Conformance Checking over Uncertain Event Data

Marco Pegoraro ⬤✉ [1], Merih Seran Uysal ⬤ [1], and Wil M.P. van der Aalst ⬤ [1]

[1] *Chair of Process and Data Science (PADS), Department of Computer Science,*
RWTH Aachen University, Aachen, Germany
`{pegoraro, uysal, vwdaalst}@pads.rwth-aachen.de`

Abstract

The strong impulse to digitize processes and operations in companies and enterprises have resulted in the creation and automatic recording of an increasingly large amount of process data in information systems. These are made available in the form of *event logs*. Process mining techniques enable the process-centric analysis of data, including automatically discovering process models and checking if event data conform to a given model. In this paper, we analyze the previously unexplored setting of uncertain event logs. In such event logs uncertainty is recorded explicitly, i.e., the time, activity and case of an event may be unclear or imprecise. In this work, we define a taxonomy of uncertain event logs and models, and we examine the challenges that uncertainty poses on process discovery and conformance checking. Finally, we show how upper and lower bounds for conformance can be obtained by aligning an uncertain trace onto a regular process model.

Keywords: Process Mining · Uncertain Data · Partial Order.

Colophon

Correspondence to:
Marco Pegoraro, Chair of Process and Data Science (PADS), Department of Computer Science,
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany
Website: http://mpegoraro.net/ · Email: pegoraro@pads.rwth-aachen.de · ORCID: 0000-0002-8997-7517

Content: 52 pages, 12 figures, 10 tables, 42 references. Typeset with pdfLaTeX, Biber, and BibLaTeX.
Please do not print this document unless strictly necessary.

1   Introduction

Over the last decades, the concept of *process* has become more and more central in formally describing the activities of businesses, companies and other similar entities, structured in specific steps and phases. A process is thus defined as a well-structured set of activities, potentially performed by multiple actors (*resources*), which contribute to the completion of a specific task or to the achievement of a specific goal. In this context, a very important notion is the concept of *case*, that is, a single instance of a process. For example, in a healthcare process, a case may be a single hospitalization of a patient, or the patient themself; if the process belongs to a credit institution, a case may be a loan application from a customer, and so on. The case notion allows us to define a process as a procedure that defines the steps needed to handle cases from inception to completion. A *process model* defines such a procedure, and can be expressed in a number of different formalisms (transition systems, Petri nets, BPMN and UML diagrams, and many more). Consequently, the study and adoption of analysis techniques specifically customized to deal with process data and process models has enabled the bridging of business administration and data science and the development of dedicated disciplines like *business intelligence* and *Business Process Management* (BPM).

The processes that govern the innards of business companies are increasingly supported by software tools. Performing specific activities is both aided and recorded by *Process-Aware Information Systems* (PAISs), which support the definition and management of processes. The information regarding the execution of processes can then be extracted from PAISs in the form of an *event log*, a database or file containing the digital footprint of the operations carried out in the context of the execution of a process and recorded as *events*. Event logs can vary in form, and contain differently structured information depending on the information system that enacted data collection in the organization. Although many different event attributes can be recorded, it is typically assumed that three basic features of an event are available in the log: the time in which the event occurred, the activity that has been performed, and the case identifier to which the event belong. This last attribute allows to group events in clusters belonging to the same case, and these resulting clusters (usually organized in sequences sorted by timestamp) are called *process traces*. The discipline of *process mining* is concerned with the automatic analysis of event logs, with the goal of extracting knowledge regarding e.g. the structure of the process, the conformity of events to a specific normative process model, the performance of the agents executing the process, the relationships between groups of actors in the process.

In this paper, we will consider the analysis of a specific class of event logs: logs that contain *uncertain event data*. Uncertain events are recordings of executions of specific activities in a process which are enclosed with an indication of uncertainty in the event attributes. Specifically, we consider the case where the attributes of an event are not

recorded as a precise value but as a range or a set of alternatives.

Uncertain event data are common in practice, but uncertainty is often not explicit. The *Process Mining Manifesto* [5] describes a fundamental property of event data as *trustworthiness*, the assumption that the recorded data can be considered correct and accurate. In a general sense, uncertainty—as defined here—is an explicit absence of trustworthiness, with an indication of uncertainty recorded together with the event data. In the taxonomy of event data proposed in the Manifesto, the logs at the two lower levels of quality frequently lack trustworthiness, and thus can be uncertain. This encompasses a wide range of processes, such as event logs of document and product management systems, error logs of embedded systems, worksheets of service engineers, and any process recorded totally or partially on paper. There are many possible causes of uncertainty:

- *Incorrectness*. In some instances, the uncertainty is simply given by errors that occurred while recording the data themselves. Faults of the information system, or human mistakes in a data entry phase can all lead to missing or altered event data that can be subsequently modeled as uncertain event data.

- *Coarseness*. Some information systems have limitations in their way of recording data - often tied to factors like the precision of the data format - such that the event data can be considered uncertain. A typical example is an information system that only records the date, but not the time, of the occurrence of an event: if two events are recorded in the same day, the order of occurrence is lost. This is an especially common circumstance in the processes that are, partially or completely, recorded on paper and then digitalized. Another factor that can lead to uncertainty in the time of recording is the information system being overloaded and, thus, delaying the recording of data. This type of uncertainty can also be generated by the limited sensibility of a sensor.

- *Ambiguity*. In some cases, the data recorded is not an identifier of a certain event attribute; in these instances, the data needs to be interpreted, either automatically or manually, in order to obtain a value for the event attribute. Uncertainty can arise if the meaning of the data is ambiguous and cannot be interpreted with precision. Examples include data in the form of images, text, or video.

These factors cause the presence of *implicit* uncertainty in the event log. It is important to note that, in order to be analyzed, these indications of imprecision or incorrectness have to be translated into *explicit* uncertainty. Explicit uncertainty is contained directly in the event log in the form of event attributes. It is possible to think of explicit uncertainty as metadata complementing the information regarding events. This metadata describes the type and magnitude of the imprecision affecting some event attributes, which might be part of the control-flow perspective or an additional data perspective present in the event log.

Aside from the possible causes, we can individuate other types of uncertain event logs based on the frequency of uncertain data. Uncertainty can be *infrequent*, when a specific attribute is only seldomly recorded together with explicit uncertainty; the uncertainty is rare enough that uncertain events can be considered outliers. Conversely, *frequent* uncertain behavior of the attribute is systematic, pervasive in a high number of traces, and thus not to be considered an outlier. The uncertainty can be considered part of the process itself. These concepts are not meant to be formal, and are laid out to distinguish between logs that are still processable regardless of the uncertainty, and logs where the uncertainty is too invasive to analyze them with existing process mining techniques.

In some contexts, the causes of uncertainty in event data can be resolved at the source, by acting directly on the process and on the tools supporting operations within it. For instance, a natural way to eliminate uncertainty in data recording is to automate tasks within the process as much as possible. This is a popular solution in applications like industry and manufacturing, where the actual tasks already involve machinery or automated systems. While we discuss the possibility of employing automation tools in processes where the majority of agents are humans in Section 7.2, supporting a process with automation or auditing software that oversees and records the actions of agents is very challenging, for both technical and ethical reasons. Automatically recording data across different platforms, formats, and information systems by different producers and hosted by different service providers is often unfeasible, and there are legal reasons that might prevent it, such as breaches of confidentiality. In some jurisdictions, legally valid documents must be on paper, making real-time automatic data recording outright impossible. For these reasons, while eliminating uncertainties through automation is an advisable choice, analyzing data containing a description of such uncertainties is sometimes the only analysis technique that can deliver approximate but trustworthy results.

The diagram in Figure 1 shows an overview of the main elements of process mining over uncertainty. The schema shows some additional elements with respect to classical process mining: we can see that we can combine raw process data from information systems (containing implicit uncertainty) with domain knowledge provided by a process expert to obtain an uncertain event log, which contains explicit uncertainty. The data in an uncertain event log can be abstracted in a graph representation, which enables the inspection of its causes. Lastly, the graph representations also allows to perform the tasks of process discovery and conformance checking on uncertain event data.

In this paper, we propose a taxonomy of the different types of explicit uncertainty in process mining, together with a formal, mathematical formulation. As an example of practical application, we will consider the case of conformance checking [15], and we will apply it to uncertain data by assessing what are the upper and lower bounds on the conformance score for possible values of the attributes in an uncertain trace.

The main drivers behind this work is to provide the means to treat uncertainty as
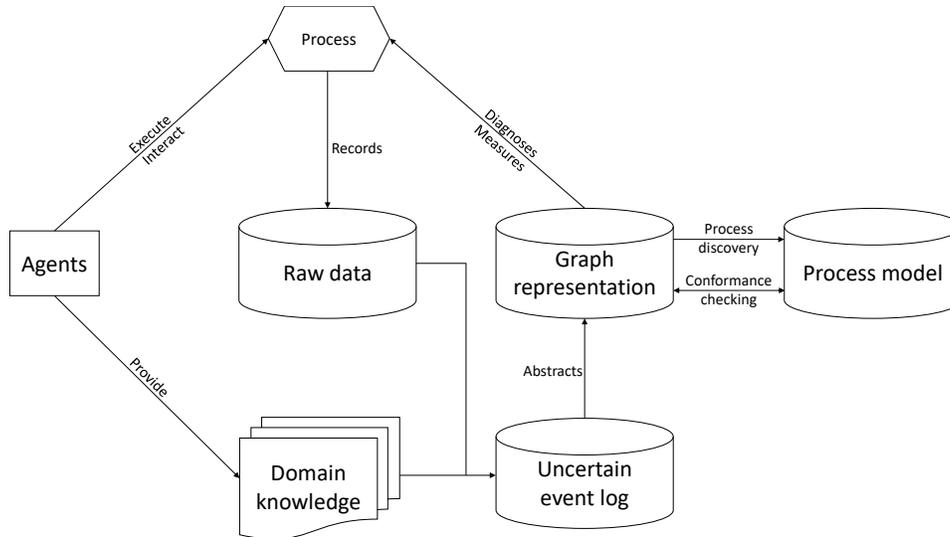
Figure 1: The overall schema for process mining over uncertain event data.

a relevant part of a process; thus, we aim not to filter it out but to model and explain it. In conclusion, there are two novel aspects regarding uncertain data that we intend to address in this work. The first novelty is the *explicitness of uncertainty*: we work with the underlying assumption that the actual value of the uncertain attribute, while not directly provided, is described formally. This is the case when meta-information about the uncertainty in the attribute is available, either deduced from the features of the information system(s) that record the logs or included in the event log itself. Note that, as opposed to all previous work on the topic, the fact that uncertainty is explicit in the data means that the concept of uncertain behavior is completely separated from the concept of infrequent behavior. The second novelty is the *explicit modeling of uncertainty*: we consider uncertainty part of the process. Instead of filtering or cleaning the log, we introduce the uncertainty perspective in process mining by extending the currently available techniques to incorporate it.

In summary, the paper aims to discuss the following research questions:

- What is uncertainty, and in which ways can it manifest within event data?

- How can uncertain event data be processed to retain all information contained within it?

- How can we exploit this representation to solve classical process mining tasks, such as conformance checking?

The rest of this paper is organized as follows. Section 2 proposes a taxonomy of the different possible types of uncertain process data. Section 3 contains the formal definitions needed to manage uncertainty. Section 4 presents the main contribution of this paper, a framework able to describe an array of types and classifications of uncertain behavior. Section 5 describes a practical application of process mining over uncertain event data, the case of conformance checking through alignments. Section 6 shows experimental results on computing conformance checking scores for synthetic uncertain data, as well as a case of application on real-life data. Section 7 discusses previous and related work on the management of uncertain data and on the topic of conformance checking. Finally, Section 8 concludes the paper and discusses future work.

2   A Taxonomy of Uncertain Event Data

The goal of this section of the paper is to propose a categorization of the different types of uncertainty that can appear in process mining. In process management, a central concept is the distinction between the data perspective (the event log) and the behavioral perspective (the process model). The first one is a static representation of process instances, the second summarizes the behavior of a process. Both can be extended with a concept of explicit uncertainty: this concept also implies an extension of the process mining techniques that have currently been implemented.

In this paper, we will focus on uncertainty in event data, rather than applying the concept of uncertainty to models. Specifically, we will consider computing the conformance score of uncertain process data on classical models, extending the approach shown in [29]. An application of process discovery in the setting of uncertain event data has been presented in [31].

We can individuate two different notions of uncertainty:

- *Strong uncertainty*: the possible values for the attributes are known, but the probability that the attribute will assume a certain instantiation is unknown or unobservable.

- *Weak uncertainty*: both the possible values of an attribute and their respective probabilities are known.

In the case of a discrete attribute, the strong notion of uncertainty consists on a set of possible values assumed by the attribute. In this case, the probability for each possible value is unknown. Vice-versa, in the weak uncertainty scenario we also have a discrete probability distribution defined on that set of values. In the case of a continuous attribute, the strong notion of uncertainty can be represented with an interval for the variable. Notice that an interval does not indicate a uniform distribution; there is no information on the likelihood of values in it. Vice-versa, in the weak uncertainty scenario

	Weak uncertainty	Strong uncertainty
Discrete data	Discrete probability distribution	Set of possible values $\{x, y, z, \dots\}$
Continuous data	Probability density function	Interval $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
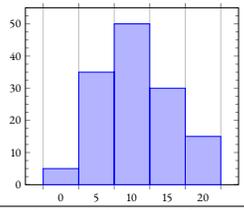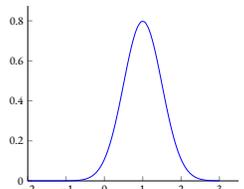
Table 1: The four different types of uncertainty.

we also have a probability density function defined on a certain interval. Table 1 summarizes these concepts. This leads to very simple representations of explicit uncertainty.

In this paper, we consider only the control flow and time perspective of a process—namely, the attributes of the events that allow us to discover a process model. These are the unique identifier of a process instance (case ID), the timestamp (often represented by the distance from a fixed origin point, e.g. the *Unix Epoch*), and the activity identifier of an event. Case IDs and activities are values chosen from a finite set of possible values; they are discrete variables. Timestamps, instead, are represented by numbers and thus are continuous variables.

Table 2: An example of a strongly uncertain trace. For the sake of clarity, the timestamp field only reports dates.

Case ID	Timestamp	Activity	Indet. event
$\{\mathrm{ID}_{327}, \mathrm{ID}_{412}\}$	2011-12-05	A	!
ID_{327}	2011-12-07	$\{B, C, D\}$	!
ID_{327}	$[\text{2011-12-06, 2011-12-10}]$	D	?
ID_{327}	2011-12-09	$\{A, C\}$	!
$\{\mathrm{ID}_{327}, \mathrm{ID}_{412}, \mathrm{ID}_{573}\}$	2011-12-11	E	?

We will also describe an additional type of uncertainty, which lies on the event level

Table 3: An example of a weakly uncertain trace. For the sake of clarity, the timestamp field only reports dates.

Case ID	Timestamp	Activity	Indet. event
{ID313:0.9, ID370:0.1}	2011-12-05	A	!
ID313	2011-12-07	{B:0.7, C:0.3}	!
ID313	$N(2011\text{-}12\text{-}08, 2)$	D	?:0.5
ID313	2011-12-09	{A:0.2, C:0.8}	!
{ID313:0.4, ID370:0.6}	2011-12-11	E	?:0.7

Table 4: Summary of the types of uncertainty that can affect a log over the attributes of its events. The last column provides an encoding for each type of uncertainty.

Attribute	Attribute type	Uncertainty type	Encoding
Event (indeterminacy)	Discrete	Weak	$[E]_\mathbb{W}$
		Strong	$[E]_\mathbb{S}$
Case	Discrete	Weak	$[C]_\mathbb{W}$
		Strong	$[C]_\mathbb{S}$
Activity	Discrete	Weak	$[A]_\mathbb{W}$
		Strong	$[A]_\mathbb{S}$
Timestamp	Continuous	Weak	$[T]_\mathbb{W}$
		Strong	$[T]_\mathbb{S}$
Other attribute	Discrete	Weak	$[ATD]_\mathbb{W}$
		Strong	$[ATD]_\mathbb{S}$
	Continuous	Weak	$[ATC]_\mathbb{W}$
		Strong	$[ATC]_\mathbb{S}$

rather than the attribute level:

- *Indeterminate event*: the event may have not taken place even though it was recorded in the event log. Indeterminate events are indicated with a ? symbol, while determinate (regular) events are marked with a ! symbol.

Examples of strongly and weakly uncertain traces are shown in Tables 2 and 3 respectively. Additionally, we present a time diagram of the trace in Table 2: this representation shows the time relationship between events in the trace in absolute scale. This diagram is shown in Figure 2

The taxonomy presented in this section is summarized in Table 4. This table encodes all types of uncertainty illustrated here. Through this taxonomy, we can indicate the types of uncertainty that might affect an uncertain event log.
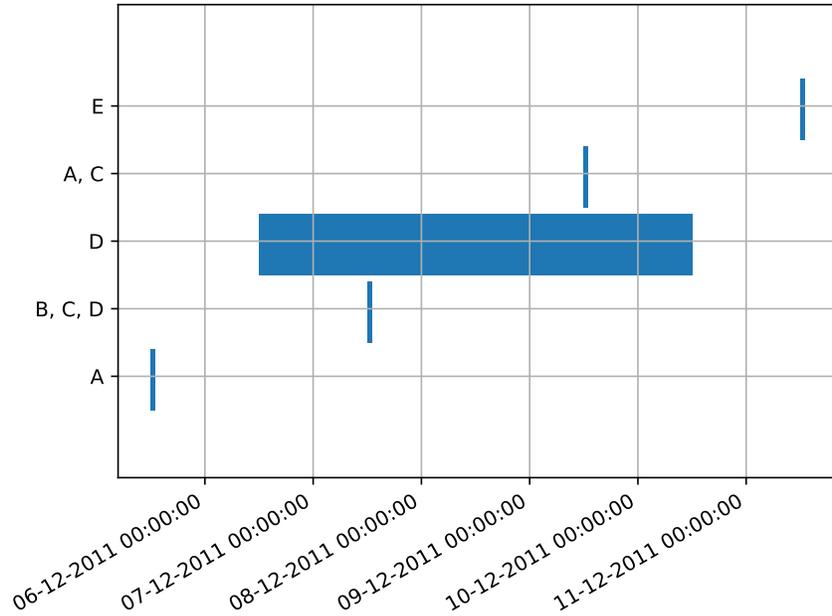
Figure 2: Time diagram of the trace in Table 2. This diagram shows the time information of an uncertain trace in an absolute scale. Note that some types of uncertainty (namely, indeterminate events and uncertainty on case IDs) are not depicted.

More types of uncertainty can be combined to describe an uncertain event log. For example, an event log with strong uncertainty on events, activities and timestamps would be an $[E, A, T]_\mathbb{S}$-type log. An uncertain log can also be characterized by different types of uncertainty on different attributes: a log with strong uncertainty on events and weak uncertainty on activities is a $[E]_\mathbb{S}[A]_\mathbb{W}$-type log.

In the next section, we will describe these different types of uncertainty in a mathematical framework that will, in turn, enable process mining analyses on uncertain event data.

3   Preliminaries

Let us introduce some preliminary definitions in order to describe uncertainty in process mining in a formal way. These definitions will provide the means to represent the behavior contained in uncertain data, and enable process mining tasks such as process discovery and conformance checking.

3.1   Basic Definitions

Firstly, we will define some basic mathematical structures.

Definition 1 (Power Set). *The* power set *of a set A is the set of all possible subsets of A, and is denoted with $\mathcal{P}(A)$. $\mathcal{P}_{NE}(A)$ denotes the set of all the non-empty subsets of A: $\mathcal{P}_{NE}(A) = \mathcal{P}(A) \setminus \{\emptyset\}$.*

Definition 2 (Multiset). *A* multiset *is an extension of the concept of set that keeps track of the cardinality of each element. $\mathcal{B}(A)$ is the set of all multisets over some set A. Multisets are denoted with square brackets, e.g. $b_1 = [\ ]$ (the empty multiset), $b_2 = [a, a, b]$, $b_3 = [a, b, c]$, $b_4 = [a, b, c, a, a, b]$ are all multisets over $A = \{a, b, c\}$. In multiset the order of representation of the elements is irrelevant, and they can also be denoted with the cardinality of their elements, e.g. $b_4 = [a, b, c, a, a, b] = [a^3, b^2, c]$. We denote with $b(x)$ the cardinality of element $x \in A$ in b, e.g. $b_4(a) = 3$, $b_4(c) = 1$, and $b_4(d) = 0$.*

   We can extend to multiset standard set operators such as membership (e.g. $a \in b_2$ and $c \notin b_2$), union (e.g. $b_2 \uplus b_3 = b_4$), difference (e.g. $b_4 \setminus b_3 = b_2$) and total cardinality (e.g. $|b_4| = 6$).

Definition 3 (Sequence, Subsequence and Permutation). *Given a set X, a finite* sequence *over X of length n is a function $s \in X^* : \{1, \ldots, n\} \to X$, and it is written as $s = \langle s_1, s_2, \ldots, s_n \rangle$. We denote with $\langle\ \rangle$ the empty sequence, the sequence with no elements and of length 0. Over the sequence s we define $|s| = n$, $s[i] = s_i$ and $x \in s \Leftrightarrow \exists_{1 \leq i \leq n} s = s_i$. The concatenation between two sequences is denoted with $\langle s_1, s_2, \ldots, s_n \rangle \cdot \langle s'_1, s'_2, \ldots, s'_m \rangle = \langle s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_m \rangle$. Given two sequences $s = \langle s_1, s_2, \ldots, s_n \rangle$ and $s' = \langle s'_1, s'_2, \ldots, s'_m \rangle$, s' is a* subsequence *of s if and only if there exists a sequence of strictly increasing natural numbers $\langle i_1, i_2, \ldots, i_m \rangle$ such that $\forall_{1 \leq j \leq m} s_{i_j} = s'_j$. We indicate this with $s' \subseteq s$. A* permutation *of the set X is a sequence x_S that contains all elements of X without duplicates: $x_S \in X$, $X \in x_S$, and for all $1 \leq i \leq |x_S|$ and for all $1 \leq j \leq |x_S|$, $x_S[i] = x_S[j] \to i = j$. We denote with S_X all such permutations of set X.*

Definition 4 (Sequence Projection). *Let X be a set and $Q \subseteq X$ one of its subsets. $\upharpoonright_Q : X^* \to Q^*$ is the sequence projection function and is defined recursively: $\langle\ \rangle \upharpoonright_Q = \langle\ \rangle$ and for $\sigma \in X^*$ and $x \in X$:*

$$(\langle x \rangle \cdot \sigma) \upharpoonright_Q = \begin{cases} \sigma \upharpoonright_Q & \text{if } x \notin Q \\ \langle x \rangle \cdot \sigma \upharpoonright_Q & \text{if } x \in Q \end{cases}$$

For example, $\langle y, z, y \rangle \upharpoonright_{\{x,y\}} = \langle y, y \rangle$.

Definition 5 (Applying Functions to Sequences). *Let $f: X \nrightarrow Y$ be a partial function. f can be applied to sequences of X using the following recursive definition: $f(\langle\,\rangle) = \langle\,\rangle$ and for $\sigma \in X^*$ and $x \in X$:*

$$f(\langle x\rangle \cdot \sigma) = \begin{cases} f(\sigma) & \text{if } x \notin dom(f) \\ \langle f(x)\rangle \cdot f(\sigma) & \text{if } x \in dom(f) \end{cases}$$

Next, so as to manage the possible different orders between events in a trace with uncertain timestamps, we introduce formalisms to denote strict partial orders.

Definition 6 (Transitive Relation and Correct Evaluation Order). *Let X be a set of objects and R be a binary relation $R \subseteq X \times X$. R is* transitive *if and only if for all $x, x', x'' \in X$ we have that $(x, x') \in R \wedge (x', x'') \in R \Rightarrow (x, x'') \in R$. A* correct evaluation order *is a permutation $s \in S_X$ of the elements of the set X such that for all $1 \le i < j \le |s|$ we have that $(s[j], s[i]) \notin R$.*

Definition 7 (Strict Partial Order). *Let S be a set of objects. Let $s, s' \in S$. A* strict partial order $\prec$ *over S is a binary relation that satisfies the following properties:*

- *Irreflexivity: $s \prec s$ is false.*

- *Transitivity: see Definition 6.*

- *Antisymmetry: $s \prec s'$ implies that $s' \prec s$ is false. Implied by irreflexivity and transitivity [22].*

Definition 8 (Directed Graph). *A* directed graph *G is a tuple (V, E) where V is the set of vertices and $E \subseteq V \times V$ is the set of directed edges. The set U_G is the* graph universe. *A* path *in a directed graph $G = (V, E)$ is a sequence of vertices $p \in V$ such that for all $1 < i < |p| - 1$ we have that $(p_i, p_{i+1}) \in E$. We denote with P_G the set of all such possible paths over the graph G. Given two vertices $v, v' \in V$, we denote with $p_G(v, v')$ the set of all paths beginning in v and ending in v': $p_G(v, v') = \{p \in P_G \mid p[1] = v \wedge p[|p|] = v'\}$. v and v' are* connected *(and v' is* reachable *from v), denoted by $v \overset{G}{\mapsto} v'$, if and only if there exists a path between them in G: $p_G(v, v') \ne \emptyset$. Conversely, $v \overset{G}{\not\mapsto} v' \Leftrightarrow p_G(v, v') = \emptyset$. We omit the superscript G if it is clear from the context. A directed graph G is* acyclic *if there exists no path $p \in P_G$ satisfying $p[1] = p[|p|]$.*

Definition 9 (Topological Sorting). *Let $G = (V, E) \in U_G$ be an acyclic directed graph. A* topological sorting *[25] $o_G \in S_V$ is a permutation of the vertices of G such that for all $1 \le i < j \le |o_G|$ we have that $o_G[j] \not\mapsto o_G[i]$. We denote with $O_G \subseteq S_V$ all such possible topological sortings over G.*

Definition 10 (Transitive Reduction). *A transitive reduction of a graph $G = (V, E) \in U_G$ [10] is the function $\varrho \colon U_G \to U_G$ such that for the graph $\varrho(G) = (V, E_r)$ we have $E_r \subseteq E$ and every pair of vertices connected in $\varrho(G)$ is not connected by any other path: for all $(v, v') \in E_r$, $p_G(v, v') = \{\langle v, v' \rangle\}$. $\varrho(G)$ is the graph with the minimal number of edges that maintain the reachability between edges of G. The transitive reduction of a directed acyclic graph always exists and is unique [10].*

3.2 Process Mining Definitions

Let us now define the basic artifacts needed to perform process mining.

Definition 11 (Universes). *Let U_I be the set of all the* event identifiers. *Let U_C be the set of all the* case ID identifiers. *Let U_A be the set of all the* activity identifiers. *Let U_T be the totally ordered set of all the* timestamp identifiers.

Definition 12 (Events and event logs). *Let us denote with $E_C = U_I \times U_C \times U_A \times U_T$ the universe of* certain events. *A* certain event log *is a set of events $L_C \subseteq E_C$ such that every event identifier in L_C is unique.*

Definition 13 (Simple certain traces and logs). *Let $\{(e_1, c_1, a_1, t_1), (e_2, c_2, a_2, t_2), \ldots, (e_n, c_n, a_n, t_n)\} \subseteq L_C$ be a set of certain events such that $c_1 = c_2 = \cdots = c_n$ and $t_1 < t_2 < \cdots < t_n$. A* simple certain trace *is the sequence of activities $\langle a_1, a_2, \ldots, a_n \rangle \in U_A{}^*$ induced by such a set of events. $T = U_A{}^*$ denotes the universe of certain traces. $L \in \mathcal{B}(T)$ is a* simple certain log. *We will drop the qualifier "simple" if it is clear from the context.*

As a preliminary application of process mining over uncertain event data, we will consider conformance checking. Starting from an event log and a process model, conformance checking verifies if the event data in the log conforms to the model, providing a diagnostic of the deviations. Conformance checking serves many purposes, such as checking if process instances follow a specific normative model, assessing if a certain execution log has been generated from a specific model, or verifying the quality of a process discovery technique.

The conformance checking algorithm that we are applying in this paper is based on *alignments*. Introduced by Adriansyah [7], conformance checking through alignments finds deviations between a trace and a Petri net model of a process by creating a correspondence between the sequence of activities executed in the trace and the firing of the transitions in the Petri net. The following definitions are partially from [3].

Definition 14 (Petri Net). *A Petri net is a tuple $N = (P, T, F)$ with P the set of places, T the set of transitions, $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation. A Petri net $N = (P, T, F)$ defines a directed graph (V, E) with vertices $V = P \cup T$ and edges $E = F$. A marking $M \in \mathcal{B}(P)$ is a multiset of places.*

A marking defines the state of a Petri net, and indicates how many *tokens* each place contains. For any $x \in P \cup T$, $\overset{N}{\bullet} x = \{x' \mid (x', x) \in F\}$ denotes the set of input nodes and $x \overset{N}{\bullet} = \{x' \mid (x, x') \in F\}$ denotes the set of output nodes. We omit the superscript N if it is clear from the context.

A transition $t \in T$ is *enabled* in marking M of net N, denoted as $(N, M)[t\rangle$, if each of its input places $\bullet t$ contains at least one token. An enabled transition t may *fire*, i.e., one token is removed from each of the input places $\bullet t$ and one token is produced for each of the output places $t\bullet$. Formally: $M' = (M \setminus \bullet t) \uplus t\bullet$ is the marking resulting from firing enabled transition t in marking M of Petri net N. $(N, M)[t\rangle(N, M')$ denotes that t is enabled in M and firing t results in marking M'.

Let $\sigma_T = \langle t_1, t_2, \ldots, t_n \rangle \in T^*$ be a sequence of transitions. $(N, M)[\sigma_T\rangle(N, M')$ denotes that there is a set of markings $M_0, M_1, \ldots, M_n$ such that $M_0 = M$, $M_n = M'$, and $(N, M_i)[t_{i+1}\rangle(N, M_{i+1})$ for $0 \leq i < n$. A marking M' is *reachable* from M if there exists a σ_T such that $(N, M)[\sigma_T\rangle(N, M')$.

Definition 15 (Labeled Petri Net). *A labeled Petri net $N = (P, T, F, l)$ is a Petri net (P, T, F) with labeling function $l\colon T \nrightarrow U_A$ where U_A is some universe of activity labels. Let $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in U_A^*$ be a sequence of activities. $(N, M)[\sigma \triangleright (N, M')$ if and only if there is a sequence $\sigma_T \in T^*$ such that $(N, M)[\sigma_T\rangle(N, M')$ and $l(\sigma_T) = \sigma$.*

If $t \notin dom(l)$, it is called *invisible*. To indicate invisible transitions, we use the place-holder symbol $\tau \notin U_A$; for any invisible transition t we define $l(t) = \tau$. An occurrence of visible transition $t \in dom(l)$ corresponds to observable activity $l(t)$.

Definition 16 (System Net). *A system net is a triplet $SN = (N, M_{init}, M_{final})$ where $N = (P, T, F, l)$ is a labeled Petri net, $M_{init} \in \mathcal{B}(P)$ is the initial marking, and $M_{final} \in \mathcal{B}(P)$ is the final marking. U_{SN} is the universe of system nets. Over a system net we define the following:*

- $T_v(SN) = dom(l)$ *is the set of* visible transitions *in SN,*

- $A_v(SN) = rng(l)$ *is the set of corresponding* observable activities *in SN,*

- $T_v^u(SN) = \{t \in T_v(SN) \mid \forall_{t' \in T_v(SN)} \, l(t) = l(t') \Rightarrow t = t'\}$ *is the set of* unique visible transitions in SN (i.e., there are no other transitions having the same visible label),*

- $A_v^u(SN) = \{l(t) \mid t \in T_v^u(SN)\}$ *is the set of corresponding* unique observable activities *in SN,*

- $\phi(SN) = \{\sigma \mid (N, M_{init})[\sigma \triangleright (N, M_{final})]\}$ *is the set of* visible *traces starting in M_{init} and ending in M_{final}, and*

- $\phi_f(SN) = \{\sigma_T \mid (N, M_{init})[\sigma_T\rangle(N, M_{final})\}$ *is the corresponding set of complete firing sequences.*

Figure 3 shows a system net with initial and final markings M_{init} = [*start*] and M_{final} = [*end*]. Given a system net, $\phi(SN)$ is the set of all possible *visible* activity sequences, i.e., the labels of complete firing sequences starting in M_{init} and ending in M_{final} projected onto the set of observable activities. Given the set of activity sequences $\phi(SN)$ obtainable via complete firing sequences on a certain system net, we can define a perfectly fitting event log as a set of traces which activity projection is contained in $\phi(SN)$.

3.3 Conformance Checking Definitions

The task of conformance checking consist in comparing an event log and a model, in order to assess the deviations of event data with respect to the expected behavior of the process. This is usually done to verify if the process conforms to a *de iure* model designed by process experts, which describes how the process should ideally run. We will now describe a conformance checking technique, in order to extend it to the uncertain setting.

Definition 17 (Perfectly Fitting Log). *Let $L \in \mathcal{B}(T)$ be a certain event log and let $SN = (N, M_{init}, M_{final}) \in U_{SN}$ be a system net. L is perfectly fitting SN if and only if $\{\sigma \in L\} \subseteq \phi(SN)$.*

The definitions described so far allow us to build *alignments* in order to compute the fitness of trace on a certain model. An alignment is a correspondence between a sequence of activities (extracted from the trace) and a sequence of transitions with the relative labels (fired in the model while replaying the trace). The first sequence indicates the "moves in the log" and the second indicates the "moves in the model". If a move in the model cannot be mimicked by a move in the log, then a "≫" ("no move") appears in the top row; conversely, if a move in the log cannot be mimicked by a move in the model, then a "≫" ("no move") appears in the bottom row. "no moves" not corresponding to invisible transitions point to deviations between the model and the log. A *move* is a pair $(x, (y, t))$ where the first element refers to the log and the second element to the model. A "≫" in the first element of the pair indicates a move on the model, while a "≫" in the second element indicates a move on the log.

Definition 18 (Legal Moves). *Let $L \in \mathcal{B}(T)$ be a certain event log, let $A \subseteq U_A$ be the set of activity labels appearing in the event log, and let $SN = (N, M_{init}, M_{final}) \in U_{SN}$ be a system net with $N = (P, T, F, l)$. $A_{LM} = \{(x, (x, t)) \mid x \in A \land t \in T \land l(t) = x\} \cup \{(\gg, (x, t)) \mid t \in T \land l(t) = x\} \cup \{(x, \gg) \mid x \in A\}$ is the set of legal moves.*

$$\gamma_1 = \begin{array}{|c|c|c|c|c|} \hline a & d & b & e & h \\ \hline a & d & b & e & h \\ \hline \end{array} \qquad\qquad \gamma_2 = \begin{array}{|c|c|c|c|c|c|} \hline a & \gg & d & b & e & h \\ \hline a & b & d & b & e & h \\ \hline \end{array}$$
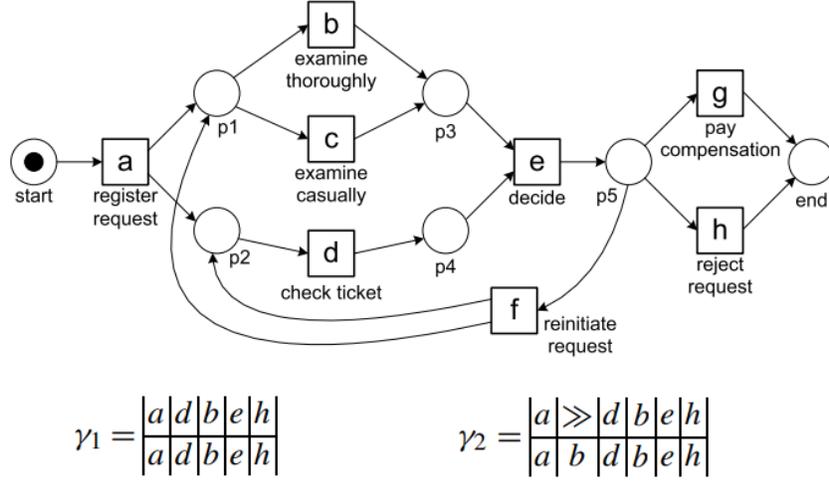
Figure 3: Example of alignments on a system net. The alignment γ_1 shows that the trace $\langle a, d, b, e, h \rangle$ is perfectly fitting the net. The alignment γ_2 shows that the trace $\langle a, b, d, b, e, h \rangle$ is misaligned with the net in one point, indicated by "$\gg$". Partially from [3].

An alignment is a sequence of legal moves such that after removing all "$\gg$" symbols, the top row corresponds to a trace in the log and the bottom row corresponds to a firing sequence starting in M_{init} and ending in M_{final}. Notice that if $t \notin dom(l)$ is an invisible transition, the activation of t is indicated by a "$\gg$" on the log in correspondence of t and the placeholder label τ. Hence, the middle row corresponds to a visible path when ignoring the τ steps. Figure 3 shows a system net with two examples of alignments, σ_1 of a fitting trace and σ_2 of a non-fitting trace.

Definition 19 (Alignment). *Let $\sigma \in L$ be a certain trace and $\sigma_T \in \phi_f(SN)$ a complete firing sequence of system net SN. An* alignment *of σ and σ_T is a sequence $\gamma \in A_{LM}{}^*$ such that the projection on the first element (ignoring "$\gg$") yields σ and the projection on the last element (ignoring "$\gg$" and transition labels) yields σ_T.*

A trace and a model can have several possible alignments. In order to select the most appropriate one, we introduce a function that associates a *cost* to undesired moves - the ones associated with deviations.

Definition 20 (Cost of Alignment). *Cost function $\delta \colon A_{LM} \to \mathbb{N}$ assigns costs to legal moves. The* cost *of an alignment $\gamma \in A_{LM}{}^*$ is the sum of all costs: $\delta(\gamma) = \sum_{(x,y) \in \gamma} \delta(x, y)$.*

Moves where log and model agree have no costs, i.e., $\delta(x, (x, t)) = 0$ for all $x \in A$. Moves on model only have no costs if the transition is invisible, i.e., $\delta(\gg, (\tau, t)) = 0$

if $l(t) = \tau$. $\delta(\gg, (x, t)) > 0$ is the cost when the model makes an "x move" without a corresponding move of the log (assuming $l(t) = x \neq \tau$). $\delta(x, \gg) > 0$ is the cost for an "x move" only on the log. In this paper, we often use a standard cost function δ_S that assigns unit costs: $\delta_S(x, (x, t)) = 0$, $\delta_S(\gg, (\tau, t)) = 0$, and $\delta_S(\gg, (x, t)) = \delta_S(x, \gg) = 1$ for all $x \in A$.

Definition 21 (Optimal Alignment). *Let $L \in \mathcal{B}(T)$ be a certain event log and let $SN \in U_{SN}$ be a system net with $\phi(SN) \neq \emptyset$.*

- *For $\sigma \in L$, we define: $\Gamma_{\sigma,SN} = \{\gamma \in A_{LM}{}^* \mid \exists_{\sigma_T \in \phi_f(SN)} \gamma$ is an alignment of σ and $\sigma_T\}$.*

- *An alignment $\gamma \in \Gamma_{\sigma,SN}$ is* optimal *for trace $\sigma \in L$ and system net SN if for any $\gamma' \in \Gamma_{\sigma,SN}$: $\delta(\gamma') \geq \delta(\gamma)$.*

- *$\lambda_{SN}: T \to A_{LM}{}^*$ is a deterministic mapping that assigns any trace σ to an optimal alignment, i.e., $\lambda_{SN}(\sigma) \in \Gamma_{\sigma,SN}$ and $\lambda_{SN}(\sigma)$ is optimal.*

- *$costs(L, SN, \delta) = \sum_{\sigma \in L} \delta(\lambda_{SN}(\sigma))$ are the* misalignment costs *of the whole event log.*

$\sigma \in L$ is a (perfectly) fitting trace for the system net SN if and only if $\delta(\lambda_{SN}(\sigma)) = 0$. L is a (perfectly) fitting event log for the system net SN if and only if $costs(L, SN, \delta) = 0$.

The technique to compute the optimal alignment [7] is as follows. Firstly, it creates an *event net*, a sequence-structured system net able to replay only the trace to align. The transitions in the event net have labels corresponding to the activities in the trace. Then, a *product net* should be computed. A product net is the union of the event net and the model together, with synchronous transitions added. These additional transitions are paired with transitions in the event net and in the process model that have the same label. Then, they are connected with arcs from the input places and to the output places of those transitions. The product net is able to represent moves on log, moves on model and synchronous moves by means of firing transitions. In fact, the transitions of the event net correspond to moves on log, the transitions of the process model correspond to moves on model, the added synchronous transitions correspond to synchronous moves. The union of the initial and final markings of the event net and the process model constitute respectively the initial and final marking of the product net, while every complete firing sequence on the product net corresponds to a possible alignment. Lastly, the product net is translated to a state space, and a state space exploration via the $\mathbb{A}^*$ algorithm is performed in order to find the complete firing sequence that yields the lowest cost.

Let us define formally the construction of the event net and the product net:

Definition 22 (Event Net). *Let $\sigma \in T$ be a certain trace. The* event net *$en : T \to U_{SN}$ of σ is a system net $en(\sigma) = (P, T, F, l, M_{init}, M_{final})$ such that:*

- $P = \{p_i \mid 1 \le i \le |\sigma| + 1\}$,

- $T = \{t_i \mid 1 \le i \le |\sigma|\}$,

- $F = \bigcup_{1 \le i \le |\sigma|} \{(p_i, t_i), (t_i, p_{i+1})\}$

- $l\colon T \to U_A$ such that for all $1 \le i \le |\sigma|$, $l(t_i) = \sigma[i]$,

- $M_{init} = [p_1]$,

- $M_{final} = [p_{|P|}]$.

Note that the labeling function l of an event net is a total function: no invisible transitions are contained in an event net, since for each event we generate a transition labeled with the corresponding activity label.

Definition 23 (Product of two Petri Nets [41]). *Let* $S_1 = (P_1, T_1, F_1, l_1, M_{init_1}, M_{final_1})$ *and* $S_2 = (P_2, T_2, F_2, l_2, M_{init_2}, M_{final_2})$ *be two system nets. The* product net *of* S_1 *and* S_2 *is the system net* $S = S_1 \otimes S_2 = (P, T, F, l, M_{init}, M_{final})$ *such that:*

- $P = P_1 \cup P_2$,

- $T \subseteq (T_1 \cup \{\gg\} \times T_2 \cup \{\gg\})$ *such that* $T = \{(t_1, \gg) \mid t_1 \in T_1\} \cup \{(\gg, t_2) \mid t_2 \in T_2\} \cup \{(t_1, t_2) \in (T_1 \times T_2) \mid l_1(t_1) = l_2(t_2) \ne \tau\}$,

- $F \subseteq (P \times T) \cup (T \times P)$ *such that*

$$
\begin{aligned}
F = \ &\{(p_1, (t_1, \gg)) \mid p_1 \in P_1 \wedge t_1 \in T_1 \wedge (p_1, t_1) \in F_1\} \cup \\
&\{((t_1, \gg), p_1) \mid t_1 \in T_1 \wedge p_1 \in P_1 \wedge (t_1, p_1) \in F_1\} \cup \\
&\{(p_2, (t_2, \gg)) \mid p_2 \in P_2 \wedge t_2 \in T_2 \wedge (p_2, t_2) \in F_2\} \cup \\
&\{((t_2, \gg), p_2) \mid t_2 \in T_2 \wedge p_2 \in P_2 \wedge (t_2, p_2) \in F_2\} \cup \\
&\{(p_1, (t_1, t_2)) \mid p_1 \in P_1 \wedge (t_1, t_2) \in T \cap (T_1 \times T_2) \wedge (p_1, t_1) \in F_1\} \cup \\
&\{(p_2, (t_1, t_2)) \mid p_2 \in P_2 \wedge (t_1, t_2) \in T \cap (T_1 \times T_2) \wedge (p_2, t_2) \in F_2\} \cup \\
&\{((t_1, t_2), p_1) \mid p_1 \in P_1 \wedge (t_1, t_2) \in T \cap (T_1 \times T_2) \wedge (t_1, p_1) \in F_1\} \cup \\
&\{((t_1, t_2), p_2) \mid p_2 \in P_2 \wedge (t_1, t_2) \in T \cap (T_1 \times T_2) \wedge (t_2, p_2) \in F_2\}
\end{aligned}
$$

- $l\colon T \to U_A$ *such that for all* $(t_1, t_2) \in T$, $l((t_1, t_2)) = l_1(t_1)$ *if* $t_2 = \gg$, $l((t1, t2)) = l_2(t_2)$ *if* $t_1 = \gg$, *and* $l((t_1, t_2)) = l_1(t_1)$ *otherwise,*

- $M_{init} = M_{init_1} \uplus M_{init_2}$,

- $M_{final} = M_{final_1} \uplus M_{final_2}$.

4   Uncertainty in Process Mining

In this section, we will extend the definitions of event, trace, and event log to the uncertain case. Let us first define the identifiers necessary to express event indeterminacy.

Definition 24 (Determinate and indeterminate event qualifiers). *Let* $U_O =$ $\{!, ?\}$, *where the "!" symbol denotes* determinate events, *and the "?" symbol denotes* indeterminate events.

For strong uncertainty, attribute values are replaced by a set of possible values. In the case of weak uncertainty, a continuous function f provides the probability density for the combinations of attribute values in the uncertain event. Notice that the total mass of probabilities described by f might be lower than 1: this is so we can aptly represent the case of an indeterminate event.

Definition 25 (Uncertain events). *Let* $E_S = U_I \times \mathcal{P}_{NE}(U_C) \times \mathcal{P}_{NE}(U_A) \times \mathcal{P}_{NE}(U_T) \times U_O$ *denote the universe of* strongly uncertain events. $E_W = \{(e_i, f) \in U_I \times ((U_C \times U_A \times U_T) \nrightarrow [0,1]) \mid \sum_{(c,a,t)\in dom(f)} f(c, a, t) \leq 1\}$ *is the universe of* weakly uncertain events[1].

The probability of a weakly uncertain event of having been recorded but not happening in reality is equal to $1 - \sum_{(c,a,t)\in dom(f)} f(c, a, t)$.

Now that the definitions of strongly and weakly uncertain events are given, let us aggregate them in uncertain event logs.

Definition 26 (Uncertain event logs). *A* strongly uncertain event log *is a set of events* $L_S \subseteq E_S$ *such that every event identifier in* L_S *is unique. A* weakly uncertain event log *is a set of events* $L_W \subseteq E_W$ *such that every event identifier in* L_W *is unique.*

For a strongly uncertain event $e = (e_i, c_s, a_s, t_s, o) \in L_S$ *we define the following projection functions:* $\pi_c^{L_S}(e) = c_s \in \mathcal{P}_{NE}(U_C)$, $\pi_a^{L_S}(e) = a_s \in \mathcal{P}_{NE}(U_A)$, $\pi_t^{L_S}(e) = t_s \in \mathcal{P}_{NE}(U_T)$ *and* $\pi_o^{L_S}(e) = o \in U_O$.

A weakly uncertain event log $L_W \subseteq E_W$ has a corresponding strongly uncertain

---

[1] We assume here that $dom(f)$ is finite. It is easy to generalize to the infinite case by employing an integral.

event log $\overline{L_W} = L_S \subseteq E_S$ such that

$$
\begin{aligned}
L_S = \{(e_i, c_s, a_s, t_s, o) \in E_S \mid {}& \exists_{(e_{i'}, f) \in L_W} e_i = e_i' \wedge \\
& c_s = \{c \in U_C \mid \exists_{a,t} (c, a, t) \in dom(f) \wedge f(c, a, t) > 0\} \wedge \\
& a_s = \{a \in U_A \mid \exists_{c,t} (c, a, t) \in dom(f) \wedge f(c, a, t) > 0\} \wedge \\
& t_s = \{t \in U_T \mid \exists_{c,a} (c, a, t) \in dom(f) \wedge f(c, a, t) > 0\} \wedge \\
& (o = {!} \Leftrightarrow \sum_{(c,a,t) \in dom(f)} f(c, a, t) = 1) \wedge \\
& (o = {?} \Leftrightarrow \sum_{(c,a,t) \in dom(f)} f(c, a, t) < 1)\}.
\end{aligned}
$$

Notice that representing the density of probability for combinations of values of case ID, time and activity with a single function f is an approximation that assumes probabilistic independence between event attributes.

Definition 27 (Realization of an event log). *$L_C \subseteq E_C$ is a* realization *of $L_S \subseteq E_S$ if and only if:*

- *For all $(e_i, c, a, t) \in L_C$ there is a distinct $(e_i', c_s, a_s, t_s, o) \in L_S$ such that $e_i = e_i'$, $c \in c_s$, $a \in a_s$ and $t \in t_s$;*

- *For all $(e_i, c_s, a_s, t_s, o) \in L_S$ with $o = {!}$ there is a distinct $(e_i', c, a, t) \in L_C$ such that $e_i = e_i'$, $c \in c_s$, $a \in a_s$ and $t \in t_s$.*

$\mathcal{R}_L(L_S)$ is the set of all such realizations of the log L_S.

Note that these definitions allow us to transform a weakly uncertain log into a strongly uncertain one, and a strongly uncertain one in a set of certain logs.

In this paper, we focus on three types of uncertainty:

- Strong uncertainty on the activity;

- Strong uncertainty on the timestamp;

- Strong uncertainty on indeterminate events.

All three can happen concurrently. Following the taxonomy presented in Section 2, this setting corresponds to a $[E, A, T]_{\mathbb{S}}$-type log. It is worth noting that the specific case of uncertainty on the case ID causes a problem; since an event can have many possible case IDs, it can belong to different traces. In data format where the events are already aggregated into traces, such as the very common XES standard, this means that the information related to a trace can be *non-local* to the trace itself, but can be stored in some other

points of the log. We will focus on the problem of uncertainty on the case ID attribute in future work.

Firstly, we will lay down some simplified notation in order to model the problem at hand in a more compact way.

Definition 28 (Simple uncertain events, traces and logs). *Let $e_i \in U_I$, $a_s \in \mathcal{P}_{NE}(U_A)$, $t_{min} \in U_T$, $t_{max} \in U_T$ and $o \in U_O$ such that $t_{min} < t_{max}$. $e_U^S = (e_i, a_s, t_{min}, t_{max}, o)$ is a* simple uncertain event. *Let us denote with $E_U^S \subseteq U_I \times \mathcal{P}_{NE}(U_A) \times U_T \times U_T \times U_O$ the universe of all simple uncertain events. $\sigma_U \subseteq E_U^S$ is a* simple uncertain trace *if all the event identifiers in σ_U are unique. T_U denotes the universe of simple uncertain traces. $L_U \in \mathcal{P}(T_U)$ is a* simple uncertain log *if all the event identifiers in L_U are unique. For $\sigma_U \in L_U$ and $e_U^S = (e_i, a_s, t_{min}, t_{max}, o) \in \sigma_U$ we define the following projection functions: $\pi_a^{L_U}(e_U^S) = a_s \in \mathcal{P}_{NE}(U_A)$, $\pi_{t_{min}}^{L_U}(e_U^S) = t_{min} \in U_T$, $\pi_{t_{max}}^{L_U}(e_U^S) = t_{max} \in U_T$ and $\pi_o^{L_U}(e_U^S) = o \in U_O$.*

In a simple uncertain event $e_U^S = (e_i, a_s, t_{min}, t_{max}, o)$, the true activity label of the event is one of the labels contained in the set a_s, the true timestamp is one of the values contained in the closed interval $[t_{min}, t_{max}]$, while the indeterminacy symbol o indicates whether the event has certainly occurred, or if it is possible that it did not occur even though it has been recorded in an event log.

Simple uncertain events are best illustrated with a running example. Let us consider the following process instance, a simplified version of anomalies that are actually occurring in processes of the healthcare domain. An elderly patient enrolls in a clinical trial for an experimental treatment against myeloproliferative neoplasms, a class of blood cancers. The enrollment in this trial includes a lab exam and a visit with a specialist; then, the treatment can begin. The lab exam, performed on the 8th of July, finds a low level of platelets in the blood of the patient, a condition known as thrombocytopenia (TP). At the visit, on the 10th of July, the patient self-reports an episode of night sweats on the night of the 5th of July, prior the lab exam: the medic notes this, but also hypothesized that it might not be a symptom, since it can be caused not by the condition but by external factors (such as very warm weather). The medic also reads the medical records of the patient and sees that, shortly prior to the lab exam, the patient was undergoing a heparine treatment (a blood-thinning medication) to prevent blood clots. The thrombocytopenia found with the lab exam can then be primary (caused by the blood cancer) or secondary (caused by other factors, such as a drug). Finally, the medic finds an enlargement of the spleen in the patient (splenomegaly). It is unclear when this condition has developed: it might have appeared at any moment prior to that point. The medic decides to admit the patient in the clinical trial, starting 12th of July. These events are collected and recorded in the trace shown in Table 5 in the information system of the hospital. For readability, the timestamp field only indicates the day of the month. This

Table 5: The uncertain trace of an instance of healthcare process used as a running example. For sake of clarity, we have further simplified the notation in the timestamps column, by showing only the day of the month.

Case ID	Event ID	Timestamp	Activity	Indet. event
ID192	e_1	5	*NightSweats*	?
ID192	e_2	8	$\{PrTP, SecTP\}$	!
ID192	e_3	$[4, 10]$	*Splenomeg*	!
ID192	e_4	12	*Adm*	!

trace includes all types of uncertainty contained in a $[E, A, T]_{\mathbb{S}}$-type log, the setting we are considering for the application of conformance checking.

In the notation of Definition 28, the trace σ_U in Table 5 is denoted as:

$$\sigma_U = \{(e_1, \{NightSweats\}, 5, 5, ?), (e_2, \{PrTP, SecTP\}, 8, 8, !),$$
$$(e_3, \{Splenomeg\}, 4, 10, !), (e_4, \{Adm\}, 12, 12, !)\}.$$

We can also draw the time diagram of this example of uncertain trace, which can be seen in Figure 4.
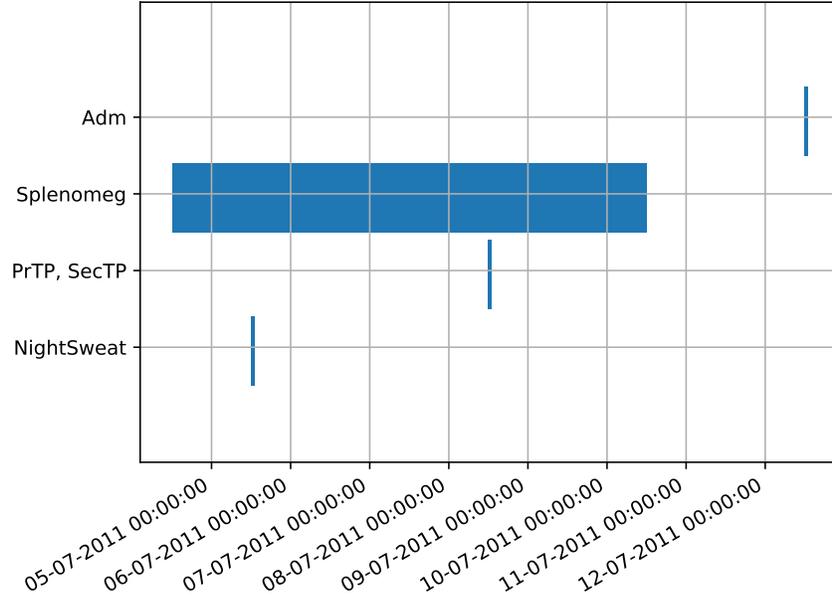
Figure 4: Time diagram of the trace in Table 5.

In the remainder of the paper, when defining simple uncertain traces and events,

we always assume that these belong to a corresponding simple uncertain log. Thus, for simplicity, we will omit the qualifier "L_U" when denoting the corresponding projection functions.

These simplified traces and logs can be related to the more general framework described in the previous section through the following transformation: let $L_S \subseteq E_S$ be a strongly uncertain log and let $g \colon U_I \nrightarrow U_C$ be a function mapping event identifiers onto cases such that $dom(g) = \{e_i \mid (e_i, c_s, a_s, t_s, u) \in L_S\}$ and for all $(e_i, c_s, a_s, t_s, u) \in L_S$, $g(e_i) \in c_s$. Thus, for $c \in rng(g), g^{-1}(c) = \{e_i \in U_I \mid g(e_i) = c\}$. The simple uncertain event log defined by g on L_S is given as $L_U = \{\{(e_i, \pi_a^{L_S}(e), min(\pi_t^{L_S}(e)), max(\pi_t^{L_S}(e)), \pi_o^{L_S}(e)) \mid e_i \in g^{-1}(c) \wedge \pi_i^{L_S}(e) = e_i\} \mid c \in rng(g)\}$.

In order to more easily work with timestamps in simple uncertain events, let us frame their time relationship as a strict partial order.

Definition 29 (Strict partial order over simple uncertain events). *Let $e, e' \in E_U^S$ be two simple uncertain events. $\prec_E$ is a strict partial order defined on the universe of strongly uncertain events E_U^S as:*

$$e \prec_E e' \Leftrightarrow \pi_{t_{max}}(e) < \pi_{t_{min}}(e')$$

Proposition 1 ($\prec_E$ is a strict partial order).

Proof. All properties characterizing strict partial orders are fulfilled by $\prec_E$. For all e, e', $e'' \in E_U^S$ we have:

- Irreflexivity: this property is always verified, since $\pi_{t_{max}}(e) < \pi_{t_{min}}(e)$ is false (see Definition 25).

- Transitivity: since $\pi_{t_{max}}(e) < \pi_{t_{min}}(e') \leq \pi_{t_{max}}(e') < \pi_{t_{min}}(e'')$ and U_T is totally ordered, we have that $\pi_{t_{max}}(e) < \pi_{t_{min}}(e'')$ and this property is always verified.

$\square$

Lemma 1 (Uncomparable events share possible timestamp values). *Let $e, e' \in E_U^S$ be two strongly uncertain events. e and e' are uncomparable with respect to the strict partial order $\prec_E$ (i.e., neither $e \prec_E e'$ nor $e' \prec_E e$ are true) if and only if e and e' share some possible values of their timestamp.*

Proof.
($\Rightarrow$) From Definition 29, it follows that two events $e, e' \in E_U^S$ are comparable if and only if either $\pi_{t_{max}}(e) < \pi_{t_{min}}(e')$ or $\pi_{t_{max}}(e') < \pi_{t_{min}}(e)$. If both are false, then $\pi_{t_{min}}(e') \leq \pi_{t_{max}}(e)$ and $\pi_{t_{min}}(e) \leq \pi_{t_{max}}(e')$. If we assume that $\pi_{t_{min}}(e) \leq \pi_{t_{min}}(e')$ then $\pi_{t_{min}}(e) \leq \pi_{t_{min}}(e') \leq \pi_{t_{max}}(e)$, while if $\pi_{t_{min}}(e) > \pi_{t_{min}}(e')$ then $\pi_{t_{min}}(e') < \pi_{t_{min}}(e) \leq \pi_{t_{max}}(e')$. In both cases, there are values common to both uncertain timestamps.

($\Leftarrow$) If the two events share timestamp values, it follows that at least one of the extremes of one event is encompassed by the extremes of the other. Assume that e encompasses at least one of the extremes of e' (the other case is symmetric): then either $\pi_{t_{min}}(e) \leq \pi_{t_{min}}(e') \leq \pi_{t_{max}}(e)$ or $\pi_{t_{min}}(e) \leq \pi_{t_{max}}(e') \leq \pi_{t_{max}}(e)$. In the first case, considering that U_T is totally ordered and that $\pi_{t_{min}}(e') \leq \pi_{t_{max}}(e')$, we have that both $\pi_{t_{min}}(e') \leq \pi_{t_{max}}(e)$ and $\pi_{t_{min}}(e) \leq \pi_{t_{max}}(e')$ are true, and e and e' are uncomparable. The second case is proved analogously. $\qquad\square$

Definition 30 (Realizations of simple uncertain traces). *Let $\sigma_U \in T_U$ be a simple uncertain trace. An* order-realization *$\sigma_O \in S_{\sigma_U}$ is a permutation of the events in σ_U such that for all $1 \leq i < j \leq |\sigma_O|$ we have that $\sigma_O[j] \not\prec_E \sigma_O[i]$, i.e., σ_O is a correct evaluation order for σ_U over $\prec_E$, and the (total) order in which events are sorted in σ_O is a linear extension of the strict partial order $\prec_E$. We denote with $\mathcal{R}_O(\sigma_U)$ the set of all such order-realizations of the trace σ_U.*

Given an order-realization $\sigma_O \in \mathcal{R}_O(\sigma_U)$, the sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in U_A{}^*$ is a realization *of σ_O if there exists a total function $f : \{1, 2, \ldots, n\} \rightarrow \sigma_O$ such that:*

- *For all $1 \leq i \leq n$, $a_i \in \pi_a(f(i))$,*

- *$\langle f(1), f(2), \ldots, f(n) \rangle$ is a subsequence of σ_O,*

- *For all $e \in \sigma_O$ with $\pi_o(\sigma_O) = !$ there exists $1 \leq i \leq n$ such that $f(i) = e$.*

We denote with $\mathcal{R}'(\sigma_O) \subseteq U_A{}^*$ the set of all such realizations of the order-realization σ_O. We denote with $\mathcal{R}(\sigma_U) \subseteq U_A{}^*$ the union of the realizations obtainable from all the order-realizations of σ_U: $\mathcal{R}(\sigma_U) = \bigcup_{\sigma_O \in \mathcal{R}_O(\sigma_U)} \mathcal{R}'(\sigma_O)$.

Let us see some examples of realizations of uncertain traces. Let σ_U be the uncertain trace shown in Table 5. We then have that σ_U has three order-realizations:

$$\mathcal{R}_O(\sigma_U) = \{\langle e_3, e_1, e_2, e_4 \rangle, \langle e_1, e_3, e_2, e_4 \rangle, \langle e_1, e_2, e_3, e_4 \rangle\}$$

We can then compute the realizations of one of the order-realizations of σ_U:

$$\mathcal{R}'(\langle e_1, e_2, e_3, e_4 \rangle) = \{\langle NightSweats, PrTP, Splenomeg, Adm \rangle,$$
$$\langle NightSweats, SecTP, Splenomeg, Adm \rangle,$$
$$\langle PrTP, Splenomeg, Adm \rangle,$$
$$\langle SecTP, Splenomeg, Adm \rangle\}$$

Simple uncertain traces and logs carry less information than their certain counterparts. Nevertheless, it is possible to extend existing process mining algorithms to extract the information in a simple uncertain log to design a process model that describes its possible behavior, or verify that it conforms to a given normative model.

5   Conformance Checking on Uncertain Event Data

Depending on the possible values for a_s, t_{min}, t_{max}, and u there are multiple possible realizations of a trace. This means that, given a model, a simple uncertain trace could be fitting for certain realizations, but non-fitting for others. The question we are interested in answering is: given a simple uncertain trace and a Petri net process model, is it possible to find an *upper and lower bound* for the conformance score? Usually we are interested in the optimal alignments (the ones with the minimal cost). However, we are now interested in the minimum and maximum cost of alignments in the realization set of a simple uncertain trace.

Definition 31 (Upper and Lower Bound on Alignment Cost for a Trace).
Let $\sigma_U \in T_U$ be a simple uncertain trace, and let $SN \in U_{SN}$ be a system net. The upper bound for the alignment cost *is a function $\delta_{max} \colon T_U \to \mathbb{N}$ such that $\delta_{max}(\sigma_U) = \max_{\sigma \in \mathcal{R}(\sigma_U)} \delta(\lambda_{SN}(\sigma))$. The* lower bound for the alignment cost *is a function $\delta_{min} \colon T_U \to \mathbb{N}$ such that $\delta_{min}(\sigma_U) = \min_{\sigma \in \mathcal{R}(\sigma_U)} \delta(\lambda_{SN}(\sigma))$.*

A simple way to compute the upper and lower bounds for the cost of any uncertain trace is using a brute-force approach: enumerating the possible realizations of the trace, then searching for the costs of optimal alignments for all the realizations, and picking the minimum and maximum as bounds. We now present a technique which improves the performance of calculating the lower bound for conformance cost with respect to a brute-force method.

We will produce a version of the event net that embeds the possible behaviors of the uncertain trace. We define a *behavior net*, a Petri net that can replay all and only the realizations of an uncertain trace. As an intermediate step in order to obtain such a Petri net, we first build the *behavior graph*, a dependency graph representing the uncertain trace. This graph contains a vertex for each uncertain event in the trace and contains an edge between two vertices if the corresponding uncertain events happen one directly after the other in at least one realization of the uncertain trace.

Definition 32 (Behavior Graph). *Let $\sigma_U \in T_U$ be a simple uncertain trace. A behavior graph $\beta \colon T_U \to U_G$ is the transitive reduction of a directed graph $\varrho(G)$, where $G = (V, E) \in U_G$ is defined as:*

- *$V = \{e \in \sigma_U\}$,*

- *$E = \{(v, w) \mid v, w \in V \wedge v \prec_E w\}$.*

The behavior graph provides a structured representation of the uncertainty on the timestamp: when a specific vertex has two or more outbound edges, the events corresponding to the destination vertices can occur in any order, concurrently with each other. We can see the result on the example trace in Figures 5 and 6.
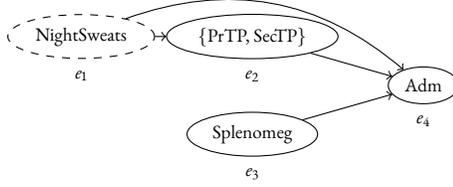
Figure 5: The graph of the trace in Table 5 before applying the transitive reduction. All the nodes in the graph are pairwise connected based on precedence relationships; pairs of nodes for which the order is unknown are not connected. The dashed node represents an indeterminate event.
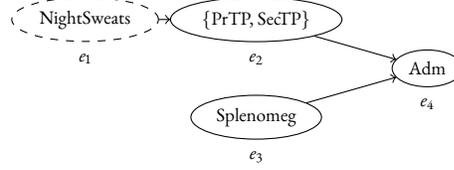
Figure 6: The behavior graph of the trace in Table 5. The transitive reduction removed the arc between e_1 and e_4, since they are reachable through e_2. This graph has a minimal number of arcs while conserving the same reachability relationship between nodes.

Theorem 1 (Correctness of behavior graphs). *Let $\sigma_U \in T_U$ be a simple uncertain trace and $bg(\sigma_U) = (V, E)$ be its behavior graph. The behavior graph $bg(\sigma_U)$ is acyclic; additionally, the set of all topological sortings of the behavior graph corresponds to the set of order-realizations of σ_U: $O_{bg(\sigma_U)} = \mathcal{R}_O(\sigma_U)$.*

Proof. From Proposition 1 we know that $\prec_E$ is a strict partial order. Let $p = \langle p_1, p_2, \ldots, p_m \rangle \in P_{bg}$ be a path in the behavior graph: if p was a cycle, that means that according to Definition 32 we have $p_1 \prec_E p_2 \prec_E \cdots \prec_E p_m \prec_E p_1$. Since $\prec_E$ is transitive, we have that $p_1 \prec_E p_m$ and $p_m \prec_E p_1$, which would violate the antisymmetry property in Definition 7 and would contradict Proposition 1. Thus the behavior graph is necessarily acyclic.

The result $O_{bg(\sigma_U)} = \mathcal{R}_O(\sigma_U)$ immediately follows from Definitions 9, 30 and 32, and from Proposition 1. ☐

Lemma 2 (Semantics of behavior graphs). *Events connected by paths in a given behavior graph have a precedence relationship; events not connected by any paths share possible values for their timestamps and thus might have happened in any order.*

Proof. Immediately follows from Proposition 1, Theorem 1, and from Lemma 1. ☐

We then obtain a *behavior net* by replacing every vertex in the behavior graph with one or more transitions in an XOR configuration, each representing an activity contained in the π_a set of the corresponding uncertain event. Every edge of the behavior graph becomes a place in the behavior net, connected from and to the transitions corresponding to, respectively, its source and target nodes in the graph.

Definition 33 (Behavior Net). *Let $\sigma_U \in T_U$ be a simple uncertain trace, and let $bg(\sigma_U) = (V, E)$ be the corresponding behavior graph. A behavior net $bn: T_U \to U_{SN}$ is a system net $bn(\sigma_U) = (P, T, F, l, M_{init}, M_{final})$ such that:*
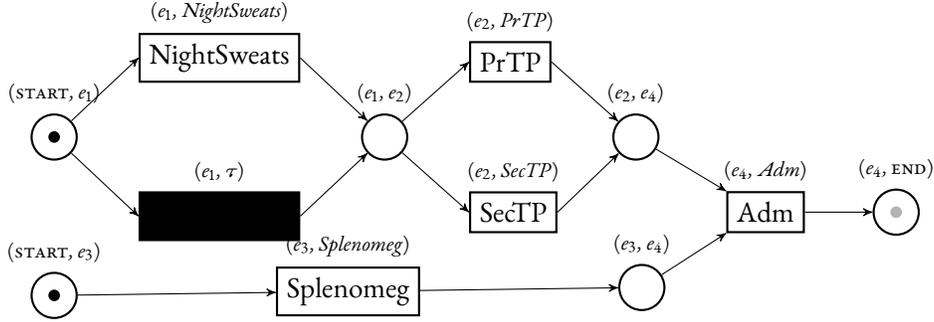
Figure 7: The behavior net corresponding to the uncertain trace in Table 5. The labels show the objects involved in the construction of Definition 33. The initial marking is displayed; the gray "token slot" represents the final marking.

- $P = E \cup$
  $\{(START, v) \mid v \in V \wedge \nexists_{v' \in V}(v', v) \in E\} \cup$
  $\{(v, END) \mid v \in V \wedge \nexists_{v' \in V}(v, v') \in E\}$,

- $T = \{(v, a) \mid v \in V \wedge a \in \pi_a(v)\} \cup \{(v, \tau) \mid v \in V \wedge \pi_o(v) = ?\}$,

- $F = \{((START, v_1), (v_2, a)) \in E \times T \mid v_1 = v_2\} \cup$
  $\{((v_1, a), (v_2, w)) \in T \times E \mid v_1 = v_2\} \cup$
  $\{((v, w_1), (w_2, a)) \in E \times T \mid w_1 = w_2\} \cup$
  $\{((v_1, a), (v_2, END)) \in T \times E \mid v_1 = v_2\}$,

- $l = \{((v, a), a) \mid (v, a) \in T \wedge a \neq \tau\}$,

- $M_{init} = [(START, v) \in P \mid v \in V]$,

- $M_{final} = [(v, END) \in P \mid v \in V]$.

In Figure 7, we can see the behavior net corresponding to the uncertain trace in Table 5. It is important to note that every set of edges in the behavior graph with the same source vertex generates an AND split in the behavior net, and a set of edges with the same destination vertex generates an AND join. At the same time, the transitions whose labels correspond to different possible activities in an uncertain event will appear in an XOR construct inside the behavior net.

Thus, in the behavior net, every set of events which timestamps share some possible values will be represented by transitions inside an AND construct, and will then be able to execute in any order allowed by their uncertain timestamp attributes. In the same fashion, an event with uncertainty on the activity will be represented by a number of

transitions in an XOR construct. This allows replaying any possible choice for the activity attribute. It follows that, by construction, for a certain simple uncertain trace σ_U we have that $\phi(bn(\sigma_U)) = \mathcal{R}_{\llcorner}(\sigma_U)$.

We can use the behavior net of an uncertain trace σ_U in lieu of the event net to compute alignments with a model $SN \in U_{SN}$; the search algorithm returns an optimal alignment, a sequence of moves $(x, (y, t))$ with $x \in U_A, y \in U_A$ and t transition of the model SN. After removing all "$\gg$" symbols, the sequence of first elements of the moves will describe a complete firing sequence σ_{bn} of the behavior net. Since σ_{bn} is complete, $\sigma_{bn} \in \phi(bn(\sigma_U))$ and, thus, $\sigma_{bn} \in \mathcal{R}_{\llcorner}(\sigma_U)$. It follows that σ_{bn} is a realization of σ_U, and the search algorithm ensures that σ_{bn} is a realization with optimal conformance cost for the model SN: $\delta(\lambda_{SN}(\sigma_{bn})) = \min_{\sigma \in \mathcal{R}_{\llcorner}(\sigma_U)} \lambda_{SN}(\sigma) = \delta_{min}(\sigma_U)$.

Theorem 2 (Correctness of behavior nets). *Let $\sigma_U \in T_U$ be a simple uncertain trace and let $bg(\sigma_U) = (V, E)$ be its behavior graph. The corresponding behavior net $bn(\sigma_U) = (P, T, F, l, M_{init}, M_{final})$ can replay all and only the realizations of σ_U: $\phi(bn(\sigma_U)) = \mathcal{R}_{\llcorner}(\sigma_U)$.*

Proof. Let $(v, v') \in E$ be an edge of the behavior graph, which also defines a place in the behavior net: $(v, v') = p_{v,v'} \in P$. Let us denote with $\mathbb{T}_v$ the set of transitions in the behavior net generated from the vertex v: $\mathbb{T}_v = \{(v', a) \in T \mid v' = v\}$.

$(\subseteq)$ Let $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in \phi(bn(\sigma_U))$ be any certain trace accepted by $bn(\sigma_U)$. Let $\sigma_T = \langle t_1, t_2, \ldots, t_n \rangle \in \phi_f(bn(\sigma_U))$ be a complete firing sequence of $bn(\sigma_U)$ yielding σ, i.e., $l(\sigma_T) \restriction_{U_A} = \sigma$. Let $\langle v_1, v_2, \ldots, v_n \rangle$ be a sequence of vertices in $bg(\sigma_U)$ such that $t_1 = (v_1, a_1), t_2 = (v_2, a_2), \ldots, t_n = (v_n, a_n)$ and $t_1 \in \mathbb{T}_{v_1}, t_2 \in \mathbb{T}_{v_2}, \ldots, t_n \in \mathbb{T}_{v_n}$. Let $\mathcal{U}$ be the set of all such sequences; by the flow relation in Definition 33 there must exist a sequence $\sigma_O = \langle v_1, v_2, \ldots, v_n \rangle \in \mathcal{U}$ such that $((v_1, a_1), (v_1, v_2)) \in F, ((v_1, v_2), (v_2, a_2)) \in F, ((v_2, a_2), (v_2, v_3)) \in F, ((v_2, v_3), (v_3, a_3)) \in F, \ldots, ((v_{n-1}, a_{n-1}), (v_{n-1}, v_n)) \in F, ((v_{n-1}, v_n), (v_n, a_n)) \in F$. This implies that $(v_1, v_2) \in E, (v_2, v_3) \in E, \ldots, (v_{n-1}, v_n) \in E$. From Definition 32 we then have that $v_1 \not\succ_E v_2 \not\succ_E \cdots \not\succ_E v_n$. Furthermore, since there exist a $\mathbb{T}_v$ for all $v \in V$ and for all $1 \leq i \leq n$ exactly one transition $t_i \in \mathbb{T}_{v_i}$ has to fire to complete the firing sequence, we have that for all $v \in V, v \in \sigma_O$ and is unique. Thus, $\sigma_O \in S_V$ is a permutation of the vertices in $bg(\sigma_U)$. Because all vertices in σ_O are sorted by a linear extension of $\prec_E$, we also have that $\sigma_O \in O_{bg(\sigma_U)}$ is a topological sorting of the vertices in $bg(\sigma_U)$. By Definition 32, we then have that σ_O is an order-realization of σ_U: $\sigma_O \in \mathcal{R}_Q(\sigma_U)$. Since, by construction, $l(t_i) \in \pi_a(v_i)$ if $\pi_o(v_i) = !$ and $l(t_i) \in \pi_a(v_i) \cup \{\tau\}$ if $\pi_o(v_i) = ?$, we have that $\sigma = l(\sigma_T) \restriction_{U_A} \in \mathcal{R}_{\llcorner}(\sigma_U)$. Since this construction is valid for any $\sigma \in \phi(bn(\sigma_U))$, every complete firing sequence of the behavior net is a realization of σ_U: $\phi(bn(\sigma_U)) \subseteq \mathcal{R}_{\llcorner}(\sigma_U)$.

$(\supseteq)$ Let $\sigma_O \in \mathcal{R}_Q(\sigma_U)$ be any order-realization of σ_U, and let $n = |\sigma_U|$. Since $\sigma_O[1] \prec_E$

$\sigma_O[2] \prec_E \cdots \prec_E \sigma_O[n]$ (by Definition 30), there exists a path $p \in P_{bg(\sigma_U)}$ such that $p = \langle v_1, v_2, \ldots, v_n \rangle = \langle \sigma_O[1], \sigma_O[2], \ldots, \sigma_O[n] \rangle$ (by Theorem 1). Let $p_{1,2} = (v_1, v_2)$, $p_{2,3} = (v_2, v_3)$, and so on. Let $t_1 \in \mathbb{T}_{v_1}, t_2 \in \mathbb{T}_{v_2}, \ldots, t_n \in \mathbb{T}_{v_n}$ and let $\sigma_T = \langle t_1, t_2, \ldots, t_n \rangle$. By the construction in Definition 33, in $bn(\sigma_U) = N$ we have that

$$(N, M_{init})[t_1\rangle(N, M_{1,2})[t_2\rangle(N, M_{2,3})[t_3\rangle, \ldots, [t_{n-1}\rangle(N, M_{n-1,n})[t_n\rangle(N, M_{final})$$

where:

$$M_{1,2} = (M_{start} \setminus [(start, v_1)]) \uplus [p_{1,2}]$$
$$M_{2,3} = (M_{1,2} \setminus [p_{1,2}]) \uplus [p_{2,3}]$$
$$\cdots$$
$$M_{n-1,n} = (M_{n-2,n-1} \setminus [p_{n-2,n-1}]) \uplus [p_{n-1,n}]$$
$$M_{final} = (M_{n-1,n} \setminus [p_{n-1,n}]) \uplus [(v_n, end)]$$

This construction implies that $(N, M_{init})[\sigma_T \triangleright (N, M_{final})$ and therefore $\sigma_T \in \phi_f(bn(\sigma_U))$.

The definition of the labeling function in the behavior net is such that, for all $1 \leq i \leq n$, we have that $(v_i, a) \in \mathbb{T}_{v_i} \Leftrightarrow a \in \pi_a(v_i)$. By Definition 30, the labeling of the sequence $\langle t_1, t_2, \ldots, t_n \rangle$ projected on the universe of activities is then a realization of the uncertain trace σ_U obtained from the possible activity labels of σ_O: $l(\sigma_T){\upharpoonright}_{U_A} = \mathcal{R}_\angle(\sigma_U)$. Since this construction is valid for any $\sigma_O \in \mathcal{R}_O(\sigma_U)$, the behavior net can replay any realization of σ_U: $\mathcal{R}_\angle(\sigma_U) \subseteq \phi(bn(\sigma_U))$. $\square$

Theorem 3 (Correctness of uncertain alignments). *Let $\sigma_U \in T_U$ be a simple uncertain trace and let $SN \in U_{SN}$ be a system net. Computing an alignment using the product net between SN and the behavior net $bn(\sigma_U)$ yields the alignment with the lowest cost among all realizations of σ_U: $\delta(\lambda_{SN}(\sigma_{bn})) = \min_{\sigma \in \mathcal{R}_\angle(\sigma_U)} \lambda_{SN}(\sigma) = \delta_{min}(\sigma_U)$.*

Proof. Recall from Definition 21 that $\lambda_{SN} \colon T \to A_{LM}{}^*$ is a deterministic mapping that assigns any trace σ to an optimal alignment. Adriansyah [7] details how to compute such a function λ_{SN} through a state-based $\mathbb{A}^*$ search over a state space defined by the reachable markings of the product net $SN \otimes en(\sigma)$ between a reference system net SN and the event net a certain trace $\sigma \in T$. As per Definition 19, this search retrieves an alignment which is optimal with respect to a certain cost function δ and, ignoring "$\gg$", is composed by a complete firing sequence of the system net $\sigma_T \in \phi_f(SN)$ and the only complete firing sequence of the event net $en(\sigma)$, which corresponds to σ by construction. Given a system net $SN \in U_{SN}$, an uncertain trace $\sigma_U \in T_U$ and its respective behavior net $bn(\sigma_U)$, the same search algorithm for λ_{SN} over $SN \otimes bn(\sigma_U)$ yields an optimal alignment containing a complete firing sequence for the reference system net

$\sigma_T \in \phi_f(SN)$ and a complete firing sequence for the behavior net of the uncertain trace $\sigma \in \phi(bn(\sigma_U))$. Since λ_{SN} minimizes the cost and $\sigma \in \mathcal{R}(\sigma_U)$ is a valid realization of σ due to Theorem 2, the resulting alignment has the minimal cost possible over all the possible realizations of the uncertain trace. $\qquad\qquad\square$

6   Experiments

The framework for computing conformance bounds for uncertain event data illustrated in this paper raises some research questions that need to be addressed in a practical and empirical manner. The questions that we aim to answer are:

- *Q1*: how do conformance bounds behave when computed on uncertain data?

- *Q2*: what is the impact of different deviating behavior and different types of uncertain behaviors on the conformance score of uncertain event logs?

- *Q3*: what is the impact on the efficiency of computing uncertain alignments utilizing the behavior net as opposed to the baseline method of enumerating and aligning all realizations?

- *Q4*: what is the impact on the efficiency of computing uncertain alignments utilizing the behavior net on different types of uncertain behavior?

- *Q5*: how do trace length and the amount of uncertain events impact the intrinsic variability (i.e., the number of realizations) of uncertain event data?

- *Q6*: is it possible to apply uncertain alignments to real-life data to obtain a best- and worst-case scenario for the execution of process instances?

The technique to compute conformance for strongly uncertain traces and to create the behavior net hereby described has been implemented in the Python programming language, thanks to the facilities for log importing, model creation and manipulation, and alignments provided by the library PM4Py [13]. Uncertainty has been represented in the XES standard through meta-attributes and constructs such as lists, such that any XES importer can read an uncertain log file. The algorithm was designed to be fully compatible with any event log in the XES format (both including and not including uncertainty); the meta-attributes for uncertainty were designed to be backward compatible with other process mining algorithms—meta-attributes describing the possible values for an uncertain activity or the interval of an uncertain timestamp can also specify a "fallback value" which other process mining software will read as (certain) activity or timestamp value.

6.1   Qualitative and Quantitative Experiments on Synthetic Data

The first four research questions listed above have been addressed by tests on synthetic uncertain event logs. To this end, we implemented the following software components necessary to the experiments:

- a *noise generator*, to introduce deviations in a controlled way in an event log. This component allows to alter the activity label, swap the order of events or add redundant events to an event log with a given probability or frequency.

- an *uncertainty generator*, to alter the XES attributes present in the log by appending additional meta-information which is then interpreted as uncertainty. The component introduces uncertainty information in an event log, with the possibility to add any of the strongly uncertain attributes described in the taxonomy of Section 2. This also allows for exporting the generated uncertain event log through the XES exporter of the PM4Py library.

- a number of smaller extensions to PM4Py functionalities, also useful for other process mining applications. Examples are the generation of all possible process variants (language) of a PM4Py Petri net, and a memoized version of alignments, which allows to trade off space in memory in order to speed up the computation of the conformance of an event log and a model.

The synthetic data generation and the software tools necessary to compute conformance bounds on uncertain event data are available within the PRocess mining OVer uncErtain Data (PROVED) project [30]. A specific branch of the repository hosting the project is dedicated to the experiments presented in this paper, making them readily reproducible[2].

In order to answer *Q1* and *Q2*, we set up an experiment with the goal to inspect the quality of bounds for conformance scores as increasingly more uncertainty is added to an event log. We ran the tests on synthetic event logs where we added simulated uncertainty. In this way, we can control the amounts and types of uncertainty in event data.

Every iteration of this experiment is as follows:

1. We generate a random Petri net with a fixed dimension ($n = 10$ transitions) through the ProM plugin *"Generate block-structured stochastic Petri nets"*.

2. We play out an event log consisting of 100 traces generated from the Petri net.

3. We randomly alter the activity label of a specific percentage d_a of events, swapping it with another label sampled from the universe of activities.

---

[2]https://github.com/proved-py/proved-core/tree/Conformance_Checking_over_Uncertain_Event_Data

4. We randomly swap a specific percentage d_s of events with their successor. For each event sampled for the swap, we randomly select either the predecessor or the successor (with 50% probability each), and we swap the timestamps of the two events, effectively inverting their order. We skip the selection of the swap direction if we select the first event in a trace (which is swapped with the second) or the last event in a trace (which swaps with the second to last).

5. We randomly duplicate a specific percentage d_d of events. For each event selected for duplication, we create a new event in the trace with identical case ID and activity label, and with timestamp equal to the average between the timestamp of this selected event and the timestamp of the following event. If we select the last event in a trace for duplication, we simply add a fixed delta to the timestamp of the duplicate.

6. We randomly introduce uncertainty in activity labels for a specific percentage u_a of events. Each event selected for uncertainty on activity labels receives one additional activity label, different from the one it already has, sampled from the universe of activity labels.

7. We randomly introduce uncertainty in timestamps for a specific percentage u_t of events. For each event sampled for timestamp uncertainty we randomly choose either the predecessor or the successor (with 50% probability each); the timestamp of the sampled event becomes an interval which extremes are the original timestamp and the timestamp of the predecessor or successor, effectively causing them to mutually overlap. In case the sampled event is the first (resp., last) event in a trace, we skip the selection of the predecessor or successor and we directly consider the successor (resp., predecessor) for the extremes of the uncertain timestamp.

8. We randomly transform a specific percentage u_i of events in indeterminate events. To these sampled events, we add the "?" attribute, in order to mark them as indeterminate.

9. We measure upper and lower bounds for conformance score with increasing percentage p of uncertainty.

All sampling operations mentioned in the previous list are performed over a uniform probability distribution over the possible values.

In terms of amount of deviation to be considered in each configuration, we aimed at recreating a situation where there is significant deviating behavior with respect to the normative model; for each kind of deviation considered, we introduced anomalous behavior in 30% of events. Thus, we consider four different settings for the addition of deviating behavior to events logs: *Activity labels = {d_a = 30%, d_s = 0%, d_d = 0%}, Swaps*

$= \{d_a = 0\%, d_s = 30\%, d_d = 0\%\}$, *Extra events* $= \{d_a = 0\%, d_s = 0\%, d_d = 30\%\}$ and *All* $= \{d_a = 30\%, d_s = 30\%, d_d = 30\%\}$.

We consider four different settings for the addition of uncertain behavior to events logs: *Activities* $= \{u_a = p, u_t = 0\%, u_i = 0\%\}$, *Timestamps* $= \{u_a = 0\%, u_t = p, u_i = 0\%\}$, *Indeterminate events* $= \{u_a = 0\%, u_t = 0\%, u_i = p\}$ and *All* $= \{u_a = p, u_t = p, u_i = p\}$. We test all four different configurations of deviation against each of the four configurations of uncertainty, with increasing values of p, for a total of 16 separate experiments.

Figure 8 summarizes our findings. The plots on this figure represent the average of 10 runs as described above.

We can observe that, in general, all plots show the expected behavior: the upper and lower bound for conformance coincide at percentage of uncertain events $p = 0$ for all experiments, to then diverge while p increases. A number of additional observations can be made looking at individual configurations for deviation or uncertainty, or at specific scatter plots. When only uncertainty on activity labels is added to the event log, we see a deterioration of the upper bound for conformance cost, but the lower bound does not improve—in fact, it is essentially constant. This can be attributed to the fact that, since to generate uncertainty on activity label we sample from the set of labels randomly, the chances of observing a realization of a trace where an uncertain activity label matches the alteration introduced by the deviations are small. Uncertainty on timestamps makes the lower bound decrease only when the introduced deviations are swaps: as expected, the possibility of changing the order of pairs of events does not have a sensible improvement in the lower bound for deviation when extra events are added or activity labels of existing events are altered.

Conversely, the possibility to "skip" some critical events has a positive effect on the lower bound of all possible configurations for deviations: in fact, when marking some events as indeterminate in a log where extra events were added as deviations, the average conformance cost drops by 30.61% at $p = 16\%$, the largest drop among all the experiments. The experiment with all three types of uncertainty and extra events as deviations essentially displays the same effect (improvement in lower bound is slightly lower, but not significantly, with a decrease in deviation of 29.38% at $p = 16\%$).

For the experiments where all types of deviations were added at once, we can see that, as could be anticipated, the differences in deviation scores on the two bounds become smaller in relative terms (because of the very high amount of deviations at $p = 0\%$), but larger in absolute terms. As per the previous experiments, the largest contributor in decreasing the conformance cost of the lower bound is the addition of indeterminate events, which by itself decreases the deviation cost by 13.92% at $p = 16\%$. In general, the vast variability in measuring the conformance of an uncertain log shows that, if all types of uncertainty can occur with high frequency in a process, the business owner should act on the uncertainty sources, since they will be a major obstacle in obtaining accurate
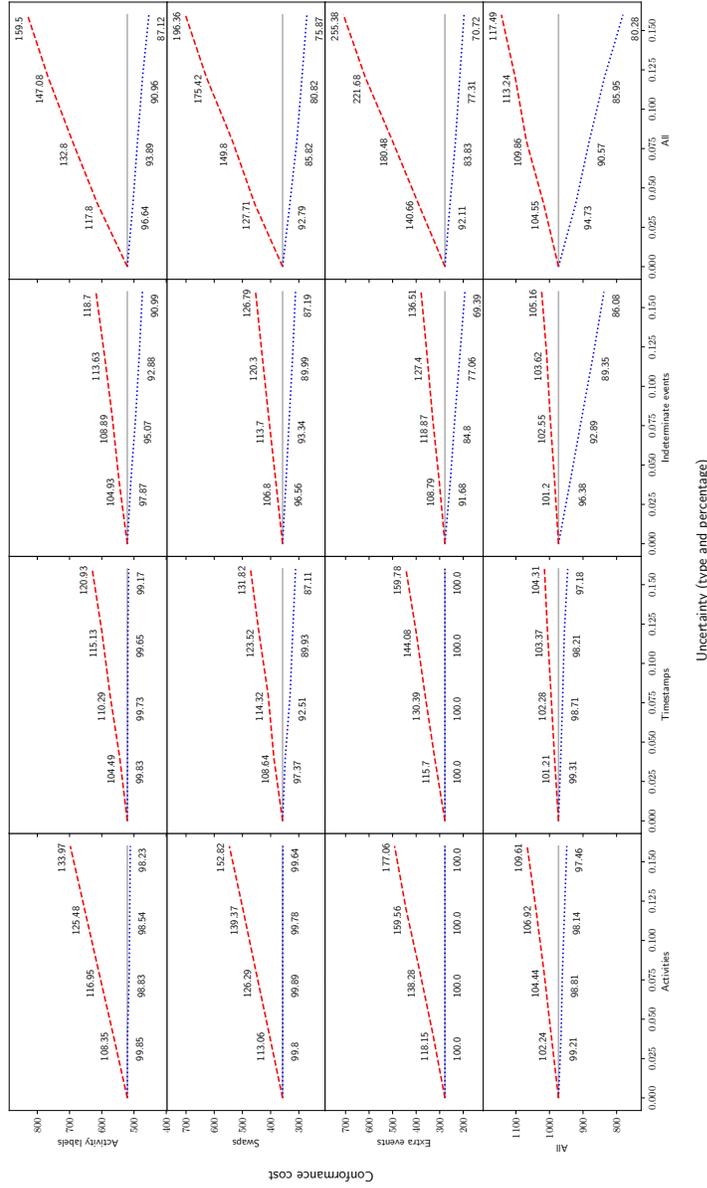
Figure 8: Upper (red, dashed) and lower (blue, dotted) bound for conformance cost for synthetic event logs with increasing uncertainty. Every plot shows a different configuration of deviation added to the log and types of uncertainty simulated in the event data. The x-axis shows the percentage of uncertainty p added to the logs; the y-axis shows the amount of deviations, computed with alignments. The labels inside the graph indicate the relative change in deviation score with respect to $p = 0$, as a percentage. The gray continuous lines indicate the amounts of deviations at $p = 0$ as a reference.
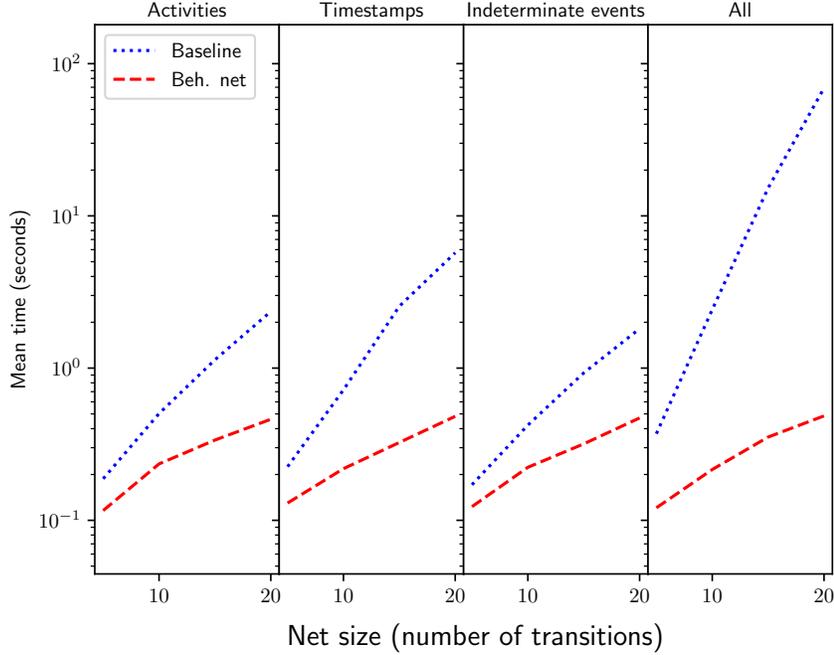
Figure 9: Effect on the time performance of calculating the lower bound for conformance cost with the brute-force method (blue, dotted) vs. the behavior net (red, dashed) on four different configurations for uncertain events.

measurements of process conformance. Vice versa, in the case of limited occurrences of uncertainty in event data, the algorithm here proposed is able to provide actionable bounds for conformance score, together with descriptions of best- and worst-case scenarios of process conformance for a given trace.

The second experiment we set up aims to answer questions Q_3 and Q_4, and is concerned with the performance of calculating the lower bound of the cost via the behavior net versus the brute-force method of listing all the realizations of an uncertain trace, evaluating all of them through alignments, then picking the best value. We used a constant percentage of uncertain events of $p = 5\%$ and logs of 100 traces for this test, with progressively increasing values of n. We ran 4 different experiments, each with one of the four configurations for uncertain behavior *Activities*, *Timestamps*, *Indeterminate events* and *All* illustrated above.

Figure 9 summarizes the results. As the diagram shows, the difference in time between the two methods tends to diverge quickly even on a logarithmic scale. The largest model we could test was $n = 20$, a Petri net with 20 transitions, which is comparatively
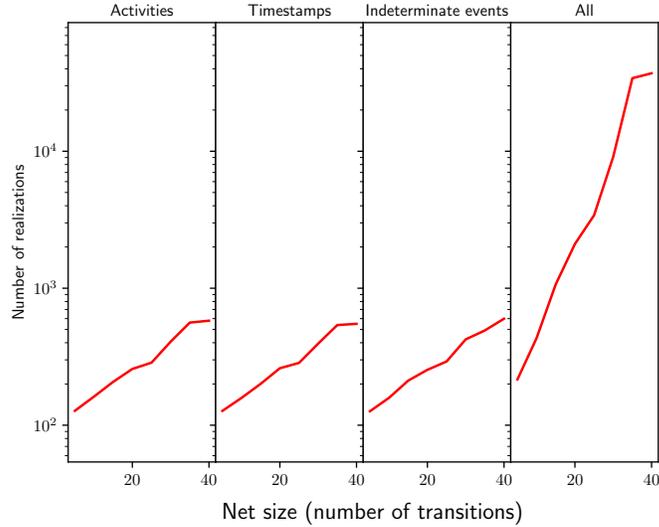
Figure 10: Number of realizations (average per log over 10 randomly generated logs of 100 traces) in an uncertain event log in function of the size of the Petri net used to generate it. The logs are generated through simulation with 10 different and randomly wired Petri nets of increasing size. Four different configurations for uncertainty are shown: on activities, timestamps, indeterminate events, and all three combined. Uncertainty is introduced in the events within the log in a fixed proportion of 5%.

tiny in practical terms; however, even at these small scales the brute-force method takes roughly 3 orders of magnitude more than the time needed by the behavior net, when all the types of uncertainty are added with $p = 5\%$.

This shows a very large improvement in the computing time for the lower bound computation; thus, the best-case scenario for the conformance cost of an uncertain trace can be obtained efficiently thanks to the structural properties of the behavior net. This graph also shows the dramatic impact on the number of realizations of a behavior net—and thus, the time needed to perform a brute-force computation of alignments—when the effects of different kinds of uncertainty are compounded.

Let us now answer $Q5$. In order to assess the impact of uncertainty on the variability of event data and, consequently, on the performance of uncertain process mining techniques, we computed the total number of realizations in an uncertain event log of 100 traces. Figure 10 shows the change in number of realizations with the increase in size of the Petri net used to generate the log, while the percentage of uncertain events is constant (5%). Conversely, Figure 11 shows the change in the number of realizations with the increase in percentage of uncertain events, while the size of the Petri nets is fixed (10 transitions).
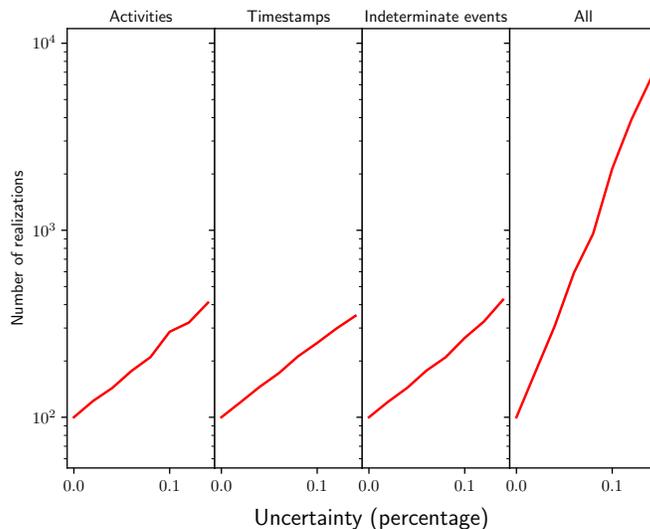
Figure 11: Number of realizations (average per log over 10 randomly generated logs of 100 traces) in an uncertain event log in function of the percentage of uncertain events within the log. The logs are generated through simulation with 10 different and randomly wired Petri nets with a fixed size of 10 transitions. Four different configurations for uncertainty are shown: on activities, timestamps, indeterminate events, and all three combined.

Figures 10 and 11 justify the results of the experiments on performance shown in Figure 9: there is a clear exponential relation between the number of realizations resulting from an uncertain event log and both trace length and percentage of uncertain events in the log. Both factors, when increasing, induce an exponential increase in the total number of realizations even when considered separately. Specifically, the comparison of Figures 9 and 10 highlights the cause of the inefficiency of computing alignments for every realization of an uncertain trace, evidently showing the linear relationship between the number of realizations in an uncertain log and the time expenditure of the brute-force alignments approach.

6.2   Applications on Real-Life Data

As illustrated in Section 1, uncertainty in event data can originate from a number of different causes in real-world applications. One prominent source of uncertainty is missing data: attribute values not recorded in an event log can on occasions be described by uncertainty, through domain knowledge provided by process owners or experts. Then, as described in this paper, it is possible to obtain a detailed analysis of the deviations of a best- and worst-case scenario for the conformance to a process model.
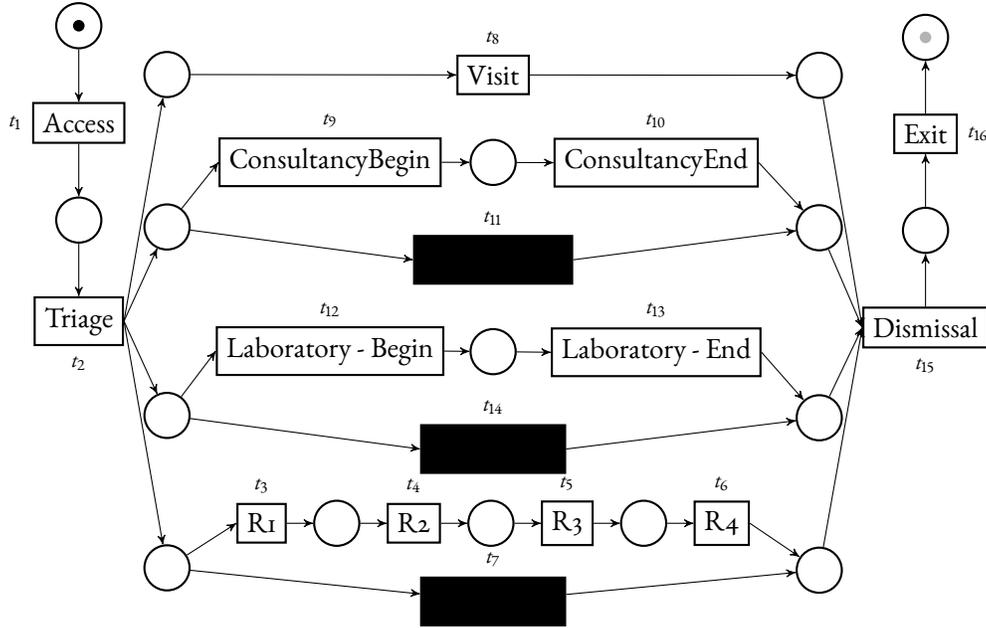
Figure 12: The Petri net that models the process related to the treatment of patients in the ICU ward of an Italian hospital. The activities R1 through R4 are abbreviations for the four phases of a radiology exam: respectively, *Radiology - Submitted Request, Radiology - Accepted Request, Radiology - Exam, Radiology - Results.*

To seek to answer research question *Q6* through a direct application of conformance checking over uncertainty, let us consider a process related to the medical procedures performed in the Intensive Care Unit (ICU) of a hospital. Figure 12 shows a ground truth model for the process.

An execution log containing events that concern this ICU process is available. Throughout the process, some anomalies with attribute values can be spotted—namely, a number of anomalies affecting the timestamp attributes. This is a $[E]_{\mathbb{S}}$-type uncertain log.

The alterations on the timestamps in this event log happen for a number of reasons. The domain experts reported that human error is a frequent source of anomaly, which is worsened by the fact that operators often do not input data in real-time, but the information is recorded after a certain delay (e.g., at the end of a shift). Moreover, the information systems of the ICU ward and other wards (such as radiology, for instance) do not allow for automatic transmission of data between one another, so in some occurrences the timestamp of visits by specialists is not recorded in the ICU information system.

Tables 6 and 7 show two examples of traces with anomalous timestamp behavior. We can see that in the trace of Table 6 the event *Triage* has an imprecise timestamp—only

Table 6: Events related to one case of the ICU process. The timestamp of the "Triage" event is imprecise: through domain knowledge, we are able to represent this uncertainty in an explicit way within the event attributes in the log.

Event ID	Raw Timestamp	Preprocessed Timestamp	Activity
e_1	2017-02-20 23:59:31	2017-02-20 23:59:31	*Access*
e_2	2017-02-21 00:02:58	2017-02-21 00:02:58	*Visit*
e_3	2017-02-21 00:06:30	2017-02-21 00:06:30	*ConsultancyBegin*
e_4	2017-02-21 00:29:12	2017-02-21 00:29:12	*R1*
e_5	2017-02-21 00:41:00	2017-02-21 00:41:00	*R2*
e_6	2017-02-21 00:41:00	2017-02-21 00:41:00	*R3*
e_7	2017-02-21 01:02:00	2017-02-21 01:02:00	*R4*
e_8	2017-02-21 01:56:26	2017-02-21 01:56:26	*ConsultancyEnd*
e_9	2017-02-21 02:01:37	2017-02-21 02:01:37	*Dismissal*
e_{10}	2017-02-21 02:02:36	2017-02-21 02:02:36	*Exit*
e_{11}	2017-02-21	[2017-02-21 00:00:00, 2017-02-21 23:59:59]	*Triage*

Table 7: Events related to one case of the ICU process. Some of the timestamp attributes are missing: through domain knowledge, we are able to represent them with uncertainty within a small interval of time. The timestamps in bold and italic of the "Raw Timestamp" column are used to set the interval boundaries for uncertain timestamps.

Event ID	Raw Timestamp	Preprocessed Timestamp	Activity
e_1	2017-08-27 11:47:46	2017-08-27 11:47:46	*Access*
e_2	2017-08-27 **11:47:53**	2017-08-27 11:47:53	*Triage*
e_3	2017-08-27 12:14:25	2017-08-27 12:14:25	*Visit*
e_4	2017-08-27 12:33:24	2017-08-27 12:33:24	*R1*
e_5	2017-08-27 13:04:11	2017-08-27 13:04:11	*ConsultancyBegin*
e_6	2017-08-27 *13:04:53*	2017-08-27 13:04:53	*Dismissal*
e_7	2017-08-27 13:08:07	2017-08-27 13:08:07	*Exit*
e_8	**NULL**	[2017-08-27 **11:47:53**, 2017-08-27 *13:04:53*]	*ConsultancyEnd*
e_9	**NULL**	[2017-08-27 **11:47:53**, 2017-08-27 *13:04:53*]	*R2*
e_{10}	**NULL**	[2017-08-27 **11:47:53**, 2017-08-27 *13:04:53*]	*R3*
e_{11}	**NULL**	[2017-08-27 **11:47:53**, 2017-08-27 *13:04:53*]	*R4*

the day has been recorded. This can be modeled with an uncertain timestamp encompassing a range of 24 hours. The column *Preprocessed Timestamp* shows the results of this preprocessing step.

Some of the events in the trace of Table 7 are missing the timestamp value entirely. In this case, we can resort to domain knowledge provided by the process owners: it is known that events related to the *Radiology* exams happen after the *Triage* event, and before the *Dismissal* event. This allows the representation of the timestamps with ranges of possible values. Notice that such a small interval of time, obtainable from the domain

Table 8: A valid alignment for both traces of Tables 6 and 7. This alignment has a deviation cost equal to 0, and corresponds to a best-case scenario for conformance between the process model and both uncertain traces.

Access	Triage	Visit	ConsultancyBegin	R1	R2	R3	R4	ConsultancyEnd	≫	Dismissal	Exit
Access	Triage	Visit	ConsultancyBegin	R1	R2	R3	R4	ConsultancyEnd	τ	Dismissal	Exit
t_1	t_2	t_8	t_9	t_3	t_4	t_5	t_6	t_{10}	t_{14}	t_{15}	t_{16}

Table 9: A valid alignment for the trace of Table 6. This alignment has a deviation cost equal to 2 (1 move on log and 1 move on model), and corresponds to a worst-case scenario for conformance between the process model and the uncertain trace.

Access	≫	Visit	ConsultancyBegin	R1	R2	R3	R4	ConsultancyEnd	≫	Dismissal	Exit	Triage
Access	Triage	Visit	ConsultancyBegin	R1	R2	R3	R4	ConsultancyEnd	τ	Dismissal	Exit	≫
t_1	t_2	t_8	t_9	t_3	t_4	t_5	t_6	t_{10}	t_{14}	t_{15}	t_{16}	

knowledge available, is preferable to larger possible intervals (e.g., 2017-08-27 00:00:00 to 2017-08-27 23:59:59), since it minimizes the amount of possible overlaps in time with other events in the trace. In turn, this means that the number of possible realizations of the uncertain trace is smaller, granting a faster conformance checking. As before, the results of modeling timestamp uncertainty are shown in the column *Preprocessed Timestamp*.

Once uncertainty is made explicit using the event log formally defined in this paper, it is possible to apply conformance checking over uncertainty. The technique of alignments illustrated here provides two results, corresponding to the lower and upper bound for the conformance score. The traces shown in Tables 6 and 7 have a best-case scenario alignment in common, which is shown in Table 8; aligning through the behavior net of these traces has allowed the algorithm to select a value for the uncertain timestamps of the traces (translated in a specific ordering) such that the deviations between data and model is the smallest possible. For both traces, the best-case scenario has a cost equal to 0, thus, no deviations occur in that case.

Let us now look at the worst-case scenarios. One of the alignments with the worst possible score for the trace in Table 6 is shown in Table 9. In this scenario, the deviations are one move on model (the *Triage* activity should have occurred after the *Access* but did not), and one move on log (the activity *Triage* occurs in the data at an unexpected moment in the process).

A worst-case scenario for the trace in Table 7 is illustrated in Table 10. In this case, the deviation is equal to 6, given by the wrong order of the event related to the *Radiology* exam. Note that, in this example, we assume that every deviation has a unit cost, but the alignment technique allows to define different costs for different types of deviations based on impact in the process. For example, a patient that exits the hospital without official dismissal might have a worse impact than an unauthorized laboratory exam. For simplicity, in this case, we assume that all types of deviation have a unit cost.

Uncertain alignments provide novel insights, not obtainable through existing con-

Table 10: A valid alignment for the trace of Table 7. This alignment has a cost equal to 6 (3 moves on log and 3 moves on model), and corresponds to the worst-case scenario for conformance between the process model and the uncertain trace.

Access	Triage	Visit	ConsultancyBegin	$\gg$	$\gg$	$\gg$	R_4	R_3	R_2	R_1	ConsultancyEnd	$\gg$	Dismissal	Exit
Access	Triage	Visit	ConsultancyBegin	R_1	R_2	R_3	R_4	τ	τ	τ	ConsultancyEnd	τ	Dismissal	Exit
t_1	t_2	t_8	t_9	t_3	t_4	t_5	t_6				t_{10}	t_{14}	t_{15}	t_{16}

formance techniques. The process owner can utilize these results to gain insights and decide actions in regard of the process. In situations where quantified uncertainty is present or can be uncovered using domain knowledge in a pre-processing step, the potential violation shown in the worst-case scenario for traces such as the one in Table 6 can be investigated, as well as the source of said uncertainty; the process owner can, furthermore, decide whether the consequences and the likelihood of the worst-case scenario are indicative of a need for a process restructuration, or whether the risk of such potential violation of the normative process model are not critical for the process execution. Conversely, if uncertainty in the event log remains non-quantified and the affected trace is treated as a regular process trace, the subsequent analysis will only consider one possible realization of the uncertain trace, possibly sampled at random. In this case, taking process management decisions that account for the best- and worst-case scenarios is not possible.

Lastly, it is important to notice an additional implication of the qualitative experiment described in this section. For the events e_8 through e_{11} of the trace in Table 7 we determined suitable bounds for uncertain timestamps through domain knowledge. In absence of such domain knowledge, it is still possible to apply process mining techniques for uncertain data to traces with missing timestamps: the lower (resp., upper) bound of such timestamps can be set to be smaller (resp., larger) than any other timestamp appearing in the uncertain trace. This models an event that, in the real process, might have occurred in any point in the trace. Thus, the resulting pre-processed event will be able to be considered by process mining techniques operating on uncertain events such as the alignment technique presented in this paper[3]. An analogous pre-processing procedure can be utilized on events with a missing activity label, by assigning an uncertain label containing all labels appearing in the event log. While these pre-processing techniques allow to apply process mining techniques to traces and events with missing attributes, it is important to bear in mind the consequences this might have in terms of performance. As demonstrated by the results of quantitative performance experiments shown in Figure 11, a small percentage of uncertain events in a log induces a large amount of realizations. Thus, modeling missing timestamps or activity labels through uncertainty

---

[3]This is limited to techniques that only consider the control-flow perspective of event data. Additional perspectives might not be modeled by this pre-processing technique (namely, in this case, the time/performance perspective).

without restricting them with domain knowledge might be unfeasible when applied to substantial amounts of event data.

7   Related Work

This section discusses existing literature relevant to the problem of computing a conformance score between historical event data and a process model, as well as research addressing types of anomalies in recorded data similar to the notion of uncertainty presented in this paper.

7.1   Conformance Checking

The discipline of conformance checking, a subfield of process mining, is concerned with defining metrics to compare how well an event log matches a given process model. The input for this task consists of an execution log and a process model (most commonly a labeled Petri net) and the output is a measurement of the distance—that is, the deviation—between the model and the log, or the traces that compose the log. The two main goals of conformance checking are measuring the quality of a process discovery algorithm by comparing the discovered process model with the source event log, to verify the extent to which the model fits the log; and comparing an execution log with a normative process model (often defined partially or completely by hand) in order to verify the deviations between the rules governing the process and the tasks carried out in reality. Often, the conformance measure defined between logs (or traces) and models includes not only a distance in absolute terms, but also an indication of where and what deviated from the norm in the process. Conformance checking was introduced by Rozinat and Van der Aalst [35], who obtained a conformance measure by tracking counts of tokens during replay of traces in a Petri net. Despite the elevated computational complexity, state-of-the-art approaches are mostly based on alignments, introduced by Adriansyah et al. [8].

The topic of conformance checking includes previous work that examines concepts connected to probability on the model side. The stochastic Petri net is an important extension of the Petri net model which probabilistically describes the time distance between the activation of transitions. Richter et al. [34] utilize these models to extend conformance checking so the conformance score of event data can be finely tuned to account for deviations in the time dimension with respect to a reference stochastic Petri net. Another formalism involving probabilities is the Fork/Join network, a scheduling model that can represent the actions of resources in a process, complete with probabilities. More recently, Leemans et al. [26] devised a conformance checking technique able to measure deviations between stochastic Petri nets and event logs. Their method hinges on converting both the stochastic model and the event log in a so-called stochastic language, i.e., a probability distribution over process variants. They then employ the

Earth Movers' Distance (EMD) to compute the difference between stochastic languages in term of distance between probability distributions. This allows to account for routing probabilities in the model, improving the reliability of conformance scores.

Senderovich et al. [36] show how to discover Fork/Join networks from an event log and a corresponding schedule for the process—i.e., a description of the tasks involved in a process and an assignment between agents and tasks—and utilize them to perform conformance checking. This allows to measure the predictive capabilities of the reference schedule—or, alternatively, to quantify the deviation from the schedule present in an historical event log. Additionally, the authors complement this conformance checking approach on schedules with a process improvement algorithm that shortens the expected delay between tasks in the process.

It is important, however, to note that stochastic conformance checking is a concept that fundamentally differs from uncertainty as presented in this paper. It is essential to understand that strong uncertainty involves non-determinism, and the behavior contained in a strongly uncertain trace is completely probability-agnostic. Moreover, existing approaches for stochastic conformance checking assume the presence of probability information on the reference model, which is then compared with classic process traces; conversely, uncertainty specifically considers anomalies within recorded data, regardless of the nature or the semantics of the corresponding process model.

7.2   Event Data Uncertainty

As mentioned, the occurrence of data containing uncertainty—in a broad sense—is common both in more classic disciplines like statistics and data mining [24] and in process mining [5]; and logs that show an explicit uncertainty in the control flow perspective can be classified in the lower levels of the quality ranking proposed in the process mining manifesto.

To historically position the topic of uncertain data, let us mention some previous work in the domain of data mining. A survey work offering a panoramic view of mining uncertain data is the one by Aggarwal and Philip [9], which focuses with particular attention on the problem of uncertain data querying. Such data is represented on the basis of probabilistic databases [38], a foundational notion in the setting of uncertain data mining. A branch of data mining particularly related to process mining is frequent itemsets mining: an efficient algorithm to search for frequent itemsets over uncertain data, the U-Apriori, have been presented by Chui et al. [16].

Within process mining, there exist various techniques to deal with a kind of uncertainty different, albeit closely related, from the one that we analyze here: missing or incorrect data. This can be considered as a form of non-explicit uncertainty: no measure or indication on the nature of the uncertainty is given in the event log. The work of Suriadi et al. [39] provides a taxonomy for such issues in event logs, laying out a series

of data patterns that model errors in process data. In these cases, and if this behavior is infrequent enough to allow the event log to remain meaningful, the most common way for existing process mining techniques to deal with missing data is by filtering out the affected traces and performing discovery and conformance checking on the resulting filtered event log. A case study illustrating such situation is, e.g., the work of Benevento et al. [12]. While filtering out missing values is straightforward, various methodologies of event log filtering have been proposed in the past to solve the problem of incorrect event attributes: the filtering can take place thanks to a reference model, which can be given as process specification [40], or from information discovered from the frequent and well-formed traces of the same event log; for example extracting an automaton from the frequent traces [17], computing conditional probabilities of frequent sequences of activities [20], or discovering a probabilistic automaton [42]. In the latter cases, the noise is identified as infrequent behavior.

Some previous work attempt to repair the incorrect values in an event log. Conforti et al. [18] propose an approach for the restoration of incorrect timestamps based on a log automaton, that repairs the total ordering of events in a trace based on correct frequent behavior. Fani Sani et al. [21] define outlier behavior as the unexpected occurrence of an event, the absence of an event that is supposed to happen, and the incorrect order of events in the trace; then, they propose a repairing method based on probabilistic analysis of the context of an outlier (events preceding or following the anomalous event). Again, both of these methods define anomalous/incorrect behavior on the basis of the frequency of occurrence.

The definition of uncertainty on activity labels as defined in the taxonomy of Section 2 has not been, to the best of our knowledge, previously employed in the field of process mining. There are, however, related examples of anomalies or uncertainties on activity labels of events: for instance, the problem of matching event identifiers to normative activity labels [11]. In this case, an event is associated with only one activity label, but this association is not known. There are a number of techniques to estimate the correct association, included some that consider the data perspective, together with the control flow perspective [36]. Using this setting, Van der Aa et al. [1] proposed a technique to estimate bounds of conformance scores for event logs with unknown or partially known event-to-activity mapping. Another related domain is the many-to-one abstraction from low-level events to a higher order of activity labels, which can be performed via clustering events in minimal conflict groups [23] or representing low-level patterns with data Petri nets which then discovers high-level activities by matching patterns through alignments [28].

A kind of anomaly in event data which is even more related to uncertainty as discussed in this paper is incompleteness in the order of events in a trace. This occurs when total ordering among events is lost or not available, and only a partial order is known. In the field of concurrent and distributed systems, the absence of a total order among

logged activities has historically been relevant by virtue of being both caused by, and a necessary condition for, the presence of concurrency in a system (refer e.g. to Beschastnikh et al. [14]). An important concept at the base of this paper is the representation of uncertainties in the timestamp dimension through directed acyclic graphs, which express these partial orders. This intuition was first presented by Lu et al. [27], also in the context of conformance checking, in order to produce partially ordered alignments. More recently, Van der Aa et al. [2] proposed a technique to resolve such order uncertainty, through estimates based on probabilistic inference aided by a normative process model.

In process mining, a notion well known for a long time is the fact that in many cases the definition of the case is not part of the normative information immediately accessible to the process analyst, so there needs to be a decision on which attribute or attributes constitutes the case of the process. In some cases, multiple definitions of cases are possible and analysis on a subset of them is desirable. This specific setting, which can be interpreted as uncertainty on the case notion, has a long history both in terms of mathematical formalization and in terms of implementation and definition of data standards. For an introduction to this subfield of process mining we refer to [van2019object].

This paper presents an extended version of the preliminary analysis on uncertain event data in process mining shown in [4], in which we presented a preliminary description of uncertain event data and their taxonomy, as well as a description of an approach to find upper and lower bound for the conformance score of an uncertain process trace through alignments. We elaborate on this previous work adding an extended formalization, proving theorems on uncertainty in process mining, and reporting on new experiments. The framework for uncertain data proposed in this paper has also been expanded by providing an algorithm capable of process discovery on uncertain event data through the definition of directly-follows relationship in uncertain settings and the computation of an uncertain directly-follows graph, which enables process discovery techniques [31]. On the topic of efficient uncertain data management, we presented an improved algorithm that allows to preprocess uncertain traces into behavior graphs in quadratic time, enabling fast uncertainty analysis [32]. The exploration of uncertain event data can also be facilitated by a memory-efficient representation method and the definition of the concept of uncertain process variants [33].

Lastly, it is important to mention some of the technological advancements that decrease the likelihood of the presence of uncertainty in the data. As discussed in Section 1, some of the most prominent causes of uncertainty are the human factor involved in the process, and the intrinsic limitations of legacy information systems. Besides more classical concepts like workflow automation [37] and the deployment of process-aware information systems [19], a recent innovation that aims to mitigate both problems is *Robotic Process Automation* (RPA) [6], a technology that aids user operations in processes by learning repeated patterns of actions, and subsequently automate them, while interacting with human operators through the same GUIs they are utilizing. Introducing a high

level of automation within the process naturally helps towards the accurate recording of process data, especially if such automation assists the human agents involved in the process.

8   Conclusion

As the need to quickly and effectively analyze process data has arisen in the recent past and is growing to this day, many new types of information regarding events are recorded; this calls for new techniques able to provide an adequate interpretation of the new data. Not only more and more event data is available to the analyst, but these data are accessible in association with a wealth of information and meta-information about the process, the resources that executed activities, data about the outcome of those actions, and many other types of knowledge about the nature of events, activities, and the process as a whole. In this paper, we presented a new paradigm for process mining applied to event data: explicit uncertainty. We described the possible form it can assume, building a taxonomy of different types of uncertainty, and we provided examples of how uncertainty can originate in a process, and how uncertainty information can be inferred from the available data and from domain knowledge provided by process experts. We then designed a framework to define the various flavors of uncertainty shown in the taxonomy. Then, in order to assess the practical applications of the uncertainty framework, we applied it to a well-consolidated technique for conformance checking: aligning data to a reference Petri net. This application of uncertainty analysis is integrated by theorems that prove the correctness of the techniques developed and illustrated here within the framework previously described. The results can provide insights on the possible violations of process instances recorded with uncertainty against a normative model. The behavior net provides an efficient way to compute the lower bound for the conformance cost—i.e., the best-case scenario for conformity of uncertain process data—with a large improvement in time performance with respect to a brute-force procedure.

The approaches shown here can be extended in a number of ways. From a performance perspective, to improve the usability of alignments over uncertainty the computation of the upper bound of the conformance cost should either be optimized, or replaced by an approximate algorithm. Another direction for future work is extending the conformance checking technique to logs with weak uncertainty, weighting the deviation by means of the probability distributions attached to activities, timestamps and indeterminate events. This includes the case in which probability distributions contained in weakly uncertain events are not necessarily independent, or where the assumption of independence is unrealistic for the process being analyzed. Furthermore, a limitation affecting the techniques presented in this paper is that using a graphical representation in lieu of process traces requires to process the entire trace. This implies that uncertain

alignments can only be applied to data available in batches, while they do not support event data in streams. Future research might include the development of graphless (i.e., not reliant on graph structures) representations towards online process mining over uncertain event data.

Additionally, investigation on real-life data is an important milestone for this line of research, and it is vital to analyze in depth a complete use case in real life of process mining in the presence of uncertain event data.

Acknowledgements

References

[1]   van der Aa, Han, Henrik Leopold, and Hajo A. Reijers. "Efficient Process Conformance Checking on the Basis of Uncertain Event-to-Activity Mappings". In: *IEEE Transactions on Knowledge and Data Engineering* 32.5 (2020), pp. 927–940. DOI: 10.1109/TKDE.2019.2897557.

[2]   van der Aa, Han, Henrik Leopold, and Matthias Weidlich. "Partial order resolution of event logs for process conformance checking". In: *Decision Support Systems* 136 (2020), p. 113347. DOI: 10.1016/j.dss.2020.113347.

[3]   van der Aalst, Wil M. P. "Decomposing Petri nets for process mining: A generic approach". In: *Distributed Parallel Databases* 31.4 (2013), pp. 471–507. DOI: 10.1007/s10619-013-7127-5.

[4]   van der Aalst, Wil M. P. "Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data". In: *Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings*. Ed. by Ölveczky, Peter Csaba and Gwen Salaün. Vol. 11724. Lecture Notes in Computer Science. Springer, 2019, pp. 3–25. DOI: 10.1007/978-3-030-30446-1_1.

[5]   van der Aalst, Wil M. P., Arya Adriansyah, Ana Karla Alves de Medeiros, et al. "Process Mining Manifesto". In: *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*. Ed. by Daniel, Florian, Kamel Barkaoui, and Schahram Dustdar. Vol. 99. Lecture Notes in Business Information Processing. Springer, 2011, pp. 169–194. DOI: 10.1007/978-3-642-28108-2_19.

[6]   van der Aalst, Wil M. P., Martin Bichler, and Armin Heinzl. "Robotic Process Automation". In: *Business Information Systems Engineering* 60.4 (2018), pp. 269–272. DOI: 10.1007/s12599-018-0542-4.

[7]   Adriansyah, Arya. "Aligning observed and modeled behavior". PhD thesis. Eindhoven University of Technology, 2014. DOI: 10.6100/IR770080.

[8]   Adriansyah, Arya, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. "Towards Robust Conformance Checking". In: *Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers*. Ed. by zur Muehlen, Michael and Jianwen Su. Vol. 66. Lecture Notes in Business Information Processing. Springer, 2010, pp. 122–133. DOI: 10.1007/978-3-642-20511-8_11.

[9]   Aggarwal, Charu C. and Philip S. Yu. "A Survey of Uncertain Data Algorithms and Applications". In: *IEEE Transactions on Knowledge Data Engineering* 21.5 (2009), pp. 609–623. DOI: 10.1109/TKDE.2008.190.

[10]   Aho, Alfred V., M. R. Garey, and Jeffrey D. Ullman. "The Transitive Reduction of a Directed Graph". In: *SIAM Journal on Computing* 1.2 (1972), pp. 131–137. DOI: 10.1137/0201008.

[11]   Baier, Thomas and Jan Mendling. "Bridging Abstraction Layers in Process Mining by Automated Matching of Events and Activities". In: *Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Ed. by Daniel, Florian, Jianmin Wang, and Barbara Weber. Vol. 8094. Lecture Notes in Computer Science. Springer, 2013, pp. 17–32. DOI: 10.1007/978-3-642-40176-3_4.

[12]   Benevento, Elisabetta, Prabhakar M. Dixit, Mohammadreza Fani Sani, et al. "Evaluating the Effectiveness of Interactive Process Discovery in Healthcare: A Case Study". In: *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*. Ed. by Di Francescomarino, Chiara, Remco M. Dijkman, and Uwe Zdun. Vol. 362. Lecture Notes in Business Information Processing. Springer, 2019, pp. 508–519. DOI: 10.1007/978-3-030-37453-2_41.

[13]   Berti, Alessandro, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. "Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science". In: *ICPM Demo Track (CEUR 2374)*. 2019, pp. 13–16. URL: http://ceur-ws.org/Vol-2374/paper4.pdf.

[14]   Beschastnikh, Ivan, Yuriy Brun, Michael D. Ernst, et al. "Mining temporal invariants from partially ordered logs". In: *ACM SIGOPS Operatin Systems Review* 45.3 (2011), pp. 39–46. DOI: 10.1145/2094091.2094101.

[15]   Carmona, Josep, Boudewijn F. van Dongen, Andreas Solti, et al. *Conformance Checking - Relating Processes and Models*. Springer, 2018. ISBN: 978-3-319-99413-0. DOI: 10.1007/978-3-319-99414-7.

[16]   Chui, Chun-Kit, Ben Kao, and Edward Hung. "Mining Frequent Itemsets from Uncertain Data". In: *Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference, PAKDD 2007, Nanjing, China, May 22-25, 2007, Proceedings*. Ed. by Zhou, Zhi-Hua, Hang Li, and Qiang Yang. Vol. 4426. Lecture Notes in Computer Science. Springer, 2007, pp. 47–58. DOI: 10.1007/978-3-540-71701-0_8.

[17]   Conforti, Raffaele, Marcello La Rosa, and Arthur H. M. ter Hofstede. "Filtering Out Infrequent Behavior from Business Process Event Logs". In: *IEEE Transactions on Knowledge and Data Engineering* 29.2 (2017), pp. 300–314. DOI: 10.1109/TKDE.2016.2614680.

[18]   Conforti, Raffaele, Marcello La Rosa, Arthur H. M. ter Hofstede, et al. "Automatic Repair of Same-Timestamp Errors in Business Process Event Logs". In: *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*. Ed. by Fahland, Dirk, Chiara Ghidini, Jörg Becker, et al. Vol. 12168. Lecture Notes in Computer Science. Springer, 2020, pp. 327–345. DOI: 10.1007/978-3-030-58666-9_19.

[19]   Dumas, Marlon, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede, eds. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, 2005. ISBN: 978-0-471-66306-5. DOI: 10.1002/0471741442.

[20]   Fani Sani, Mohammadreza, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. "Improving Process Discovery Results by Filtering Outliers Using Conditional Behavioural Probabilities". In: *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*. Ed. by Teniente, Ernest and Matthias Weidlich. Vol. 308. Lecture Notes in Business Information Processing. Springer, 2017, pp. 216–229. DOI: 10.1007/978-3-319-74030-0_16.

[21]   Fani Sani, Mohammadreza, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. "Repairing Outlier Behaviour in Event Logs". In: *Business Information Systems - 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings*. Ed. by Abramowicz, Witold and Adrian Paschke. Vol. 320. Lecture Notes in Business Information Processing. Springer, 2018, pp. 115–131. DOI: 10.1007/978-3-319-93931-5_9.

[22]   Flaška, Václav, Jaroslav Ježek, Tomáš Kepka, et al. "Transitive closures of binary relations. I." In: *Acta Universitatis Carolinae. Mathematica et Physica* 48.1 (2007), pp. 55–69.

[23]　Günther, Christian W. and Wil M. P. van der Aalst. *Mining activity clusters from low-level event logs*. Beta, Research School for Operations Management and Logistics, 2006. URL: https://www.persistent-identifier.nl/urn: nbn:nl:ui:25-5e7c14a5-bcb7-42a9-ab4d-25b41b1fb6c3.

[24]　Han, Jiawei, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011. ISBN: 978-0123814791. URL: http://hanj.cs.illinois.edu/bk3/.

[25]　Kalvin, Alan D. and Yaakov L. Varol. "On the Generation of All Topological Sortings". In: *Journal of Algorithms* 4.2 (1983), pp. 150–162. DOI: 10.1016/0196-6774(83)90042-1.

[26]　Leemans, Sander JJ, Wil MP van der Aalst, Tobias Brockhoff, et al. "Stochastic process mining: Earth movers' stochastic conformance". In: *Information Systems* (2021), p. 101724. DOI: 10.1016/j.is.2021.101724.

[27]　Lu, Xixi, Dirk Fahland, and Wil M. P. van der Aalst. "Conformance Checking Based on Partially Ordered Event Data". In: *Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers*. Ed. by Fournier, Fabiana and Jan Mendling. Vol. 202. Lecture Notes in Business Information Processing. Springer, 2014, pp. 75–88. DOI: 10.1007/978-3-319-15895-2_7.

[28]　Mannhardt, Felix, Massimiliano de Leoni, Hajo A. Reijers, et al. "From Low-Level Events to Activities - A Pattern-Based Approach". In: *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*. Ed. by Rosa, Marcello La, Peter Loos, and Oscar Pastor. Vol. 9850. Lecture Notes in Computer Science. Springer, 2016, pp. 125–141. DOI: 10.1007/978-3-319-45348-4_8.

[29]　Pegoraro, Marco and Wil M. P. van der Aalst. "Mining Uncertain Event Data in Process Mining". In: *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*. IEEE, 2019, pp. 89–96. DOI: 10.1109/ICPM.2019.00023.

[30]　Pegoraro, Marco, Merih Seran Uysal, and Wil M P van der Aalst. "PROVED: A Tool for Graph Representation and Analysis of Uncertain Event Data". In: *International Conference on Application and Theory of Petri Nets and Concurrency*. Springer. 2021.

[31]　Pegoraro, Marco, Merih Seran Uysal, and Wil M. P. van der Aalst. "Discovering Process Models from Uncertain Event Data". In: *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*. Ed. by Francescomarino, Chiara Di, Remco

M. Dijkman, and Uwe Zdun. Vol. 362. Lecture Notes in Business Information Processing. Springer, 2019, pp. 238–249. DOI: [10.1007/978-3-030-37453-2_20](10.1007/978-3-030-37453-2_20).

[32] Pegoraro, Marco, Merih Seran Uysal, and Wil M. P. van der Aalst. "Efficient Construction of Behavior Graphs for Uncertain Event Data". In: *Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings*. Ed. by Abramowicz, Witold and Gary Klein. Vol. 389. Lecture Notes in Business Information Processing. Springer, 2020, pp. 76–88. DOI: [10.1007/978-3-030-53337-3_6](10.1007/978-3-030-53337-3_6).

[33] Pegoraro, Marco, Merih Seran Uysal, and Wil M. P. van der Aalst. "Efficient Time and Space Representation of Uncertain Event Data". In: *Algorithms* 13.11 (2020), p. 285. DOI: [10.3390/a13110285](10.3390/a13110285).

[34] Richter, Florian, Janina Sontheim, Ludwig Zellner, et al. "TADE: Stochastic Conformance Checking Using Temporal Activity Density Estimation". In: *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*. Ed. by Fahland, Dirk, Chiara Ghidini, Jörg Becker, et al. Vol. 12168. Lecture Notes in Computer Science. Springer, 2020, pp. 220–236. DOI: [10.1007/978-3-030-58666-9_13](10.1007/978-3-030-58666-9_13).

[35] Rozinat, Anne and Wil M. P. van der Aalst. "Conformance checking of processes based on monitoring real behavior". In: *Information Systems* 33.1 (2008), pp. 64–95. DOI: [10.1016/j.is.2007.07.001](10.1016/j.is.2007.07.001).

[36] Senderovich, Arik, Andreas Rogge-Solti, Avigdor Gal, et al. "The ROAD from Sensor Data to Process Instances via Interaction Mining". In: *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*. Ed. by Nurcan, Selmin, Pnina Soffer, Marko Bajec, et al. Vol. 9694. Lecture Notes in Computer Science. Springer, 2016, pp. 257–273. DOI: [10.1007/978-3-319-39696-5_16](10.1007/978-3-319-39696-5_16).

[37] Stohr, Edward A. and J. Leon Zhao. "Workflow Automation: Overview and Research Issues". In: *Information Systems Frontiers* 3.3 (2001), pp. 281–296. DOI: [10.1023/A:1011457324641](10.1023/A:1011457324641).

[38] Suciu, Dan, Dan Olteanu, Christopher Ré, et al. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. DOI: [10.2200/S00362ED1V01Y201105DTM016](10.2200/S00362ED1V01Y201105DTM016).

[39] Suriadi, Suriadi, Robert Andrews, Arthur H. M. ter Hofstede, et al. "Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs". In: *Information Systems* 64 (2017), pp. 132–150. DOI: [10.1016/j.is.2016.07.011](10.1016/j.is.2016.07.011).

[40] Wang, Jianmin, Shaoxu Song, Xuemin Lin, et al. "Cleaning structured event logs: A graph repair approach". In: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. Ed. by Gehrke, Johannes, Wolfgang Lehner, Kyuseok Shim, et al. IEEE Computer Society, 2015, pp. 30–41. DOI: 10.1109/ICDE.2015.7113270.

[41] Winskel, Glynn. "Petri Nets, Algebras, Morphisms, and Compositionality". In: *Information and Computation* 72.3 (1987), pp. 197–238. DOI: 10.1016/0890-5401(87)90032-0.

[42] van Zelst, Sebastiaan J., Mohammadreza Fani Sani, Alireza Ostovar, et al. "Filtering Spurious Events from Event Streams of Business Processes". In: *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings*. Ed. by Krogstie, John and Hajo A. Reijers. Vol. 10816. Lecture Notes in Computer Science. Springer, 2018, pp. 35–52. DOI: 10.1007/978-3-319-91563-0_3.