A Foundational Approach for Managing Process Variability

Matthias Weidlich[1], Jan Mendling[2], and Mathias Weske[1]

[1] Hasso Plattner Institute at the University of Potsdam, Germany
{Matthias.Weidlich,Mathias.Weske}@hpi.uni-potsdam.de
[2] Humboldt-Universität zu Berlin, Germany
Jan.Mendling@wiwi.hu-berlin.de

Abstract. A business process often shows different variations in a large organisation, due to different legal requirements in different countries, deviations in the IT infrastructure, or organisational differences. These variants are documented in separate independent process models. Management of these variants imposes various challenges. Invariant behaviour needs to be identified and redundancies among the variants have to be avoided. In this paper, we address these questions by defining a set-algebra for behavioural profiles. These profiles represent a behavioural abstraction of process models that can be computed efficiently. We trace back many questions of process variability management to set-theoretic operations and relations defined for behavioural profiles. As a validation, we apply our approach to an industry model collection.

1 Introduction

In large organisations, a business process often exists in many variations. Those stem from different legal requirements in different countries, deviations in the IT infrastructure, or organisational differences [1]. The existence of *process variants* is inevitable – a certain degree of variability is needed to meet the concerns of a specific organisational unit. Variants are often documented in independent process models. As the variability is not made explicit, synergies between variants are hard to explore and process harmonisation is impeded.

Recently, approaches to control the creation of process variants based on well-defined change patterns [2,3,4] or configurations [5,6] have been presented. However, enforcement of such a controlled variability of processes is hard to achieve once process variants are created independently in different organisational units. Therefore, we focus on the use case of managing decoupled process variants. To this end, identification of commonalities and differences between process variants is of central importance. Those have to be made explicit to allow for a comparison of the variants. This is a prerequisite for any process harmonisation effort that aims at reducing the amount of allowed process variability.

There are two fundamental challenges towards efficient management of process variants: the appropriate specification of formal operations to support reasoning with process variants, and the foundation of such operations upon an appropriate

behavioural abstraction of processes. Throughout this paper, we will demonstrate that the essential variant management questions can be answered based on set algebraic operations of complementation, intersection, and union, and the relations of set equivalence and inclusion. Although there are several works on particular subsets of these aspects such as inclusion (or behaviour inheritance) [7,8,9] and union (or merging behaviour) [10,11,12,13], we currently miss an overarching set algebra which provides the means to efficiently calculate with behaviour.

There is a wide spectrum of behavioural abstraction available upon which algebraic operations could be defined. Existing work on behaviour inheritance [8,9,14] builds on an adapted notion of branching bisimilarity. That is, activities that are without counterpart can either be *blocked* or *hidden* when assessing behavioural equivalence. Such an approach has the drawback that the underlying notion of branching bisimilarity is computationally hard as it is based on state space analysis, cf., [15]. Therefore, we base our set algebra on behavioural profiles, a behavioural abstraction of process models that can be computed efficiently for a broad class of models. Behavioural profiles capture the essential behaviour of the set of traces of a process in terms of order constraints, and they are insensitive to skipping of an activity. Differences related to causal constraints are often observed among process variants, and behavioural profiles have been found to provide a suitable abstraction to reason about consistency of process variants [16].

Our contribution is a formal definition of a set algebra for calculating with behavioural profiles of process variants. Our approach is inspired by work on a set algebra for service interaction [17]. The concepts are applied to an industry model collection, for which we identify variant clusters and analyse behavioural commonalities. In this way, our work informs formal research on behavioural inheritance as much as engineering approaches towards merging of behaviour.

The remainder of this paper is structured as follows. The next section illustrate the use case in more detail and presents formal preliminaries. Section 3 defines a set algebra for behavioural profiles, which is applied in Section 4 to address various questions on managing process variability. We present findings from a case study with industry models in Section 5. Finally, Section 6 discusses our approach in the light of related work, before Section 7 concludes the paper.

2 Background

This section introduces the background of our work. First, Section 2.1 elaborates on the use case of managing decoupled process variants. Subsequently, Section 2.2 presents our formal model and Section 2.3 defines behavioural profiles.

2.1 Challenges for Managing Decoupled Process Variants

We illustrate our use case using the two process models depicted in Fig. 1. Both models capture the process of registering a newborn and are adapted versions of the models obtained in different Dutch municipalities, cf., [18]. The administrative processes of Dutch municipalities are rather similar due to legal regulations and

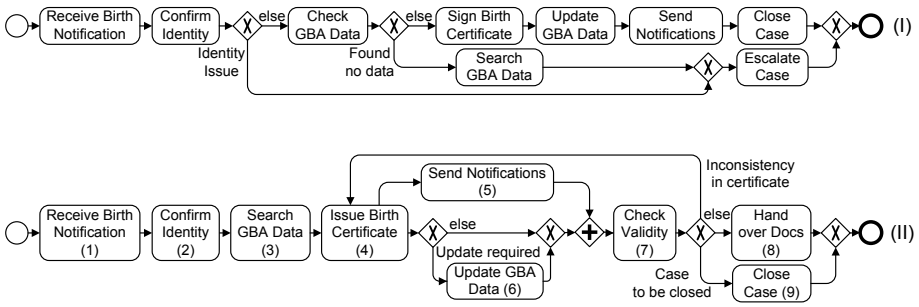


Fig. 1. Variants of the process of registering a newborn

a non-normative reference model. Nevertheless, these processes show a certain degree of variation caused by, for instance, the size of the municipality or the used information systems, and evolve independently from each other.

Identification of best practices or reducing of the number of process variants among the municipalities can be seen as driven factors for process harmonisation. This requires the identification of semantically corresponding activities of different process models based on linguistic or structural analysis [19,20,21]. Those techniques may reveal that the activity *'Sign Birth Certificate'* in model (I) corresponds to the activity *'Issue Birth Certificate'* in model (II). For this paper, we assume correspondences between activities to be given.

Once we identify correspondences between activities of various pairs of models, we can define clusters of related models that capture variants of the same process. The following challenges must be addressed to support process harmonisation efforts for a particular cluster of process variants.

C1: Given a set of variants, what are their commonalities in terms of shared behaviour? This question relates to the challenge of identifying the behaviour that is invariably agreed on by all variants, i.e., the implemented invariant behaviour.

C2: Given a set of variants, what is the most general allowed behaviour? The challenge when answering this question is to integrate the behaviour defined by all process variants, such that the most general behaviour becomes visible. This is required to come to a notion of a configurable model known from [5,6], which subsumes the complete behaviour defined in single variants.

C3: Given a set of variants, what are their commonalities in terms of shared forbidden behaviour? Similar to the first question, also the shared forbidden behaviour is of interest for managing process variants.

C4: Given a set of variants, which variants are redundant in terms of the specified behaviour? The challenge behind this question is to identify variants for which the specified behaviour is completely subsumed by another process variant. In that case, the existence of the former variant may be challenged.

Although these questions may be approached on a structural level by investigating the set of shared activities, more detailed conclusions can only be drawn once the behavioural perspective is considered. For instance, the activities *'Update GBA Data'* and *'Send Notifications'* show different behavioural constraints in the

models in Fig. 1. To consider such differences, we approach the aforementioned questions based on a set algebra for behavioural profiles. Behavioural profiles are a behavioural abstraction that focusses on order constraints and is insensitive to causal constraints between activities, such as skipping an activity. Fig. 1 illustrates that such differences are often observed among process variants due to additional process entries and exits. Behavioural profiles have been found to provide a suitable abstraction to reason about consistency of process variants [16].

2.2 Formal Model

We use a notion of a process model that is based on a graph containing activity nodes and control nodes. It captures the commonalities of process description languages. For illustration purposes, we use a subset of BPMN and EPCs.

Definition 1 (Process Model).
A *process model* is a tuple $P = (A, s, e, C, F, T)$ where:

- A is a finite non-empty set of activity nodes,
- C is a finite set of control nodes,
- $N = A \cup C$ is a finite set of nodes with $A \cap C = \emptyset$,
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph,
- $\bullet n = \{n' \in N | (n', n) \in F\}$ and $n\bullet = \{n' \in N | (n, n') \in F\}$ denote direct predecessors and successors, we require $\forall\ a \in A : |\bullet a| \leq 1 \wedge |a \bullet| \leq 1$,
- $s \in A$ is the only start node, such that $\bullet s = \emptyset$,
- $e \in A$ is the only end node, such that $e\bullet = \emptyset$,
- $T : C \rightarrow \{and, xor\}$ associates each control node with a type.

The start and end activity nodes are assumed to carry no semantic meaning but indicate initialisation and termination of a process. Refactoring may be applied to arrive at these start and end activity nodes [22]. We assume trace semantics for process models. Execution semantics of a process model is defined by a translation into a Petri net following on common formalisations, cf., [23]. As our notion of a process model comprises a dedicated start and end activity nodes, the resulting Petri net is a workflow net (WF-net) [24]. All control nodes are of type *and* or *xor*, such that the WF-net is free-choice [24]. The translation into WF-nets defines the behaviour of a process model $P = (A, s, e, C, F, T)$, which is captured by a set of *traces* $\mathcal{T}_P$. It comprises a set of lists of the form $s \cdot A^*$, which represent the execution order of activities.

2.3 Behavioural Profiles

A behavioural profile captures behavioural characteristics of a process model by three relations between pairs of activity nodes. These relations are based on the notion of *weak order*. Two activities of a process model are in weak order, if there exists a trace in which one activity occurs after the other. Note that we require only the *existence* of such a trace.

Definition 2 (Weak Order Relation). Let $P = (A, s, e, C, F, T)$ be a process model and $\mathcal{T}_P$ its set of traces. The *weak order relation* $\succ_P \subseteq A \times A$ contains all pairs (x, y), such that there is a trace $\sigma = n_1, \ldots, n_m$ in $\mathcal{T}_P$ with $j \in \{1, \ldots, m-1\}$ and $j < k \leq m$ for which holds $n_j = x$ and $n_k = y$.

Two activity nodes of a process model might be related by weak order in three different ways, which define the characteristic relations of the behavioural profile.

Definition 3 (Behavioural Profile). Let $P = (A, s, e, C, F, T)$ be a process model. A pair $(x, y) \in A \times A$ is in one of the following relations:

- The *strict order relation* $\rightsquigarrow_P$, if $x \succ_P y$ and $y \not\succ_P x$.
- The *exclusiveness relation* $+_P$, if $x \not\succ_P y$ and $y \not\succ_P x$.
- The *interleaving order relation* $||_P$, if $x \succ_P y$ and $y \succ_P x$.

The set $\mathcal{B}_P = \{\rightsquigarrow_P, +_P, ||_P\}$ of all three relations is the *behavioural profile* of P.

We illustrate the relations of the behavioural profile for the lower model in Fig. 1. For instance, it holds (1) $\rightsquigarrow$ (5) as both activities are ordered whenever they occur together in a trace. It holds (8) + (9) as both activities will never occur in a single trace, and (5)||(6) due to the potential concurrent execution. The relations of the behavioural profile are mutually exclusive. Along with the reverse strict order $\rightsquigarrow^{-1} = \{(x, y) \in (A \times A) \mid (y, x) \in \rightsquigarrow\}$, the relations partition the Cartesian product of activities. An activity is either said to be *exclusive to itself* (e.g., (1) + (1) in Fig. 1) or in *interleaving order to itself* (e.g., (6)||(6)). The former holds, when an activity cannot be repeated, whereas the latter implies that the activity may be executed multiple times. The behavioural profile is a behavioural abstraction. Hence, there may be process models that show different trace semantics but have equal behavioural profiles. Moreover, behavioural profiles may be lifted from activities to labels of activities. Still, this may result in serious information loss as two occurrences of activities with equal labels would not be distinguished. The behavioural profile $\mathcal{B}_P$ of a process model $P = (A, s, e, C, F, T)$ may be restricted to a subset of activity nodes $A' \subset A$ by restricting the definition of the three relations $\rightsquigarrow_P$,$+_P$, and $||_P$ to $A' \times A'$. We refer to this restricted profile as the behavioural profile over A'.

Computation of the behavioural profile of a process model is done efficiently under the assumption of soundness. Soundness is a correctness criteria that guarantees the absence of behavioural anomalies, such as deadlocks [25]. It has been defined for WF-nets, so that it can be directly applied to our notion of a process model. We are able to reuse techniques for the computation of behavioural profiles introduced for sound free-choice WF-nets [16]. Using these results, behavioural profiles are computed in $O(n^3)$ time with n as the number of nodes of the respective WF-net.

3 A Set Algebra for Behavioural Profiles

To introduce a set algebra for behavioural profiles, we first discuss a notion of strictness for profile relations in Section 3.1. Then, Section 3.2 introduces set-theoretic relations, while Section 3.3 turns the focus on set-theoretic operations.

3.1 Strictness of Behavioural Relations

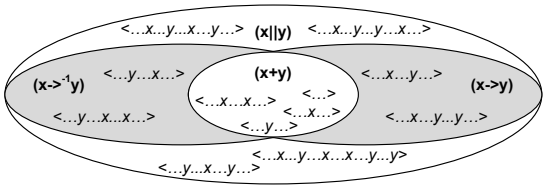


Fig. 2. Sets of traces induced by the relations of the behavioural profile for two activities x and y

The relations of the behavioural profile can be classified according to their *strictness*, as they allow different levels of freedom for the occurrences of activities in a trace. Interleaving order can be seen as the absence of any restriction on the order of potential occurrence – the activities are allowed to appear in an arbitrary order. In contrast, (reverse) strict order defines a particular order of execution for two activities, whereas exclusiveness completely prohibits the occurrence of two activities together in one trace. Therefore, we consider interleaving order to be the weakest relation, while exclusiveness is the strictest relation. For a dedicated pair of activities (x, y), this strictness is also reflected in the containment hierarchy of traces that show either none, one, or both activities, illustrated in Fig. 2. Here, all traces that conform to a specific behavioural relation are part of the encircled set of traces. A process model that defines interleaving order between two activities x and y allows for any trace, i.e., it may contain none, one, or both activities in any order. A model that imposes exclusiveness for both activities is most restrictive. It allows for traces that comprise none or only one of the activities. This set is a proper subset of the traces induced by interleaving order.

This notion of strictness is the foundation for the definition of a set algebra for behavioural profiles. Our operations and relations are *not* defined based on sets of traces, but on the relations of the behavioural profile. Still, the strictness of behavioural relations illustrated above with sets of traces is taken into account.

3.2 Set-Theoretic Relations

We start by introducing three set relations, i.e., equivalence, inclusion, and emptiness for behavioural profiles. Most use cases require the application of these concepts for behavioural profiles of different models for which a separate relation identifies corresponding pairs of activities. To keep the formalisation concise, we abstract from such correspondences and assume corresponding activities to be identical. As partially overlapping sets of activities do not impose serious challenges, we restrict the discussion to the behavioural aspects and assume identical sets of activities once more than one behavioural profile is considered.

Equivalence. Two behavioural profiles are equivalent, if they enforce equal behavioural constraints for the shared activities. Equivalence of behavioural profiles does not imply equal trace semantics for the shared activities, cf., [16].

Definition 4 (Equivalence). Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, ||_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, ||_2\}$ be two behavioural profiles over a set of activities A. $\mathcal{B}_1$ *equals* $\mathcal{B}_2$, denoted by $\mathcal{B}_1 = \mathcal{B}_2$, if and only if their relations are equal for all activities, i.e., $\rightsquigarrow_1 = \rightsquigarrow_2$, $+_1 = +_2$, and $||_1 = ||_2$.

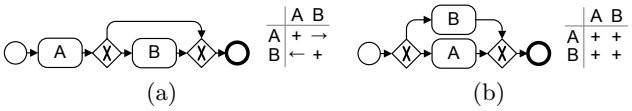


Fig. 3. The behavioural profile of (a) includes the one of (b)

Inclusion. An inclusion holds between two behavioural profiles, if one profile completely subsumes the behavioural constraints of another profile for shared activities according to the notion of strictness discussed in the previous section.

Definition 5 (Inclusion). Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, ||_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, ||_2\}$ be two behavioural profiles over a set of activities A. $\mathcal{B}_1$ *includes* $\mathcal{B}_2$, denoted by $\mathcal{B}_1 \subseteq \mathcal{B}_2$, if and only if for all pairs of activities $(a_1, a_2) \in A \times A$ it holds:

- $a_1 +_1 a_2$ implies $a_1 +_2 a_2$,
- $a_1 \rightsquigarrow_1 a_2$ implies $a_1 +_2 a_2$ or $a_1 \rightsquigarrow_2 a_2$,
- $a_1 \rightsquigarrow_1^{-1} a_2$ implies $a_1 +_2 a_2$ or $a_1 \rightsquigarrow_2^{-1} a_2$.

If $\mathcal{B}_1 \subseteq \mathcal{B}_2$, but not $\mathcal{B}_1 = \mathcal{B}_2$, we speak of proper inclusion, denoted by $\mathcal{B}_1 \subset \mathcal{B}_2$.

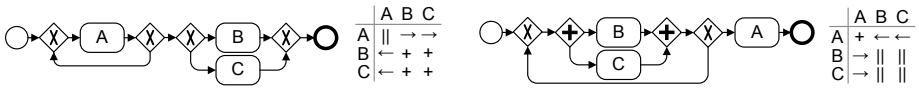
Fig. 3 illustrates inclusion of behavioural profiles (relations for start and end nodes are omitted). The profile of model 3(a) includes the one of model 3(b) as the former is less restrictive. It allows for ordered execution of activities A and B, whereas model 3(b) forbids any occurrence of both activities in the same trace. Due to the assumed behavioural abstraction, inclusion of the behavioural profiles does not imply inclusion of the respective sets of traces. Still, the behavioural abstraction allows us to cope with variations as they are visible for activities *'Search GBA Data'* and *'Update GBA Data'* in our example in Fig. 1. Model (II) defines strict order for both activities, whereas model (I) completely disallows joint occurrence of both activities. Hence, the behavioural constraints imposed by model (I) are included in the constraints imposed by model (II). Any trace-based assessment will fail to address such cases.

Emptiness. A behavioural profile is empty, if it defines all pairs of activities, except for the start and end activity, to be exclusive. Such a profile forbids the execution of any activity other than the start and the end activity. As those activities are assumed to carry no semantic meaning but indicate initialisation and termination of the process, such a trace is considered to be empty.

Definition 6 (Emptiness). Let $\mathcal{B} = \{\rightsquigarrow, +, ||\}$ be a behavioural profile over a set of activities A with $s, e \in A$ being start and end activities. $\mathcal{B}$ is *empty*, if and only if all activity pairs $(a_1, a_2) \in (A \times A) \setminus \{(s, e), (e, s)\}$ are exclusive, $a_1 + a_2$.

3.3 Set-Theoretic Operations

We introduce three set operations for behavioural profiles, i.e., complementation, intersection, and union. Again, we abstract from a correspondence relation, assume that corresponding activities are identical, and focus on the constraints for shared activities once multiple behavioural profiles are considered.



(a) (b)

Fig. 4. Process models with complementary behavioural profiles

Complementation. The complement operation is defined for a single behavioural profile and returns a profile that specifies reverse relations for all pairs of activities of the original behavioural profile.

Definition 7 (Complement). Let $\mathcal{B} = \{\rightsquigarrow, +, ||\}$ be a behavioural profile over a set of activities A. The *complement* $\overline{\mathcal{B}} = \{\rightsquigarrow_C, +_C, ||_C\}$ of $\mathcal{B}$ is a behavioural profile over A, such that for all pairs of activities $(a_1, a_2) \in A \times A$ it holds:

- $a_1 +_C a_2$ if and only if $a_1 \mid\mid a_2$,
- $a_1 \rightsquigarrow_C a_2$ if and only if $a_1 \rightsquigarrow^{-1} a_2$,
- $a_1 ||_C a_2$ if and only if $a_1 + a_2$.

The complement operation is illustrated in Fig. 4. Model 4(a) and model 4(b) show complementary behavioural profiles (neglecting start and end nodes) – the profile of the left model is the complement of the profile of the right model, and vice versa. For instance, there is a strict order constraint between activities A and B in model 4(a), whereas both activities are in reverse strict order in model 4(b). Further, activity A may occur multiple times in model 4(a), whereas it is exclusive to itself in model 4(b).

Intersection. Given two behavioural profiles, the intersection operation yields a third behavioural profile that combines the strictest relations of the behavioural profile for all shared pairs of activities. Therefore, the intersection represents the behavioural constraints that are shared by both profiles.

Definition 8 (Intersection). Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, ||_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, ||_2\}$ be two behavioural profiles over a set of activities A. The *intersection* $\cap$ of these profiles is a behavioural profile $\mathcal{B}_3 = \{\rightsquigarrow_3, +_3, ||_3\}$ over A, denoted by $\mathcal{B}_1 \cap \mathcal{B}_2 = \mathcal{B}_3$, such that for all pairs of activities $(a_1, a_2) \in A \times A$ it holds:

- $a_1 +_3 a_2$ if and only if either $a_1 +_1 a_2$, $a_1 +_2 a_2$, $(a_1 \rightsquigarrow_1 a_2 \wedge a_1 \rightsquigarrow_2^{-1} a_2)$, or $(a_1 \rightsquigarrow_1^{-1} a_2 \wedge a_1 \rightsquigarrow_2 a_2)$,
- $a_1 \rightsquigarrow_3 a_2$ if and only if either $(a_1 \rightsquigarrow_1 a_2 \wedge (a_1 \rightsquigarrow_2 a_2 \vee a_1 ||_2 a_2))$ or $(a_1 \rightsquigarrow_2 a_2 \wedge (a_1 \rightsquigarrow_1 a_2 \vee a_1 ||_1 a_2))$,
- $a_1 ||_3 a_2$ if and only if $a_1 ||_1 a_2$ and $a_1 ||_2 a_2$.

We illustrate the intersection of behavioural profiles with the models in Fig. 5. The lower model (c) shows a behavioural profile that corresponds to the intersection of the behavioural profiles of the upper two models (a) and (b). Consider, for instance, activities A and B. While model (a) allows for interleaving order between both activities, model (b) is more restrictive and enforces strict order. Hence, the intersection also defines strict order for both activities. Due to the assumed behavioural abstraction, again, model (c) does not represent the intersection of

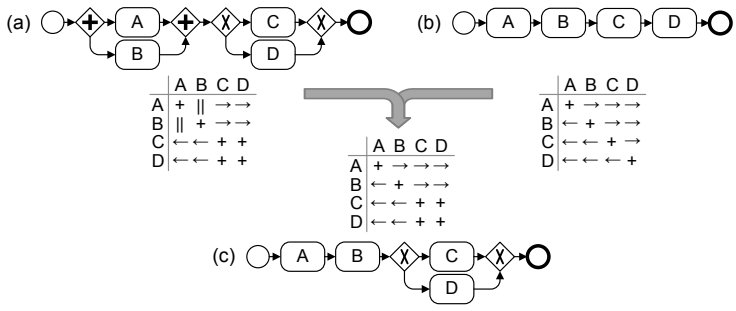


Fig. 5. The behavioural profile of model (c) corresponds to the intersection of the profiles of models (a) and (b)

the sets of traces of models (a) and (b). Referring to our example in Fig. 1, the set of shared complete traces is empty for both models. The evident behavioural commonalities of both models are not revealed by a trace-based assessment. The intersection of behavioural profiles allows to address such scenarios.

Union. The union operation for two behavioural profiles yields a third behavioural profile that combines the weakest constraints of the two profiles given as input parameters for all pairs of activities. We trace back the definition of the union operation to the complement and intersection operations using De Morgan's rule. The union $\mathcal{B}_3$ of two behavioural profiles $\mathcal{B}_1$ and $\mathcal{B}_2$ over a set of activities A, denoted by $\mathcal{B}_3 = \mathcal{B}_1 \cup \mathcal{B}_2$ is defined as $\mathcal{B}_3 = \overline{\overline{\mathcal{B}_1} \cap \overline{\mathcal{B}_2}}$.

4 Managing Process Variability

In this section, we discuss how the set algebra for behavioural profiles is applied to address the questions raised in Section 2.1 regarding the management of decoupled process variants. Let $P_1, \ldots, P_n$ be a set of models that were found to capture different variants of a business process with A being the shared activities.

C1: Shared behaviour: Greatest Common Divisor. The shared behaviour may be referred to as the Greatest Common Divisor (GCD), cf., [14]. Given a set of process variants, the GCD is characterised by a behavioural profile $\mathcal{B}_{GCD}$ over A that is derived by computing the intersection of all profiles $\mathcal{B}_{P_1}, \ldots, \mathcal{B}_{P_n}$. Hence, the GCD integrates the constraints shared by all variants. The profile $\mathcal{B}_{GCD}$ may be checked for emptiness. If it is empty, all variants impose contradicting constraints for the shared activities. A model representing the GCD can be synthesized from the profile $\mathcal{B}_{GCD}$ following the approach introduced in [26]. Note that the synthesises imposes certain consistency requirements on the behavioural profile. Further, the behavioural commonality between a single variant P_i and the GCD is quantified as the relative share of activity pairs that have equal relations in $\mathcal{B}_{GCD}$ and $\mathcal{B}_{P_i}$. This measure quantifies how much additional behaviour the variant allows for, relative to the behaviour shared with all variants.

Fig. 6 depicts the GCD for the variants of our initial example in Fig. 1. The behavioural profile of the GCD comprises all constraints from model (I) as they

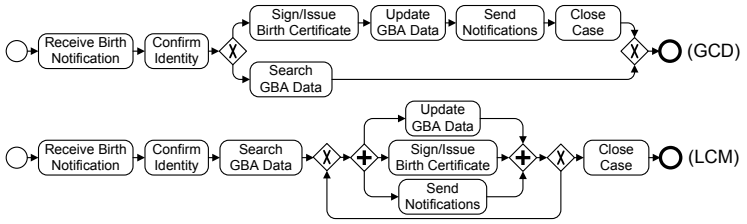


Fig. 6. Synthesised GCD and LCM for the scenario of Fig. 1

are more restrictive than those imposed by model (II). The GCD visualises the basic ordering constraints of both variants. Still, it is not identical to model (I) as certain causalities are abstracted by the behavioural profile.

C2: Most general behaviour: Least Common Multiple. The most general behaviour is referred to as the Least Common Multiple (LCM) of a set of variants. It is characterised by a behavioural profile $\mathcal{B}_{LCM}$ over A that is derived by computing the union of all profiles $\mathcal{B}_{P_1}, \ldots, \mathcal{B}_{P_n}$. The LCM imposes solely the weakest constraints for a pair of activities in the set of variants. Again, a model is derived from $\mathcal{B}_{LCM}$ using the synthesis approach for behavioural profiles.

Fig. 6 depicts the LCM for the variants of our initial example. The parallel execution of three activities is caused by the interleaving order imposed by the profile of the LCM. As all of them may be executed multiple times (interleaving order as a self-relation), they are also part of a control flow cycle. Note that model synthesis for these activities includes various design decisions on how to represent interleaving order (by concurrency or by cyclic structures) [26]. Hence, the synthesis approach may be adapted so that an LCM with a different structure, but identical behavioural profile is created.

C3: Shared forbidden behaviour: Complementary LCM. In order to characterise the behaviour that is forbidden by all variants for shared activities, a behavioural profile $\mathcal{B}_{SFB}$ over A is created as the complement of the LCM, $\mathcal{B}_{SFB} = \overline{\mathcal{B}_{LCM}}$. However, this profile does not directly capture all constraints that are not implemented in any variant due to the strictness of behavioural relations, cf., Section 3.1. Conclusions are drawn solely from the (reverse) strict order and interleaving order relations of the profile $\mathcal{B}_{SFB}$. If $\mathcal{B}_{SFB}$ defines interleaving order between two activities, all variants show exclusiveness for these activities. Hence, the potentially arbitrary order implied by the interleaving order constraint is forbidden in all variants. Similar conclusions are drawn for (reverse) strict order constraints in $\mathcal{B}_{SFB}$.

We illustrate this concept with the activities *'Update GBA Data'* and *'Send Notifications'* of our initial example. For both activities, the LCM defines interleaving order, which yields exclusiveness in the complement. As exclusiveness is the strictest relation, we cannot draw any conclusions on shared forbidden behaviour. For the activities *'Confirm Identity'* and *'Update GBA Data'*, however, the LCM defines a strict order constraint from the former to the latter. Therefore, the reverse strict order in the complement is shared forbidden behaviour and prevents execution of activity *'Update GBA Data' before 'Confirm Identity'*.

Table 1. Variant clusters in the SAP reference model

# Shared Nodes (Min)	4	6	8	10	12	14	16	18	20	22	24	26
# Variant Clusters	84	48	33	23	23	21	15	11	9	2	1	0
Avg Size of Clusters	3	3	3	3	3	2.6	2.4	2.3	2.3	2	2	0
Max Size of Clusters	10	9	8	8	7	6	4	4	4	2	2	0
# Models in Clusters	212	124	88	63	56	47	36	25	21	4	2	0
# Subsumed Models	127	73	55	41	35	28	20	14	12	2	1	0

C4: Redundancy of variants: Inclusion of variants. Given the behavioural profiles of two process variants P_i and P_j, the question whether the behaviour of one variant is captured in the other variant for the shared activities is traced back to the inclusion of their behavioural profiles, i.e., $\mathcal{B}_{P_i} \subseteq \mathcal{B}_{P_j}$. In this case, all constraints imposed by P_i would be equal or less strict than the constraints imposed by P_j, so that the behaviour (according to the behavioural profile) of P_j is covered by P_i. That suggests integration of variant P_j into variant P_i.

Investigating the two models given in Fig. 1, the behavioural profile of model (II) includes the profile of model (I) for the shared activities. Hence, when aiming at reducing the number of variants, the existence of model (I) may be challenged.

5 Case Study

Our approach to managing process variability has been evaluated based on the SAP reference model [27]. This model collection describes the functionality of the SAP R/3 system and comprises 604 process diagrams, which are expanded to 737 models in EPC notation as some diagrams contain multiple disconnected EPCs. These EPC models capture different functional aspects of an enterprise, such as sales or accounting. However, the models are not fully orthogonal. Various models show an overlap, such that events and functions with identical labels occur in multiple models. These models, therefore, represent process variants.

For our analysis, we excluded all models that showed behavioural anomalies, such as deadlocks and livelocks [28], or ambiguous instantiation semantics [29]. We also normalised multiple start and end events, and replaced block-structured OR-split and OR-join connectors with AND connectors, which does not impact on the behavioural profile. These selections and transformations led to a set of 493 EPC models, that are grounded on our formal model introduced in Section 2.2. Hence, we were able to leverage the efficient techniques mentioned in Section 2.3, so that behavioural profiles were computed in milliseconds.

Table 1 gives on overview of the observed clusters of process variants in the SAP reference model. Given a threshold of shared nodes, we derived all process model clusters of maximal size for which the set of shared nodes was equal or larger than the threshold. For these clusters, Table 1 depicts the number of clusters, their average and maximal size, and the number of considered models. For instance, requiring variants to share at least 12 nodes led to 23 clusters comprising 56 distinct models. The average size of the clusters is three models,

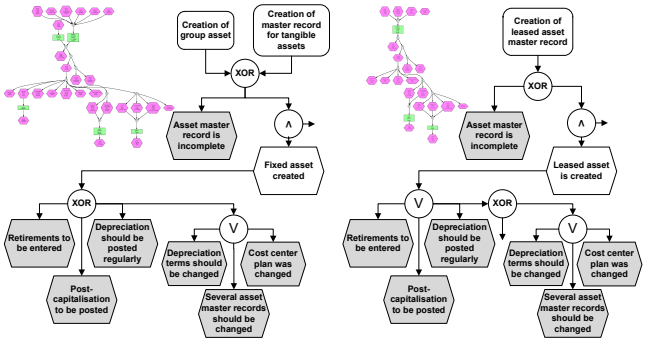


Fig. 7. Excerpts of two process variants in the SAP reference model

the maximum size is seven models. The derived values witness the existence of a large number of process variants in the model collection.

In order to analyse the commonalities for all identified model clusters, we implemented the presented set algebra and derived the LCM. This analysis revealed that the behavioural profile of the LCM was often equal to the profile of at least one model in the cluster. Based on this observation, we checked for all clusters of variants whether the behavioural profile of one variant includes the profiles of all other variants for the shared nodes. In fact, this was the case for all but two clusters. The identified LCM models could be used as a starting point for harmonising the reference model by removing redundant variants and integrating their non-shared nodes into the former model. Consider, for instance, the excerpts of two encountered process variants in Fig. 7. Both models share 16 nodes, some of them are highlighted in the excerpt. For some shared nodes both models define different behavioural constraints. Still, the behaviour of the right model includes the behaviour of the left model for shared nodes, which suggests integration of the processing defined by the left model into the right one.

Using this approach, more than half of the identified variants could be removed from the collection as indicated by the last row in Table 1. Depending on the number of required shared nodes, the reference model could be reduced by up to 127 process models.

6 Related Work

Research related to our work is conducted mainly in three areas: comparison of behaviour, modelling of process variants, and behaviour synthesis.

The comparison of behaviour as defined by the operators in this paper is related to various notions of behavioural equivalence of the linear time – branching time spectrum [30]. Behavioural profiles provide an abstraction which approximates trace equivalence at the weaker end of this spectrum. Notions of inclusion are discussed in work on behavioural equivalence [7,8,9]. Recent works calculate a degree of behavioural similarity between process models based on linguistic, graph-matching, and state-based concepts, see [31] for an overview. Our work instead

provides a precise definition of how to determine the behavioural intersection of two process models. There is also work that aims to determine the union of the behaviour captured by two process models [10,11,12,13]. None of these works has been integrated and extended towards the definition of an algebra.

The requirements for representing variants of a process have been addressed by different modelling approaches. In our work we build on the assumption that the union or intersection of two process models is again a process model. This approach is in line with work on the configuration of workflow models [6]. Other approaches use dedicated elements for capturing variation points on the level of the modelling language. Such languages include Configurable EPCs [5,32], aggregated EPCs [33], Provop [3], or variant rich process models [34], which pick up ideas and concepts from modelling of software product families [35] and feature diagrams [36]. These approaches typically assume that variation is identified and explicitly represented by a human modeller while our algebraic operations permit the calculation of intersection or union from two model variants.

Similar relations, but not exactly those of behavioural profiles, are used in a pre-processing step of approaches to synthesize a process model [37]. Please refer to [38] for a discussion of the conceptual differences between the relations used in [37] and those of the behavioural profile. Our general idea of defining an algebra for managing process variants is inspired by an algebra for operating guidelines, a formal concept for the synthesis of interaction partners for a process [17].

7 Conclusion

In this paper, we addressed two fundamental challenges of managing process variants, the specification of formal operations for reasoning with variants, and the foundation of such operations upon an appropriate behavioural abstraction. As a solution, we proposed a set algebra for behavioural profiles that enables conclusions on behavioural commonalities and differences using set-theoretic relations and operations. We showed how these concepts are used to address the problem of managing process variants. Our case study illustrated the potential of our approach for harmonisation of process model collections.

Having defined a complete algebra for the (abstracted) behaviour of processes allows for behavioural analysis way beyond similarity measurement. For instance, a process model may be used to encode forbidden behaviour. Using our algebraic operations, a model collection may then be analysed whether it allows for the forbidden behaviour. Our operations may also be used during the design of process models. As an example, subsumption of behavioural profiles may hint at the creation of model fragments that can be found in a model collection already.

We also have to reflect on some limitations of our approach. We focus on the control flow perspective and neglect data and resource assignments. We foresee that our set algebraic approach may be extended in these directions. As a starting point, the standard set theoretic operations may be applied to sets of data artefacts and resources. However, operations that consider the interplay between the perspectives (i.e., the intersection of data access constraints for a

certain role) can be assumed to provide even more value. Another restriction is our assumption of elementary 1:1 correspondences between activities of two behavioural profiles. Differences in the abstraction level are likely to be observed in practice, in particular if models are created for different purposes (e.g., process analysis vs. process design). In future work, we aim at lifting our approach to the level of n:m correspondences between activities.

References

1. Wijnhoven, F., Spil, T., Stegwee, R., Fa, R.: Post-merger IT integration strategies: An IT alignment perspective. The Journal of Strategic Information Systems **15**(1) (2006) 5–28
2. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with ADEPT2. In: ICDE, IEEE Computer Society (2005) 1113–1114
3. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. Journal of Software Maintenance **22**(6-7) (2010) 519–546
4. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66**(3) (2008) 438–466
5. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1) (2007) 1–23
6. Gottschalk, F., van der Aalst, W., Jansen-Vullers, M., Rosa, M.L.: Configurable workflow models. International Journal of Cooperative Information Systems (IJCIS) **17**(2) (June 2008) 177–221
7. Ebert, J., Engels, G.: Observable or Invocable Behaviour - You Have to Choose. Technical Report 94-38, Leiden University (December 1994)
8. Schrefl, M., Stumptner, M.: Behavior-consistent specialization of object life cycles. ACM Trans. Softw. Eng. Methodol. **11**(1) (2002) 92–148
9. Basten, T., van der Aalst, W.M.P.: Inheritance of Behavior. Journal of Logic and Algebraic Programming (JLAP) **47**(2) (2001) 47–145
10. Preuner, G., Conrad, S., Schrefl, M.: View integration of behavior in object-oriented databases. Data & Knowledge Engineering **36**(2) (2001) 153–183
11. Mendling, J., Simon, C.: Business Process Design by View Integration. In: BPM Workshops. Volume 4103 of LNCS. Springer (2006) 55–64
12. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H.: Merging event-driven process chains. In: OTM Proceedings, Part I. Volume 5331 of LNCS. Springer (2008) 418–426
13. Rosa, M.L., Dumas, M., Uba, R., Dijkman, R.M.: Merging business process models. In: OTM Proceedings, Part I. Volume 6426 of LNCS. Springer (2010) 96–113
14. van der Aalst, W.M.P.: Inheritance of business processes: A journey visiting four notorious problems. In: Petri Net Technology for Communication-Based Systems. Volume 2472 of LNCS. Springer (2003) 383–408
15. van Glabbeek, R.J., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Inf. **37**(4/5) (2001) 229–327
16. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioural profiles of process models. IEEE Transactions on Software Engineering (2010) To appear.

17. Kaschner, K., Wolf, K.: Set algebra for service behavior: Applications and constructions. In: BPM. Proceedings. Volume 5701 of LNCS. Springer (2009) 193–210
18. Gottschalk, F.: Configurable Process Models. PhD thesis, Eindhoven University of Technology, The Netherlands (December 2009)
19. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S.M., Zave, P.: Matching and merging of statecharts specifications. In: ICSE, IEEE Computer Society (2007) 54–64
20. Dijkman, R.M., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: EDOC, IEEE Computer Society (2009) 45–53
21. Weidlich, M., Dijkman, R.M., Mendling, J.: The icop framework: Identification of correspondences between process models. In: CAiSE. Volume 6051 of LNCS. Springer (2010) 483–498
22. Vanhatalo, J., Völzer, H., Leymann, F., Moser, S.: Automatic workflow graph refactoring and completion. In: ICSOC. Volume 5364 of LNCS. Springer (2008) 100–115
23. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri net transformations for business processes - a survey. TOPNOC **2** (2009) 46–63
24. Aalst, W.: The application of Petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1) (1998) 21–66
25. Aalst, W.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: BPM. Volume 1806 of LNCS. (2000) 161–183
26. Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In: ICSOC. Volume 6470 of LNCS. Springer (2010) 1–16
27. Curran, T.A., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice-Hall (1997)
28. Mendling, J., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P., Neumann, G.: Detection and prediction of errors in EPCs of the SAP reference model. Data Knowl. Eng. **64**(1) (2008) 312–329
29. Decker, G., Mendling, J.: Process instantiation. Data Knowl. Eng. **68** (2009) 777–792
30. van Glabbeek, R.: The linear time - brancing time spectrum I. The semantics of concrete, sequential processes. In: Handbook of Process Algebra. Elsevier (2001) 3–99
31. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. **36**(2) (2011) 498–516
32. La Rosa, M., Dumas, M., ter Hofstede, A., Mendling, J.: Configurable multi-perspective business process models. Inf. Syst. **36**(2) (2011) 313–340
33. Reijers, H., Mans, R., van der Toorn, R.: Improved model management with aggregated business process models. Data Knowl. Eng. **68**(2) (2009) 221–243
34. Schnieders, A., Puhlmann, F.: Variability mechanisms in e-business process families. In: BIS. Volume 85 of LNI. GI (2006) 583–601
35. Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering: foundations, principles, and techniques. Springer New York Inc (2005)
36. Schobbens, P.Y., Heymans, P., Trigaux, J.C., Bontemps, Y.: Generic semantics of feature diagrams. Computer Networks **51**(2) (2007) 456–479
37. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9) (2004) 1128–1142
38. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient computation of causal behavioural profiles using structural decomposition. In: Petri Nets. Volume 6128 of LNCS. Springer (2010) 63–83