving*; plus an additional *co-occurrence* relation in the case of causal behavioral profiles. Both techniques are incomplete as they mishandle several types of constructs, e.g., task skipping (silent transitions), duplicate tasks, and cycles. In this case, two processes can have identical BPs despite not being behaviorally equivalent.

Alpha relations [13] are another representation of processes using binary behavioral relations (direct causality, conflict and concurrency), proposed in the context of process mining. Alpha causality is not transitive (i.e. causality has a localized scope) making alpha relations unsuitable for behavior comparison [14]. Moreover, alpha relations cannot capture so-called "short loops" and hidden tasks (including task skipping). *Relation sets* [15] are a generalization of alpha relations. Instead of one matrix, the authors use k matrices (with a variable k). In each matrix, causality is computed with a different look-ahead. It is shown that 1-lookahead matrices induce trace equivalence for a restricted family of Petri nets. The authors claim that using k matrices improves accuracy. But it is unclear how human-readable diagnostics of behavioral differences could be extracted from two sets of k matrices and it is unclear to what notion of equivalence would the diagnostics correspond.

3 Preliminaries

This section introduces some fundamental notions on *Petri nets*, *branching processes* and *event structures* that will be used in subsequent parts of the paper.

3.1 Petri nets

Definition 1 (Petri net, Net system). *A tuple* $N = (P, T, F)$ *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 $S = (N, M_0)$ *is a Petri net* $N = (P, T, F)$ *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 is defined in terms of markings. A marking M *enables* a transition t if $\forall p \in \bullet t : M(p) > 0$, denoted as $(N, M)[t\rangle$. 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 $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$. The set of all the markings reachable from a marking M is denoted $[M\rangle$. 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 $\{p \in P \mid M(p) = 1\}$.

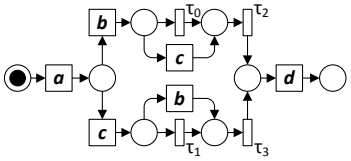


Fig. 2. N_2 of Fig. 1(b)

A labeled Petri net $N = (P, T, F, \lambda)$ has a function $\lambda : P \cup T \to \Lambda \cup \{\tau\}$ that associates a node with a label. A transition x is said to be *observable* if $\lambda(x) \neq \tau$, otherwise x is *silent*. A labeled net system $S = (N, M_0, \lambda)$ is similarly defined. An example of a labeled net system is shown in Fig. 2, the transitions display their corresponding label inside the rectangle if they are observable.

3.2 Deterministic and branching processes

The partial order semantics of a net system can be formulated in terms of runs or, more precisely, prefixes of runs that are referred to as *deterministic processes*[4] [16]. A process can be represented as an acyclic net with no branching nor

[4] Here and in the rest of this section, the term *process* refers to a control-flow abstraction of a business process based on a partial order semantics.

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* [3], which can 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 Petri net and $x, y \in P \cup T$ two nodes in N.*

- *x and y are in* causal *relation, denoted $x <_N y$, iff $(x, y) \in F^+$. The* inverse causal *relation is denoted $>_N$. By $\leq_N$ we denote the* reflexive causal *relation.*
- *x and y are in* conflict*, denoted $x \#_N y$, 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 $x \#_N y$.*

We can now provide a formal definition for branching process.

Definition 3 (Branching process). *Let $S = (P, T, F, M_0)$ be a net system. The* branching process *$\mathcal{U}(S) = (B, E, G, \rho)$ of S is the net (B, E, G) defined by the inductive rules in Fig. 3. The rules also define the function $\rho : B \cup E \to P \cup T$ that maps each node in the branching process $\mathcal{U}(S)$ to a node in S. We write $\varrho(B)$ as a shorthand for $\bigcup_{b \in B \cup E} \rho(b)$.*

In a branching process $\mathcal{U}(S) = (B, E, G, \rho)$, B represents the set of conditions (places) and E the set of events (transitions). Let $\beta = \mathcal{U}(S)$ be a branching process, thus $Min(\beta)$ 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 S, i.e., $\varrho(Min(\beta)) = M_0$. A *co-set* is a set of conditions $B' \subseteq B$ such that for all $b, b' \in B'$ it holds $b \parallel b'$. A *cut* is a maximal co-set w.r.t. set inclusion.

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

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

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

Fig. 3. Branching process, inductive rules

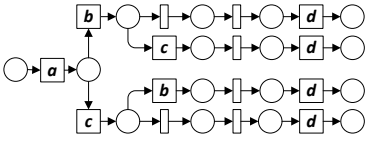


Fig. 4. $\mathcal{U}(N_2)$ (Fig. 1(b))

One characteristic of a branching process is that it does not contain merging conditions. As a result, some nodes in the net system need to be represented more than once in the branching process. For example, the branching process in Fig. 4 contains multiple instances of b, c and d, which come from a single transition in the net system shown in Fig. 2.

Definition 4 (Configuration and deterministic process). *Let* $\beta = (B, E, G, \rho)$ *be a branching process.*

- *A* configuration C *of* β *is a set of events,* $C \subseteq E$*, which is*
  i) causally closed, i.e., $\forall e' \in E, e \in C : e' \leq_\beta e \Rightarrow e' \in C$*, and*
  ii) conflict free, i.e., $\forall e, e' \in C, \neg(e \#_\beta e')$*.*
  We denote by $Conf(\beta)$ *the set of configurations of the branching process* β*, whereas* $MaxConf(\beta)$ *refers to the maximal configurations w.r.t. set inclusion.*
- *A* local configuration *of an event* $e \in E$ *is denoted as* $\lfloor e \rfloor = \{e' \mid e' \leq e\}$*, such that it is unique for any event* $e \in E$*. In the same vein, by* $\lfloor e)$ *we denote the set of strict causes of an event* $e \in E$*, i.e.,* $\lfloor e) = \lfloor e \rfloor \backslash \{e\}$
- *A* deterministic process $\pi = (B_\pi, E_\pi, G_\pi, \rho)$ *is the net induced by a configuration* C*, where* $B_\pi = \bigcup_{c \in C} (\bullet c \cup c\bullet)$*,* $E_\pi = C$*, and* $G_\pi = G \cap (B_\pi \times E_\pi \cup E_\pi \times B_\pi)$*.*

A cut for a configuration C of a branching process $\beta = \mathcal{U}(S)$ is defined as $Cut(C) = (Min(\beta) \cup \bigcup_{c \in C} c\bullet) \backslash (\bigcup_{c \in C} \bullet c)$; whereas $\varrho(Cut(C))$ is a reachable marking in S, denoted by $Mark(C)$, i.e, $Mark(C) \in [M_0\rangle$. Let C and C' be configurations of β, such that $C \subset C'$, and let π and π' be their corresponding deterministic branching processes. If $X = C' \setminus C$, then we write $\pi' = \pi \oplus X$ and we say that π' is an *extension* of π.

Throughout this paper, we use *visible-pomset equivalence* [17] as the notion of behavioral equivalence. A pomset is a tuple $\langle X, \leq |_X \rangle$, where X is a set of events and $\leq |_X$ is the projection of the causal relation over X. We use $X^\Lambda = \{e \in X \mid \lambda(e) \neq \tau\}$ to denote the restriction of X to observable events. With abuse of notation, we write X^Λ to denote the restriction of the pomset induced by X, restricted to observable behavior, and it is called the *visible pomset* underlying X. Moreover, we denote by $Conf(\mathbb{P})^\Lambda$ the set of visible pomsets underlying its configurations, i.e., $Conf(\beta)^\Lambda = \{C^\Lambda : C \in Conf(\beta)\}$.

A function f is an *isomorphism* between pomset p and pomset q, iff it is a label-preserving order-isomorphism, i.e., $f : E_p \to E_q$ is a bijection, $\lambda_p = \lambda_q \circ f$, and $e <_p e' \Leftrightarrow f(e) <_q f(e')$ for all $e, e' \in E_p$. Armed with the concepts above, we can now formally define visible-pomset equivalence:

Definition 5 (Visible-pomset equivalence [17]). *Let* β *and* β' *be the branching processes of the net systems* N *and* N'*. Then* N visible-pomset approximates N'*, written* $N \sqsubseteq_{pt} N'$*, iff every visible-pomset* $X^\Lambda \in Conf(\beta)^\Lambda$ *is isomorphic with at least one visible-pomset* $Y^\Lambda \in Conf(\beta')^\Lambda$*. Moreover,* N *and* N' *are* visible pomset equivalent*, denoted* $\mathbb{N}_1 \equiv_{vp} \mathbb{N}_2$*, iff each is* $\sqsubseteq_{vp}$ *to the other.*

3.3 Event structures

This section introduces two variants of event structures, which are the cornerstones of our comparison technique, *prime* and *asymmetric event structures.*

Definition 6 (Prime Event Structure [3]). *Let* $S = (N, M_0)$ *be a net system, where* $N = (P, T, F, \lambda_N)$*, and* $\beta = (B, E, G, \rho)$ *be its branching process. The*

labeled Prime Event Structure *(PES) of β is defined as* $\mathbb{P} = \langle E, \leq_{\mathbb{P}}, \#_{\mathbb{P}}, \lambda_{\mathbb{P}} \rangle$, *where* $\leq_{\mathbb{P}} = \leq_{\beta} \cap E^2$ *and* $\#_{\mathbb{P}} = \#_{\beta} \cap E^2$. *Finally,* $\lambda_{\mathbb{P}} = \lambda_N \circ \rho$ *is a labeling function that associates each event* $e \in E$ *with the label of its corresponding transition* $t \in T$, *i.e.,* $\rho(e) = t \Rightarrow \lambda_{\xi}(e) = \lambda_N(t)$.

The conflict relation $\#_{\mathbb{P}}$ *is* hereditary *w.r.t.* $\leq_{\mathbb{P}}$, *i.e.* $e \#_{\mathbb{P}} e' \wedge e' \leq_{\mathbb{P}} e'' \Rightarrow e \#_{\mathbb{P}} e''$ *for all* $e, e', e'' \in E$.

As stated before, we focus only on observable behavior. Therefore, we use $\bar{\mathbb{P}}$ to denote a PES with observable events, that is, with all its invisible events being abstracted away. Figure 5 shows the PES with all the observable behavior of the net system N_2 from Fig. 2. In this graphical representation, solid arrows represent causality, and annotated dotted lines represent conflict. It is common practice not to include transitive relations in the graphical representation of a PES, for the sake of readability.

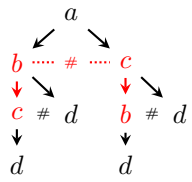


Fig. 5. PES $\bar{\mathbb{P}}$

The set of configurations of a PES $\mathbb{P}$ coincides with the set of configurations of its originative branching process. We will denote this set as $Conf(\mathbb{P})$.

We now turn our attention to *Asymmetric Event Structures* (AESs).

Definition 7 (Asymmetric Event Structure [4]). *An AES is a triplet* $\mathbb{A} = \langle E, \leq, \nearrow \rangle$, *where* E *represents the set of events,* $\leq$ *is the causality relation and* $\nearrow$ *is the asymmetric conflict relation. Moreover, for all* $e, e', e'' \in E$ *the following holds: (1)* $\lfloor e \rfloor = \{e' \mid e' \leq e\}$ *is finite, (2)* $e < e' \Rightarrow e \nearrow e'$, *(3) if* $e \nearrow e'$ *and* $e' < e''$ *then* $e \nearrow e''$, *(4)* $\nearrow |_{\lfloor e \rfloor}$ *is acyclic, (5) if* $\nearrow |_{\lfloor e \rfloor \cup \lfloor e' \rfloor}$ *is cyclic then* $e \nearrow e'$.
Finally, let $\Psi_{\mathbb{A}} = (<, \nearrow)$ *denote the behavior relations of* $\mathbb{A}$.

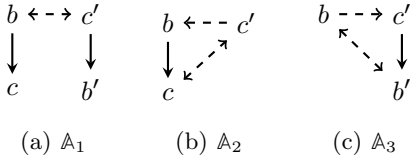
This type of event structure replaces the conflict relation in PESs with an asymmetric relation. Graphically, causality is represented by a solid arrow and asymmetric conflict with a dashed arrow. Intuitively, the statement $a \nearrow b$ has two interpretations: (i) the occurrence of b *prevents* the occurrence of a, or (ii) a *precedes* b in all computations where both events occur. By (ii), asymmetric conflict can be seen as a form of weak causality. Interestingly, asymmetric conflict is also hereditary w.r.t. causality. As for PESs, two events are said *concurrent* when they are neither in causal nor in asymmetric conflict relation.

In the case of PESs, the set inclusion relation defines a order over configurations referred to as *configuration extension*. This does not apply to the case of AESs. Consider the AES presented in Fig. 6. Note that $\{a, b, c\}$ is an extension of $\{a, b\}$, but it is not an extension of $\{a, c\}$, because the occurrence of c prevents that of b. Formally, a configuration of $\mathbb{A} = \langle E, \leq, \nearrow, \lambda \rangle$ is a set of events $C \subseteq E$ such that 1) for any $e \in C$, $\lfloor e \rfloor \subseteq C$ (causal closedness) 2) $\nearrow |_C$ is acyclic (conflict free). Moreover, if $C_1, C_2 \in Conf(\mathbb{A})$ are cofigurations, we say that C_2 extends C_1, written $C_1 \sqsubseteq C_2$, if

Fig. 6. $\mathbb{A}_0$

$C_1 \subseteq C_2$ and for all $e \in C_1,\ e' \in C_2 \setminus C_1, \neg(e' \nearrow e)$. The set of all configurations of $\mathbb{A}$ is denoted by $Conf(\mathbb{A})$.

The AES formalism is more expressive than PESs and it can provide a more compact representation for a given set of configurations. In fact, any PES can be seen as a special case of an AES [4] where the conflict relation is replaced with asymmetric conflict relations in both directions. Consider the AES shown in Fig. 7. $\mathbb{A}_1$ can be seen as the direct translation of a PES, hence requiring duplication. $\mathbb{A}_2$ and $\mathbb{A}_3$ are smaller, but still visible-pomset equivalent, versions of $\mathbb{A}_1$. Observe that there is no smaller AES representation for the same behavior and, in that sense, both $\mathbb{A}_2$ and $\mathbb{A}_3$ are minimal.



(a) $\mathbb{A}_1$ (b) $\mathbb{A}_2$ (c) $\mathbb{A}_3$

Fig. 7. Equivalent AESs

In [5], we introduced a technique for behavior-preserving minimization of AESs. Moreover, we found that the technique may lead to different representations for the same behavior, depending on the order on which the folding operation is applied on the input AES. For instance, $\mathbb{A}_2$ (Fig. 7(b)) comes after folding events b and b', whereas $\mathbb{A}_3$ (Fig. 7(c)) comes after folding events c and c'. In the next section, we will address the problem of canonical folding of AESs.

4 Comparison of process models

This section describes our approach to process model differencing with Asymmetric Event Structures. The first part addresses the problem of the non-canonicity of the folding of an AES by leveraging the notion of canonical labelling of graphs. The second part extends the method to support the comparison of process models with cycles. Finally, the section presents a differencing operator and an approach to verbalizing the differences found while comparing pairs of processes. The proofs are available at [18].

4.1 Canonicity

Any reliable comparison method requires that its input is provided in a canonical representation. In our context, if we consider that the folding operation is behavior preserving, we would like that the AESs obtained from two isomorphic PESs are also isomorphic. As shown in Fig. 7, this not always the case. In order to address this problem, we leverage some concepts from graph theory.

Our solution to the problem of non-canonicity relies on the concept of canonical labelling of a graph [19], that is an approach to deciding graph isomorphism. We say that $Canon(G)$ is a function that maps a graph G to a canonical label. In this way, if graphs G and H are given, we expect $Canon(G) = Canon(H)$ to

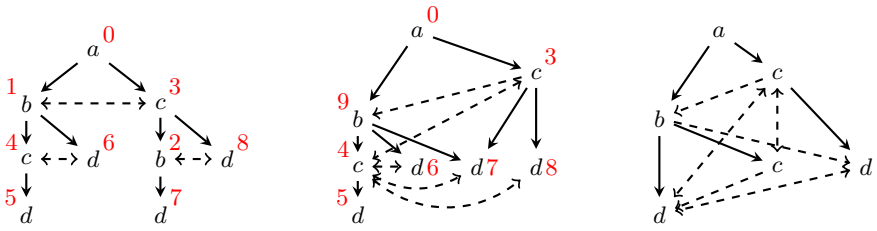
hold iff H and G are isomorphic. If we use the string representation of the adjacency matrix of a graph, then a canonical label for a graph G can be determined by computing all permutations of its adjacency matrix and selecting the largest lexicographical exemplar among them[5]. Clearly, this naïve approach is computationally expensive, but state-of-the-art software implement several heuristics to compute canonical labels in a reasonable time. In our context, we are interested in the order of the vertices associated to the adjacency matrix of the canonical exemplar. Formally, let $G = (V, A)$ be a graph, where V is the set of vertices and A the set of arcs. Moreover, let $M(G)$ be the adjacency matrix of G, $\gamma = (0, 1, ...|V|)$ be an order over the set of vertices, and $STR(M(G)^{\gamma})$ be the string linear representation of the adjacency matrix G given the order γ. Then the canonical label of G is the string induced by order $\hat{\gamma}$, s.t., $STR(M(G)^{\hat{\gamma}}) \geq_{lex} STR(M(G)^{\gamma_\pi})$ holds for every possible permutation γ_π of γ.

In our implementation, we use **nauty** (`http://pallini.di.uniroma1.it/`) for computing the graph canonical label and, more precisely, the order on the vertices of the canonical exemplar. Nauty and other similar tools work on graphs with unlabeled edges. To overcome this limitation, we adapted a transformation introduced in [20]. Briefly, this transformation maps a fully labeled graph (both nodes and edges carry a label) into a node-labeled graph and has been proved to be isomorphism preserving. The reader is referred to [20] to get more details about this transformation. By leveraging this result and the notion of canonical label of a graph we can now establish an order on the folding that yields a minimal and canonical AES for a PES.

Intuitively, the folding starts with an AES that is isomorphic to the PES of a business process. Thus, we carry the order $\hat{\gamma}$ computed over the nauty graph of the PES. In every iteration we select a set of events that can be merged without changing the behavior of the AES. We use $\mathbb{A}_{/X}$ to denote the folding of a combinable set of events X on an AES $\mathbb{A}$. In [5] we show that the folding defines a morphism $f : AES \to AES$ that preserves visible-pomset equivalence. In this context, since there might be multiple candidate sets of events for folding, we use $\hat{\gamma}$ for establishing a total order on the folding operations. For space restriction, we do not include the details on how the whole set of combinable sets of events is computed and refer the reader to [5] for a full description. Therefore, we will assume that the combinable sets of events are given.

Definition 8 (Deterministic folding). *Let $\mathbb{A} = \langle E, \leq, \nearrow, \lambda \rangle$ be an AES, and $\hat{\gamma} : E \to \mathbb{N}_0$ be the canonical order of events given by nauty. Let $X, Y \subseteq E$ be combinable sets of events. Then the precedence of X over Y in a deterministic folding is defined by the following conditions, listed in decreasing relevance: (i) $\lambda(e) >_{lex} \lambda(e')$ where $e' \in Y$ and $e \in X$, or (ii) $\lambda(e) =_{lex} \lambda(e') \wedge |X| > |Y|$, or (iii) $\lambda(e) =_{lex} \lambda(e') \wedge |X| = |Y| \wedge \hat{\gamma}(X) >_{lex} \hat{\gamma}(Y)$. Hence, $\mathbb{A}_{/X} = \langle E_{/X}, \leq_{/X}, \nearrow_{/X}, \lambda_{/X} \rangle$ is a folding of $\mathbb{A}$, s.t. $e_X \in E_{/X}$ is the event representing $X \subseteq E$, and the canonical labeling function is $\hat{\gamma}_{\mathbb{A}_{/X}} = \hat{\gamma}[e_X \mapsto Ran(\hat{\gamma}) + 1]$. Finally, $f(\mathbb{A})^{+}_{/X}$ denotes the folding induced by $\hat{\gamma}$ that cannot be further minimized, i.e., the* minimal canonical folding.

[5] Some authors prefer the smallest lexicographic string.



(a) $\mathbb{A}_4$ (b) $f(\mathbb{A}_4)_{/\{b,b\}}$ (c) $f(\mathbb{A}_4)^+_{/X}$

Fig. 8. Canonical labeling and folding

Figure 8 illustrates the canonical folding of $\mathbb{A}_4$, which corresponds to the PES $\bar{\mathbb{P}}$ in Fig. 5. $\mathbb{A}_4$ shows the order $\hat{\gamma}$ assigned by nauty. The combinable sets of events in $\mathbb{A}_4$ are $\{\{b(1), b(2)\}, \{c(3), c(4)\}, \{d(5), d(6)\}, \{d(7),\ d(8)\}\}$, and from Definition 8 we know that $\{b(1), b(2)\}$ takes precedence over the others. The folding of $\{b(1), b(2)\}$ is depicted in Fig. 8(b). Note that a fresh event b is added, replacing the set $\{b(1), b(2)\}$, and the value 9 is associated to this event in $\hat{\gamma}$. The values added to $\hat{\gamma}$ are monotonically increased. Finally, Fig. 8(c) depicts the minimal and canonical AES. In this particular case, it was necessary to keep two events with label c and two with label d to preserve the behavior. The following proposition shows that the folding of an AES is canonical.

Proposition 1. *Let $\mathbb{A}_1 = \langle E_1, \leq_1, \nearrow_1, \lambda_1 \rangle$ and $\mathbb{A}_2 = \langle E_2, \leq_2, \nearrow_2, \lambda_2 \rangle$ be two isomorphic AESs and, $\hat{\gamma}_1 : E_1 \to \mathbb{N}_0$ and $\hat{\gamma}_2 : E_2 \to \mathbb{N}_0$ be the canonical order for E_1 and E_2, correspondingly. Then the deterministic folding of $\mathbb{A}_1$ and $\mathbb{A}_2$ produces a canonical AES, such that $f(\mathbb{A}_1)^+_X$ is isomorphic to $f(\mathbb{A}_2)^+_X$.*

4.2 Finite representation of cyclic behavior

A fundamental problem with cyclic process models is that their branching processes may easily get unboundedly large. Engelfriet [16] showed that every Petri net has a unique maximal branching process up to isomorphism, the so-called unfolding of the net. McMillan [21] and then Esparza et al. [22] introduced sophisticated strategies for truncating the unfolding to a finite level, thus getting what is referred to as the *complete unfolding prefix (CP)*. Later, the authors in [23] introduced a framework to generalize previous work and to defined the notion of canonical unfolding prefixes. Our own work relies on such a framework. In the following we restrict ourselves to Petri nets without duplicate tasks.

Consider the net system N_1 and the complete unfolding prefix β_1 presented in Fig. 9. Note that both b_1 and b_4 correspond to the place p_1 in N_1. To compute the complete unfolding prefix, we start applying the inductive rules described in Fig. 3. In this case, however, it is possible to stop unfolding once we reach b_2 and b_4 because any addition to the prefix would duplicate information already represented there. For this reason, events b and c are called *cutoff events*. Although it has been proved that the complete unfolding prefix represents all the

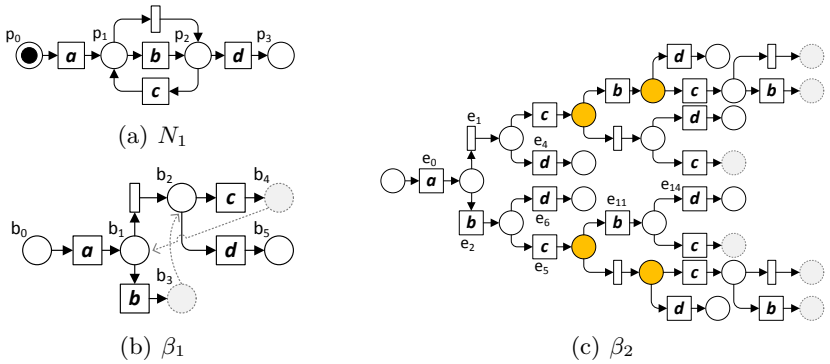


Fig. 9. Petri net and two different unfoldings

behavior of the original net system [22], this prefix does not explicitly contain the information that we require to diagnose the behavioral differences of business processes. For instance, the fact that c causally precedes b and d is not explicitly represented in the prefix. Therefore, we require a larger prefix of the branching process that makes explicit all the causal relations. In the case of the net system N_1 in Fig. 9(a) the required unfolding prefix is β_2, (Fig. 9(c)).

In order to compute a unfolding prefix as the one required for comparison of process models, we define new criteria to identify cutoff events. To this end, we use the notion cutting context introduced in [23]. The cutting context is formally defined as the tuple $\Theta = (\approx, \lhd, \mathcal{C})$ where $\approx$ is an equivalence relation over configurations, $\lhd$ is a total order over configurations, and $\mathcal{C}$ is the set of configurations used at the time of the computation of the unfolding prefix. E.g., the cutting context used in McMillan [21] is $\Theta_{McMillan} = (\approx_{mark}, \lhd_{size}, \mathcal{C}_{loc})$, where $\approx_{mark}$ equates two configurations when they produce the same marking, $\lhd_{size}$ is the total order induced by the size of configurations, and $\mathcal{C}_{loc} = \{\lfloor e \rfloor \mid e \in E\}$ is the set of local configurations. Note that, the complete unfolding prefix β_1 can also be computed by using McMillan's cutting context. In fact, if we consider the local configurations $\lfloor c \rfloor = \{a, \tau, c\}$ and $\lfloor a \rfloor = \{a\}$, then one can easily check that $Mark(\lfloor a \rfloor) = Mark(\lfloor c \rfloor) = \{p_1\}$. Moreover, since $|\lfloor a \rfloor| < |\lfloor c \rfloor|$, then one should conclude that event c is a cutoff event. The cutting context in Esparza et al. [22], denoted $\Theta_{ERV} = (\approx_{mark}, \lhd_{slf}, \mathcal{C}_{loc})$, differs from that in [21] only for the definition of the partial order $\lhd_{slf}$, which is refined by considering action labels thus leading to more cut-offs and smaller prefixes (see [22] for details). For our purposes, consider a cutting context which is a modification of Θ_{ERV} with a refined equivalence relation over configurations.

Definition 9 ($\approx_{Pred}$). *Let $\beta = (B, E, G, \rho)$ be a branching process. A pair of configurations $C_1, C_2 \in Conf(\beta)$ are equivalent, represented as $C_1 \approx_{Pred} C_2$, iff $eMark(C_1) = eMark(C_2)$, where*

- $eCut(C) = \{\langle b, \lfloor \bullet b \rfloor \rangle \mid b \in Cut(C)\}$, *and*
- $eMark(C) = \{\langle \rho(b), \rho(\lfloor \bullet b \rfloor) \rangle \mid \langle b, \lfloor \bullet b \rfloor \rangle \in eCut(C)\}$.

We define our cutting context as $\Theta_{Pred} = (\approx_{Pred}, \lhd_{slf}, \mathcal{C}_{loc})$. Khomenko et al. [23] also offers a framework for showing that the unfolding prefix generated by a cutting context ensures canonicity, finiteness and completeness. To this end, we need to prove that the equivalence $\approx_{Pred}$ and the adequate order $\lhd_{slf}$ are preserved by finite configuration extensions. Esparza et al [22] showed that this property holds for $\lhd_{slf}$. The following proposition shows that the property also holds for $\approx_{Pred}$.

Proposition 2. *Let $\beta = (B, E, G, \rho)$ be the branching process of a net system $S = (N, M_0)$ and $C, C' \in Conf(\beta)$ be a pair of configurations, s.t. that $C \approx_{Pred} C'$. Therefore, for every suffix V of C, there exists a finite suffix V' of C' s.t.:*

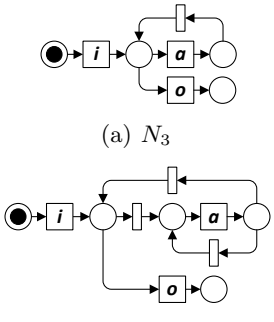
$$C' \oplus V' \approx_{Pred} C \oplus V$$

The following proposition shows that the canonical unfolding prefix constructed with Θ_{Pred} contains all the causal relations that would be exhibited in the (possibly infinite) unfolding of a business process with cycles.

Proposition 3 (Completeness of transitive causal relation). *Let $\beta = (B, E, G, \rho)$ be the full branching process of a net system $S = (N, M_0)$, $\Theta_{Pred} = (\approx_{Pred}, \lhd_{slf}, \mathcal{C}_{loc})$ be the cutting context and $\beta_\Theta = (B', E', G', \rho_\Theta)$ be the CP unfolding constructed by Θ_{Pred}. Finally, let E''_Θ be the set of cut-offs computed by the cutting context. Then, the unfolding prefix β_Θ contains the distinct transitive causal dependencies, such that for any pair of events $e_1, e_2 \in E : e_1 < e_2$ then*

$$\exists e'_1, e'_2 \in E' : e'_1 < e'_2, \text{ where } \rho(e_1) = \rho_\Theta(e'_1) \text{ and } \rho(e_2) = \rho_\Theta(e'_2).$$

Unfortunately, the cutting context Θ_{Pred} does not always produce a prefix that is canonical for business process comparison. For instance, the two net systems presented in Figure 10 are visible-pomset equivalent. However, the presence of silent transitions leads to unfolding prefixes with larger duplication in case of N_4. In the current form, the behavior-preserving folding technique that we rely on will not merge causally related events, e.g. as in the unfolding of N_4, because it prevents cycles in the AES. The problem of computing a canonical folding of such cycles is left as future work.



(a) N_3

(b) N_4

Fig. 10. Sample net systems

Repetitions. We now show how to identify the repetitive behavior, given the canonical unfolding prefix induced by Θ_{Pred}. Intuitively, we can say that a transition t in a net system is part of repetitive behavior iff there exists at least one configuration on which two events associated to transition t occur in causal relation. This intuition is captured in the following definition:

Definition 10 (Repetitive behavior). *Let $\beta = (B, E, G, \rho)$ be the unfolding prefix induced by Θ_{Pred} for a net $N = (P, T, F, \lambda)$. The* repetitive behavior *of N is defined as $\mathcal{R} = \{\rho(e_1) \mid \exists C \in Conf(\beta).\ e_1, e_2 \in C\ \wedge\ \rho(e_1) = \rho(e_2) \wedge e_1 < e_2\}$.*

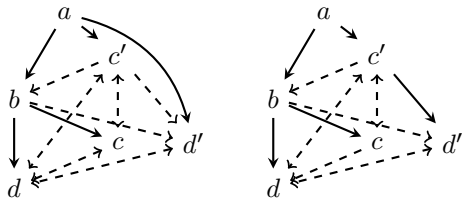
It can be easily checked that the sets $\{e_0, e_1, e_4\}$, $\{e_0, e_2, e_6\}$ and $\{e_0, e_2, e_5, e_{11}, e_{14}\}$ are configurations in the unfolding prefix β_2 from Fig. 9(c). From the discussion above, we can conclude that b is part of repetitive behavior, in spite of the fact that there is a configuration that contains a single event carrying the label b. Moreover, we can also see that there exist at least one configuration on which no event labeled b occurs. This means that the task b will be observed zero or more times. In general, tasks participating in repetitive behavior can be observed either "0 or more times" (denoted as "$*$") or "1 or more times" (denoted as "+"). We will use the marker "0" for tasks that do not participate in repetitive behavior. The following definition captures the intuition above.

Definition 11 (Partitions of repetitive behavior). *Let $\beta = (B, E, G, \rho)$ be the unfolding prefix induced by Θ_{Pred} for net system $S = (N, M_0)$. The* constant behavior $\mathcal{K}$ *is defined as $\mathcal{K} = \varrho(\bigcap MaxConf(\beta))$. Therefore, the partitions of repetitive behavior are defined as:*

- $0 = \{e \mid e \in E\ \wedge\ \rho(e) \notin \mathcal{R}\}$
- $+ = \{e \mid e \in E\ \wedge\ \rho(e) \in \mathcal{R} \cap \mathcal{K}\}$
- $* = \{e \mid e \in E\ \wedge\ \rho(e) \in \mathcal{R}\ \wedge\ e \notin +\}$

4.3 Comparison

The comparison of process models happens on a subset of events, with the remaining events being discarded. We keep all the events which carry labels that are present in both process models. Moreover, since a single task may have multiple events associated, we compute an optimal matching, with well-known methods [1], on the set of events from both process models and discard the subset of events that does not make part of the matching. As discussed before, if two AESs are isomorphic then they must be diagnosed as behaviorally equivalent. In this work we adopt *visible pomset equivalence* [17][6]. It



(a) $\mathbb{A}_5$ (b) $\mathbb{A}_6$

Fig. 11. Foldings and optimal matching (hinted by the position) of the AESs of to processes in the running example

[6] Due to the presence of silent transitions, we use a weaker notion of equivalence than the one adopted in [5].

is only when two AESs are not isomorphic that we have to diagnose the differences. Once the optimal matching is computed, the comparison of business processes happens in three stages: (1) diagnosis over the set of events in the optimal matching, (2) diagnosis on repetitive behavior, and (3) diagnosis on the set of unmatched events.

The diagnostic of the differences of behavior can be represented in a square matrix of order n, where n is the number of events in the optimal matching. To this end, we define the following differencing operator.

Definition 12 (Symmetric difference of AES behavior relations). *Let* $\mathbb{A}_1 = (E_1, \leq_1, \nearrow_1, \lambda_1)$ *and* $\mathbb{A}_2 = (E_2, \leq_2, \nearrow_2, \lambda_2)$ *be labeled event structures, and let* $\Psi_{\mathbb{A}_1}$ *and* $\Psi_{\mathbb{A}_2}$ *be their corresponding behavior relations. Let* $\mathcal{I} : E_1' \to E_2'$ *is the mapping function from* $\mathbb{A}_1$ *to* $\mathbb{A}_2$ *given by the graph matching algorithm, such that* $E_1' \subseteq E_1$ *and* $E_2' \subseteq E_2$.

Let $(e_1, e_2), (e_1', e_2') \in \mathcal{I}$ *be event matchings. The* symmetric difference *of* $\Psi_{\mathbb{A}_1}$ *and* $\Psi_{\mathbb{A}_2}$*, denoted* $\Psi_{\mathbb{A}_1} \vartriangle \Psi_{\mathbb{A}_2}$*, is defined as follows:*
$\Psi_{\mathbb{A}_1} \vartriangle \Psi_{\mathbb{A}_2}[(e_1, e_2), (e_1', e_2')] =$

$$\begin{cases} \cdot & \text{if } \Psi_{\mathbb{A}_1}[e_1, e_1'] = \Psi_{\mathbb{A}_2}[e_2, e_2'] \\ (\Psi_{\mathbb{A}_1}[e_1, e_1'], \Psi_{\mathbb{A}_2}[e_2, e_2']) & \text{if } \Psi_{\mathbb{A}_1}[e_1, e_1'] \neq \Psi_{\mathbb{A}_2}[e_2, e_2'] \end{cases}$$

Figure 11 shows the AESs of the sample process models in Fig. 1, projected to the subset of events (and behavior relations) in the optimal matching. Figure 12, in turn, shows the symmetric differencing of $\mathbb{A}_5$ and $\mathbb{A}_6$. Since cycles of asymmetric conflict hint a sort of symmetric conflict, in the matrix we prefer to use symmetric conflict to highlight this subtle difference, e.g., $\Psi_{\mathbb{A}_5} \vartriangle \Psi_{\mathbb{A}_5}[(c, d), (c, d)] = (\#, \nearrow)$.

	a	**b**	**c**	**c′**	**d**	**d′**
a	.	.	.	.	.	.
b	.	$(*, 0)$	.	.	.	.
c	.	.	$(*, 0)$	.	$(\#, \nearrow)$	.
c′	.	.	.	$(*, 0)$	.	$(\nearrow, <)$
d	.	.	.	.	.	.
d′	.	.	.	.	.	.

Fig. 12. $\Psi_{\mathbb{A}_5} \vartriangle \Psi_{\mathbb{A}_6}$

Given the intuitive interpretation of the behavior relations represented in an AES, it is possible to use the following statements to describe the eventual differences in behavior:

- Causality: *"task* **a** *occurs before task* **b***"*.
- Asymmetric conflict: *"task* **a** *can occur before task* **b***, or* **a** *can be skipped"*.
- Conflict: *"task* **a** *and task* **b** *are mutually exclusive"*.
- Concurrency: *"task* **a** *and task* **b** *occur in parallel"*.

Similarly, for repetitive activities, we say:

- 0: *"it is not repeated any time"*,
- +: *"activitiy a can occur 1 or more times"*, and
- $*$: *"activity a can occur 0 or more times"*.

It is often the case that the feedback requires further information to understand the context on which a behavior difference arises. One possibility would

be to present the set of runs on which a particular event occurs. However, the amount of information might be overwhelming. Therefore, we include in the feedback only the set of events that are in direct causal relation with the event giving rise to the difference. Based on the above considerations, the following are examples of verbalization for differences encountered from the matrix in Fig. 12:

- $\mathbf{c}, \mathbf{d} = (\#, \nearrow)$: *In model 1, there is a state after the execution of* **c** *where* **d** *and* **c** *are mutually exclusive; whereas in model 2, there is a state after the execution of* **b** *where* **c** *can occur before* **d**, *or* **c** *can be skipped*
- $\mathbf{c}', \mathbf{d}' = (\nearrow, <)$: *In model 1, there is a state after the execution of* **a** *where* **c** *can occur before* **d**, *or* **c** *can be skipped; whereas in model 2, there is a state after the execution of* **a** *where* **c** *precedes* **d**
- $\mathbf{b}(*, 0)$: *Task* **b** *may occur many times in model 1; whereas in model 2, it is not repeated any time*
- $\mathbf{c}(*, 0)$: *Task* **c** *may occur many times in model 1; whereas in model 2, it is not repeated any time*

In the case of tasks with repetitive behavior, one event is randomly chosen and the feedback is generated with respect to this event (note that the feedback from other instances would be the same). In the last step, we need to produce the feedback for the set of unmatched events. In this case, we also include the set of direct causally preceding events to give a context in the feedback. For the running example, the feedback would be:

- *There is an occurrence of* **b** *after* **c** *in model 1 but not in model 2*
- *There is an occurrence of* **c** *after* **b** *in model 1 but not in model 2*

5 Conclusions and future work

We present a method for comparing business process models based on behavioral relations, specifically those supported by AES. The contributions of the paper are threefold. First, we propose a method to calculate a canonically reduced AES from an acyclic Petri net. Second, we propose a technique to compute a finite representation for repetitive behavior that preserves casual dependencies, although the latter representation is not canonical as in the acyclic case. Finally, we propose a verbalization technique to generate difference diagnostics between process models. The presented techniques are implemented in a prototype tool available at `https://code.google.com/p/fdes/`. This tool takes pairs of process models captured in BPMN notation as input and produces a textual diagnostic of their differences.

As avenues for future research, we want fine tune the techniques to improve scalability of the tool. We also foresee an empirical study to assess the usability of the diagnostics produced by our tool. Finally, we aim at investigating further the problem of comparison of process models with cycles.

References

1. 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

2. Rosa, M.L., Clemens, S., ter Hofstede, A.H.M., Russell, N.: Appendix A. The Order Fulfillment Process Model. In: Modern Business Process Automation 2010
3. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. Theoretical Computer Science **13** (1981) 85–108
4. Baldan, P., Corradini, A., Montanari, U.: Contextual Petri Nets, Asymmetric Event Structures, and Processes. Information and Computation 2001 **171** 1–49
5. Armas, A., Baldan, P., García-Bañuelos, L.: Reduction of event structures under hp-bisimulation. Technical report `http://arxiv.org/abs/1403.7181`.
6. Polyvyany, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. Information Systems **37**(6) (2012) 518–538
7. van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica **37** (2001) 229–327
8. Cleaveland, R.: On automatically explaining bisimulation inequivalence. In: CAV. LNCS 531. Springer (1991) 364–372
9. Sokolsky, O., Kannan, S., Lee, I.: Simulation-Based Graph Similarity. In: TACAS. LNCS 3920. Springer (2006) 426–440
10. Dijkman, R.: Diagnosing Differences between Business Process Models. In: BPM. Volume 5240 of LNCS 5240. Springer (2008) 261–277
11. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. IEEE TSE **37**(3) (2011) 410–429
12. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal Behavioural Profiles. Fundamenta Informaticae **113**(3-4) (2011) 399–435
13. 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
14. Badouel, E.: On the α-Reconstructibility of Workflow Nets. In Haddad, S., Pomello, L., eds.: ATPN. LNCS 7347. Springer (2012)
15. Weidlich, M., van der Werf, J.: On Profiles and Footprints-Relational Semantics for Petri Nets. In: ATPN. LNCS 7347. Springer (2012) 148–167
16. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica **28** (1991) 575–591
17. van Glabbeek, R., Goltz, U.: Equivalence notions for concurrent systems and refinement of actions. Volume 379 of LNCS. Springer (1989) 237–248
18. Armas, A., Baldan, P., Dumas, M., García-Bañuelos, L.: Behavioral comparison of process models based on canonically reduced event structures. Technical report It is available at `http://math.ut.ee/~abela`.
19. McKay, B.D.: Practical graph isomorphism. Department of Computer Science, Vanderbilt University (1981)
20. Kant, G.: Using canonical forms for isomorphism reduction in graph-based model checking. Technical report, CTIT University of Twente, Enschede (July 2010)
21. McMillan, K.L., Probst, D.K.: A Technique of State Space Search Based on Unfolding. Formal Methods in System Design **6**(1) (1995) 45–65
22. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. Formal Methods in System Design **30**(2) (2002) 285–310
23. Khomenko, V., Koutny, M., Vogler, W.: Canonical prefixes of Petri net unfoldings. Acta Informatica **40**(2) (2003) 95–118