Efficient Process Conformance Checking on the Basis of Uncertain Event-to-Activity Mappings

Han van der Aa, Henrik Leopold, Hajo A. Reijers

Abstract—Conformance checking enables organizations to automatically identify compliance violations based on the analysis of observed event data. A crucial requirement for conformance-checking techniques is that observed events can be mapped to normative process models used to specify allowed behavior. Without a mapping, it is not possible to determine if an observed event trace conforms to the specification or not. A considerable problem in this regard is that establishing a mapping between events and process model activities is an inherently uncertain task. Since the use of a particular mapping directly influences the conformance of an event trace to a specification, this uncertainty represents a major issue for conformance checking. To overcome this issue, we introduce a probabilistic conformance-checking technique that can deal with uncertain mappings. Our technique avoids the need to select a single mapping by taking the entire spectrum of possible mappings into account. A quantitative evaluation demonstrates that our technique can be applied on a considerable number of real-world processes where existing conformance-checking techniques fail.

Index Terms—Business process management, Business Process Monitoring

1 Introduction

In many organizational contexts, employees are required to execute tasks of business processes in conformance with certain rules. For example, employees of a bank must check the credit history of a customer before granting a loan or the ground staff at an airport must verify the identity of a flight passenger before checking in the passenger's luggage. So-called *conformance-checking techniques* play an important role in evaluating such rules [1]. They compare the actual behavior of employees, as recorded by information systems, to allowed process behavior that is specified in a normative *process model* [2]. In this way, conformance-checking techniques can automatically identify non-compliant actions and prevent potentially negative consequences such as delays in the process execution or fines imposed by authorities.

A fundamental requirement for conformance checking is that observed process behavior, commonly represented in the form of an *event log*, can be related to the normative process specification. This means that events in an event log must be mapped to the activities of a process model [3]. Without knowing the relations between events and activities, it is not possible to determine if observed behavior conforms to the allowed behavior specified by the process model, which makes conformance checking impossible. Typically, however, such a so-called *event-to-activity mapping* is not readily available [4], [5]. Furthermore, *manually* obtaining a mapping is often unfeasible because business analysts rarely possess the necessary knowledge on the details of a process implementation [6], whereas *automated* mapping techniques suffer from the high uncertainty caused by cryptic event names, non-compliant behavior, and noisy data [7]. As a result, existing mapping techniques (cf. [4], [8], [9]) fail to provide a definite solution to the mapping problem. Instead, they represent probabilistic methods that aim to select the *best* mapping from a number of potential ones, rather than providing a deterministic solution [10]. This comes with the considerable risk that the selected mapping does not capture the true relations between the events and activities. This can be particularly harmful in the context of conformance checking: if the selected mapping is incorrect, conformance-checking results based on this mapping will be incorrect as well.

Recognizing that the problem of mapping uncertainty compromises the successful application of conformance checking in practice, we use this paper to follow up on a proposal for a conformance-checking technique that can be applied in spite of mapping uncertainty [11]. The core idea of our technique is to take the entire spectrum of potential event-to-activity mappings into account and store them in a so-called *behavioral space*. In this way, we capture the implications of the different mappings in a structured way. Conformance checks based on this behavioral space, therefore, allow us to provide trustworthy conformance-checking results that take all potential mappings into account. Furthermore, by building on a hierarchical decomposition of process models, we are also able to compute conformance-checking results in a more efficient manner.

In comparison to [11], this paper provides several novel contributions. First, while the approach from [11] only provided conformance insights at a *process level*, the approach presented here is able to identify the specific parts of a process where conformance issues arise. As a result, the conformance-checking results are much more detailed and, furthermore, provide the foundation for process improvement endeavors. Second, we introduce a novel approach to efficiently perform conformance checks. Whereas the original approach required a full enumeration of all *possible worlds* implied by mapping uncertainty, we present

Han van der Aa is with the the Humboldt-Universität zu Berlin, Germany (han.van.der.aa@hu-berlin.de).
Henrik Leopold is with the Kühne Logistics University in Hamburg, Germany, and the Hasso Plattner Institute, University of Potsdam, Germany (henrik.leopold@the-klu.org)
Hajo A. Reijers is with Utrecht University and the Eindhoven University of Technology, The Netherlands
(h.a.reijers@uu.nl)

a computation approach that improves the computational efficiency of the approach by up to 70%. Third and finally, this paper presents the results of various additional evaluation experiments that demonstrate the applicability of our approach to a variety of process model types.

The remainder of the paper is structured as follows. Section 2 illustrates the problem of mapping uncertainty using an example. Section 3 introduces basic definitions. Section 4 presents the conceptual basis for our conformance-checking technique using behavioral spaces. Section 5 discusses how to efficiently obtain conformance-checking results using our approach. Section 6 presents the evaluation of our approach with a set of 598 real-world and 650 synthetic process models. Finally, Section 7 elaborates on related work, before Section 8 concludes the paper.

2 Problem Illustration

In this section, we illustrate the problem of mapping uncertainty in the context of conformance checking. The starting point is an *event log* and a normative *process model*. The event log consists of a number of *event traces*, each referring to a single process execution. Since these event traces have been recorded by an information system during the process execution, the event log corresponds to the *actual* behavior. The normative process model, by contrast, specifies the *allowed* behavior of the process.

A crucial prerequisite for conformance checking is that the events from the event log can be related to the activities of the process model. To illustrate this requirement, consider the process model M and an event log L from [12], corresponding to a real-world order handling process, depicted in Figure 1 and Table 1, respectively. The order handling process starts when an order is received (A). An order can be changed an arbitrary number of times (B), before it is processed (C). Afterwards, the products are shipped (D) and, in the meantime, an invoice will be sent (E), followed by either a digital (F) or physical notification (G) to the customer. Once all the previous activities have been completed, the order is archived (H), which completes the process. Looking at the traces in the event log L, we realize that there is only limited information that allows for us to compare the traces in L to the activities in model M. Without knowing that, for instance an event with the label L_SM corresponds to the activity "*Send notification e-mail*", it is impossible to understand which activities have occurred in reality and, thus, whether or not traces from L conform to M. Therefore, a so-called *event-to-activity mapping* is required that clearly specifies the relations between the events from the log and the activities of the process model.

At its heart, this problem of identifying event-to-activity mappings represents an *alignment problem*, which is often also referred to as a *matching problem*. In fact, the alignment and integration of different representations of reality is a long-standing problem in computer science. In particular, it is the core problem in data integration [13]. However, the problem of identifying matching data elements has also been recognized in other domains. For example, there are various techniques for matching ontologies [14], [15], process models [16], [17], and also event data [18], [19]. One of the central challenges of alignment problems is to identify relations with certainty. In the context of event-to-activity mappings, factors such as cryptic event names (e.g., *CDHDR*, *L_SM*), non-compliant behavior, and noisy data [7] lead to a lack of reliable information that can be utilized to accurately identify event-activity pairs. As a result, even the state-of-the-art mapping technique by Baier et al. reports a mapping accuracy of only 50.8% on real-world data [4], which demonstrates that the mapping problem is highly uncertain. As a result of this *mapping uncertainty*, event-to-activity mapping techniques rather construct a number of *potential* mappings without being able to determine with certainty which mapping is correct.

TABLE 1
Event log L corresponding to process model M (from [12])

Trace ID	Label sequence
τ_a	⟨O_CHK, O_PRC, L_SM, P_SP, O_ARC ⟩
τ_b	⟨O_CHK, O_RCO, O_CHK, O_PRC, P_SP, P_NOT, L_SM, O_ARC ⟩
τ_c	⟨O_CHK, O_PRC, P_SP, P_NOT, L_SM, O_ARC ⟩
τ_d	⟨O_CHK, O_PRC, L_SM, P_NOT, P_SP, O_ARC ⟩
τ_e	⟨O_CHK, O_PRC, P_SP, L_SM, P_NOT, O_ARC ⟩

The unique issue in the context of event-to-activity mappings is that existing conformance-checking techniques can only be performed on a single, certain event-to-activity mapping. Therefore, in case of mapping uncertainty, these techniques require the selection of a single mapping from the set of potential mappings. This, however, comes with the considerable risk that the selected mapping is incorrect and that, consequently, conformance-checking results based on the selected mapping are incorrect as well.

To illustrate the risks of selecting a single, possibly incorrect, mapping, consider the trace τ_d from log L in Table 1 and the process model M. By comparing the behavioral relations from the traces in L to the behavior of model M, behavior-based mapping techniques such as [12], [20] can identify two potential mapping relations for the trace τ_d, which differ in their mapping of the P_NOT and P_SP events . In this case, one mapping (referred to as $\sim_1$) will lead to the corresponding activity sequence[1] $\sigma_1 = \langle a, c, d, e, f, h\rangle$ and the other mapping (i.e., $\sim_2$) to $\sigma_2 = \langle a, c, d, f, e, h\rangle$. The task sequence σ_1 conforms to model M, whereas σ_2 does not, because in this sequence, a notification e-mail is sent (event f) *before* the invoice (event e), instead of *after*. Since the conformance of these two activity sequences differs for this scenario, the assessment of whether or not τ_d conforms to the process model M depends on the selection of a mapping relation. If $\sim_1$ is chosen, conformance-checking techniques will determine that τ_d conforms to M, whereas the opposite holds if $\sim_2$ is selected. As a result, the conformance of τ_d fully depends on the ability to select a correct mapping in a situation where it is inherently uncertain what that mapping is.

This example illustrates that conformance-checking results based on the selection of a single, potentially incorrect mapping are not trustworthy. To provide a compre-

1. Note that we use lowercase letters to refer to execution instances of an activity, i.e. a lowercase a denotes an instance of activity A being executed.

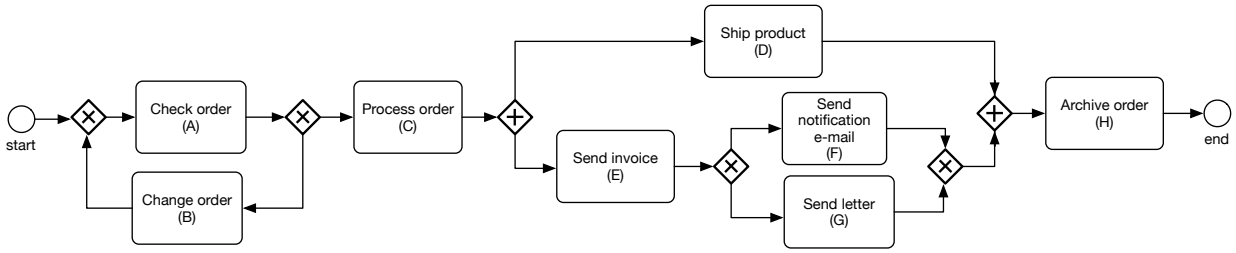


Fig. 1. Process model M of an order handling process (from [12])

hensive solution to this problem, this paper introduces a conformance-checking technique that eliminates the need to select a single, possibly incorrect mapping and circumvents one of the central issues of alignment problems.

3 Preliminaries

The inputs for conformance checking are a process model and an event log. In the following we introduce formal notions for both. Definition 1 first introduces a formalization of a process model, adapted from [21]. It captures the commonalities of widely-used process modeling languages such as the Business Process Model and Notation (BPMN) and Event-driven Process Chains (EPCs). The execution semantics of such a process model are given by a translation into a Petri net following common formalizations, cf. [22], [23]. The process model depicted in Figure 1 corresponds to this formal notion.

Definition 1 (Process Model) A process model is a tuple $M = (A_M, E, G, N, F, t)$, where

- A_M is a finite set of activities,
- E is a finite set of events,
- G is a finite set of gateways,
- $N = A_M \cup E \cup G$ is a finite set of nodes,
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph,
- $t : G \to \{and, xor\}$ is a mapping that associates each gateway with a type.[2]

In this paper, we decompose process models into single-exit single-entry (SESE) fragments in order to perform conformance checks. A SESE fragment refers to any part of a process model with exactly one *entry* and one *exit* node. The theoretical notions of SESE fragments in the context of directed graphs were developed by Hopcroft and Tarjan [25]. Vanhatalo et al. [26] and Polyvyanyy et al. [27] defined how a respective SESE decomposition can be obtained for process models. Definition 2 formally describes the notion of a SESE fragment we build on in this paper.

Definition 2 (SESE Fragment) *Given a process model M with its set of nodes N and its flow relation F, a SESE (single-entry single-exit) fragment S has exactly one entry node and one exit node. A SESE fragment is* trivial *if it is composed of a single flow relation. A SESE fragment S is* canonical *if it does not partially overlap with any other SESE fragment S' that can be derived from M, i.e. S and S' are either nested ($S \subseteq S' \vee S' \subseteq S$) or they are disjoint ($S \cap S' = \emptyset$).*

2. Please note that we disregard inclusive OR-gateways because of their marginal relevance in industrial process models [24].

In the remainder of this paper, we only consider so-called *valid decompositions* of process models [28], which is a decomposition in which fragments only share activities or gateways. Any SESE decomposition can be turned into a valid decomposition by using the technique presented in [29]. Figure 2a provides a decomposition for the running example into a root fragment S_0 and 9 smaller fragments. This figure also highlights the hierarchy that exists between the SESE fragments in a decomposition. The so-called *Refined Process Structure Tree* (RPST) can be used to capture this hierarchy between canonical SESE fragments, as visualized in Figure 2b, and defined as follows.

Definition 3 (RPST) *Let M be a process model. The Refined Process Structure Tree (RPST) of M is the tree composed of the set of all its canonical SESE fragments such that the parent of a canonical SESE fragment S is the smallest canonical SESE fragment that contains S. The root of the tree is the entire process model, and the leaves are the trivial SESE fragments. The set of all the nodes of the tree is denoted as $\mathcal{S}$.*

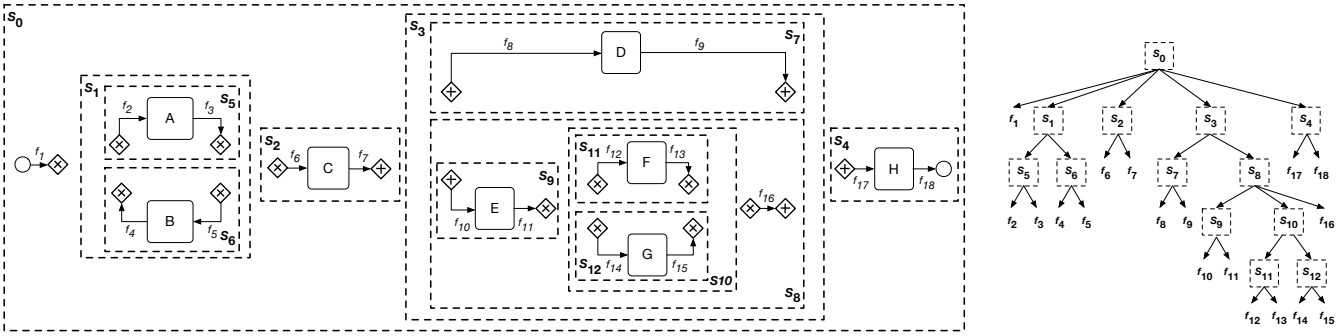
In the remainder of the paper, we will refer to canonical SESE fragments resulting from the RPST decomposition simply as SESE fragments. Also note that the SESE fragments are defined as a set of flow relations. For simplicity, however, we will use the term SESE fragment to also refer to the fragment of a process model that is induced by those relations. For instance, we shall use S_1 to denote the fragment containing the activities A and B, the *xor*-gateway following B, and the flow relations that connect these nodes.

Finally, we define event logs and event traces according to notions from [30]. Definition 4 formally defines an event log.

Definition 4 (Event Log) *Let $\mathcal{C}$ be a finite set of event classes. A log L is defined as $L = (\mathcal{E}, \mathcal{I}, E, I, <)$, where*

- *$\mathcal{E}$ is the set of events,*
- *$\mathcal{I}$ is the set of case identifiers,*
- *$E : \mathcal{E} \to \mathcal{I}$ a surjective function linking events to cases,*
- *$I : \mathcal{E} \to \mathcal{C}$ a surjective function linking events to event classes,*
- *$< \subseteq \mathcal{E} \times \mathcal{E}$ a strict total ordering over the events.*

Given an event log L according to Definition 4, we shall use the shorthand notation $\tau = \langle e_1, \ldots, e_n \rangle$ in the remainder of this paper to refer to an event trace that consists of n events with an identical case identifier. Furthermore, for



(a) SESE decomposition (b) RPST representation

Fig. 2. Decompositions of the process model for the order handling process

any pair of events e_i and e_j with $i < j$, it holds that $e_i < e_j$ according to the strict total ordering of the events in log L.

4 Conformance Checking using Behavioral Spaces

This section describes the conceptual basis of our conformance-checking technique. It takes as input an event trace, a process model, and an uncertain event-to-activity mapping. Note that the question of how to obtain an uncertain mapping, which consists of a number of potential event-to-activity mappings, is not the focus of this paper. Potential mappings can be obtained using one or more mapping techniques, such as [4], [8], [9]. In the remainder, Section 4.1 first describes the notion of a behavioral space, which we use to capture the impact of mapping uncertainty on the process behavior described by trace τ. Then, Section 4.2 introduces the conformance-checking metrics that build on the obtained behavioral spaces. Finally, Section 4.3 discusses the diagnostic conformance-checking information that can be obtained for processes with our technique.

4.1 Capturing Mapping Uncertainty using Behavioral Spaces

Mapping uncertainty results from multiple views on which behavior, in terms of process model activities, is described by a single event trace. This uncertainty manifests itself through the existence of multiple possible event-to-activity mappings. A single event-to-activity mapping captures relations between events in an event trace τ and the activities in a process model M, as defined in Definition 5.

Definition 5 (Event-to-Activity Mapping) *Let $\tau = \langle e_1, \ldots, e_n \rangle$ be an event trace with a set of events E_τ and $M = (A_M, E, G, N, F, t)$ a process model. An event-to-activity mapping is a surjective relation $\sim \subseteq E_\tau \times A_M$. Elements of the relations are referred to as correspondences, where a correspondence $e \sim a \in (E_\tau \times A_M)$ denotes a mapping relation between an event e and an activity a.*

In Definition 5, the relation $\sim$ is defined as *surjective*, because we assume that each event in a trace τ is always mapped to an activity. Furthermore, this implies that a N:1 relation can exist between events and activities. This cardinality captures the notion that events are typically more fine granular than activities [31]. As an illustration, consider a trace $\tau = \langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle$ and a mapping $\{e_1 \sim a, e_2 \sim c, e_3 \sim c, e_4 \sim d, e_5 \sim e, e_6 \sim f\}$. Given that both e_2 and e_3 are aligned to activity c, this mapping indicates that the trace τ, consisting of six events, corresponds to a sequence of only five activities: $\langle a, c, d, e, f \rangle$.

Mapping uncertainty leads to the existence of multiple potential event-to-activity mappings. Here, we capture this spectrum in the form of an *uncertain event-to-activity mapping* $\mathbb{EA}(\tau, M)$, as defined in Definition 6.

Definition 6 (Uncertain Event-to-Activity Mapping) *Let $\tau = \langle e_1, \ldots, e_n \rangle$ be an event trace and M a process model with an activity set A_M. An uncertain event-to-activity mapping is a tuple $\mathbb{EA}(\tau, M) = (\mathbb{M}, \phi)$, with:*

- *$\mathbb{M}$: a set of event-to-activity mappings between τ and M;*
- *$\phi : \mathbb{M} \to [0, 1]$: a function that assigns a probability to each event-to-activity mapping $\mathcal{M}(\tau, M) \in \mathbb{M}$. For this function, it holds that the cumulative probability is equal to 1, i.e., $\sum_{\mathcal{M} \in \mathbb{M}} \phi(\mathcal{M}) = 1$.*

In this definition, each mapping $\mathcal{M}(\tau, M) \in \mathbb{M}$ represents a potential way to map the events in τ to the activities in A_M. The probability function ϕ assigns a probability p_i to each mapping $\mathcal{M}(\tau, M) \in \mathbb{M}$. These probabilities generally follow from the confidence of an event-to-activity mapping technique. For instance, a technique based on semantic similarity scores, such as [4], can quantify the probability as the product of the similarity scores associated with each correspondence in the mapping. In this way, mappings with a higher semantic similarity receive a higher probability than the ones with a lower score. If no probabilities are available, the most straightforward solution is to assign an equal probability $p_= \frac{1}{|\mathbb{M}|}$ to each mapping.

Given such an uncertain event-to-activity mapping $\mathbb{EA}(\tau, M)$, we define the notion of a probabilistic behavioral space as a means to capture all process model behavior conveyed by trace τ according to the mapping $\mathbb{EA}(\tau, M)$, i.e. the sequences of process model activities that follow from the different possible mappings. We shall refer to such a sequence of process model activities as a *trace translation* of event trace τ, because it represents a translation of the trace's events into process model activities. We denote a trace translation of τ with $\sigma(\tau)$.

Definition 7 (Trace Translation) *Let $\tau = \langle e_1, \ldots, e_n \rangle$ be an event trace with a set of events E_τ, $M = (A_M, E, G, N, F, t)$ a process model, and $\mathcal{M}(\tau, M)$ an event-to-activity mapping*

between τ and the activity set of process model M. A trace translation $\sigma(\tau)$ represents a sequence of activities $\langle a_1, \ldots, a_m \rangle$ according to the mapping $\mathcal{M}(\tau, M)$.

Since an uncertain mapping $\mathbb{EA}(\tau, M)$ consists of multiple event-to-activity mappings, a mapping $\mathbb{EA}(\tau, M)$ results in different trace translations for τ. For instance in Section 2, we described an example where a single trace had two trace translations, $\sigma_1(\tau) = \langle a, c, d, e, f, h \rangle$ and $\sigma_2(\tau) = \langle a, c, d, f, e, h \rangle$, which resulted from two possible mappings. Together, the translations of a trace represent the spectrum of process behavior potentially conveyed by τ, i.e. the behavioral space of an event trace. Since each mapping can be associated with a probability, we include a probabilistic component in our definition of a behavioral space, as captured in Definition 8.

Definition 8 (Probabilistic Behavioral Space) *Let $\tau = \langle e_1, \ldots, e_n \rangle$ be an event trace with a set of events E_τ, $M = (A_M, E, G, N, F, t)$ a process model, and $\mathbb{EA}(\tau, M)$ an uncertain event-to-activity mapping between τ and the activity set of process model M. We define a* probabilistic behavioral space *as a tuple $PBS_\tau = (\Sigma(\tau), \phi)$, with:*

- *$\Sigma(\tau)$: the set of* trace translations *of trace τ over the activity set A as given by the event-to-activity mappings in $\mathbb{EA}(\tau, M)$;*
- *$\phi : \Sigma_\tau \rightarrow [0, 1]$: a function that assigns a probability to each trace translation in $\Sigma(\tau)$. For this function, it holds that the cumulative probability is equal to 1, i.e., $\sum_{\sigma \in \Sigma_t} \phi(\sigma) = 1$.*

The set $\Sigma(\tau)$ comprises potential trace translations of trace τ over the activity set A_M, where each translation $\sigma_i \in \Sigma(\tau)$ is based on a mapping $\mathcal{M}(\tau, M)$ contained in $\mathbb{EA}(\tau, M)$. This set, together with the probabilities provided by the function ϕ, provides the basis for the probabilistic conformance metric described next.

4.2 Probabilistic Conformance

The goal of conformance checking is to determine if observed behavior in a trace τ is allowed by the behavioral specification of a process model M. Since uncertain event-to-activity mappings lead to multiple views on the process model behavior described by a trace (i.e. its trace translations), all these views need to be checked against the model M. In this section, we demonstrate how to perform a conformance check given a probabilistic behavioral space in order to obtain insightful conformance results and diagnostic information.

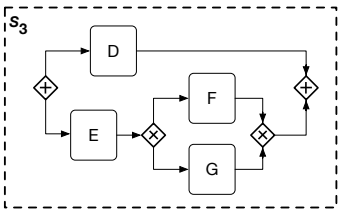
To perform this conformance checks, we build on the approach for decomposed conformance checking defined by Munoz-Gama et al. [29]. It splits up a process model into a set of canonical SESE fragments $\mathcal{S}$, as described and depicted in Section 3, and then determines for each fragment in $\mathcal{S}$ whether it conforms to a given activity sequence or not. We use this approach as the basis for our conformance checks of an entire behavioral space against a model. We kindly refer the interested reader to [29] for an in-depth explanation of the conformance checks between a single activity sequence and a SESE fragment and proofs of the guarantees it provides regarding the correctness of decomposed conformance-checking results.

Building on the approach from [29], we introduce a probabilistic conformance metric that quantifies the conformance of a probabilistic behavioral space to a SESE fragment. The metric combines the conformance assessments for individual trace translations with probabilistic information. Specifically, the metric corresponds to the total probability associated with the trace translations that conform to a certain SESE fragment $S \in \mathcal{S}$. In this way, the metric defined in Definition 9 represents the probability that τ conforms to fragment S.

Definition 9 (Behavioral Space Conformance) *Let τ be a trace with a probabilistic behavioral space $PBS(\tau) = (\Sigma(\tau), \phi)$, M a process model, and S a fragment of the SESE decomposition of M. Then we define:*

- *$\Sigma_S(\tau) \subseteq \Sigma(\tau)$ as the set of trace translations in $\Sigma(\tau)$ conforming to fragment S;*
- *$ProbConf(\tau, S) = \sum_{\sigma \in \Sigma_C(\tau)} \phi(\sigma)$: as the* behavioral space conformance *of trace τ to fragment S, where $\phi(\sigma)$ captures the probability of trace translation σ.*

Because of the probabilistic nature of the $ProbConf$ metric, the metric yields a different kind of result than traditional conformance-checking techniques. In traditional conformance-checking scenarios, i.e. without mapping uncertainty, a trace either conforms or does not conform to (a fragment of) a process model. By contrast, when using our technique, traces are either conforming, non-conforming, or *potentially conforming*. Potentially conforming traces are those traces for which some trace translations conform to a process model, whereas others do not. The conformance of these traces is associated with a certain probability $0 < p < 1$. Take, for instance, the process model fragment S_3 and a trace τ_1 with two trace translations $\sigma_1(\tau_1)$ and $\sigma_2(\tau_1)$, as depicted in Figure 3. Assume that $\sigma_1(\tau_1)$ is associated with a probability of 0.7 and $\sigma_2(\tau_1)$ with probability 0.3.



$\tau_1 = \langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle$

$\sigma_1(\tau_1) = \langle a, c, d, e, f, h \rangle$
$\sigma_2(\tau_1) = \langle a, c, d, e, f, g \rangle$

$\phi(\sigma_1(\tau_1)) = 0.7$
$\phi(\sigma_2(\tau_1)) = 0.3$

Fig. 3. Process model fragment S_3 and two trace translations

While trace translation $\sigma_1(\tau_1)$ conforms to S_3, this does not apply for $\sigma_2(\tau_1)$. This latter translation executes both of the mutually exclusive activities F and G. This leads to a conflict between the conformance of the different translations of τ_1 with respect to S_3. Therefore, the trace τ_1 is said to be potentially conforming with S_3 with a probability of 0.7, i.e. $ProbConf(\tau_1, S_3) = 0.7$. This shows that, even though we cannot make certain statements about the conformance of τ_1 to S_3, we do know that τ_1 is more likely to conform than not. Furthermore, we also know the mapping conditions under which τ_1 is conforming or non-conforming. Namely, τ_1 conforms to S_3 if the correspondence $e_6 \sim f$ holds, whereas the trace is non-conforming if $e_6 \sim g$ is true. This type of diagnostic information is very useful because it provides insights into which aspects of an uncertain

mapping lead to uncertainty in the conformance-checking results for observed behavior.

It is important to note that the $ProbConf$ metric, despite its probabilistic nature and the presence of mapping uncertainty, can often still produce non-probabilistic (or deterministic) conformance-checking results. To illustrate this, reconsider the process model fragment S_3, as well as a trace τ_2 with the following trace translations:

$$\sigma_1(\tau_2) = \langle a, c, d, e, f, h \rangle$$
$$\sigma_2(\tau_2) = \langle a, c, d, e, g, h \rangle$$

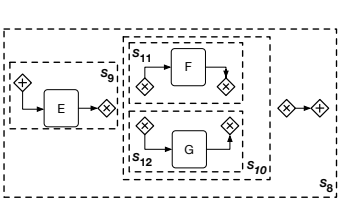
In this case, mapping uncertainty has resulted in two trace translations that differ with respect to their fifth activity: F for $\sigma_1(\tau_2)$ and G for $\sigma_2(\tau_2)$. Despite this uncertainty, we can still with certainty state that τ_2 conforms to S_3. The reason is that both trace translations conform to S_3, since S_3 allows for the execution of *either* activity F or G. As a result, $ProbConf(\tau_2, S_3) = 1.0$, thus yielding a deterministic result. In a similar fashion, we can determine for certain that some traces are non-conforming, despite having multiple translations.

4.3 Hierarchical Conformance Insights

Thus far we focused on conformance checking for a single process fragment using the $ProbConf$ metric. This metric can be applied to obtain conformance information for a process model as a whole, yielding a single $ProbConf$ value indicating the likelihood that a trace conforms to the entire process. However, an important benefit of decomposed conformance checking is that it can provide conformance results at various levels of detail. In particular, we can exploit the hierarchical relations between the SESE fragments of a decomposition in order to obtain hierarchical conformance-checking results.

Consider, for instance, the SESE fragments depicted in Figure 4. The fragments S_9 and S_{10} are subsumed by their parent, S_8, whereas, in turn, S_{10} subsumes fragments S_{11} and S_{12}. By applying $ProbConf$ on all fragments, we can obtain conformance-checking results at different levels of detail. Using the trace translations of τ_4 depicted in the same figure, we can observe some interesting properties. For instance, even though the conformance of the fragments S_{11} and S_{12} are both equal to 1.0, the conformance of their parent fragment, i.e., $ProbConf(\tau_4, S_{10})$, is only 0.1. These results reveal that the conformance problems for this trace do not relate to either S_{11} or S_{12} individually, but rather to their inter-relation. This example illustrates the usefulness of decomposed conformance checking in combination with behavioral spaces because this diagnosis could only be observed by considering both levels of detail.

We can expand this hierarchical view on conformance checking to consider the entire hierarchical structure of SESE fragments in a decomposition. For this, we can adapt the RPST representation of a decomposition, which we provided in Figure 2b, to incorporate conformance-checking results obtained by the $ProbConf$ metric. Figure 5 illustrates this for the running example with conformance-checking results based on trace τ_4. This figure clearly shows how the probabilistic conformance of a trace can differ among



$$\sigma_1(\tau_4) = \langle a, c, d, f, g, h \rangle$$
$$\sigma_2(\tau_4) = \langle a, c, d, e, f, g \rangle$$
$$\sigma_3(\tau_4) = \langle a, c, d, e, f, h \rangle$$

$$\phi(\sigma_1(\tau_4)) = 0.4$$
$$\phi(\sigma_2(\tau_4)) = 0.5$$
$$\phi(\sigma_3(\tau_4)) = 0.1$$

Fig. 4. Process model fragment S_8 with subsumed fragments

the various fragments and levels of detail. While the entire process model has a probability of only 0.1 to be conforming (following from the only fully conforming trace translation $\sigma_3(\tau_4)$), the figure clearly shows that this low probability largely results from problems in the latter part of the process, related to the fragment S_8.

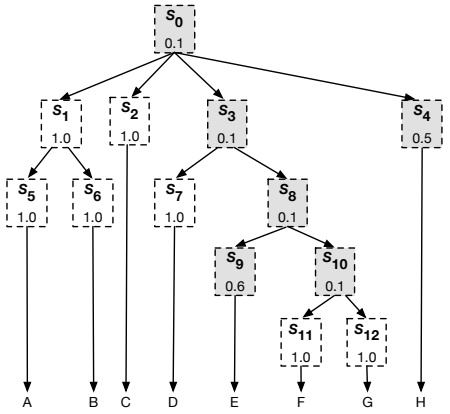


Fig. 5. Hierarchical conformance results for τ_4

Despite these apparent differences between probabilistic conformance values, the relation between the likelihoods of a fragment and its sub-fragments defines a clear bound for conformance values. Namely, the $ProbConf$ value of a fragment S_i with sub-fragments $\mathcal{S}_{S_i} = \{S_1^i, \ldots, S_n^i\}$ can never be higher than the minimum of its child-fragments, i.e. $ProbConf(\tau, S_i) \leq \min\{ProbConf(\tau, S_1^i), \ldots, ProbConf(\tau, S_n^i\}$. Intuitively, this means that a child-fragment can never be less conforming than its parent.

By obtaining conformance insights for all process model fragments at various levels of granularity, our conformance-checking technique provides the foundation for root-cause analysis of conformance problems. The conformance levels of traces to particular fragments reveals which parts of a process can be considered most problematic. As such, these insights can be used in order to take measures to avoid these problems in the future, thereby improving the process. In the next section, we describe how the concepts introduced in this section can be used to obtain the conformance-checking results in an efficient manner.

5 Efficient Conformance Checking

In this section, we define an approach that can be used to efficiently compute the conformance-checking results described in the previous section. The ability to obtain these results efficiently is important, because the computational

complexity of conformance-checking techniques represents a key issue for their applicability in industrial settings [32]. In the context of this paper, this complexity is particularly problematic, since mapping uncertainty can exponentially increase the number of conformance checks that are required to be performed per trace. For instance, if traces are associated with 10 trace translations, the execution time could be an order of magnitude larger than in a situation without mapping uncertainty. However, because mapping uncertainty may only relate to specific parts of a process, it is not always necessary to recompute the conformance of all process model fragments for every trace translation. By recognizing this, we can define a technique that obtains conformance-checking results in a considerably more efficient manner.

5.1 Approach Overview

Our computation approach builds on the idea that mapping uncertainty often only relates to specific parts of a process. For example, in the running example, mapping uncertainty might affect the parts of the process related to billing (i.e. activities E, F, and G), whereas other parts of the process, related to order processing and shipment, are not affected by uncertainty. By utilizing this knowledge, we define an efficient conformance-checking technique that only performs repeated checks (for different trace translations) for those parts of a process affected by mapping uncertainty. For other parts, a single check suffices to determine the conformance of all trace translations in a behavioral space.

We achieve this through the approach visualized in Figure 6. The approach takes as input an event trace τ, a process model M, and an uncertain event-to-activity mapping $\mathbb{EA}(\tau, M)$. In our approach we first decompose a process model according to the method from [27], briefly described in Section 3. Furthermore, we compute a behavioral space PBS_τ by generating a single trace translation $\sigma(\tau)$ for each mapping in $\mathbb{EA}(\tau, M)$, according to the method described in Section 4.1.

After these preliminary steps, we conduct three further steps that enable us to conduct conformance checks in a more efficient manner. Based on the established behavioral space, our approach identifies those activities that are associated with mapping uncertainty. Then, we determine which process fragments of the decomposed process model are affected by uncertainty. Finally, we perform an efficient conformance check by only recomputing conformance values for those process model fragments that are actually affected by mapping uncertainty. In the remainder of this section, we describe each of these latter three steps in detail.

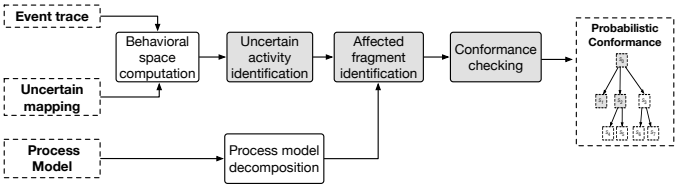


Fig. 6. Overview of our computation approach

5.2 Uncertain Activity Identification

In the first step of our approach, we determine the activities affected by mapping uncertainty. Algorithm 1 describes this step. The method used to identify activities affected by mapping uncertainty depends on whether the event-to-activity mapping was established using a class-level or instance-level mapping technique. Class-level techniques, such as [7], [12], establish a mapping between event *classes* and process model activities. As a result, any occurrence of an event of a certain class will be mapped to the same activity or, in case of uncertainty, the same activities. By contrast, instance-level techniques, such as [4], may map events from the same event class to different activities in different traces. In Algorithm 1, we use `isClassLevel()` on line 4 and `isInstanceLevel()` on line 11 to distinguish among these two options.

Algorithm 1 Identifying activities affected by mapping uncertainty

```
1:  function IdentifyUncertainActivities
2:      Input: PBS_τ = (Σ(τ), φ)            ▷ Behavioral space of τ
3:      Input: 𝔼𝔸(τ, M)                      ▷ Uncertain mapping
4:      if isClassLevel(𝔼𝔸(τ, M)) then
5:          U_M = ∅
6:          for E ∈ C do
7:              A_E = getMappedActivities(E, 𝔼𝔸(τ, M))
8:              if |A_E| > 1 then
9:                  U_M = U_M ∪ A_E
10:         U_τ = U_M ∩ A_τ
11:     else if isInstanceLevel(𝔼𝔸(τ, M)) then
12:         U_τ = ∅
13:         for σ1(τ) ∈ Σ_τ do
14:             for σ2(τ) ∈ Σ_τ do
15:                 δ = String.diff(σ1(τ), σ2(τ))
16:                 U_τ = U_τ ∪ δ
17:     return U_τ
```

A class-level mapping directly provides insights into the activities that differ among the trace translations. In such a scenario, we can, for instance, observe that an event class $E \in \mathcal{C}$ is always mapped to either activity B or activity C. From this, we know that activities B and C are affected by uncertainty. By identifying the activities associated with all event classes subject to mapping uncertainty, see lines 5–9 in Algorithm 1, we then obtain a set $\mathcal{U}_M \subset A_M$ of uncertain activities for the process model M. For a specific behavioral space PBS_τ, the set of uncertain activities $\mathcal{U}_\tau$ is then given as the subset of $\mathcal{U}_\tau$ that is contained in any of the trace translations in BS_τ, i.e. using A_τ to denote the set of activities contained in the behavioral space of trace τ, we get $\mathcal{U}_\tau = \mathcal{U}_M \cap A_\tau$, as described in line 10.

Computing the differences for mappings at the instance-level is more complex. Specifically, this requires a comparison of the trace translations with each other. By abstracting from process model activities to a set of symbols of an alphabet $\mathcal{A}$ (as we have done throughout this work), each trace translation represents a sequence of characters, i.e. a string, from $\mathcal{A}$, i.e. $\sigma(t) = \langle a, b, c, d \rangle$ equals the string $abcd$. Given such character sequences, we can employ a string-difference algorithm by Myers [33] to efficiently compute the difference between two character sequences, which find the difference between two strings by identifying the shortest path in an edit graph (line 15).

For instance, sequences $abde$ and $acde$ yield the difference $\delta = \{B, C\}$. The union of all differences that exist among trace translations in Σ_τ then represents the set of activities affected by mapping uncertainty (lines 15–16).

5.3 Affected Fragment Identification

Given a set $\mathcal{U}_\tau$ of activities affected by uncertainty, the next step is to determine to which SESE fragments of a decomposed process model these activities relate. To do this, we check which fragments do not contain any activities from the set U_τ. Formally, we define the set of *certain* (i.e. fragments unaffected by uncertainty) fragments as $\mathcal{S}_c(\tau) = \{S \in \mathcal{S} \mid T(S) \cap U_\tau = \emptyset\}$. The set $\mathcal{S}_c(\tau)$ can be efficiently computed in a top-down manner, as illustrated in Algorithm 2. Given a fragment S in the RPST, we first determine if the fragment does not contain any activity in the set of uncertain activities $\mathcal{U}_\tau$ (line 4). If this is the case, then we know that S is not affected by uncertainty and should be included in $\mathcal{S}_c(\tau)$, but we also know that none of the descendants (i.e. elements below S in the RPST) contains any activities from $\mathcal{U}_\tau$. Therefore, also these descendants are included in $\mathcal{S}_c(\tau)$ (line 5). In case S does contain activities from $\mathcal{U}_\tau$, the algorithm recursively moves to the child-nodes of S in lines 8–10.

Algorithm 2 Identifying fragments affected by mapping uncertainty

```
1:  function IDENTIFYAFFECTEDFRAGMENTS
2:      Input: U_τ                          ▷ Result of Algorithm 1
3:      Input: S                            ▷ SESE fragment as RPST node
4:      S_c(τ) = ∅
5:      if A_S ∩ U_τ = ∅ then
6:          S_c(τ) = {S} ∪ S.getDescendants()
7:      else
8:          for child ∈ S.getChildren() do
9:              S'_c(τ) = IdentifyAffectedFragments(U_τ, child)
10:             S_c(τ) = S_c(τ) ∪ S'_c(τ)
11:     return S_c(τ)
```

For example, given a set of uncertain activities $\mathcal{U}_\tau = \{E, F\}$, we obtain the set of unaffected fragments $\mathcal{S}_c(\tau) = \{S_1, S_2, S_4, S_5, S_6, S_7, S_{12}\}$. This set does not include the fragments S_9 and S_{11}, which, respectively, encompass activities E and F and it neither includes any of their super-fragments (S_{10}, S_8, S_3, and S_0). The complement of the set $\mathcal{S}_c(\tau)$, which we denote as $\mathcal{S}_u(\tau)$, contains all fragments that are affected by uncertainty. We take these two sets as input to the final step of our approach.

5.4 Conformance Checking

After determining which fragments of a process model are affected by mapping uncertainty, we can perform the necessary conformance checks. Here, we apply the notion that for fragments *not* affected by uncertainty, i.e. the set $\mathcal{S}_c(\tau)$, we only have to check the conformance of a single trace translation $\sigma(\tau) \in \Sigma(\tau)$ in order to determine the conformance of all translations in $\Sigma(\tau)$. Because these fragments are not affected by uncertainty, all translations are either conforming or all are non-conforming to fragments in $\mathcal{S}_c(\tau)$. As described in Section 4.2, we perform the conformance check between a single trace translation and a fragment by employing the method from [29]. Naturally, when all trace translations have the same conformance, the $ProbConf$ metric will yield a value of either 0.0 or 1.0 for the fragments in $\mathcal{S}_c(\tau)$.

For fragments that are affected by mapping uncertainty, in the set $\mathcal{S}_u(\tau)$, we need to determine the conformance of all trace translations in order to obtain a correct $ProbConf$ value. Therefore, we determine the conformance of all trace translations in $\Sigma(\tau)$ with respect to a fragment S and then compute the probabilistic conformance as $ProbConf(\tau, S) = \sum_{\sigma \in \Sigma(\tau)} \phi(\sigma)$. After computing $ProbConf(\tau, S)$ for all fragments in $\mathcal{S}_c(\tau)$ and $\mathcal{S}_u(\tau)$, we have obtained the necessary results. These results can, for instance, be visualized in the manner depicted in Figure 5. These results represent the final output of our conformance-checking technique.

6 Evaluation

This section presents a two-stage evaluation in which we assess the usefulness of behavioral space-based conformance checking and the efficiency of our computation approach. In the first part of the evaluation, we assess how the impact of mapping uncertainty on conformance checking can be reduced by using behavioral spaces as a basis. In particular, we observe for how many traces our technique is able to provide non-probabilistic conformance-checking results as compared to traditional conformance-checking techniques. In the second part of the evaluation, we analyze the computational efficiency of our computation approach and compare it to a benchmark.

In the remainder of this section, Section 6.1 first introduces the test collections used for the evaluation experiments. Section 6.2 presents the first part of the evaluation, focusing on the usefulness of our proposed technique. Section 6.3 presents the second part, focusing on the efficiency of our computation approach. Finally, Section 6.4 discusses limitations of the conformance-checking technique and its evaluation.

6.1 Test Collections

To perform the evaluation, we combine a collection of real-world business process models with a collection of synthetic, generated models.

Real-world models. As a real-world collection, we employ the BIT process library, which comprises 886 process models that were created in process automation projects in various industry domains, such as financial services, automotive, telecommunications, construction, supply chain, health care, and customer relationship management [34]. We picked this process model collection because it has already been used in a variety of related evaluations, among others to test several event-to-activity mapping approaches [7], [12]. Hence, we believe that results obtained by using this collection present a realistic view on the applicability of the event-to-activity mapping approach against which we compare our conformance-checking technique. Following the same filtering steps used in [7], we omitted any process models with soundness issues such as *deadlocks* or *livelocks*. Furthermore, we omitted six models for which the employed event-to-activity mapping approach was not able to

TABLE 2
Characteristics of the test collections

Models with	Real-world	Synthetic	Total
Loops	63	63	126
Skips	0	139	139
Non-free choice	0	104	104
Total	**598**	**650**	**1248**

	Real-world		Synthetic		Total
Node Type	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**
Places	8.8	34	17.7	96	13.4
Transitions	7.4	40	17.7	100	12.8
And-splits	0.7	5	4.3	62	2.6
Xor-splits	0.8	12	4.5	33	2.6
Silent steps	2.2	29	6.1	76	2.7
Skips	0	0	0.2	4	0.1

produce a mapping due to memory shortage. As a result of the filtering, a collection of 598 process models remains available for usage in our evaluation.

Synthetic models. We have generated a collection of 650 synthetic models that allows us to assess the impact of model characteristics such as loops, arbitrary skips, and non-free choice constructs on the performance of our approach. We employed the state-of-the-art process model generation technique from [35] due to its ability to also generate non-structured models, e.g., models with non-free choice constructs [36]. For the generation, we employed the default parameters used by the developers of the approach in their evaluation.

As shown in Table 2, the test collections contain models that differ considerably in their size, complexity, and characteristics. For this reason, as well as due to the widely-established relevance of the BIT process library, we believe that our test collection is well-suited to achieve a high external validity of the results.

6.2 Deterministic Conformance Evaluation

This section presents our evaluation regarding the utility of behavioral space-based conformance checking. To assess the utility of our approach, we observe for how many traces our approach is able to provide non-probabilistic, i.e. deterministic, conformance-checking results. We compare this to the number of traces for which traditional conformance-checking techniques can provide the same results. We note that these results are independent of the choice for a particular traditional conformance-checking technique, since none of the other techniques are able to provide trustworthy results in the presence of mapping uncertainty.

6.2.1 Setup

Figure 7 depicts the three main steps of our evaluation setup. To perform these steps, we employ the *ProM6 framework*, which provides a vast amount of so-called *plug-ins* that implement process mining techniques.[3] In the first two steps of this evaluation, we build on existing plug-ins for event-to-activity mapping techniques, as described in [7].

3. See www.promtools.org for more information and to download the framework.

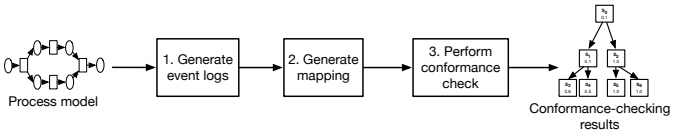


Fig. 7. Overview of the evaluation setup

For the third step, we implemented the generation of behavioral spaces and our proposed technique for conformance checking as a plug-in, which is available as part of the *BehavioralSpaces* package in ProM6. In this implementation, we employ the technique for decomposed conformance checking, as described in [29], available in the *JorgeMunozGama* package.

In step 1 of the evaluation, we first generate an event log for each of the 1248 process models in the test collection. In line with the evaluation in [7], we generate a log containing 1000 traces for each model. For process models that include loops, we generate traces with a maximum length of 1000 events. Since we are interested in conformance checking, we transform these fully conforming logs into logs containing non-conforming behavior. We achieve this by using a *noise-insertion* plug-in in ProM.[4] This plug-in randomly adds noise to a log (i.e. non-conforming behavior) by shuffling, duplicating, and removing events for a given percentage of traces. In this manner, we generate 11 different event logs, containing 0%, 10%, . . . , 90%, 100% noise.

In step 2, we take a process model and an accompanying event log and use the mapping technique from [7] to establish an event-to-activity mapping. We have selected this particular technique for the real-world model collection because it returns all potential mappings in case of uncertainty. Furthermore, the technique is relatively robust in the context of non-conforming behavior. For the synthetic data collection, we use the positional-based matcher from [20], since the technique presented in [7] is not suitable in the presence of non-free choice constructs. In case the approach can compute a single mapping, i.e. there is no mapping uncertainty, we can conclude that for this process model and event log, traditional conformance-checking techniques suffice to determine the conformance of all traces in the log. If the mapping approach returns multiple possible mappings, i.e. there is mapping uncertainty, we continue with the third step of the evaluation.

In step 3, we first construct a behavioral space for a trace based on the uncertain event-to-activity mapping $\mathbb{EA}$ established in the previous step. We obtain a behavioral space by creating a trace translation for each of the potential event-to-activity mappings included in $\mathbb{EA}$, as described in Section 4.1. Afterwards, we perform a conformance check between the obtained behavioral space and the process model using the approach described in this paper. If this method returns a $ProbConf$ value of 0.0 or 1.0 for the entire process model, we can conclude that our technique is able to provide a non-probabilistic conformance-checking result for the trace. This means that we know whether or not τ is conforming with certainty. For other values, also the con-

4. Provided by the *Event2ActivityMatcher* package.

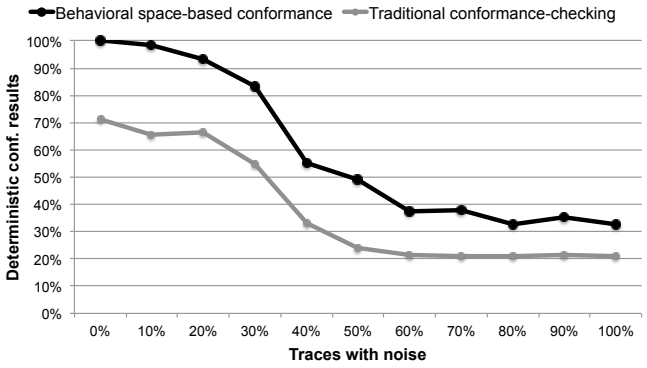


Fig. 8. Deterministic conformance-checking results for the real-world collection

sideration of behavioral spaces does not suffice to be sure about the conformance of τ. Nevertheless, our technique still obtains probabilistic results and diagnostic information, whereas traditional conformance-checking techniques cannot provide any trustworthy results for these cases.

6.2.2 Results

In this section, we first consider the overall performance of our technique in practical settings based on the evaluation results obtained for the real-world model collection. Subsequently, we investigate the impact of control-flow constructs on the performance based on the collection of synthetic models.

Real-world collection. Figure 8 presents the results of our evaluation experiments for the real-world model collection. The figure illustrates for which percentage of traces deterministic conformance-checking results are obtained by our and traditional techniques.

For noise level 0, where all traces in the event logs conform to the process models, we observe that the mapping approach establishes a single event-to-activity mapping for 71% of the models in the collection, which means that traditional techniques can provide (deterministic) results 71% of the traces. Because these logs do not contain non-conforming behavior, the inability to establish mapping for certain models is caused by activities which are behaviorally identical to each other. Such cases can be seen for activities F and G of the running example. Because of these issues, traditional conformance-checking techniques cannot assess the conformance of 29% of the traces. However, by using behavioral spaces, we can still determine the conformance of a trace when mapping uncertainty is caused by such behavioral equivalent activities. Hence, by using our proposed conformance-checking technique, we can provide deterministic conformance-checking results for all traces.

For increasing noise levels, the ability of the mapping approach to establish a single event-to-activity mapping diminishes. For logs with 10% and 20% noisy traces the approach can still establish a mapping for approximately 66% of the process models, as indicated by the 66% deterministic conformance results in Figure 8. However, this percentage sharply drops to 33% certain results for event logs with 40% noise, followed by only 21% certainty for noise levels above 60%. As a result of these steep drops, the ability of traditional conformance-checking techniques to provide trustworthy conformance-checking results also sharply decreases. Although our behavioral space-based technique can also provide less deterministic results when the level of noise increases, this decrease is considerably less severe than for the benchmark. For instance, at 30% noise, our technique can still provide deterministic results for 83%, whereas the benchmark can only provide such results for 55% of the cases. For the highest noise levels, our technique can still provide deterministic results for approximately 30% of the traces, which means that the technique outperforms the benchmark by close to 50%.

In summary, traditional conformance-checking techniques become less and less useful. For high noise levels, they can provide results for as little as 21% of the traces. Although the deterministic results obtainable through conformance checking with behavioral spaces is also affected by the increased levels of noise, the impact is much smaller. Therefore, we can conclude that in practical scenarios our conformance-checking technique is much wider applicable than traditional conformance-checking techniques. Furthermore, a crucial aspect in favor of our conformance-checking technique is that even in cases where also our technique cannot provide deterministic conformance-checking results, our technique still provides trustworthy conformance-checking information in the form of probabilistic results and diagnostic insights.

TABLE 3
Fraction of traces with deterministic conformance-checking results for the synthetic collection

Models with	N	0%	20%	40%	60%	80%	100%
Loops	63	96.5	75.5	57.0	38.4	21.2	3.9
Skips	130	81.8	60.5	45.9	33.2	19.2	5.5
N.-free choice	104	73.0	53.0	41.2	31.7	18.5	5.9
All models	650	89.8	68.9	52.1	36.1	19.9	4.0

Synthetic collection. Table 3 provides an overview of the performance of our technique for the collection of synthetic models. Given that these models are all generated based on the same parameters, this collection allows us to accurately assess how the presence of certain control-flow constructs affects our technique's performance. In particular, the table depicts the performance of the technique on the subset of models that contain loops, skips, and non-free choice constructs. From these results, we can observe that in particular non-free choice constructs affect the performance of our technique. For models with non-free choice constructs, the fraction of traces for which we can compute deterministic conformance-checking results is lower than for the overall collection, e.g. 0.73 versus 0.90 for 0% noise and 0.41 versus 0.52 for 40% noise. The presence of skips has also been shown to affect the performance of the approach, though the impact is lower in this case. However, the performance on models with loops is slightly higher when compared to the average. This is most likely because the presence of loops results in repeated occurrences of subtraces, which can be particularly helpful when establishing event-to-activity

mappings, resulting in less mapping uncertainty.

6.3 Computational Efficiency

In the second part of our evaluation, we evaluate the computational efficiency of our computation approach. As described in Section 5, our approach analyzes which fragments of a decomposed process model actually need to be recomputed for all trace translations, as a means to reduce the necessary number of conformance checks. To determine the gains obtained in this way, we compare the computation time to a benchmark. This benchmark is obtained by performing the conformance check for all trace translations for every fragment of a decomposition, rather than considering which fragments actually need to be recomputed for all translations. This part of the evaluation is conducted on the collection of real-world models.

6.3.1 Setup

For this evaluation experiment we focused on the time required to perform conformance checks using our technique and the benchmark. We performed this evaluation using the same sets of event logs, with varying noise levels, as used in the first part of the evaluation. Considering the setup depicted in Figure 7, we start the measurement after we have decomposed a process model in step 3 of the evaluation. This means that we measure the time used for the actual conformance check, not for establishing the event-to-activity mapping and for process model decomposition. We performed the evaluation on an HP desktop with an 8 $\times$ 3.60 GHz Intel Core i7 processor and 8 GB RAM, running on 64-bit Ubuntu 16.04 and Java Virtual Machine 1.8. As a benchmark, we also measure the computation time that is required when all trace translations are compared against all process model fragments, i.e., the time that would be required without our proposed approach for efficient conformance checking.

6.3.2 Results

Figure 9 depicts the efficiency gains that can be obtained by using our computation approach when compared to the benchmark. The percentage efficiency gained is here computed as $(1 - \frac{time(approach)}{time(benchmark)}) \times 100\%$.

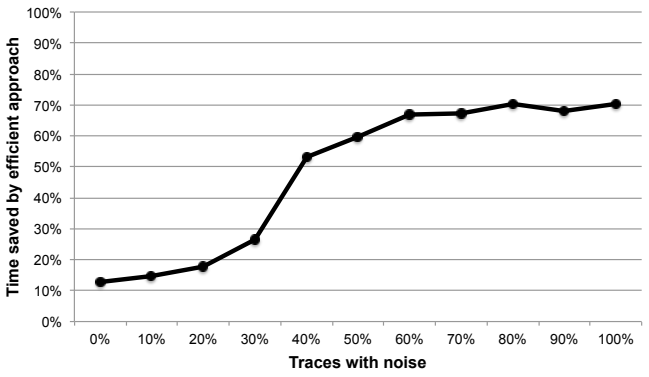


Fig. 9. Computational efficiency of conformance-checking approach

The figure shows that these efficiency gains increase for higher levels of noise. This is the case because for higher levels of noise, the number of possible mappings generated by a mapping approach typically increases. In these cases, more time can be saved when performing conformance checks in a manner that does not require a full enumeration of all possible trace translations for every fragment of a decomposition. In particular, at 0% noise, our efficient approach only saves approximately 11% of the computation time on average. For this noise level, the mapping approach most often generates only a single mapping and an average of approximately 10 mappings per process. By contrast, for noise levels above 40%, our computation approach leads to a reduction of more than half of the throughput time. At higher noise levels, our approach saves approximately 70% of the computation time. This reduction can be explained by the large number of mappings generated by the mapping approach: an average of 136 mappings per process for event logs with a 100% noise level.

In summary, these results clearly illustrate that the computation approach described in Section 5 can yield significant gains in computational efficiency. In this way, the approach makes conformance checking more applicable in realistic settings.

6.4 Limitations

Our evaluation experiments demonstrate that our technique provides useful and efficient conformance-checking results in realistic settings. However, these results need to be considered against the background of certain limitations. In particular, we identify limitations related to the technique itself, limitations with respect to the expected input, and limitations related to the presented evaluation.

Our proposed technique has to be considered against the limitation that the obtained conformance-checking results are dependent on the quality of the event-to-activity mappings on which our approach builds, i.e. the results depend on the quality of the utilized mapping techniques. Most importantly, its results can be negatively affected if the correct mapping is not included in the set of potential mappings generated by a mapping approach. However, these issues rarely occurred during our evaluation experiments, especially for traces with relatively low levels of noise. This illustrates that the utilized mapping technique is well-suited for our purposes.

As for the expected input of our technique, it is important to note that the level of block-structuredness of the input model affects how detailed the computed conformance insights are. In case the input model consists of a single unstructured fragment, conformance-checking results can be only obtained on that level of granularity. However, since both industrial as well as academic modeling guidelines strongly advocate the use of block-structured models [37], [38], we do not expect this to represent a common issue. What is more, there are automatic techniques available to structure certain types of non-block-structured process models [39].

A limitation related to the evaluation of our technique is that our test collection consisted of partially synthetic data. While the process models used for the evaluation

were obtained from a collection of real-world models, the event logs were automatically generated. This means that the obtained results may not fully reflect the situation in practice, where event logs related to the process models may have different characteristics. Nevertheless, we generated logs with varying levels of noisy behavior, which means that the utilized test collection contains behavior that can be observed in a variety of situations. Since our evaluation results show that our conformance-checking technique outperforms traditional techniques for all noise levels, we are confident that the improved performance also holds for real-world event logs.

7 Related Work

This section discusses three streams of work related to our conformance-checking technique. Section 7.1 discusses application scenarios and techniques for conformance checking. Section 7.2 considers techniques for the establishment of event-to-activity mappings. Finally, Section 7.3 considers existing works that deal with data uncertainty in other application contexts.

7.1 Conformance Checking

Process conformance checking involves the comparison of observed process behavior to a process specification. These techniques are applied in various application scenarios, including process querying [40], legal compliance [41], and auditing [42]. Most conformance-checking techniques focus on the comparison of observed process behavior, as captured in event traces, to a process specification in the form of a process model, cf. [29], [32], [43], [44], [45], [46]. Recently, however, we also developed a technique for conformance checking for process specifications in the form of natural language texts [47].

The goal of conformance checking is primarily to determine if a trace of observed events conforms to the process specified by a process model. However, most conformance-checking techniques go beyond determining whether or not a trace conforms to a process model. They analyze non-conforming traces in order to determine their degree of non-conformance, as well as to investigate to which parts of a process the trace does and does not conform. Different types of conformance-checking techniques have been developed for this purpose. Examples include the seminal *replay-based* [2], [45] and *alignment-based* [32], [43] techniques, as well as the decomposition-based technique [29] utilized in this paper. While these techniques focus on conformance from a control-flow perspective, developments have also been made that check conformance with respect to other process perspectives, such as time-based [48] and data-based conformance [49].

Despite the vast number of existing conformance-checking techniques, all these techniques require the existence of a known event-to-activity mapping, a limitation which we overcome with the conformance-checking technique presented in this paper.

7.2 Mapping Events to Activities

The task of establishing a mapping between events and activities represents a so-called *matching problem*. Matching problems are addressed by *matching techniques* that set out to automatically identify relations between two artifacts [50]. A plethora of matching techniques have been developed and applied in various fields, including schema matching (cf. [13], [51], [52]), ontology alignment (c.f.. [15], [53], [54]), and process model matching (cf. [20], [55], [56]).

A variety of matching techniques [4], [8], [9], [12] have recently been developed that apply concepts from related matching fields to the task of matching events to activities. These techniques aim to identify events and workflow activities with similar characteristics. To achieve this, they consider a variety of information. For instance, techniques presented in [4], [8] compare the labels associated with events and tasks in order to determine their similarity. This supports the identification of events and activities with equal labels, as well pairs with similar labels, e.g. "*ship product*" and "*product sent*". Other information taken into accounts by matchers are structural properties (indicating the relations that exist between different events and activities) [7], [12], work instructions associated with process models [4], and additional data associated with events [9]. A distinction between mapping approaches that is relevant for this paper exists between approaches that map at the event class-level and approaches that map events at an instance-level. Approaches of the former type, such as [7], map all occurrences of a particular event class E_x to the same activity class. By contrast, instance-level techniques might map events of a particular class to different activities for different traces. For instance, an event of type E_y might be mapped to activity C for a trace τ, but to activity D for trace τ'.

7.3 Representing Data Uncertainty

To be able to reason about process conformance in the context of mapping uncertainty, we introduce the notion of *probabilistic behavioral spaces* in this paper. This concept is used to capture the implications of uncertainty in a manner that enables probabilistic reasoning about process conformance. In this way, our work relates to streams of research for reasoning in the presence of data uncertainty.

Data uncertainty is inherent to various application contexts, typically caused by data randomness, incompleteness, or limitations of measuring equipment [57]. This has created a need for algorithms and applications for uncertain data managements. As a result, the modeling of uncertain data has been studied extensively [58]. This has, lead to the development of various probabilistic and uncertain databases cf. [59], [60], [61], as well as a variety of querying and analysis algorithms for these data structures [62], [63].

Our notion of probabilstic behavioral spaces shares characteristics with solutions for probabilistic and uncertain databases. Most prominently, our notion of probabilistic behavioral spaces builds on a concept of *possible worlds* used in databases to capture the various possible implications of data uncertainty for database instances [63]. We use a similar notion to capture the impact of mapping uncertainty on the process behavior that may be conveyed by an event trace. Furthermore, the process-orientation of behavioral spaces implicitly capture the dependencies that exist between uncertain events for a single trace. This is similar to the use

of conditions to capture dependencies between uncertain database values, such as used by [61]. Despite these similarities, the application contexts of these uncertain data models, mostly querying and data integration [58], differ considerably from the process-oriented view of proabilstic behavioral spaces used for conformance checking.

8 Conclusion

In this paper, we introduced a conformance-checking technique that can be used in the presence of uncertain event-to-activity mappings. Our technique provides conformance-checking results without the need to select a single, possibly incorrect mapping to base conformance checks on. This is achieved by considering the entire spectrum of possible mappings generated by event-to-activity mapping techniques and capturing this spectrum in a behavioral space. Our probabilistic conformance-checking metric then provides insights into the fraction of compliant mappings, as well as useful diagnostic information. Therefore, our conformance-checking technique avoids the risk of drawing incorrect conclusions about process conformance. A quantitative evaluation based on a large collection of real-world process models demonstrated that our technique can be used to obtain results in a vast number of cases where existing conformance-checking techniques fail to do so. Furthermore, we presented a computation approach that is shown to considerably reduce the time required to obtain conformance-checking results.

Acknowledgment

Han van der Aa is funded by a postdoctoral grant from the Alexander von Humboldt Foundation.

References

[1] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Inf. Syst.*, vol. 54, pp. 209–234, 2015.

[2] S. K. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, "Determining process model precision and generalization with weighted artificial negative events," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1877–1889, 2014.

[3] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose *et al.*, "Process mining manifesto," in *BPM*. Springer, 2011, pp. 169–194.

[4] T. Baier, J. Mendling, and M. Weske, "Bridging abstraction layers in process mining," *Inf. Syst.*, vol. 46, pp. 123–139, 2014.

[5] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communication. ACM*, vol. 55, no. 2, pp. 55–61, 2012.

[6] M. Weidlich, H. Ziekow, A. Gal, J. Mendling, and M. Weske, "Optimizing event pattern matching using business process models," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2759–2773, 2014.

[7] T. Baier, C. Di Ciccio, J. Mendling, and M. Weske, "Matching of events and activities-an approach using declarative modeling constraints," in *International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer, 2015, pp. 119–134.

[8] ——, "Matching events and activities by integrating behavioral aspects and label analysis," *Soft. Syst. Mod.*, pp. 1–26, 2017.

[9] A. Senderovich, A. Rogge-Solti, A. Gal, J. Mendling, and A. Mandelbaum, "The road from sensor data to process instances via interaction mining," in *CAiSE*. Springer, 2016, pp. 257–273.

[10] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy, "Representing and reasoning about mappings between domain models," *AAAI/IAAI*, vol. 2002, pp. 80–86, 2002.

[11] H. van der Aa, H. Leopold, and H. A. Reijers, "Checking process compliance on the basis of uncertain event-to-activity mappings," in *CAiSE*. Springer, 2017, pp. 79–93.

[12] T. Baier, A. Rogge-Solti, M. Weske, and J. Mendling, "Matching of events and activities-an approach based on constraint satisfaction," in *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 2014, pp. 58–72.

[13] A. Doan and A. Y. Halevy, "Semantic integration research in the database community: A brief survey," *AI magazine*, vol. 26, no. 1, p. 83, 2005.

[14] J. Euzenat, P. Shvaiko *et al.*, *Ontology matching*. Springer, 2007, vol. 18.

[15] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 158–176, 2013.

[16] C. Klinkmüller, H. Leopold, I. Weber, J. Mendling, and A. Ludwig, "Listen to me: Improving process model matching through user feedback," in *BPM*. Springer, 2014, pp. 84–100.

[17] H. Leopold, M. Niepert, M. Weidlich, J. Mendling, R. Dijkman, and H. Stuckenschmidt, "Probabilistic optimization of semantic process model matching," in *BPM*. Springer, 2012, pp. 319–334.

[18] Y. Gao, S. Song, X. Zhu, J. Wang, X. Lian, and L. Zou, "Matching heterogeneous event data," *IEEE Trans. Knowl. Data Eng.*, 2018.

[19] S. Song, Y. Gao, C. Wang, X. Zhu, J. Wang, and P. Yu, "Matching heterogeneous events with patterns," *IEEE Transactions on Knowledge & Data Engineering*, no. 1, pp. 1–1, 2017.

[20] H. Van der Aa, A. Gal, H. Leopold, H. A. Reijers, T. Sagi, and R. Shraga, "Instance-based process matching using event-log information," in *International Conference on Advanced Information Systems Engineering (In press)*. Springer, 2017.

[21] S. Smirnov, M. Weidlich, and J. Mendling, "Business process model abstraction based on behavioral profiles," in *Service-Oriented Computing*. Springer, 2010, pp. 1–16.

[22] W. M. Van der Aalst, "The application of petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

[23] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in bpmn," *Information and Software technology*, vol. 50, no. 12, pp. 1281–1294, 2008.

[24] M. Zur Muehlen and J. Recker, "How much language is enough? theoretical and practical use of the business process modeling notation," in *Seminal Contributions to Information Systems Engineering*. Springer, 2013, pp. 429–443.

[25] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM Journal on Computing*, vol. 2, no. 3, pp. 135–158, 1973.

[26] J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through sese decomposition," in *International Conference on Service-Oriented Computing*. Springer, 2007, pp. 43–55.

[27] A. Polyvyanyy, J. Vanhatalo, and H. Völzer, "Simplified computation and generalization of the refined process structure tree," in *International Workshop on Web Services and Formal Methods*. Springer, 2010, pp. 25–41.

[28] W. M. Van der Aalst, "Decomposing petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[29] J. Munoz-Gama, J. Carmona, and W. M. Van Der Aalst, "Single-entry single-exit decomposed conformance checking," *Inf. Syst.*, vol. 46, pp. 102–122, 2014.

[30] R. Conforti, M. La Rosa, and A. H. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, 2017.

[31] M. Weidlich and J. Mendling, "Perceived consistency between process models," *Inf. Syst.*, vol. 37, no. 2, pp. 80–98, 2012.

[32] W. M. P. Van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[33] E. W. Myers, "An o (nd) difference algorithm and its variations," *Algorithmica*, vol. 1, no. 1, pp. 251–266, 1986.

[34] D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf, "Analysis on demand: Instantaneous soundness checking of industrial business process models," *Data Knowl. Eng.*, vol. 70, no. 5, pp. 448–466, 2011.

[35] T. Jouck and B. Depaire, "Generating artificial data for empirical analysis of control-flow discovery algorithms: A process tree and log generator," *BISE*, pp. 1–18, 2018.
[36] J. Desel and J. Esparza, *Free choice Petri nets*. Cambridge university press, 2005, vol. 40.
[37] J. Mendling, H. A. Reijers, and W. M. van der Aalst, "Seven process modeling guidelines (7pmg)," *Information and Software Technology*, vol. 52, no. 2, pp. 127–136, 2010.
[38] B. Silver, *BPMN Method and Style, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press Aptos, 2011.
[39] A. Polyvyanyy, L. García-Bañuelos, and M. Dumas, "Structuring acyclic process models," in *BPM*. Springer, 2010, pp. 276–293.
[40] A. Awad, G. Decker, and M. Weske, "Efficient compliance checking using bpmn-q and temporal logic," in *BPM*. Springer, 2008, pp. 326–341.
[41] S. Sadiq, G. Governatori, and K. Namiri, "Modeling control objectives for business process compliance," in *BPM*. Springer, 2007, pp. 149–164.
[42] R. Accorsi and T. Stocker, "On the exploitation of process mining for security audits: the conformance checking case," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 1709–1716.
[43] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC*. IEEE, 2011, pp. 55–64.
[44] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *BPM*. Springer, 2010, pp. 211–226.
[45] A. Rozinat and W. M. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.
[46] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske, "Process compliance analysis based on behavioural profiles," *Inf. Syst.*, vol. 36, no. 7, pp. 1009–1025, 2011.
[47] H. Van der Aa, H. Leopold, and H. A. Reijers, "Dealing with behavioral ambiguity in textual process descriptions," in *BPM*. Springer, 2016, pp. 271–288.
[48] A. Senderovich, M. Weidlich, L. Yedidsion, A. Gal, A. Mandelbaum, S. Kadish, and C. A. Bunnell, "Conformance checking and performance improvement in scheduled processes: A queueing-network perspective," *Inf. Syst.*, vol. 62, pp. 185–206, 2016.
[49] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. Van der Aalst, "Balanced multi-perspective checking of process conformance," *Computing*, vol. 98, no. 4, pp. 407–437, 2016.
[50] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "Semantic matching," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 2561–2566.
[51] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *the VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001.
[52] A. Gal, "Uncertain schema matching," *Synthesis Lectures on Data Management*, vol. 3, no. 1, pp. 1–97, 2011.
[53] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.
[54] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with coma++," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 906–908.
[55] R. M. Dijkman, M. Dumas, B. F. Van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Inf. Syst.*, vol. 36, no. 2, pp. 498–516, 2011.
[56] M. Weidlich, R. M. Dijkman, and J. Mendling, "The ICoP framework: Identification of correspondences between process models," in *CAiSE*. Springer, 2010, pp. 483–498.
[57] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 15–26.
[58] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 609–623, 2009.
[59] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working models for uncertain data," in *22nd International Conference on Data Engineering*. IEEE, 2006, pp. 7–7.
[60] L. Peng and Y. Diao, "Supporting data uncertainty in array databases," in *SIGMOD*. ACM, 2015, pp. 545–560.
[61] S. Abiteboul, P. Kanellakis, and G. Grahne, *On the representation and querying of sets of possible worlds*. ACM, 1987, vol. 16, no. 3.
[62] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases with x-relations," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1669–1682, 2008.
[63] R. Murthy, R. Ikeda, and J. Widom, "Making aggregation work in uncertain and probabilistic databases," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1261–1273, 2011.

Han van der Aa Han van der Aa is a post-doctoral researcher in the Process-Driven Architectures Group at the Humboldt-Universität zu Berlin. He is funded by a grant from the Alexander von Humboldt Foundation. He obtained a PhD from the Vrije Universiteit Amsterdam in 2018. His research interests include business process modeling, process mining, natural language processing, and complex event processing. His research has been published, among others, in IEEE Transactions on Knowledge and Data Engineering, Decisions Support Systems and Information Systems.

Henrik Leopold Henrik Leopold is an Assistant Professor at Kühne Logistics University (KLU) and a Senior Researcher at Hasso Plattner Institute (HPI) at the Digital Engineering Faculty, University of Potsdam. He obtained his PhD degree in Information Systems from the Humboldt University Berlin, Germany. Before joining KLU/HPI, Henrik held positions at Vrije Universiteit Amsterdam as well as WU Vienna. His research is mainly concerned with leveraging artificial intelligence to analyze and improve business processes. He has published over 60 scientific contributions, among others, in IEEE Transactions on Software Engineering, Decision Support Systems, and Information Systems. His doctoral thesis received the German Targion Award for the best dissertation in the field of strategic information management.

Hajo A. Reijers Hajo Reijers is a full professor in the Department of Information and Computing Sciences of Utrecht University, where he holds the chair in Business Process Management and Analytics. He is also a part-time, full professor in the Department of Mathematics and Computer Science of Eindhoven University of Technology, as well as an adjunct professor in the School of Information Systems of Queensland University of Technology (QUT). The focus of his academic research is on business process innovation, process analytics, robotic process automation, and enterprise IT. He published in Information Systems, the Journal of Management Information Systems, the Journal of Information Technology, the International Journal of Cooperative Information Systems, Organization Studies, and Omega, among other journals.