Control-flow Reconstruction Attacks on Business Process Models

Henrik Kirchmann[1], Stephan A. Fahrenkrog-Petersen[1,2], Felix Mannhardt[3], and Matthias Weidlich[1]

[1] Humboldt-Universität zu Berlin, Berlin, Germany
`{henrik.kirchmann, stephan.fahrenkrog-petersen, matthias.weidlich}@hu-berlin.de`
[2] Weizenbaum Institute for the Networked Society, Berlin, Germany
[3] Eindhoven University of Technology, Eindhoven, Netherlands
`f.mannhardt@tue.nl`

Abstract. Process models may be automatically generated from event logs that contain as-is data of a business process. While such models generalize over the control-flow of specific, recorded process executions, they are often also annotated with behavioural statistics, such as execution frequencies. Based thereon, once a model is published, certain insights about the original process executions may be reconstructed, so that an external party may extract confidential information about the business process. This work is the first to empirically investigate such reconstruction attempts based on process models. To this end, we propose different play-out strategies that reconstruct the control-flow from process trees, potentially exploiting frequency annotations. To assess the potential success of such reconstruction attacks on process models, and hence the risks imposed by publishing them, we compare the reconstructed process executions with those of the original log for several real-world datasets.

Keywords: Reconstruction Attacks· Process Analysis · Model Play-out

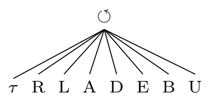
1 Introduction

Under the umbrella of process mining, event logs that have been recorded by information systems facilitate the analysis of qualitative and quantitative properties of business processes [27]. Event logs support information systems engineering through the discovery of process models [1], which are useful for understanding the flow of the process and, once annotated with performance characteristics, help to identify performance bottlenecks and improvement opportunities.

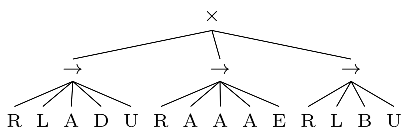
Discovery algorithms generalize and aggregate the behaviour recorded in an event log. As a consequence, individual process executions are not directly represented, when publishing the model [18], e.g., to an external party for the purpose of process certification, staff training or consulting. However, in practice, process models are not limited to the generalized control-flow of a process. Rather, they also contain summary statistics about the behaviour, most prominently execution frequencies or branching probabilities [3,4,10].

ID	Trace
1	Register (R), IV Liquid (L), Antibio (A), Rel. D (D), Return (U)
2	Register (R), Antibio (A), Antibio (A), Antibio (A), Rel. E (E)
3	Register (R), IV Liquid (L), Rel. B (B), Return (U)

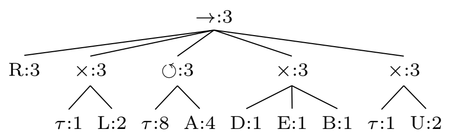
(a)



(b)



(c)



(d)

Fig. 1: (a) A log of patient treatments and three process models for it: (b) a 'flower model' describing any set of traces; (c) a 'trace model' enumerating all traces; (d) a model offering some generalization, potentially annotated with frequencies.

Once a process model is enriched with behavioural statistics, it may be a target of a reconstruction attack. That is, similar to reconstruction attacks in machine learning (ML) [14,12,23], which strive for a characterization of the data used for training the ML model, such an attack aims at deriving insights about the original process executions. Even if the exact reconstruction of the executions is not possible, which would yield severe privacy risks for process stakeholders [28], it is problematic: The combination of the control-flow of a process model with behavioural statistics may facilitate conclusions on confidential information about the underlying business process. For instance, one may reconstruct dependencies between activity executions and behavioural patterns, which reveal internal decision procedures that may be exploited for malicious purposes.

Consider the event log in Fig. 1a, which contains three traces of patient treatments in a hospital. The impact of the generalization adopted in a process model on revealing insights on the original process executions is illustrated by two extreme cases: Fig. 1b shows a 'flower model' that represents any log of traces comprising executions of the respective activities and, hence, does not enable any conclusions. Fig. 1c shows a 'trace model', which models a lossless representation of each recorded trace variant, but has no information on their probability or frequency. The model represents an infinite number of possible event logs with different frequencies of those traces. Nonetheless, all possible logs include all steps of all process executions for this procedure in the hospital. A middle ground is offered by the model in Fig. 1d, which enables control-flow reconstruction to some extent. In particular, annotating the model with frequency information reduces number of event logs modeled by this model and reveals certain insights on the treatments: We conclude that (i) antibiotics have been given at least twice to a single patient, (ii) all release types appear to be equally likely, and (iii) there is at least one patient, who returned after receiving intravenous (IV) liquid. These insights are shared across the control-flow of all possible logs that this model represents.

In this paper, we study reconstruction attacks on process models and analyse how the information contained in process models influences one's ability to reconstruct the control-flow of process executions. We formulate reconstruction attacks as play-out strategies for models given as process trees, which incorporate annotations on branching probabilities or execution frequencies and especially address the question of how to cope with repetitive behaviour in the model. Our experiments of applying the attacks to real-world datasets indicate that frequency-annotated models of structured processes are particularly vulnerable.

Below, we first review related work (Section 2), before defining preliminary notions (Section 3). We then present our approaches to control-flow reconstruction (Section 4), report on our evaluation (Section 5), and conclude (Section 6).

2 Related Work

Any attempt to reconstruct the original process executions from a process model is related to privacy risks, which received much attention in recent years. However, we notice that existing work on the quantification of privacy risks in process mining [28] and the development of a large number of related privacy-preserving techniques [11,13,21,22] has focused primarily on event logs. As such, there is a reasonable level of understanding of these risks and possible mitigation strategies.

The risks induced by process models discovered from event logs, in turn, have been described only recently in [18]. Here, the authors quantify the re-identification risk in frequency annotated block-structured process models with a two-step approach: First, a play-out strategy is used to reconstruct event logs from the process model. Second, the measures proposed in [20] are used to quantify the re-identification risk in the reconstructed log, to then assess the re-identification risk of the original log caused by the release of the process model. However, this approach is only feasible if there is a strong similarity between the reconstructed logs and the original log. This aspect is not further studied in [18], though, which is a research gap that we close with our work.

Play-out strategies and the comparison of the obtained output with a ground truth are also studied in other process mining settings: Conformance checking [6] relates the behaviour described by a process model with behaviour in an event log. Yet, often we cannot fully trust both our model and the source event log, as indicated in [25]. In our context, missing or extra behaviour in the process model as assumed in conformance checking would further impair the chance of a successful reconstruction attack. As such, we assume the process model to be a good representation of the observed process behaviour, which presents the worst case for any attempt to derive insights on the underlying business process, as it simplifies the reconstruction. Both process simulation [5] and stochastic process mining [3] aim to more accurately capture the underlying process observed in process executions. These streams of research investigate how close simulated process executions [7] or the probability distributions in stochastic process model executions [16] are to the actual observations. Unlike our work, however, these approaches do not target the reconstruction of the original log, but on representing the general process behaviour including possible future process executions.

3 Preliminaries

Below, we summarize essential notions for event logs and process trees that are used in the remainder of the paper.

Event Log. Let $\mathcal{A}$ be the universe of activities. A trace $t \in \mathcal{A}^*$, where $\mathcal{A}^*$ is the set of all finite sequences over $\mathcal{A}$, is a sequence of activities. In such a trace, each activity a denotes the recorded event of the execution of a well-defined step in a process. $\mathcal{T} = \mathcal{A}^*$ denotes the universe of traces. A trace $t \in \mathcal{T}$ is represented as $t = \langle a_1, a_2, ..., a_n \rangle$, where $a_1, a_2, ..., a_n \in \mathcal{A}$. With $|t|$ we denote the length of a trace $t \in \mathcal{T}$, i.e., the number of activities in the trace. Denoting with $\mathcal{B}(X)$ the set of all possible multisets over X, let $\mathcal{L} = \mathcal{B}(\mathcal{T})$ be the universe of event logs. An event log $l \in \mathcal{L}$ is a finite multiset of traces.

Process Tree. In this work, we consider process trees as the formal model to capture business processes. A process tree represents a process in a hierarchical (block-structured) way [4,15]. Process trees can be transformed into models of other languages for business processes, such as Petri nets or BPMN models [27]. As such, the ideas outlined in the remainder are not limited to process trees. In general, a process tree denotes a process as a rooted tree. Its leaf nodes represent activities and all other nodes represent operators. Following the aforementioned references, we formally define a process tree as follows:

Definition 1 (Process Tree). *Let $A \in \mathcal{A}$ be a finite set of activities and let $\tau \notin A$ denote the silent activity, which cannot be observed in a trace. A process tree Q, is defined recursively as:*

- *If $a \in A \cup \{\tau\}$, then $Q = a$ is a process tree.*
- *If $n \geq 1$, $Q_1, Q_2, \ldots, Q_n$ are process trees, and $\oplus \in \{\rightarrow, \times, \wedge\}$, then $Q = \oplus(Q_1, Q_2, \ldots, Q_n)$ is a process tree.*
- *If $n \geq 2$, $Q_1, Q_2, \ldots, Q_n$ are process trees, and $\oplus = \circlearrowleft$, then $Q = \oplus(Q_1, Q_2, \ldots, Q_n)$ is a process tree.*

A process tree might be annotated with information about probabilities or frequencies of the recorded behaviour. We capture such information by a weight $w \in \mathbb{R}$ that is assigned to a process tree Q, which is denoted by $Q : w$.

Consider Fig. 2, which shows the process tree $Q = \rightarrow(\wedge(a, \times(b, c)), \circlearrowleft(d, \tau))$. The $\rightarrow$ operator refers to the execution of the child nodes in sequential order, i.e., the execution of $\wedge(a, \times(b, c))$ is followed by the execution of $\circlearrowleft(d, \tau)$. The $\wedge$ operator defines the execution of all of its child nodes in any order, while the $\times$ operator specifies an exclusive choice. The $\circlearrowleft$ operator has at least two children, the first being the "do" part of a loop; all other children representing "redo" parts. The "do" part is always executed; execution of the "redo" part is optional and only one of the "redo" parts is executed, before the "do" part is executed again.

To formalize the semantics of process trees, we need the following auxiliary operators for general sequences [27]:

Definition 2 (Auxiliary Operators). *Let $\sigma_1, \sigma_2 \in A^*$ be two sequences over A and let $S_1, S_2, \ldots, S_n \subseteq A^*$. We define two operators as:*

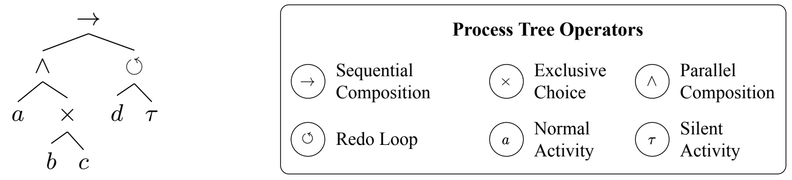


Fig. 2: Visualization of the process tree $Q = \rightarrow(\wedge(a, \times(b, c)), \circlearrowleft(d, \tau))$.

- *Concatenation:* $\sigma_1 \cdot \sigma_2 \in A^*$ *concatenates two sequences. The concatenation operator can be generalized to sets of sequences by* $S_1 \cdot S_2 = \{\sigma_1 \cdot \sigma_2 \mid \sigma_1 \in S_1 \wedge \sigma_2 \in S_2\}$ *and* $\bigodot_{1 \le i \le n} S_i = S_1 \cdot S_2 \cdots S_n$ *concatenates an ordered collection of sets of sequences.*
- *Shuffle:* $\sigma_1 \diamond \sigma_2 \in A^*$ *generates the set of all interleaved sequences. The shuffle operator can be generalized to sets of sequences by* $S_1 \diamond S_2 = \{\sigma_1 \diamond \sigma_2 \mid \sigma_1 \in S_1 \wedge \sigma_2 \in S_2\}$ *and* $\Diamond_{1 \le i \le n} S_i = S_1 \cdot S_2 \cdots S_n$ *shuffles an ordered collection of sets of sequences.*

Given two sequences $\sigma_1 = \langle a, b\rangle$ and $\sigma_2 = \langle c, d\rangle$, the operators yield $\sigma_1 \cdot \sigma_2 = \langle a, b, c, d\rangle$ as well as $\sigma_1 \diamond \sigma_2 = \{\langle a, b, c, d\rangle, \langle a, c, b, d\rangle, \langle c, a, b, d\rangle, \langle a, c, d, b\rangle, \langle c, a, d, b\rangle, \langle c, d, a, b\rangle\}$. Furthermore, we define the language of a process tree as follows.

Definition 3 (Language of a Process Tree). *Let $Q \in \mathcal{Q}$ be a process tree over a set A. $\mathcal{L}(Q)$ denotes the language of Q, i.e., the set of traces that can be generated. $\mathcal{L}(Q)$ is defined recursively:*
- $\mathcal{L}(Q) = \{\langle a\rangle\}$, *if* $Q = a \in A$,
- $\mathcal{L}(Q) = \{\langle\rangle\}$, *if* $Q = \tau$,
- $\mathcal{L}(Q) = \bigodot_{1 \le i \le n} \mathcal{L}(Q_i)$, *if* $Q = \rightarrow(Q_1, Q_2, \ldots, Q_n)$,
- $\mathcal{L}(Q) = \bigcup_{1 \le i \le n} \mathcal{L}(Q_i)$, *if* $Q = \times(Q_1, Q_2, \ldots, Q_n)$,
- $\mathcal{L}(Q) = \Diamond_{1 \le i \le n} \mathcal{L}(Q_i)$, *if* $Q = \wedge(Q_1, Q_2, \ldots, Q_n)$,
- $\mathcal{L}(Q) = \{\sigma_1 \cdot \sigma_1' \cdot \sigma_2 \cdot \sigma_2' \cdots \sigma_m \in A^* \mid m \ge 1 \ \wedge\ \forall\, 1 \le j \le m : \sigma_j \in \mathcal{L}(Q_1) \ \wedge\ \sigma_j' \in \bigcup_{2 \le i \le n} \mathcal{L}(Q_i)\}$, *if* $Q = \circlearrowleft(Q_1, Q_2, \ldots, Q_n)$.

Based on the definition above, we see that $\mathcal{L}(\circlearrowleft(Q_1, Q_2, \ldots, Q_n)) = \mathcal{L}(\circlearrowleft(Q_1, \times(Q_2, \ldots, Q_n)))$, i.e., a restriction to a single "redo" child does not lower the expressiveness of the model. As a consequence, without loss of generality, we assume that an $\circlearrowleft$ operator has only one "redo" child in the remainder, to simplify the presentation. Turning to Fig. 2, the language of Q is unbounded due to the loop operator, i.e., $\mathcal{L}(Q) = \{\langle a, b, d\rangle, \langle a, c, d\rangle, \langle b, a, d\rangle, \langle c, a, d\rangle, \langle a, b, d, d\rangle, \langle a, c, d, d\rangle \ldots\}$.

4 Control-Flow Reconstruction

As illustrated in our initial example in Fig. 1, process models may facilitate conclusions on the event log from which the model was discovered. We therefore formulate the respective control-flow reconstruction attacks as five different play-out strategies that, given a process tree, generate a reconstructed event log. We will compare in Section 5 the control-flow of the reconstructed log with the

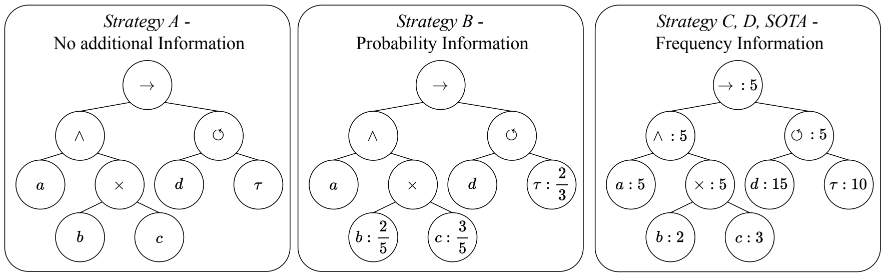


Fig. 3: The three common scenarios how a model discovered from log $L = [\langle a,b,d\rangle, \langle a,c,d\rangle, \langle c,a,d,d,d,d,d\rangle^2, \langle b,a,d,d,d\rangle]$ can be released.

control-flow of the original log. The strategies are motivated to reflect the three common ways a process model can be released, see Fig. 3, and how they utilize this information to reconstruct the control-flow. This enables us to analyze the impact different kinds of additional information, as well as different usage of this information, have on the reconstruction success. We first introduce our play-out strategies in Section 4.1. Next, in Section 4.2, we discuss specific issues related to the handling of loops.

4.1 Play-out Strategies for Process Trees

In essence, a play-out strategy defines a particular traversal of the process tree according to the control-flow structure defined by it.

Definition 4. *Given a process tree Q, a play-out strategy p is a function that, applied to Q, returns an event log $L_p \subseteq \mathcal{B}(2^{\mathcal{L}(Q)})$.*

Before we formalize the individual aspects of each play-out strategy, we define some general rules that guide all strategies and apply to any traversal of a process tree, i.e., the generation of a single trace based on the process tree:

- R_0 Start the traversal with an empty trace.
- R_1 If a non-silent leaf node (i.e., not τ) is encountered during the traversal, the respective activity is concatenated to the current trace.
- R_2 If a silent leaf node (τ) is encountered during the traversal, the current trace remains unchanged.
- R_3 Once the traversal considered all children of a node Q, it returns to and continues with the parent node. If Q is the root node, the reconstructed trace will be added to the result.

Similarly, we provide some general rules for the play-out of process trees that relate to the operators for sequential composition and parallel composition. $R_\wedge$ does not apply to the *SOTA Strategy* [18].

- $R_\rightarrow$ When $Q = \rightarrow (Q_1, \ldots, Q_n)$ is encountered, the traversal continues with the child nodes $Q_1, \ldots, Q_n$ in the respective order.

$R_\wedge$ When $Q = \wedge(Q_1, \ldots, Q_n)$ is encountered, all $U_1, \ldots, U_n$ sub-trees are executed until all sub-trees reach a leaf node or Q again, then it is chosen uniformly at random which leaf node is executed. This is repeated until all sub-trees are completely executed and back to Q. Thus, true parallelization of activities is achieved.

Based thereon, we define a first basic play-out strategy that is not based on any additional information on frequencies.

Strategy A. This strategy considers only the semantics of the operators in a process tree. For the exclusive choice operator and the loop operator, the respective control-flow choices are taken uniformly at random:

$R^A_\times$ When $Q = \times(Q_1, \ldots, Q_n)$ is encountered, traversal continues with one child Q_i, $1 \leq i \leq n$, chosen uniformly at random.
$R^A_\circlearrowleft$ When $Q = \circlearrowleft(Q_1, Q_2)$ is encountered, traversal continues with the child Q_1. Then, a choice between executing child Q_2 and then child Q_1 or ending the traversal of Q is made. This decision is made with probability $1/2$, until the option to end the traversal of Q is taken.

Strategy B. This strategy interprets the weights assigned to nodes as fixed branching probabilities. These probabilities will be derived from frequencies. For notational purposes, we let the strategy compute these probabilities using the actual frequencies. But the derived fixed probabilities that are computed and used by this strategy correspond to the probabilities a probability-annotated model would have:

$R^B_\times$ When $Q = \times(Q_1 : w_1, \ldots, Q_n : w_n) : w$ is encountered, traversal continues with one child Q_i, $1 \leq i \leq n$, chosen with probability ${}^{w_i}/_{\sum_{1\leq j\leq n} w_j}$.
$R^B_\circlearrowleft$ When $Q = \circlearrowleft(Q_1 : w_1, Q_2 : w_2) : w$ is encountered, we follow the approach from $R^A_\circlearrowleft$, but adopt the probability of $1 - {}^{w}/_{w_1}$ for the option to continue with children Q_2 and Q_1, and ${}^{w}/_{w_1}$ for the option to end Q's traversal.

Further strategies leverage the actual frequencies and interpret them in absolute terms. That is, traversal changes the respective counts, which is captured by the following rule that applies to all remaining strategies:

R_4 Upon traversal of a node $Q : w$, the value of w will be decreased by one.

Based thereon, we distinguish two strategies to incorporate the absolute frequencies in the traversal of nodes that model control-flow choices.

Strategy C. This strategy takes control-flow choices related to exclusive choice operators and loop operators, with probabilities that are determined based on the leftover frequencies:

$R^C_\times$ When $Q = \times(Q_1 : w_1, \ldots, Q_n : w_n) : w$ is encountered, traversal continues with one child Q_i, $1 \leq i \leq n$, chosen with probability ${}^{w_i}/_{\sum_{1\leq j\leq n} w_j}$. Note that this rule differs from the one of *Strategy B*, since the weights w_i are continuously updated during traversal, as mentioned above.

$R^C_{\circlearrowleft}$ When $Q = \circlearrowleft(Q_1 : w_1, Q_2 : w_2) : w$ is encountered, traversal first continues with child Q_1. If after this, $w_1 = w_2$ holds, traversal iteratively continues with children Q_2 and Q_1, until $w_1 = 0$. Intuitively, such an approach collects all leftover frequencies with the last trace that is generated. Otherwise, if $w_1 \neq w_2$, we distinguish $w_1 = 0$, in which case traversal of Q ends, and $w_1 > 0$, in which case traversal iteratively continues in the loop as in *Strategy B*, with probability $1 - {}^{w}/_{w_1}$ for the option including the children Q_2 and Q_1, and ${}^{w}/_{w_1}$ for the option to end traversal.

Strategy D with Variance *v*. This strategy denotes an adaptation of *Strategy C*. While it also takes all control-flow choices with probabilities that are determined based on the leftover frequencies, it includes a normal distribution to decide on the number of loop iterations. As usual, by $\mathcal{N}(\mu, \sigma^2)$, we denote a normal distribution with mean μ and variance σ^2. Then, the strategy replaces the rule of *Strategy C* for the loop operator by the following rule:

$R^D_{\circlearrowleft}$ When $Q = \circlearrowleft(Q_1 : w_1, Q_2 : w_2) : w$ is encountered, traversal continues the same way as in $R^C_{\circlearrowleft}$. All rules apply, except for the case when $w_1 \neq w_2$ and $w_1 > 0$. In this case, we will traverse the loop, i.e., children Q_2 and Q_1 a total of $\min(\lfloor|x|\rfloor, w_2)$-times, where x is randomly sampled from the normal distribution, $x \sim \mathcal{N}(\frac{w_2}{w}, v)$. To get a positive integer, we compute $\lfloor|x|\rfloor$. Our experiments have shown that rounding up or concatenating the functions in different order had no measurable impact. We return to the parent node after we traversed the loop $\min(\lfloor|x|\rfloor, w_2)$-times or $w_1 = 0$.

State-of-the-Art (SOTA) Strategy. This strategy was introduced in [18]. It traverses a process tree like *Strategy C* but does a sequential play-out for the parallel composition, hence $R^{SOTA}_{\wedge} = R_{\rightarrow}$, and for the exclusive choice operator:

$R^{SOTA}_{\times}$ When $Q = \times(Q_1 : w_1, \ldots, Q_n : w_n) : w$ is encountered, consider the child nodes $Q_1, \ldots, Q_n$ in their respective order. Traversal continues with the first child Q_i with a positive weight, i.e., $w_i > 0$.

4.2 Reconstructing the Number of Loop Iterations

Next, we discuss the motivation for the approach presented in *Strategy D* that determines the number of loop iterations upfront, instead of relying solely on branching probabilities.

As an illustrative example, consider the process tree $\circlearrowleft(a{:}10000, \tau{:}9000){:}1000$ discovered from event log $L = [\langle a, a, a, a, a, a, a, a, a, a\rangle^{1000}]$. Fig. 4 shows the trace length distribution of L and the normalized trace length distribution of 100 play-outs, each containing 1000 traces for each play-out strategy. The majority of traces produced by all strategies except *Strategy D* are much shorter than the traces of log L. The reason is that these play-out strategies, and modelling techniques such as [24] or partly [4], capture the execution of the "redo" child of a loop operator with some probability p. Suppose p is fixed, like in *Strategy A* and

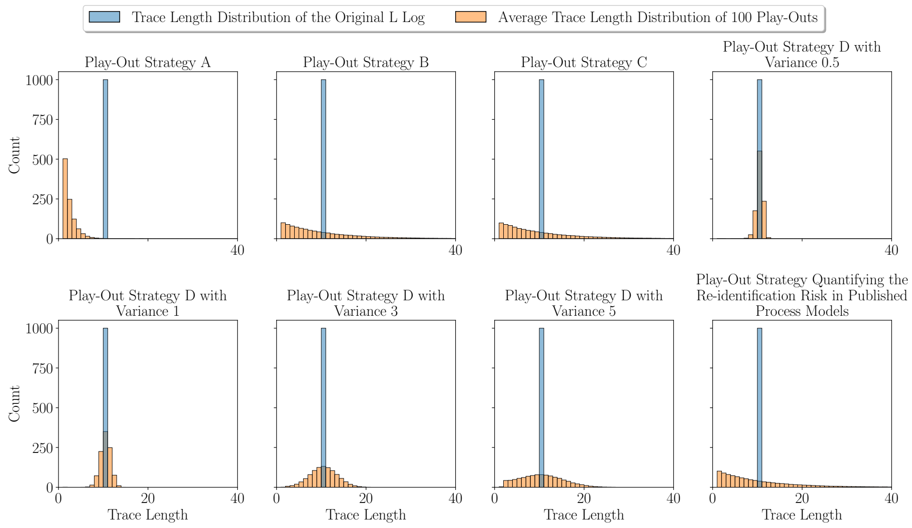


Fig. 4: The distribution of the average trace length when playing out 100 traces using each play-out strategy from $\circlearrowleft$ $(a{:}10000, \tau{:}9000){:}1000$, along with the distribution of the original log $L = [\langle a, a, a, a, a, a, a, a, a, a\rangle^{1000}]$.

B. In that case, each iteration is a Bernoulli trial with the number of iterations being a geometric variable [4]. Because in *Strategy C* and the *SOTA Strategy*, the probability changes at each loop iteration, the sequence of iterations is not a sequence of Bernoulli trials. Nonetheless, our experiments show that the resulting distributions of iterations are actually close to a geometric distribution.

To reconstruct traces with consistently more loop iterations, one must decide on the number of loop iterations upfront. When playing out a process tree $Q = \circlearrowleft (Q_1 : w_1, Q_2 : w_2) : w$, we know that the traces of the original log, took in w $\circlearrowleft$-executions, on average ${}^{w_2}/_{w}$ loop repetitions. In our example, traces took, on average, ${}^{9000}/_{1000} = 9$ loop repetitions. *Strategy D* uses this information to set the mean of the normal distribution to ${}^{w_2}/_{w}$, each time we execute the $\circlearrowleft$ node. Here, the choice of a normal distribution is motivated by the fact that, in each process execution, multiple choices on (re)entering the loop are taken. Once these choices can be assumed to be independent and identically distributed (i.i.d.), the observational error is expected to tend to a normal distribution. Even in the absence of knowledge on the variance parameter v of the distribution, we expect it to provide a suitable representation of the number of loop iterations per process execution.

Compared to *Strategy D* the other strategies will perform worse when number of loop iterations is distributed such that the values are large and the variance is low. *Strategy D* performs worse when the number of loop iterations is distributed with large variance and the values are not centered around the chosen mean.

5 Experimental Evaluation

In this section, we evaluate how well our proposed play-out strategies can reconstruct the control-flow of logs from their discovered models. We present our experimental setup in Section 5.1, and discuss evaluation measures in Section 5.2. Then, we describe our results in Section 5.3 and discuss them in Section 5.4.

5.1 Experimental Setup

Experimental Pipeline. We use the inductive miner without noise filtering to discover the process trees. The lack of noise filtering results in a perfect fitting process model, a necessary condition to be able to fully reconstruct the log from the model. In our setting, it is impossible to reconstruct control-flow information about the event log that is not present in the model. To determine the frequency of nodes, we replay each trace of the original event log on the process tree. Each time we visit a node, we increase its weight by one. For each strategy, we do 100 play-outs of each process tree to obtain the evaluation logs. For *Strategy A* and *Strategy B*, we fix the number of traces generated to the number of traces in the original log. Hence, our results for these strategies are an upper bound for the reconstruction risks, since usually the number of traces is not known to the adversary, when the model is not annotated with absolute frequencies.

Dataset. We evaluate the play-out strategies using four real-world event logs: the BPIC 2015 Municipalities log [9], the BPIC 2017 log [8], the BPIC 2013 Closed Problems log [26], and the Sepsis log [19]. In Table 1 we show certain characteristics of the logs. The logs range from unstructured (BPIC 2015) to structured (BPIC 2013) and also differ drastically in the number of their activities. In addition to different levels of structuredness, we also considered different trace lengths, since longer traces are potentially harder to reconstruct. The logs differ from having relatively short (BPIC 2013) to very long traces (BPIC 2015).

Implementation. Our implementation is available on GitHub[4]. We used the inductive miner and earth mover's distance of PM4Py [2]. The runtime of our implemented play-out strategies is fast. On a machine with an AMD Ryzen 5600G a play-out of the BPIC 2013 log is generated in under one second and in 30 seconds one play-out for the BPIC 2017 log.

5.2 Evaluation measures

Trace Length Distribution. We look into the trace length distribution to investigate the impact of how different play-out strategies handle the $\circlearrowleft$ operator. We plot the normalized distributions of each play-out strategy and the distribution of the original log as histograms for the BPIC 2017 log. The plots for the other event logs can be found in our appendix[5]. We calculate the similarity of the distributions using the normalized histogram intersection for all logs:

[4] https://github.com/henrikkirchmann/Control-Flow-Reconstruction

[5] https://github.com/henrikkirchmann/Control-Flow-Reconstruction/tree/main/Appendix

Table 1: Descriptive statistics of the event logs.

Event Log	**# of Traces**	**# of Variants**	$\frac{\text{\# of Variants}}{\text{\# of Traces}}$	**Trace Length**			**# of Activities**
				Min.	**Avg.**	**Max.**	
BPIC 2017	31509	15930	0.505	10	38.1	180	27
BPIC 2015 Municipalities	1199	1170	0.975	2	43.5	101	399
BPIC 2013 Closed Problems	1487	183	0.123	1	4.4	35	5
Sepsis Cases	1050	846	0.805	3	14.4	185	17

Definition 5 (Normalized histogram intersection (NHI) size). *Let I and M be histograms, each containing n bins, and let I_j respectively M_j denote the number of elements in bin j of I respectively of M. The normalized histogram intersection size is defined to be*

$$NHI(I, M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j}.$$

Earth Mover's Distance (EMD). We compare the reconstructed logs with the original log through the earth mover's distance (EMD) introduced in [17]. The EMD measures the distance, by the Levenshtein distance function, between the trace variant distributions of the event logs. Through the EMD, we can measure the difference between the logs in terms of their control-flow of each trace. This allows us to see if the play-out strategies produce logs that have similar control-flow to the original log, without the need for traces to be the same.

Normalized Multiset Intersection (NMI) Size. The multiset intersection size between two event logs represents the count of traces from the original log that are completely and successfully reconstructed. The normalized multiset intersection size, denoted by $NMI(L_1, L_2)$, is defined as the sum of the minimum occurrences of each trace σ in both multi sets L_1 and L_2 divided by $|L_1|$. For example, given the event logs $L_1 = [\langle a, b\rangle^2, \langle a, b, c\rangle^2]$ and $L_2 = [\langle a, b\rangle^3, \langle a, b, b\rangle]$, we have $NMI(L_1, L_2) = {}^2/_4$. Through this metric, we can determine if the play-out strategies create traces that are exactly the same as the traces of the original log.

Reconstructed Eventually Follows Relations. To check how many dependencies between activities we can reconstruct, we compare the eventually follows relations of the reconstructed logs to the original log. An eventually follows relation between two activities a and b, can be one of three types: (i) The relation between a and b is of type *always follows* when in all traces of the log activity b will occur eventually after a; (ii) *sometimes follows* when in some but not all traces of the log b will occur eventually after a; (iii) *never follows* when in no trace of the log b will eventually occur after a. To quantify the differences between the predicted eventually follows relations of our play-out strategies and the ones of the original log, we calculate the F_1-Scores, as the harmonic mean of precision and recall.

5.3 Results

Trace Length Distribution. Table 2 shows the NHI size with higher values, meaning better reconstruction of the trace length distribution. We can observe NHI values above 0.7 for 3 out of 4 of the event logs, with the exception being the BPIC 2015 log. Therefore, we can conclude that it is generally possible to mimic the trace length distribution and to rediscover general control-flow properties.

Considering the results in more detail, the success of the reconstruction might depend highly on the handling of loops. This aspect can be seen by the difference between the different settings for *Strategy D*. For BPIC 2015 the worst setting (*Strategy D with Variance 0.5*) reached a NHI of 0.19, while the best setting (*Strategy D with Variance 5*) led to a NHI value of 0.44. In Fig. 5, we can see that, compared to the other strategies, *Strategy D with Variance 0.5-3* creates considerably fewer traces of length below 20.

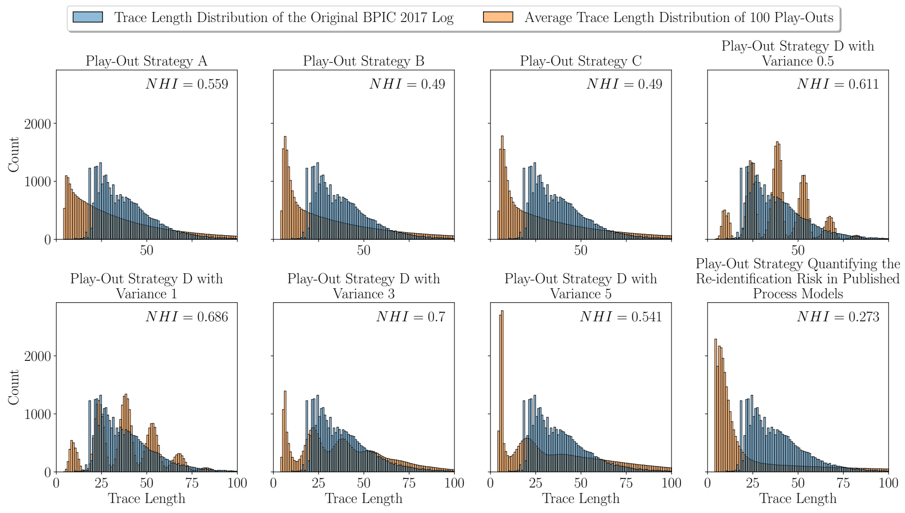


Fig. 5: The trace length distributions for the BPIC 2017 log.

Earth Mover's Distance. Table 2 shows the EMD. Smaller values, correspond to higher similarity between the control-flow of the play-outs and the original log. Unfortunately, computing the EMD for the BPIC 2017 log was not feasible. We can observe that for the BPIC 2013 log, it is possible to generate logs that can be very close to the original event log. However, the BPIC 2015 log shows that this might not be possible for all logs. We can observe that the difference between the logs is significantly larger than between the strategies. This lets us conclude that specifics of the process itself determine the chance of success for the adversary. *Strategy A* that has no additional information about the control-flow performs the worst but is followed by the *SOTA Strategy*, despite having knowledge about the absolute frequencies. The other Strategies reconstruct the control-flow of the original log with similar success in terms of the EMD.

Table 2: NHI size, EMD and NMI size of 100 play-out logs and the original log. Higher NHI/NMI values and lower EMD values denote higher reconstruction success, the values that indicate the highest reconstruction success are bold.

	BPIC17		BPIC13			BPIC15		Sepsis		Average	
Strategy	NHI	EMD	NHI	EMD	NMI	NHI	EMD	NHI	EMD	NHI	EMD
A	0.55	-	0.66	0.35	0.19	0.17	0.93	0.50	0.74	0.47	0.67
B	0.49	-	0.83	**0.10**	**0.59**	0.43	0.87	**0.70**	0.52	0.61	0.50
C	0.49	-	0.83	0.11	**0.59**	0.43	0.87	**0.70**	0.51	0.61	0.50
D Var. ½	0.61	-	0.60	0.22	0.37	0.19	**0.86**	0.54	0.51	0.48	0.53
D Var. 1	0.68	-	0.66	0.18	0.44	0.22	**0.86**	0.61	**0.50**	0.54	0.51
D Var. 3	**0.70**	-	**0.88**	**0.10**	**0.59**	0.38	0.87	0.61	0.51	**0.64**	**0.49**
D Var. 5	0.54	-	0.76	0.17	0.51	**0.44**	0.87	0.51	0.53	0.56	0.52
SOTA [18]	0.27	-	0.83	0.14	0.52	0.38	0.92	0.69	0.62	0.54	0.56
Avg. for Log	0.54	-	0.75	0.17	0.47	0.33	0.88	0.60	0.55	0.55	0.53

Normalised Multiset Intersection Size. For all logs except the BPIC 2013 log, the NMI Size was below 0.01. The values for the BPIC 2013 log are shown in Table 2. This strongly suggests that the adversary might often not be able to generate the specific traces of the original log. However, for the BPIC 2013 log, we can see that *Strategy A* performs worse than the strategies with knowledge about frequencies. The results indicate again that knowledge about relative or absolute frequencies in process models can significantly increase the reconstruction success. Also, differences between the different settings of *Strategy D* can be significant.

Reconstructed eventually follows relations. Table 3 shows the F_1-Scores of the reconstructed eventually follows relations. Regarding the reconstruction of always follows (AF) relations, all strategies perform similar, except for *A*, which performed the worst, and the *SOTA Strategy*, which is the second worst. Notably, the F_1-Scores of *SOTA* are by far the lowest in 3 out of 4 evaluated logs. For the sometimes follows (SF) relations, the *SOTA Strategy* again performs the worst, despite having access to absolute frequency information, that is not available to strategies *A* and *B*. Strategy *B* outperforms *D*, despite having only knowledge of branching probabilities and the number of traces to generate. In the case of never follows (NF) relations, the performance of all strategies, except for *A*, which performed the worst, is again very similar.

Overall, we observe a significant level of variance in the F_1-Scores, reaching from cases where no reconstruction is possible to values as high as 0.89. While it is expected that the highest values are obtained for the never follows (NF) relations, since they relate to behaviour that shall not be generated according to the process model, we also observe relatively high F_1-Scores for the always follows (AF) relations. Those can be interpreted as invariants on the presence of activity executions, and hence, are particular interesting from a reconstruction point of view. With F_1-Scores around 0.6, we conclude that a good share of these relations are reconstructed successfully.

Table 3: Average F_1-scores of 100 play-outs for the reconstructed always (AF), sometimes (SF) and never follows (NF) relations. Higher values denote higher reconstruction success, the highest values are bold.

	BPIC17			BPIC13			BPIC15			Sepsis			Average		
Strat.	AF	SF	NF	AF	SF	NF	AF	SF	NF	AF	SF	NF	AF	SF	NF
A	0.40	0.38	0.39	0.69	0.63	0.00	0.03	0.16	0.81	0.23	0.46	0.02	0.34	0.41	0.30
B	0.52	0.54	0.47	**0.75**	**0.80**	0.87	0.20	0.48	0.71	0.52	0.60	0.33	0.51	**0.61**	0.60
C	0.55	**0.55**	0.47	**0.75**	0.79	0.85	0.20	**0.49**	0.76	0.49	0.60	0.32	0.50	**0.61**	0.60
D ½	**0.64**	0.54	0.46	0.64	0.52	0.87	0.22	0.48	0.80	**0.59**	**0.61**	0.36	**0.52**	0.54	**0.62**
D 1	**0.64**	0.52	0.46	0.63	0.50	**0.89**	**0.23**	0.47	0.79	0.57	0.60	0.35	**0.52**	0.52	**0.62**
D 3	0.61	0.43	0.45	0.61	0.43	0.86	**0.23**	0.44	0.77	**0.59**	0.60	0.30	0.51	0.48	0.60
D 5	0.62	0.43	0.46	0.61	0.43	0.86	**0.23**	0.43	0.76	0.56	0.57	0.25	0.51	0.47	0.58
SOTA	0.16	0.30	**0.57**	0.71	0.71	0.41	0.01	0.06	**0.86**	0.14	0.34	**0.53**	0.44	0.35	0.59
Avg.	0.51	0.46	0.47	0.67	0.60	0.70	0.17	0.36	0.78	0.46	0.55	0.31	0.48	0.50	0.56

5.4 Discussion

Comparison of Play-out Strategies. Overall, *Strategy A* performed the worst of all play-out strategies. This is expected, since *Strategy A* lacks information about probabilities or frequencies in the process model. We conclude that it is indeed harder or even impossible to successfully reconstruct much of the control-flow of the original log the un-annotated process model was discovered from.

The play-outs from *Strategy B* and *Strategy C* were almost similar in our evaluated statistics. Knowledge of each node's left-over frequency did not help *Strategy C* to make better reconstruction decisions than *Strategy B*, when *Strategy B* knows how many traces to generate. This indicates that when a log with branching probabilities and the number of how many traces the original log contains are released, the model will reveal nearly the same amount of control-flow information as it would have done when released with absolute frequencies.

Strategy D with Variance v was unable to consistently outperform *Strategy C* or *Strategy B*. In our experiments, we could observe that setting the variance value between 1 and 3 led to good results. A limitation of this strategy is that an attacker does not know what variance to pick. When we sample from the normal distribution with a large variance, like in *Strategy D with Variance* 5 we generate traces that took numerous loop iterations. The longest trace we generated with *Strategy D with Variance* 5 for the BPIC 2017 log was 863 activities long. Those long traces consume much of the frequency weights, thus forcing the other reconstructed traces to be shorter.

The *State-of-the-Art Strategy [18]* performed worse than *Strategy B* despite knowing the left-over frequencies of each node. This indicates that we should not execute $\times$ and $\wedge$ nodes sequentially if we want to reconstruct the control-flow from the original log. We saw for example that this results in many false positive always follows relations and hence low F_1-scores.

Assessment of Reconstruction Risk. In our experiments, we observed that we were able to reconstruct control-flow properties (trace length distribution). Additionally, we were also able to generate logs with a small distance to the original log for one process and a reasonable distance for another. However, we were only able to reconstruct concrete cases from one log. Finally, we illustrated that information on the eventually follows relations of the underlying process may be reconstructed to a significant extent, revealing co-occurrences of activity executions and their mutual exclusiveness.

However, we acknowledge that, in practice, an attacker also always faces uncertainty about the reconstructed information, i.e., if a reconstructed trace was actually part of the original log. This, in general, hinders the operationalization of the insights obtained through a reconstruction attack. This leads to the following assessment in terms of the reconstruction risk of process models: In general, it is possible to retrieve traces from process models. However, this is not possible for all process models. Therefore, reconstruction risks of process models need to be considered and taken seriously, but their risk should not be overstated. Instead, it should be considered that while process models might not lead to the reconstruction of complete traces, even partially reconstructed information might be exploitable for an adversary.

6 Conclusion

To mitigate confidentially risks, one may resort to publishing a process model instead of an event log for operational analysis. In this paper, we argued that such an approach also potentially incurs risks, since some information about the original process executions may be reconstructed from the released process model. We studied this risk and formulated reconstruction attacks as play-out strategies for models given as process trees. We conclude from our experiments that the reconstruction risk for process trees modelled by the inductive miner from complex real-world event logs is very low. However, there is a considerable reconstruction risk for more structured event logs. The annotation of process trees with frequency information increases the reconstruction risk considerably. Compared to the state of the art, our approaches can consistently provide better results, even with less background knowledge.

In future work, we plan to shift our focus from the quantity of information that can be reconstructed to a more nuanced analysis. This will involve examining the specific types of information that can be reconstructed and the associated uncertainties from an attacker's point of view. Our goal is to develop algorithms capable of answering questions such as: given a process model, which traces can be reconstructed that occurred with absolute certainty in the original log.

Acknowledgements. This work was supported by the German Federal Ministry of Education and Research (BMBF), grant number 16DII133 (Weizenbaum-Institute).

References

1. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. IEEE TKDE **31**(4), 686–705 (2019)
2. Berti, A., van Zelst, S.J., Schuster, D.: Pm4py: A process mining library for python. Softw. Impacts **17**, 100556 (2023). https://doi.org/10.1016/J.SIMPA.2023.100556
3. Burke, A., Leemans, S.J., Wynn, M.T.: Stochastic process discovery by weight estimation. In: ICPM Workshops. pp. 260–272. Springer (2020)
4. Burke, A., Leemans, S.J., Wynn, M.T.: Discovering stochastic process models by reduction and abstraction. In: Petri Nets. pp. 312–336. Springer (2021)
5. Camargo, M., Dumas, M., González, O.: Automated discovery of business process simulation models from event logs. DSS **134**, 113284 (2020)
6. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018). https://doi.org/10.1007/978-3-319-99414-7
7. Chapela-Campa, D., Benchekroun, I., Baron, O., Dumas, M., Krass, D., Senderovich, A.: Can I trust my simulation model? measuring the quality of business process simulation models **14159**, 20–37 (2023). https://doi.org/10.1007/978-3-031-41620-0_2
8. van Dongen, B.: Bpi challenge 2017. 4tu. Centre for Research Data, Dataset (2017)
9. van Dongen, B.F.: Bpi challenge 2015. In: 11th International Workshop on Business Process Intelligence (BPI 2015) (2015)
10. Elkoumy, G., Pankova, A., Dumas, M.: Privacy-preserving directly-follows graphs: Balancing risk and utility in process mining. arXiv preprint arXiv:2012.01119 (2020)
11. Fahrenkrog-Petersen, S.A., Kabierski, M., van der Aa, H., Weidlich, M.: Semantics-aware mechanisms for control-flow anonymization in process mining. Information Systems p. 102169 (2023)
12. Hidano, S., Murakami, T., Katsumata, S., Kiyomoto, S., Hanaoka, G.: Model inversion attacks for online prediction systems: Without knowledge of non-sensitive attributes. IEICE Trans. Inf. Syst. **101-D**(11), 2665–2676 (2018). https://doi.org/10.1587/TRANSINF.2017ICP0013
13. Hildebrant, R., Fahrenkrog-Petersen, S.A., Weidlich, M., Ren, S.: PMDG: privacy for multi-perspective process mining through data generalization. In: CAiSE. Lecture Notes in Computer Science, vol. 13901, pp. 506–521. Springer (2023)
14. Hilprecht, B., Härterich, M., Bernau, D.: Monte carlo and reconstruction membership inference attacks against generative models. Proc. Priv. Enhancing Technol. **2019**(4), 232–249 (2019). https://doi.org/10.2478/POPETS-2019-0067
15. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - A constructive approach. In: Colom, J.M., Desel, J. (eds.) Petri Nets. LNCS, vol. 7927, pp. 311–329. Springer (2013)
16. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware precision and recall measures for conformance checking in process mining. Inf. Syst. **115**, 102197 (2023)
17. Leemans, S.J., Syring, A.F., van der Aalst, W.M.: Earth movers' stochastic conformance checking. In: BPM Forum. pp. 127–143. Springer (2019)
18. Maatouk, K., Mannhardt, F.: Quantifying the re-identification risk in published process models. In: ICPM Workshops. pp. 382–394. Springer (2021)
19. Mannhardt, F.: Sepsis cases-event log, 4tu. ResearchData. Dataset. DOI: https://doi.org/10.4121/uuid: 915d2bfb-7e84-49ad-a286-dc35f063a460 (2016)

20. Rafiei, M., van der Aalst, W.M.P.: Towards quantifying privacy in process mining. In: ICPM Workshops. LNBIP, vol. 406, pp. 385–397. Springer (2020)
21. Rafiei, M., van der Aalst, W.M.P.: Group-based privacy preservation techniques for process mining. Data Knowl. Eng. **134**, 101908 (2021)
22. Rafiei, M., Wangelik, F., Pourbafrani, M., van der Aalst, W.M.P.: Travag: Differentially private trace variant generation using gans. In: RCIS. LNBIP, vol. 476, pp. 415–431. Springer (2023)
23. Rigaki, M., García, S.: A survey of privacy attacks in machine learning. ACM Comput. Surv. **56**(4), 101:1–101:34 (2024). https://doi.org/10.1145/3624010
24. Rogge-Solti, A., van der Aalst, W.M., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: BPM Workshops. pp. 15–27. Springer (2014)
25. Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., Gal, A.: In log and model we trust? A generalized conformance checking framework. In: BPM. LNCS, vol. 9850, pp. 179–196. Springer (2016)
26. Steeman, W.: Bpi challenge 2013, closed problems. URL: https://doi.org/10.4121/uuid: c2c3b154-ab26-4b31-a0e8-8f2350ddac11. doi: doi **10** (2013)
27. Van Der Aalst, W.: Process mining: data science in action, vol. 2. Springer (2016)
28. Nuñez von Voigt, S., Fahrenkrog-Petersen, S.A., Janssen, D., Koschmider, A., Tschorsch, F., Mannhardt, F., Landsiedel, O., Weidlich, M.: Quantifying the reidentification risk of event logs for process mining: Empiricial evaluation paper. In: CAiSE. pp. 252–267. Springer (2020)