Business Process Model Abstraction based on Behavioral Profiles

Sergey Smirnov[1], Matthias Weidlich[1], and Jan Mendling[2]

[1] Hasso Plattner Institute, Potsdam, Germany
{sergey.smirnov,matthias.weidlich}@hpi.uni-potsdam.de
[2] Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract. A variety of drivers for process modeling efforts, from low-level service orchestration to high-level decision support, results in many process models describing one business process. Depending on the modeling purpose, these models differ with respect to the model granularity. Business process model abstraction (BPMA) emerged as a technique that given a process model delivers a high-level process representation containing more coarse-grained activities and overall ordering constraints between them. Thereby, BPMA reduces the number of models capturing the same business process on different abstraction levels. In this paper, we present an abstraction approach that derives control flow dependencies for activities of an abstract model, once the groups of related activities are selected for aggregation. In contrast to the existing work, we allow for arbitrary activity groupings. To this end, we employ the behavioral profile notion that captures behavioral characteristics of a process model. Based on the original model and the activity grouping, we compute a new behavioral profile used for synthesis of the abstract process model.

1 Introduction

Business process management is a methodology that allows companies to stay competitive and shorten the time to market periods of their products and services [14]. Typically, each product or service is supported by a series of operational business processes. Companies that adopt business process management use models to explicitly capture the knowledge about their processes. In large companies such initiatives often yield several thousand models. Not only the number is a challenge to maintenance of these models, but also the fact that often several models relate to the same process. This is, for instance, the case when there exists a BPEL model capturing the service orchestration, a detailed conceptual model describing the work steps, and an overview model for senior management. In this context, business process model abstraction (BPMA) emerged as a technique that works on the most detailed model. It preserves essential process properties leaving out insignificant details. In this way, maintenance can be centered around the most fine-grained model from which the more abstract models are generated.

BPMA is related to different use cases. These include, for instance, discovering the perspective of a particular collaboration partner or filtering out activities of minor interest. A user study with industry has revealed that getting a quick overview of a detailed process is urgently required in practice [26]. Technically, a more high-level process model has to be derived with more coarse-grained activities and their control flow relations. A corresponding BPMA technique has to tackle two questions. First, which activities should be grouped into more coarse-grained ones? Second, what are the control flow relations between them? Most of the existing work has made structural assumptions regarding the first question, such that the second question becomes trivial, cf. [6, 12, 16, 21]. Meanwhile, these restrictions are often not realistic given the requirements in practice [22, 29].

In this paper, we address the abstraction problem, assuming that arbitrary activity groupings are specified. Our contribution is a technique for discovering control flow relations of an abstract model given an unrestricted grouping of activities in the initial model. Our novel approach builds on behavioral profiles, a mechanism that captures control relations between each pair of activities in terms of strict order, exclusiveness, or interleaving. Furthermore, we develop an approach for the synthesis of a process model from a behavioral profile. The synthesis builds on the newly defined notion of consistency for behavioral profiles.

The rest of the paper is structured accordingly. Section 2 motivates the problem and introduces the basic notations. Section 3 presents the developed BPMA technique including the derivation of an abstract profile and the synthesis of the model. Section 4 discusses the related work. Finally, Section 5 concludes the paper and provides an outline on the future work.

2 Background

This section discusses BPMA and explains the limitations of the existing approaches supporting the argumentation with an example. Once the motivation is provided, we introduce the formalism further used in the paper.

2.1 Business Process Model Abstraction

In essence, business process model abstraction is an operation on a model which preserves essential properties by leaving out insignificant details in order to retain information relevant for a particular purpose. BPMA is realized by means of two basic abstraction operations: elimination and aggregation (respectively, inverse of the extension and refinement operations for behavior inheritance [24]). While elimination omits insignificant activities, aggregation groups several semantically related activities into one high-level activity. As the question of which activities shall be aggregated has been partially answered by prior research, e.g., cf. [25], we focus on how the ordering relations between high-level activities are derived.

Existing approaches restrict the choice of activities to be aggregated and derive the ordering relations between high-level activities analyzing the initial model control flow, cf. [6, 16, 21]. In these works each coarse-grained activity is mapped to a process model fragment. The fragments are either explicitly given by patterns or specified through properties. The latter enables aggregation of

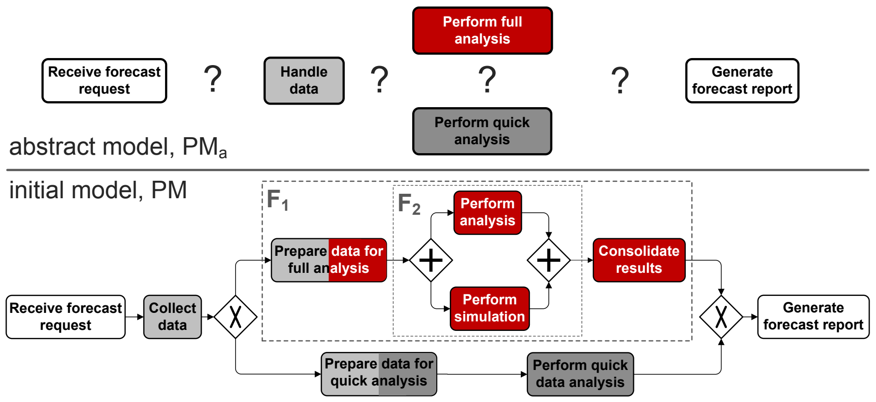

Fig. 1. Motivating example: initial model and activity grouping

fragments with an arbitrary inner structure and rests upon model decomposition techniques [21]. However, the direct analysis of the control flow has limitations. In practice, semantically related activities can be allocated in the original model independently of the control flow structure, while one activity may belong to several semantically related activity sets [22, 29]. Consider the example model PM in Fig. 1. The model describes a business process, where a forecast request is processed. Once a forecast request is received, the required data is collected. Then, there are two options: either to perform a full data analysis, or its short version. The process concludes with a forecast report creation. Model PM contains several semantically related activities that can be aggregated together into more coarse-grained ones. In model PM related activities are marked by the same background color, for instance, {*Prepare data for quick analysis, Perform quick data analysis*}. Each activity set corresponds to a respective high-level activity in the abstract model PM_a, e.g., *Perform quick analysis*. Activities *Receive forecast request* and *Generate forecast report* are not aggregated and, hence, appear in the abstract model as is. The abstraction techniques proposed in prior research allow to aggregate activities that belong, for instance, to fragment F_1 or F_2. However, none of the existing approaches is capable of suggesting the ordering constraints between activities *Handle data*, *Perform full analysis*, and *Perform quick analysis* in the abstract model. In this paper, we define a more flexible approach to determine the control structure. We utilize *behavioral profiles* [30] as the underlying formalism. Behavioral profiles capture the essential behavior of a process model in terms of strict order, exclusiveness, and interleaving order relations for each activity pair.

Further, addressing the user demand revealed in [26], our BPMA technique comprises a slider control that manages the ordering constraints loss. The slider allows the user to select an appropriate model from a spectrum of models: from the model with an arbitrary execution of high-level activities to the model where the ordering constraints of PM are best-effort preserved.

2.2 Preliminaries

For our discussion, we use a formal notion of a process model. It captures the commonalities of process modeling languages, such as BPMN or EPCs.

Definition 1 (Process Model). A tuple $PM = (A, G, F, s, e, t)$ is a *process model* , where:

- A is a finite nonempty set of activities;
- G is a finite set of gateways;
- $N = A \cup G$ is a finite set of nodes with $A \cap G = \emptyset$;
- $F \subseteq N \times N$ 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, respectively, the direct predecessors and successors of a node $n \in N$;
- $\forall\ a \in A : |\bullet a| \leq 1 \wedge |a \bullet| \leq 1$
- $s \in A$ is the only start activity, such that $\bullet s = \emptyset$;
- $e \in A$ is the only end activity, such that $e\bullet = \emptyset$;
- $t : G \rightarrow \{and, xor\}$ is a mapping that associates each gateway with a type.

The execution semantics of such a process model is given by a translation into a Petri net following on common formalizations, cf., [1, 10]. As our notion of a process model comprises a dedicated start activity and a dedicated end activity, the resulting Petri net is a workflow net (WF-net) [1]. All gateways are of type *and* or *xor*, such that the WF-net is free-choice [1]. In order to arrive at a behavioral profile, we consider the set of all complete traces (or execution sequences) from start s to end e. The set of *complete process traces* $\mathcal{T}_{PM}$ for a process model PM contains lists of the form $s \cdot A^* \cdot e$ such that a list comprises the execution order of activities. We use $a \in \sigma$ with $\sigma \in \mathcal{T}_{PM}$ to denote that an activity a is a part of a complete process trace. The behavioral profile is grounded on the notion of *weak order* between activities within this set of traces. Two activities of a process model are in weak order, if there exists a trace in which one activity occurs after the other. This relation requires the *existence* of such a trace and does not have to hold for all traces of the model.

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

Depending on how two activities of a process model are related by weak order, we define three relations forming the behavioral profile.

Definition 3 (Behavioral Profile). Let $PM = (A, G, F, s, e, t)$ be a process model. A pair $(x, y) \in (A \times A)$ is in one of the following relations:

- *strict order relation* $\rightsquigarrow_{PM}$*, if* $x \succ_{PM} y$ *and* $y \not\succ_{PM} x$*;*
- *exclusiveness relation* $+_{PM}$*, if* $x \not\succ_{PM} y$ *and* $y \not\succ_{PM} x$*;*
- *interleaving order relation* $||_{PM}$*, if* $x \succ_{PM} y$ *and* $y \succ_{PM} x$*.*

The set of all three relations is the *behavioral profile* of PM.

The relations of the behavioral profile, along with the inverse strict order $\rightsquigarrow^{-1} = \{(x, y) \in (A \times A) \mid (y, x) \in \rightsquigarrow\}$, partition the Cartesian product of activities.

The behavioral profile relations allow different levels of freedom for activities. While interleaving order relation allows the activities to appear in an arbitrary order, (inverse) strict order specifies a particular execution order, and exclusiveness prohibits appearance of two activities in one trace. Thus, we organize the relations into a hierarchy presented in Fig. 2. At the top of the hierarchy the "strictest" relation appears, while at the bottom—the least restrictive.

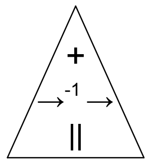

Fig. 2. Behavioral relation hierarchy

3 Abstract Model Synthesis

In this section, we describe the developed abstraction technique. We realize the technique in the following steps.
Step 1 derive the behavioral profile BP_{PM} for model PM
Step 2 construct the behavioral profile BP_{PM_a} for model PM_a
Step 3 **if** a model consistent with profile BP_{PM_a} exists
Step 4 **then** create PM_a, **else** report an inconsistency.
The remainder of this section is structured according to the above mentioned steps: each subsection discusses one step.

3.1 Deriving Behavioral Relations from a Process Model

Derivation of the behavioral profile of a process model can be done efficiently under the assumption of soundness. Soundness is a correctness criteria often used for process models that guarantees the absence of behavioral anomalies, such as deadlocks or livelocks [2]. It has been defined for WF-nets. As discussed in Section 2.2, semantics for our notion of a process model is given by a translation into free-choice WF-nets. Hence, the soundness criterion can be directly applied to a process model. Moreover, we are able to reuse techniques for the derivation of behavioral profiles that have been introduced for sound free-choice WF-nets in [30]. Based thereon, behavioral profiles can be derived in $O(n^3)$ time with n as the number of nodes of the respective WF-net. In essence, this approach establishes a relation between the structure of a net and the relations of its behavioral profile.

3.2 Abstract Model Behavioral Profile Construction

We assume that each high-level activity in $PM_a = (A_a, G_a, F_a, s_a, e_a, t_a)$ is the result of aggregation of several activities in $PM = (A, G, F, s, e, t)$. Then, the construction of coarse-grained activities is formalized by a function *aggregate*:

Definition 4 (Function Aggregate). Let $PM = (A, G, F, s, e, t)$ be a process model and $PM_a = (A_a, G_a, F_a, s_a, e_a, t_a)$—its abstract counterpart. Function $aggregate : \mathcal{A}_a \rightarrow \mathcal{P}(\mathcal{A})$ specifies a correspondence between one activity in PM_a and the set of activities in PM.

Algorithm 1 Derivation of a behavioral relation for an activity pair

```
1: deriveBehavioralRelation(Activity $x$, Activity $y$, Double $w_t$)
2: $w(x \succ_{PM_a} y) = |\{\forall (a,b) \in aggregate(x) \times aggregate(y) : a \rightsquigarrow_{PM} b \vee a ||_{PM} b\}|$
3: $w(y \succ_{PM_a} x) = |\{\forall (a,b) \in aggregate(x) \times aggregate(y) : a \rightsquigarrow^{-1}_{PM} b \vee a ||_{PM} b\}|$
4: $w(x \not\succ_{PM_a} y) = |\{\forall (a,b) \in aggregate(x) \times aggregate(y) : a \rightsquigarrow^{-1}_{PM} b \vee a +_{PM} b\}|$
5: $w(y \not\succ_{PM_a} x) = |\{\forall (a,b) \in aggregate(x) \times aggregate(y) : a \rightsquigarrow_{PM} b \vee a +_{PM} b\}|$
6: $w_{prod} = |aggregate(x)| \cdot |aggregate(y)|$
7: $w(x +_{PM_a} y) = \frac{min(w(x \not\succ_{PM_a} y), w(y \not\succ_{PM_a} x))}{w_{prod}}$
8: $w(x \rightsquigarrow_{PM_a} y) = \frac{min(w(x \succ_{PM_a} y), w(y \not\succ_{PM_a} x))}{w_{prod}}$
9: $w(x \rightsquigarrow^{-1}_{PM_a} y) = \frac{min(w(y \succ_{PM_a} x), w(x \not\succ_{PM_a} y))}{w_{prod}}$
10: $w(x ||_{PM_a} y) = \frac{min(w(x \succ_{PM_a} y), w(y \succ_{PM_a} x))}{w_{prod}}$
11: if $w(x +_{PM_a} y) > w_t$ then
12:     return $x +_{PM_a} y$
13: if $w(x \rightsquigarrow_{PM_a} y) > w_t$ then
14:     if $w(x \rightsquigarrow^{-1}_{PM_a} y) > w(x \rightsquigarrow_{PM_a} y)$ then
15:         return $x \rightsquigarrow^{-1}_{PM_a} y$
16:     else
17:         return $x \rightsquigarrow_{PM_a} y$
18: if $w(x \rightsquigarrow^{-1}_{PM_a} y) > w_t$ then
19:     return $x \rightsquigarrow^{-1}_{PM_a} y$
20: return $x ||_{PM_a} y$
```

Considering the example in Fig. 1, for instance, it holds *aggregate(Perform quick analysis)* = {*Prepare data for quick analysis, Perform quick data analysis*} and *aggregate(Handle data)*={*Collect data, Prepare data for full analysis, Prepare data for quick analysis*}. The behavioral profile of model PM_a defines the relations between each pair of activities in PM_a. To discover the behavioral profile for PM_a we analyze the relations among activities in PM and consider the function *aggregate*. For each pair of coarse-grained activities x, y, where $x, y \in A_a$, we study the relations between a and b, where $a \in aggregate(x), b \in aggregate(y)$. This study reveals a dominating behavioral relation between elements of $aggregate(x)$ and $aggregate(y)$. We assume that the behavioral relations between activity pairs of PM_a can be discovered independently from each other, i.e., the relation between x and y, where $x, y \in A_a$ depends on the relations between activities in $aggregate(x)$ and $aggregate(y)$, but does not depend on the relations between $aggregate(x)$ and $aggregate(z), \forall z \in A_a$.

Algorithm 1 formalizes the derivation of behavioral relations. The input of the algorithm is a pair of activities, x and y, and w_t—the user-specified threshold telling significant relation weights from the rest and, hence, managing the ordering constraints loss. The output of the algorithm is the relation between x and y. Algorithm 1 derives behavioral profile relations between x and y from the observable frequencies of relations between activities (a, b), where $(a, b) \in aggregate(x) \times aggregate(y)$. According to Definition 3 each of the behavioral profile relations is specified by the corresponding weak order relations. Thereby, to conclude about the behavioral profile relation between x and y we first evaluate the frequencies of weak order relations for x and y. The latter are found in the assumption that each weak order relation holding for $(a, b) \in$

$aggregate(x) \times aggregate(y)$, contributes to the weak order relation between x and y. This rationale allows to find the weight for each weak order relation between x and y (lines 2–5). The overall number of relations is stored in variable w_{prod} (line 6). Algorithm 1 continues finding the relative weight for each behavioral profile relation (lines 7–10). The relative weights of behavioral relations together with the relation hierarchy are used to choose the dominating relation (lines 11–20). The behavioral relations are ranked according to their relative weights. Threshold w_t selects significant relations, omitting those which relative weights are less than w_t. Finally, the relation hierarchy allows to choose the strictest relation among the significant ones. Notice that the input parameter w_t implements the slider concept: by means of w_t the user expresses the preferred ordering constraint loss level and obtains the corresponding behavioral relations for model PM_a.

To illustrate Algorithm 1 we refer to the motivating example and derive the behavioral relation between activities *Handle data* (HD) and *Perform quick analysis* (PQA) given the threshold $w_t = 0.5$. Following Algorithm 1, $w(HD \succ_{PM_a} PQA) = 4$, $w(PQA \succ_{PM_a} HD) = 1$, $w(HD \not\succ_{PM_a} PQA) = 2$, $w(PQA \not\succ_{PM_a} HD) = 5$, and $w_{prod} = 6$. Then, $w(HD +_{PM_a} PQA) = 2/6$, $w(HD \rightsquigarrow_{PM_a} PQA) = 4/6$, $w(HD \rightsquigarrow^{-1}_{PM_a} PQA) = 1/6$, and $w(HD \rightsquigarrow^{-1}_{PM_a} PQA) = 1/6$. The constellation of behavioral relation weights is shown in Fig. 3. Each relation weight w_r defines a segment $[0, w_r]$, where the respective behavioral relation r is valid. If the maximum weight of the relations w_{max} is less than 1, we claim that the interleaving order relation is valid in segment $[w_{max}, 1]$ (it provides most freedom in execution of two activities). While the resulting segments overlap, the relation hierarchy defines the dominating relation in a particular point of $[0, 1]$. For only $w(HD \rightsquigarrow_{PM_a} PQA) > 0.5$, we state *Handle data* $\rightsquigarrow_{PM_a}$ *Perform quick analysis.*

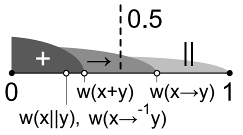

Fig. 3. Slider example

3.3 Behavioral Profile Consistency Validation

The creation of the behavioral profile for the abstract model as introduced above might yield an *inconsistent* profile. Hence, this section first introduces a notion of consistency for behavioral profiles and then elaborates on how it can be decided.

Consistent Behavioral Profiles. A behavioral profile is inconsistent, if there exists no process model that satisfies all the constraints of the behavioral profile. Whether such a process model exists depends on the applied notion of a process model and the intended structural and behavioral characteristics of the synthesized model. For instance, the strict order relation might define a cyclic dependency between three activities x, y, and z: $x \rightsquigarrow y$, $y \rightsquigarrow z$, and $z \rightsquigarrow x$. The process model fragment in Fig. 4(a) satisfies these behavioral constraints at the expense of duplicating activities. However, the result is clearly inappropriate against the background of our use case: an abstract model

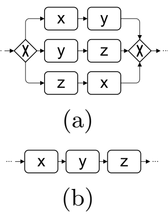

Fig. 4. Exemplary model fragments

should provide a concise and compact view on the process. Hence, extensive activity duplication should be avoided. For our notion of a process model, the aforementioned behavioral constraints cannot be satisfied as exemplified by the model in Fig. 4(b), where constraint $z \rightsquigarrow x$ is violated. Moreover, structural and behavioral properties of a process model, e.g., the absence of deadlocks, impact on the existence of a process model for a given behavioral profile.

Our notion of behavioral profile consistency is motivated by the goal of deriving a sound process model. That is, the resulting model should be free of behavioral anomalies, cf., Section 2.2.

Definition 5 (Consistent Behavioral Profile). Let $PM = (A, G, F, s, e, t)$ be a process model with the behavioral profile $\mathcal{BP} = \{\rightsquigarrow_{PM}, +_{PM}, ||_{PM}\}$. $\mathcal{R} \subseteq (A \times A)$ is a *dependency relation* derived from the behavioral profile with:
- $(x, y) \notin \mathcal{R}$ and $(y, x) \notin \mathcal{R}$, if $x +_{PM} y$.
- $(x, y) \in \mathcal{R}$ and $(y, x) \notin \mathcal{R}$, if $x \rightsquigarrow_{PM} y$.
- either $(x, y), (y, x) \in \mathcal{R}$ or $(x, y), (y, x) \notin \mathcal{R}$, if $x||_{PM}y$.

Then, $\mathcal{BP}$ is *consistent*, iff $\mathcal{R}$ is a transitive relation.

For the aforementioned example of three activities x, y, and z with $x \rightsquigarrow y$, $y \rightsquigarrow z$, and $z \rightsquigarrow x$ the profile is inconsistent: $x \rightsquigarrow y$ and $y \rightsquigarrow z$ induce $x\ \mathcal{R}\ y$ and $y\ \mathcal{R}\ z$, whereas $x \not\mathcal{R}\ z$ is derived from $z \rightsquigarrow x$.

In order to prove that consistency of a behavioral profile coincides with the existence of a sound process model that shows this profile, we need auxiliary results on the relation between behavioral and structural dependencies.

Proposition 1. Let $PM = (A, G, F, s, e, t)$ be a sound process model and $\mathcal{BP} = \{\rightsquigarrow_{PM}, +_{PM}, ||_{PM}\}$—its behavioral profile. Then, for $x, y \in A$ it holds:

`S1` there is no path between them, $x\not{F}^{+}y$ and $y\not{F}^{+}x$, if $x +_{PM} y$.
`S2` there is a path between them, $xF^{+}y$ and $y\not{F}^{+}x$, if $x \rightsquigarrow_{PM} y$.
`S3` there is either no path between them or they are a part of a control flow cycle, $(x\not{F}^{+}y) \wedge (y\not{F}^{+}x)$ or $(xF^{+}y) \wedge (yF^{+}x)$, if $x||_{PM}y$.
`S4` they are exclusive or in interleaving order, $x +_{PM} y$ or $x||_{PM}y$, if $x\not{F}^{+}y$ and $y\not{F}^{+}x$.
`S5` they are in strict order, $x \rightsquigarrow_{PM} y$, if $xF^{+}y$ and $y\not{F}^{+}x$.
`S6` they are in interleaving order, $x||_{PM}y$, if $xF^{+}y$ and $yF^{+}x$.

Proof. We have already mentioned that any sound process model can be transformed into a corresponding sound free-choice WF-net. Hence, we reuse the results that have been proven for this class of nets in [30].

`S1` Follows from Lemma 3 in [30], stating that activities that cannot be enabled concurrently are exclusive if and only if there is no path between them.

`S2` Follows from the proof of Theorem 1 in [30], which includes the statement that for activities that cannot be enabled concurrently, the path relations $xF^{+}y$ and $y\not{F}^{+}x$ coincide with strict order $x \rightsquigarrow y$.

`S3` According to Lemma 2 in [30], activities that cannot be enabled concurrently are in interleaving order, if and only if they are a part of a common control flow cycle. It remains to show that potential concurrent enabling of activities x

and y implies that there is either no path between them, $x \not F^+ y$ and $y \not F^+ x$, or they belong to a common control flow cycle, xF^+y and yF^+x. Assume that x and y can be enabled concurrently and that xF^+y and $y \not F^+ x$ hold. The path xF^+y induces a sequence of activities that can be executed starting with x and leading to a state in which y is enabled (this property has been shown for sound free-choice WF-nets in [15]). Due to $y \not F^+ x$, any token enabling y in the first state cannot impact on the execution of this activity sequence. Thereby, a state in which y is enabled concurrently to itself is reached. This contradicts the process model soundness property. Again, that follows from transferring the results shown for free-choice sound WF-nets to our setting. In [2] such nets are shown to be safe, so that no activity can be enabled concurrently to itself.

`S4` Follows directly from Lemma 1 in [30] and Lemma 3 in [30], cf. also `S1`.

`S5` Follows from the proof of Theorem 1 in [30], cf. also `S2`.

`S6` Follows directly from Lemma 2 in [30]. □

The requirements imposed by Definition 5 for relation $\mathcal{R}$ coincide with the properties of the flow relation in sound process models, cf., Proposition 1. In other words, Definition 5 ensures that the structural requirements induced by the behavioral profile are satisfiable. We see that every sound process model shows a consistent behavioral profile.

Lemma 1. *The behavioral profile of a sound process model is consistent.*

Proof. Let $PM = (A, G, F, s, e, t)$ be a sound process model with the behavioral profile $\mathcal{BP} = \{\rightsquigarrow_{PM}, +_{PM}, ||_{PM}\}$ and assume that $\mathcal{BP}$ is not consistent. According to Proposition 1 `S1`, `S2`, and `S3`, the transitive closure of the flow relation F of PM qualifies for being a dependency relation as defined in Definition 5. However, the flow relation F is transitive by definition, which yields a contradiction with our assumption of $\mathcal{BP}$ being inconsistent. □

Deciding Behavioral Profile Consistency. According to Lemma 1, consistency of the behavioral profile is a necessary condition for the existence of a sound process model. Hence, we have to clarify how to decide consistency for a given behavioral profile.

In the general case the consistency check for a behavioral profile cannot be done efficiently: interleaving order between two activities stems either from the potential concurrent enabling, or from the existence of a control flow cycle spanning both activities. Hence, to decide whether there exists a transitive relation $\mathcal{R}$ according to Definition 5, both possibilities have to be explored for each pair of activities in interleaving order. Analysis of alternatives yields an exponential time complexity of any algorithm for checking consistency.

Still, consistency can be decided efficiently under certain assumptions. Consistency of a behavioral profile comprising solely strict order and exclusiveness relations can be decided in polynomial time. In addition, two stricter notions of consistency can be applied. First, all pairs of activities x and y with $x||y$ may be required to be in the dependency relation, i.e., $(x, y), (y, x) \in \mathcal{R}$. In this case, the existence of a transitive relation $\mathcal{R}$ indicates the existence of a sound process model for the profile that does not show any concurrency of activities. That is due to the fact that interleaving order between activities stems solely from control flow cycles. We refer to this consistency as *non-concurrent consistency*. Second, all pairs of activities x and y with $x||y$ may be required not to be in the dependency relation, i.e., $(x, y), (y, x) \notin \mathcal{R}$. Then, interleaving order is expected to stem solely from concurrent enabling of activities. This consistency, referred to as *acyclic consistency*, hints at the existence of a sound process model for the profile that is acyclic. As both notions avoid to check two possibilities for interleaving pairs of activities, they can be decided in polynomial time.

Corollary 1. The following problems can be decided in $O(n^3)$ time with n as the number of activities of a behavioral profile.

1. Given a behavioral profile, to decide consistency in the absence of activities in interleaving order.
2. Given a behavioral profile, to decide non-concurrent consistency.
3. Given a behavioral profile, to decide acyclic consistency.

Proof. According to Definition 5, a dependency relation is built from the behavioral profile. In all three cases, the derivation is straight-forward as for every pair of activities it is defined whether or not it is part of the dependency relation. Hence, building the dependency relations takes $O(n^2)$ time with n as the number of activities. Then, the dependency relations is checked for transitivity, which takes $O(n^3)$ time with n as the number of activities of the process model. □

3.4 Abstract Model Synthesis from a Consistent Behavioral Profile

Once consistency of a behavioral profile is validated, we derive the abstract model structure from the behavioral profile. Due to consistency and according to Definition 5 there is a transitive dependency relation between the activities. This dependency relation induces the abstract process model flow relation. Still, to arrive at a well-structured process model in which every activity has at most one predecessor and successor, several additional steps have to be done. We outline these steps in Algorithm 2 and explain them in detail in the following paragraphs.

First, the dependency relation $\mathcal{R}$ between activities is determined. Then, a transitive reduction is performed on $\mathcal{R}$ yielding the relation $\mathcal{R}'$. Intuitively, the dependency relation corresponds to the transitive closure of the flow relation. Thereby, it is reduced by all activity pairs that can be derived through transitivity of two other activity pairs. In the case of cyclic dependencies, there are different options to remove activity pairs during transitive reduction. We can choose one pair arbitrarily, since the activity order inside a control flow cycle does not impact on the relations of the behavioral profile between these activities.

Second, we extract the set of activities of the process model, which corresponds to one of the domains of the behavioral profile relations. Further, we determine a start and an end activity as follows. If there is exactly one activity that has no predecessor in the reduced dependency relation $\mathcal{R}'$, this activity is selected as the start activity s. When there are multiple start activity candidates, an auxiliary

Algorithm 2 Deriving a process model from a consistent behavioral profile

```
1: deriveProcessModelFromBehavioralProfile($\mathcal{BP}$)
2: $\mathcal{R}$ := determineDependencyRelation($\mathcal{BP}$)
3: $\mathcal{R}'$ := doTransitiveReduction($\mathcal{R}$)
4: $A$ :=extractOneDomainOfRelation($\mathcal{BR}$)
5: $G := \emptyset$
6: $s$ := determineOrCreateStartActivity($\mathcal{R}'$)
7: $e$ := determineOrCreateEndActivity($\mathcal{R}'$)
8: $A := A \cup \{s, e\}$
9: $\mathcal{R}'$ := updateRelationBasedOnStartAndEndActivities($s, e, \mathcal{R}'$)
10: $F := \mathcal{R}'$
11: $t := \emptyset$
12: for all $a \in A$ with more than one successor in $\mathcal{R}'$ do
13:     $G_a$ := createSplitGatewaysBetweenActivityAndSuccessors($a$,$\mathcal{R}'$)
14:     $t := t \cup$ determineGatewayTypes($a$,$G_a$,$\mathcal{R}'$)
15:     $F$ := updateRelationForGatewaysAfterActivity($G, a, F$)
16:     $G := G \cup G_a$
17: for all $a \in A$ with more than one predecessor in $\mathcal{R}'$ do
18:     $G_a$ := createJoinGatewaysBetweenActivityAndSuccessors($a$,$\mathcal{R}'$)
19:     $t := t \cup$ determineGatewayTypes($a$,$G_a$,$\mathcal{R}'$)
20:     $F$ := updateRelationForGatewaysAfterActivity($G, a, F$)
21:     $G := G \cup G_a$
22: return $PM = (A, G, F, s, e, t)$
```

activity is created and added to the set of activities. Relation $\mathcal{R}'$ is updated, so that there is an entry between the auxiliary activity and all start activity candidates. In the same vein, an end activity is selected or created, respectively.

Third, the abstract model flow relation is defined as the reduced dependency relation $\mathcal{R}'$. Then, the result is a process model consisting of activities and flow arcs with dedicated start and end activities. However, activities might show multiple incoming or outgoing flow arcs. In a post-processing step, we introduce gateways to realize the splitting and joining of control flow. For an activity with multiple outgoing arcs split gateways are applied. Notice that more than one gateway might be introduced to implement the behavioral relations between succeeding activities correctly. Thus, there might be a sequence of gateways, inserted between the activity and its successors, such that the flow relation and the relation that types gateways have to be updated accordingly. For an activity a the sequence of succeeding split gateways is created as follows.

1. All successors of a, for which there is a path to a, are connected to a new *xor* split gateway. Note that these successors are part of one control flow cycle.
2. All successors of a are grouped iteratively, so that each group has the same behavioral relations to the other successors of a (or groups of successors). All these groups are either exclusive to each other, or in interleaving order.
3. All successors of a in a dedicated group are connected to a new split gateway. The gateway type is defined by the behavioral relation between the group members: exclusiveness yields a *xor* gateway, interleaving order—an *and* gateway.
4. The gateways created for all groups are chained according to the order of activity grouping. Any *xor* gateway for the activities in a control flow cycle is added as the last gateway.

We illustrate these steps by the model fragment depicted in the left part of Fig. 5. Assume there is a path from b to a and from c to a, whereas there is no path from activities d, e, and f to a. Thus, activities b and c are grouped and connected to an *xor* split. Assume that activities d and e are exclusive, while both of them are in interleaving order with activity f. Then, d and e are grouped first and connected to an *xor* split. Taking both activities as a single fragment, the second group consists of this fragment and activity f. Due to interleaving order, both are connected to an *and* split. Finally, the group containing the activities in a control flow cycle, i.e., b and c, is connected. The right part of Fig. 5 depicts the post-processing result for activity a. The approach described for activities with more than one outgoing flow arc is mirrored for activities with more than one incoming flow arc in order to introduce the respective join gateways.

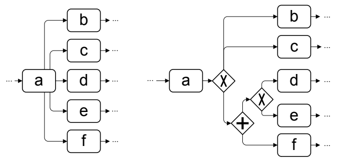

Fig. 5. Post-processing of activities with multiple outgoing flow arcs

After the activities with multiple incoming or outgoing flow arcs have been post-processed, the complete abstract model derived from the consistent behavioral profile is returned by the algorithm.

Lemma 2. *Given a consistent behavioral profile, the process model derived by Algorithm 2 shows the same behavioral profile.*

Proof. Let $\mathcal{BP} = \{\rightsquigarrow, +, ||\}$ be a consistent behavioral profile and $PM = (A, G, F, s, e, t)$ the derived process model with the behavioral profile $\mathcal{BP}' = \{\rightsquigarrow_{PM}, +_{PM}, ||_{PM}\}$. The algorithm translates the transitive dependency relation into the transitive closure of the flow relation F. Therefore, $x \rightsquigarrow y$ implies xF^+y and $y\not{F}^+x$. According to Proposition 1 S5, $x \rightsquigarrow_{PM} y$ holds. The same argument holds for interleaving activities $x||y$ that translate to xF^+y and yF^+x, yielding $x||_{PM}y$ by Proposition 1 S6. For activities x and y with $x\not{F}^+y$ and $y\not{F}^+x$, it holds either $x||y$ or $x+y$. It remains to show that in both cases the same relation can be observed in the derived process model. As the process model is a connected graph, there must be a node n for which we have a path to x and a path to y, while for two successors of n, $n_1, n_2 \in n\bullet$, it holds n_1F^+x, $n_1\not{F}^+y$, $n_2\not{F}^+x$, and n_2F^+y. As n has more than one successor, it is a gateway. The type of gateway n, *and* or *xor*, determines whether it holds $n_1 +_{PM} n_2$ or $n_1||_{PM}n_2$ and, therefore, $x +_{PM} y$ or $x||_{PM}y$. As this gateway type is selected by our algorithm based on the behavioral relation between x and y in the original profile $\mathcal{BP} = \{\rightsquigarrow, +, ||\}$, the relations of both profiles $\mathcal{BP}$ and $\mathcal{BP}'$ coincide for activities x and y. □

After we studied the relation between consistency of a behavioral profile and the existence of a sound process models in both directions, we conclude the following.

Theorem 1. *There is a sound process model, if and only if, the behavioral profile is consistent.*

Proof. The $\Rightarrow$ direction follows from Lemma 1, the $\Leftarrow$ direction from Lemma 2. □

We conclude this section returning to the motivating example presented in Section 2. Fig. 6 illustrates the complete abstract model derived from the initial model according to the developed abstraction technique. We see that the activities aggregated into *Handle data* are in strict order with most of the other activities to aggregate. Hence, the strict order relation holds also between the aggregated activities in the abstract model. For the other two aggregated activities, exclusiveness turns out to be the dominating behavioral relation.

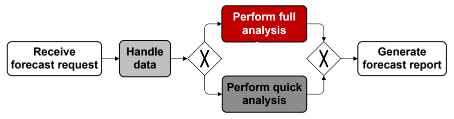

Fig. 6. Abstract model for the initial example in Fig. 1

4 Related Work

The work presented in this paper complements two research areas: business process model abstraction and process model synthesis. The former studies methods of process model transformation and criteria of model element abstraction.

The related process model transformation techniques constitute two groups. The first group builds on an explicit definition of a fragment to be transformed. Here, Petri net reduction rules preserving certain behavioral properties play an important role [19]. Such rules have also been defined for workflow graphs [23], EPCs [11, 18], and YAWL [31]. The second group of transformation techniques hierarchically decomposes a model into fragments, e.g., cf. [28]. The reduced process model can be regarded as a view in terms of [20] and typically preserves properties of behavior inheritance [3]. Unfortunately, such hierarchical decomposition is not sufficient in many scenarios, cf. [29]. The technique developed in this paper shows how the abstract model control flow can be discovered even for non-hierarchical abstractions. Model element abstraction criteria, for instance, execution cost, duration, and path frequency, have been studied in a number of works [12, 13, 26]. These works have in common that their major focus is on identifying abstraction candidates. The current paper complements this stream of research demonstrating how abstracted process models can be constructed even if aggregated activities are not structurally close to each other. There is a series of works that address the requirements of business process model abstraction. The approaches of [6, 7, 16] build on an explicit definition of a fragment that can be abstracted to provide a process overview. In [5, 21, 27] such fragments are discovered without user specification according to the model structure.

The developed method for the construction of an abstract process model from the behavioral profile extends the family of process model synthesis techniques. In process mining the alpha algorithm is used for the construction of a process model from event logs [4]. The mining relations used by the alpha algorithm differ to ours as they are only partially transitive. In this paper, we use the behavioral profile relations, which permit the reconstruction of the process model if the profile is consistent. There are further approaches to synthesis that take the state space as an input to generate process models including [8, 9, 17], which all build on Petri net formalism.

5 Conclusion and Future Work

In this paper, we have presented a novel approach to process model abstraction that addresses existing industry demand. Given a process model and sets of related activities in it, we are capable to deliver a high-level process model preserving the overall properties of the initial model or tell the user that such a model cannot be created. The suggested abstraction approach bases on an aggregation of the initial model elements and, in contrast to the available techniques, is capable of non-hierarchical aggregation. To synthesize a high-level process model we leverage behavioral profiles. Once a profile for the initial model is created, we propose how to abstract it and synthesize a high-level model out of it. Notice that in the synthesis step we assume the resulting model to be a sound process model—a reasonable assumption in practice.

This paper motivates several directions of the future work. In the context of BPMA, it is imperative to investigate criteria and methods allowing to learn which activities in the initial model are related. A corresponding solution would complement the approach developed in this paper and their combination may support the user with an automated BPMA solution. Another direction of the future work is the further research on model synthesis out of behavioral profiles. In particular, it is interesting to broaden the class of synthesized models.

References

1. W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *JCSC*, 8(1):21–66, 1998.
2. W. M. P. van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In *BPM*, volume 1806 of *LNCS*, pages 161–183, 2000.
3. W. M. P. van der Aalst and T. Basten. Life-Cycle Inheritance: A Petri-Net-Based Approach. In *ICATPN 1997*, pages 62–81, London, UK, 1997. Springer.
4. W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE TKDE*, 16(9):1128–1142, 2004.
5. A. Basu and R.W. Blanning. Synthesis and Decomposition of Processes in Organizations. *ISR*, 14(4):337–355, 2003.
6. R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007*, volume 4714 of *LNCS*, pages 88–95, Berlin, 2007. Springer.
7. J. Cardoso, J. Miller, A. Sheth, and J. Arnold. Modeling Quality of Service for Workflows and Web Service Processes. Technical report, University of Georgia, 2002. Web Services.
8. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE TC*, 47(8):859–882, August 1998.
9. J. Dehnert and W. M. P. van der Aalst. Bridging The Gap Between Business Models And Workflow Specifications. *IJCIS*, 13(3):289–332, 2004.
10. R. M. Dijkman, M. Dumas, and Ch. Ouyang. Semantics and Analysis of Business Process Models in BPMN. *IST*, 50(12):1281–1294, 2008.

11. B. van Dongen, M. Jansen-Vullers, H. Verbeek, and W. M. P. van der Aalst. Verification of the SAP Reference Models Using EPC Reduction, State-space Analysis, and Invariants. *CAIE*, 58(6):578–601, 2007.
12. R. Eshuis and P. Grefen. Constructing Customized Process Views. *DKE*, 64(2):419–438, 2008.
13. C. W. Günther and W. M. P. van der Aalst. Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007*, volume 4714 of *LNCS*, pages 328–343, Berlin, 2007. Springer.
14. M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, April 1994.
15. B Kiepuszewski, A. H. M. ter Hofstede, and W. M. P. van der Aalst. Fundamentals of Control Flow in Workflows. *Acta Informatica*, 39(3):143–209, 2003.
16. D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *ISJ*, 28(6):505–532, 2003.
17. P. Massuthe, A. Serebrenik, N. Sidorova, and K. Wolf. Can I Find a Partner? Undecidability of Partner Existence for Open Nets. *IPL*, 108(6):374–378, 2008.
18. J. Mendling and W. M. P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In *CAiSE 2007*, volume 4495 of *LNCS*, pages 439–453, Trondheim, Norway, 2007. Springer.
19. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
20. V. Pankratius and W. Stucky. A Formal Foundation for Workflow Composition, Workflow View Definition, and Workflow Normalization based on Petri Nets. In *APCCM 2005*, pages 79–88, Darlinghurst, Australia, 2005. ACS, Inc.
21. A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, pages 229–244, Ulm, Germany, 2009. Springer.
22. H. A. Reijers, J. Mendling, and R. M. Dijkman. On the Usefulness of Subprocesses in Business Process Models. BPM Center Report BPM-10-03, BPMcenter.org, 2010.
23. W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *ISJ*, 25(2):117–134, 2000.
24. M. Schrefl and M. Stumptner. Behavior-Consistent Specialization of Object Life Cycles. *ACM TOSEM*, 11(1):92–148, 2002.
25. S. Smirnov, R. Dijkman, J. Mendling, and M. Weske. Meronymy-based Aggregation of Activities in Business Process Models. In *ER 2010*, LNCS. Springer, 2010.
26. S. Smirnov, H. Reijers, Th. Nugteren, and M. Weske. Business Process Model Abstraction: Theory and Practice. Technical report, Hasso Plattner Institute, 2010. `http://bpt.hpi.uni-potsdam.de/pub/Public/SergeySmirnov/abstractionUseCases.pdf`.
27. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. *LNCS*, 3649:205–219, 2005.
28. J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. In *BPM 2008*, pages 100–115, Milan, Italy, 2008. Springer.
29. M. Weidlich, A. Barros, J. Mendling, and M. Weske. Vertical Alignment of Process Models - How Can We Get There? In *BPMDS 2009*, volume 29 of *LNBIP*, pages 71–84. Springer, 2009.
30. M. Weidlich, J. Mendling, and M. Weske. Efficient Consistency Measurement based on Behavioural Profiles of Process Models. *IEEE TSE*, 2010. to appear.
31. M. Th. Wynn, H. M. W. Verbeek, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Reduction Rules for YAWL Workflows with Cancellation Regions and OR-joins. *IST*, 51(6):1010–1020, 2009.