Modelling Data-Aware Stochastic Processes - Discovery and Conformance Checking

Felix Mannhardt[1], Sander J.J. Leemans[2], Christopher T. Schwanen[2], and Massimiliano de Leoni[3]

[1] Eindhoven University of Technology, Netherlands
[2] RWTH Aachen University, Germany
[3] University of Padova, Italy

Abstract. Process mining aims to analyse business process behaviour by discovering process models such as Petri nets from process executions recorded as sequential traces in event logs. Such discovered Petri nets capture the process behaviour observed in a log but do not provide insights on the likelihood of behaviour: the stochastic perspective. A stochastic Petri net extends a Petri net to explicitly encode the occurrence probabilities of transitions. However, in a real-life processes, the probability of a trace may depend on data variables: e.g., a higher requested loan amount will trigger additional checks. Such dependencies are not described by current stochastic Petri nets and corresponding stochastic process mining techniques. We extend stochastic Petri nets with data-dependent transition weights and provide a technique for learning them from event logs. We discuss how to evaluate the quality of these discovered models by deriving a stochastic data-aware conformance checking technique. The implementations are available in ProM, and we show on real-life event logs that the discovery technique is competitive with existing stochastic process discovery approaches, and that new types of stochastic data-based insights can be derived.

Keywords: Stochastic labelled data Petri nets, Process mining, stochastic data-aware process discovery, stochastic data-aware conformance checking

1 Introduction

The largest portion of research in Process Mining has focused on the discovery, conformance checking and enhancement of processes that do not consider the likelihood of the behavior allowed by the process model. In other words, when multiple activities are enabled according to the current state of the process model, they are assumed to have the same probability to occur. This is often unrealistic: even if multiple steps are possible as next, some are more common than others. As an example, in a loan application, when the model allows the notification of the application's acceptance or rejection as next activities, they cannot be associated with the same probability to occur.

These considerations motivate the importance of stochastic process mining, which little research has been carried on. Existing works on stochastic process discovery [3,9,21] and stochastic conformance checking [16] take the frequencies of the event log, which are a sample of the full process behaviour, into account and enable several analysis tasks, e.g., computing the occurrence probability for a trace or obtaining the probability that a marking can be reached [18].

These and other works on stochastic process mining have only focused on mining the activity occurrence probabilities on the basis of the sequence of activities that have happened beforehand. This is certainly valuable. However, in reality, the computation of the probability of an activity to occur as next within a set of enabled ones depends on the current state of the process data variables, as well. For instance, the probability to execute the activity of acceptable notification of a loan applicant will likely depend on the amount requested by the applicants, and on the his/her wealthiness.

We address this shortcoming and enable the discovery of models that, stochastically, fit better to the underlying distribution of the actual process. In particular, the methods rely on process models that are implemented as stochastic Data Petri nets, which are a variation on Data Petri nets [23] to encode the occurrence probabilities of transitions. This requires new methods for both process discovery and conformance checking. Our proposed discovery method learns data-dependant weight functions by building a set of regression problems that are fitted on the observed transition occurrences and the observed data values. To determine the quality of the resulting discovered Stochastic Labelled Data Petri nets (SLDPN), we design a new conformance checking technique that allows to compare the learned process behavior expressed by an SLDPN with that observed in an event log.

In contrast to existing work our methods leverage the information encoded in data attributes from the event log. In particular, our conformance checking technique overcomes the problem of stochastically comparing potentially infinite behaviour defined by an SLDPN with finite and sparse behaviour observed in an event log. The technique has been implemented as plug-ins of ProM, the largest open-source process mining framework. The evaluation has been carried out via a large set of publicly available event logs. For conformance checking, we illustrate that the technique follows the intuition of stochastic conformance and is a proper generalization of existing measures. For discovery of SLDPNs, the inclusion of the data variables for the computation of the activity occurrence probability is shown to improve the stochastic fitness for event logs. Of course, this holds for event logs that include data variables.

Section 2 discusses related work on stochastic process mining. Section 3 reports on the notation and concepts used in the paper. Section 4 introduces SLDPN. Sections 5 and 6 illustrates the techniques proposed for discovery and conformance checking methods of SLDPN. Section 7 reports on the evaluation with many real-life process event logs, while Section 8 concludes this paper, summarizing the paper's contributions and delineating potential future work.

2 Related Work

A large body of work exists on the discovery of data-dependent guards for activities of business process models, including transitions. This research field is often referred to as Decision Mining, starting from the seminal work by Rozinat et al. [31]. Batoulis et al. [5] focus on extracting guards for the outgoing arcs of XOR splits of BPMN models, while Bazhenova et al. [6] aims to discover Decision Model and Notation tables. The discovery of guards for causal nets is discussed in [24]. All of these approaches focus on ensuring that exactly one transition is enabled when a decision point (i.e., an XOR split) is reached. Mannhardt et al. [23] is the only approach that attempts to discover overlapping guards for Petri Nets, namely such that multiple transitions may be enabled in certain data states. However, this work does not provide a probability for transitions, such that the most reasonable assumption is that every enabled transition has the same probability to occur, whereas this paper aims to discover probabilities of transitions to fire when being given a data state. Thus, this paper does not consider guards, but generalisations of guards.

Within the realm of conformance checking, a few research works aim to check the conformance of process executions with respect to a process model represented as a Data Petri Net [22,13], but the conformance of each event-log trace is computed in isolation. This contrasts the notion of stochastic conformance checking that this paper tackles: the determination of the suitability of the overall stochastic behaviour requires the consideration of all traces together.

Stochastic process discovery aims to find a stochastic model such as a stochastic Petri net from an event log. Approaches include those that take a Petri net and estimate their weights, using alignments or frequencies [8], or based on time [29]. Our discovery technique falls into this category, but adds data awareness. Another approach starts from a model with the stochastic behaviour of the log, and reduces this into a smaller model repeatedly [9].

Examples of stochastic conformance checking techniques include the Earth Movers' Stochastic Conformance [20], Entropic Relevance [28] and Probabilistic Trace Alignments [7]. It would be challenging to adapt these to data-aware settings, as our models do not exhibit a stochastic language without data sequences as input. Stochastic models that are declarative have been proposed in [3]; these models express families of stochastic languages.

Key differentiators between stochastic process models and existing Markov-based stochastic models are concurrency, silent transitions and arc-based labels, the combination of which is not the focus of the latter [4,30]. Even though stochastic model checkers such as [15] do not typically consider these three aspects, they could still be applicable after appropriate translations.

Stochastic process discovery also relates to building a model that can compute the firing probability of each enabled transition, as a function of the sequence of fired transitions and data variables. This falls into the realm of predictive process monitoring (cf. [12,25,27]), and several techniques can be leveraged to compute the transition weights. However, the predictive monitoring techniques rely on the typical evaluation of machine-learning techniques, which looks at

each transition in isolation and cannot be used for conformance checking against stochastic process models, which is conversely a global property that looks at traces as whole.

3 Preliminaries

In this section, we introduce required existing concepts.

A multiset is a function mapping its elements to the natural numbers. For a set A, $\mathcal{M}(A)$ denotes the set of all multisets over A. For instance, $[a^2, b]$ is a multiset containing two as and one b. Let X and Y be multisets, then $X \sqsubseteq Y$ if and only if $\forall_a X(a) \leq Y(a)$. The multiset union is $\forall_a (X \uplus Y)(a) = X(a) + Y(a)$. The multiset difference is $\forall_a (X \setminus Y)(a) = \max(0, X(a) - Y(a))$. The set view $\widetilde{X} = \{a \mid X(a) > 0\}$.

Let Σ be an alphabet of activities, i.e. process tasks, such that $\tau \notin \Sigma$. A *data state* is an assignment to numeric[4] variables; let Δ be the set of all data states.

An *event* denotes the occurrence of an activity in a process, and a trace denotes the sequence of events that were executed for a particular case. A *stochastic language* is a weighted set of traces, such that their weights sum up to 1.

A *data event* is an event annotated with a data state, which indicates the data state after the event happened. A *data trace* denotes all data events belonging to a particular case. Formally, let $a_1, \ldots, a_n \in \Sigma$ and $d_0, \ldots, d_n \in \Delta$, then a data trace is a pair of lists $(\langle a_1, \ldots, a_n \rangle, \langle d_0, \ldots, d_n \rangle)$, in which each a_i indicates that event i involved activity a_i, and in which d_0 indicates the data state at the start of the trace, while subsequent $d_{i>0}$ indicate data states after occurrence of event i. Given a data trace $\sigma = (\langle a_1, \ldots, a_n \rangle, \langle d_0, \ldots, d_n \rangle)$, we refer to the sequence $\langle a_1, \ldots, a_n \rangle$ as the activity sequence (σ) and to the sequence $\langle d_0, \ldots, d_n \rangle$ as the data sequence (σ). We refer to the multisets of activity sequences and data sequences of a log L as L and L .

For instance, $(\langle a, b, c \rangle, \langle x = 10, x = 15, x = 20, x = 0 \rangle)$ indicates a data trace with three activities (a, b and c), where the variable x is 10 before a, 15 after a, 20 after b and 0 after c.

A labelled Petri net (LPN) is a tuple (P, T, F, λ, S_0), in which P is a set of places, T is a set of transitions such that $P \setminus T = \emptyset$, $F \in \mathcal{M}(P \times T \cup T \times P)$ is a flow relation, $\lambda : T \not\to \Sigma \cup \{\tau\}$ is a labelling function, and $S_0 \in \mathcal{M}(P)$ is an initial marking. For a node $n \in P \cup T$, we denote $\bullet n = [n' \mid (n', n) \in F]$ and $n \bullet = [n' \mid (n, n') \in F]$. We assume the standard semantics of Petri nets here: a marking consisting of *tokens* on places indicates the state of the net. A transition $t \in T$ is *enabled* in a marking S if $\bullet t \sqsubseteq S$. Let $E(S)$ be the set of all enabled transitions in a marking S. An enabled transition t can fire in a marking S, which changes the marking to $S' = S \setminus \bullet t \cup t \bullet$. The firing of a transition such that $\lambda(t) \neq \tau$ indicates the execution of the mapped activity. A *path* of the net is a sequence of transitions that brings the marking from S_0 to a marking in

[4] Note that our technique only considers numeric variables. Other types of variables can be mapped using a suitable encoding, such as one-hot-encoding.

which no transition is enabled. The corresponding activity sequence is obtained by mapping the path using λ, while removing all transitions mapped to τ:

$$\lambda(\langle t_1, \ldots, t_n \rangle) = \begin{cases} \langle\rangle & \text{if } n < 1 \\ \lambda(t_1) \cdot \lambda(\langle t_2, \ldots, t_n \rangle) & \text{if } \lambda(t_1) \neq \tau \\ \lambda(\langle t_2, \ldots, t_n \rangle) & \text{otherwise} \end{cases}$$

A stochastic labelled Petri net (SLPN) is a tuple $(P, T, F, \lambda, S_0, w)$ such that (P, T, F, λ, S_0) is an LPN and $w \colon T \to \mathbb{R}^+$ is a weight function. In a marking S, the probability to fire $t \in E(S)$ is $\frac{w(t)}{\sum_{t' \in E(S)} w(t')}$. Note that this probability depends on all other enabled transitions, and as such also expresses likelihoods on the order of transitions, even when they are concurrent. The probability of a path $\langle t_1, \ldots, t_n \rangle$ is, due to the independence of subsequent transitions, $\prod_{i=1}^{n} \frac{w(t_i)}{\sum_{t' \in E} w(t')}$. Note that the silent transitions make this a little-studied class of models [18].

In order to validate the quality of a stochastic model, a useful measure is the overlap in probability mass between the stochastic language of an event log and the stochastic language of the model. For stochastic process models, such a measure has been defined as the Unit Earth Movers' Stochastic Conformance (uEMSC) measure [20]. uEMSC measures the overlap in probability mass between a log and a stochastic language, by, for each trace σ of the log L, taking the positive difference between the probability of that trace in the log and the probability of that trace in the SLPN M [20]:

$$\text{uEMSC}(L, M) = 1 - \sum_{\sigma \in L} \max(L(\sigma) - M(\sigma), 0) \tag{1}$$

This rather simple formula uses the probability of a trace σ in a stochastic process model ($M(\sigma)$), which is not trivial to compute. $M(\sigma)$ indicates the sum of all paths through the model that yield the trace σ, however in case of silent transitions labelled τ there may be infinitely many such paths. A solution proposed in [18] – for bounded SLPNs – is to explicitly construct a state space of paths, and compute the trace probability using standard Markov reduction techniques.

The Earth Movers' Distance (EMD) is also known as the Wasserstein distance (W_1) of order 1. For the present special case where we consider unit distances, the EMD is also equivalent to the total variation distance (TV). A proof of the coupling between EMD and TV is for example shown in [14]. Thus, $\text{uEMSC}(L, M) = 1 - \text{TV}(L, M)$.

4 SLDPN

In this section, we extend SLPNs with data-based weight functions to Stochastic Labelled Data Petri nets (SLDPN). Syntactically, SLDPNs are similar to SLPNs, but utilise a weight function that is dependent on a data state.

Definition 1 (Stochastic Labelled Data Petri Net - syntax). *A stochastic labelled data Petri net (SLDPN) is a tuple* $(P, T, F, \lambda, S_0, \mathtt{w})$*, such that* (P, T, F, λ, S_0) *is a Petri net and* $\mathtt{w}\colon T \quad \Delta \; ! \; \mathtt{R}^+$ *is a weight function.*

The state of an SLDPN is the combination of a marking and a data state (d 2 Δ). The marking determines which transitions are enabled, while the data state influences the probabilities of transitions.

Definition 2 (Stochastic Labelled Data Petri Net - semantics). *Let* $(P, T, F, \lambda, S_0, \mathtt{w})$ *be an SLDPN, and let* $hd_0, d_1, \ldots i$ *be a data sequence. The SLDPN starts in state* (S_0, d_0)*. Suppose the SLDPN is in state* (S_i, d_i)*. The probability to re* t 2 $E(S_i)$ *is:*

$$\frac{\mathtt{w}(t, d_i)}{\sum_{t^0 2 E(S_i)} \mathtt{w}(t^0, d_i)}.$$

When a transition t *res, then the new state is* (S_{i+1}, d_{i+1}) *with* $S_{i+1} = S_i \,] \, t \;\; n \;\; t$.

An SLDPN is not executable without further data modelling: the data state influences the likelihood of decisions, but the model does neither describe how the data state is initialised, nor how it changes with the execution of transitions. Thus, an SLDPN potentially has infinitely many stochastic languages.

Furthermore, these definitions do not specify *when* the data state is considered. In a real-life process, the data state may change in between the executions of visible transitions; for instance based on temperature, blood pressure or weather events, time, etc. Our semantics abstracts from the timing of such a decision point, however assumes that a stochastic decision between transitions is made given a data state that does not change at the moment of choice. In future work, this could be extended to choices at arbitrary moments.

Example. Figure 1 shows an example of an SLDPN. The control flow of this SLDPN consists of a choice between a and b, followed by a choice between c and d. The transitions are annotated with weight functions: the weight of a and b depend on the continuous variable X, while c and d depend on the categorical variable Y.

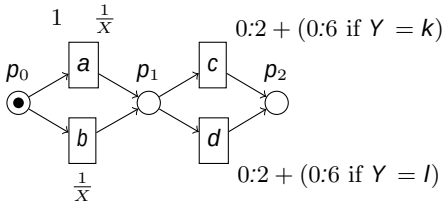


Fig. 1: Example of an SLDPN.

4.1 Trace-Based Execution Semantics & XES Logs

In order to use SLDPNs in a process mining setting, we need to further operationalise the execution semantics. To this end, in this section, we draw links with event logs of the XES standard [2] explicitly. Notice that we assume that the log fits the LPN underlying the SLDPN.

An XES log (XLog) consists of XES traces (XTraces), which are sequences of XES events (XEvents). All XLogs, XTraces and XEvents are annotated with key-value pairs of data attributes. One of the attributes of an XEvent – typically `concept:name` – is designated as the activity. There are also other attributes indicating the time of occurrence and the identifier of the process case.

The activity sequences A and data sequences D of a trace σ can be directly obtained from XES traces. The initial data state d_0 is obtained from the attributes of the XTrace. Note that in the context of our work typically a selection of considered attributes will need to be made. Only attributes that can be assumed to be available at the start of the process case should be considered; however, XTraces of real-life logs may also contain attributes that are the result of the process case executing (e.g., a decision or outcome of the case).

Subsequent data states $d_{i>0}$ are obtained by updating the previous data state with the values from the numeric attributes of that each of the XEvents provides. The activities $a_{i>0}$ are obtained from the designed activity attribute, which is not used for the data state. In our operationalisation, we assume that this data state represents the data *directly after* the event happened. This is not limiting as the mapping could be adapted for other interpretations. Finally, silent transitions are not observed in event logs; thus, there is no information about the data state at the moment of their execution. Therefore, in our operationalisation, silent transitions do not change the data state.

Example. Table 1 shows an example of an event log. In this log, the attribute X is continuously uniform distributed between 1 and 10, and Y is a categorical attribute of $\{k, l\}$ with equal likelihood. Their distribution is shown in Figure 3a. The complete log has 10 000 traces.

Table 1: Running example of an event log with two attributes.

Trace attributes	⟨event #1,	event #2⟩
$X = 5.381523$	$\langle a^{X=5.381523,Y=l},$	$d^{X=5.381523,Y=l}\rangle$
$X = 8.214670$	$\langle a^{X=8.214670,Y=l},$	$d^{X=8.214670,Y=l}\rangle$
$X = 2.463189$	$\langle b^{X=2.463189,Y=l},$	$d^{X=2.463189,Y=l}\rangle$
$X = 6.361540$	$\langle a^{X=6.361540,Y=k},$	$c^{X=6.361540,Y=k}\rangle$
$X = 3.125406$	$\langle a^{X=3.125406,Y=l},$	$d^{X=3.125406,Y=l}\rangle$
$X = 4.099525$	$\langle b^{X=4.099525,Y=k},$	$c^{X=4.099525,Y=k}\rangle$
...		

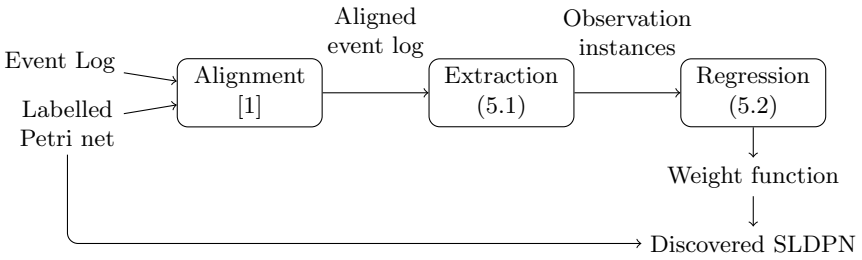


Fig. 2: The proposed method uses an alignment between a Petri net and an event log to extract observation instances for inferring a weight function through regression. This weight function extend the input Petri net to an SLDPN.

5 Data-Based Stochastic Discovery

In this section, we define a method to discover an SLDPN: Data-Based Stochastic Discovery (DSD). DSD takes as input an LPN $N = (P, T, F, \lambda, S_0)$ as well as an event log L, as indicated in Figure 2. Our discovery method learns the weight function $\mathtt{w}$ from the activity and data traces observed in the log and yields an SLDPN $= (P, T, F, \lambda, S_0, \mathtt{w})$.

The weight function needs to be learned based on the data values and transition occurrences observed in the log, i.e., the data sequences σ and their corresponding activity sequences σ for each trace $\sigma \in L$. For a transition t, the learned function $\mathtt{w}(t)$ should return a higher weight for those data states $d \in \Delta$ for which t is more likely to occur compared to other transitions that may be enabled in the same marking.

As shown in Figure 2, we transform this problem to a regression problem. The first step is to build a set of observation instances (a training set) for each transition t, where each instance is an observation in the log of t being enabled in the LPN, with the corresponding data state. The second step is to fit a regression model to each of the sets observations, and to combine the learned regression models to the weight function of the SLDPN. Both steps are detailed in the remainder of this section.

5.1 Extracting Observation Instances

To extract observation instances for the data traces in a log L and the transitions of an LPN N we firstly relate the observed activity sequences L to paths of N. Secondly, we relate the observed data states in the data traces L to sequences of transition firings.

An activity sequence $A \in L$ has no direct correspondence to a path of the LPN: there may be steps required in N that are not present in A, N may contain silent transitions, or there may be activities in A that cannot be mapped to a transition in N. Therefore, we use alignments [1] to establish a mapping between L and N. That is, each activity $a \in A$ is either mapped to a transition $t \in N$ such that $a = \lambda(t)$, or to a log move . The thus-mapped transitions must form

Table 2: Example of an alignment computed for a data trace and our example LPN (Figure 1)

(a) Alignment notation.

model (transitions)	a c
log (activity)	a
log (data)	$X = 10; Y = k$

(b) Matrix notation.

$$= \begin{bmatrix} a & c \\ a & \\ X = 10; Y = k & \end{bmatrix}$$

a path of N, and may need intermediate transitions that are not represented in A (model moves). An alignment is such a mapping , such that the number of log and model moves is minimised. We provide an example in Table 2, but do not further detail the computation of the alignments; please refer to [1] for more details. Please note that we index the matrix notation starting from 1.

Without loss of generality, we may assume that the alignment γ does not contain column vectors in which only the log has an activity, without the model having a corresponding transition ( ). That is, that the alignment contains no log moves. From such an alignment γ, we construct a data sequence that corresponds to the followed path, by taking a previous data state if none is present:

$$D(\gamma, 0) = \sigma_{0}$$

$$D(\gamma, i \quad 1) = \begin{cases} D(\gamma, i \quad 1) & \text{if } \gamma(i,1) = \quad \_\lambda(\gamma(i,1)) = \tau \\ \gamma(i,3) & \text{otherwise} \end{cases}$$

Then, we build observation instances for each transition. For a transition $t \in T$, we collect all observations (d, t') of transition t' firing while t was enabled, with the corresponding data state d. That is, here $d \in \Delta$ is the observed data state before transition t' fired. Note that t' may be the same as t. To collect observations, we define an observation instance builder $O(t)$ that provides a multiset of instances from a collection of alignments Γ.

$$O(\Gamma, t) = \biguplus_{\in \quad \wedge \ (i;1)=t \wedge t' \in E(S_i)} [(D(\gamma, i \quad 1), t')]$$

with

$$S_{i \ 1} = \begin{cases} S_{i \ 1} & \text{if } \gamma(i,0) = \\ S_{i \ 1} \rangle \gamma(i,1) \ n \ \gamma(i,1) & \text{otherwise} \end{cases} \tag{2}$$

This gives us a multiset of data states with positive and negative samples concerning transition t – that is, t was enabled and fired (positive) or t was enabled but another transition fired (negative). The multiset frequencies also inform on the occurrences of transitions.

Example. From our running example (Figure 1 and Table 1), consider the data trace $\sigma_e = (\langle a, d\rangle, \langle \{X = 5.381523\}, \{X = 5.381523, Y = l\}, \{X = 5.381523, Y =$

lgi). The observation points derived from this data trace are ($fX = 5.381523g, a$) and ($fX = 5.381523g, b$) for a; and ($fX = 5.381523, Y = lg, c$) and ($fX = 5.381523, Y = lg, d$) for d.

5.2 Learning Weight Functions

In this section, we use the multisets of observations to discover weight functions for transitions. This involves two steps for each transition: 1) choosing a weight function w, and 2) estimating the parameters of the weight function. In principle, any machine learning approach could be used, including regression and classification, that eventually provides a numeric value. The positive or negative cases with their attached data states can be used to learn the chosen weight function. The choice for a weight function w also sets the types of variables in the data states that can be supported: in principle, any data type up to images, sound and even video can be supported, as long as there is a weight function available that transforms a datum into a numeric weight.

We do not aim to cover a broad range of possible weight functions, however, in order to illustrate SLDPNs, we consider numeric, categorical and boolean variables, as such variables are typically found as attributes in event logs. As weight function, we choose the simple logistic model with parameters β_0 (the intercept) and $\beta_1, \ldots, \beta_n$ (coefficients). Let $x_1, \ldots, x_n$ be the variables of the data state, then

$$\mathsf{w}(t) = \frac{1}{1 + e^{\left({}_0 + {}_1 x_1 + \cdots + {}_n x_n \right)}} \tag{3}$$

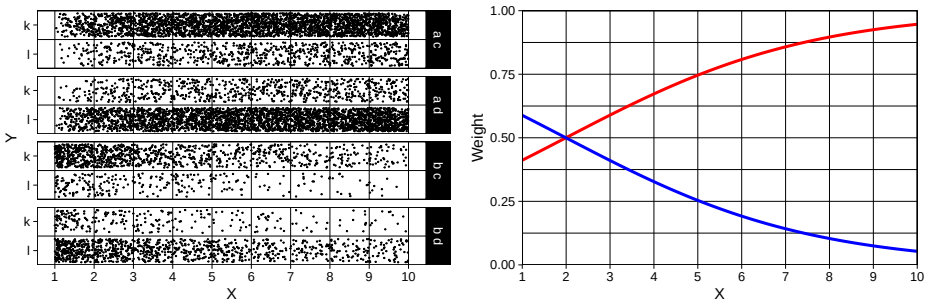
As this weight function only supports numerical variables, categorical and boolean variables are included using one-hot encoding. As such, in the remainder of this paper, we only consider numerical variables. Variables that have not been assigned a value, e.g., because they are only observed later in the process, are handled in the learning procedure through mean imputation; to distinguish these cases, an additional variable is recorded that indicates whether the variable has been assigned in the data state.

The use of the simple logistic model also implies that there is no need to consider all transitions together: global approaches could learn the entire weight function for all transitions together. Instead, a local approach learns the weight function for each transition in isolation, thereby limiting the search space considerably.

To estimate the parameters of the simple logistic weight function – one for each transition –, we leverage the observation instances. For each observation instance (d, t^0) *2* $O(\Gamma, t)$ we obtain a data point in our training set as (d, c) with the to-be predicted independent variable c encoded as:

$$c = \begin{cases} 0 & \text{if } t \text{ 6= } t^0 \\ 1 & \text{if } t = t^0 \end{cases}.$$

Using simple logistic regression, the intercept β_0 and a set of coefficients $\beta_1, \ldots, \beta_n$ are fitted.



(a) Scatter plot of the example log.

(b) Plot of the weight function of transitions *a* and *b* in the discovered SLDPN.

Fig. 3: Distributions of our running example log and SLDPN.

There may be cases in which we cannot collect observation instances for either the positive or the negative case, such as when no other transition is enabled when t is enabled, when t was never observed, or when none of the variables have been assigned (yet). In these case no sensible logistic weight function can be learned from the data states and we resolve to setting $\mathtt{w}(t)$ to the support of transition t, i.e., the relative frequency of occurrences of t when it was enabled.

Example. For our running example of (Figure 1 and Table 1), the regressed parameters for transition a are as follows: The intercept β_0 is 0.716, while the coefficient on X β_1 is 0.359. For b, this is 0.716 and 0.359, respectively. Figure 3b shows that the weight of a and b depend on X, e.g., the weight of b reduces with increasing X. Note that we started the example with a function $1 \quad \frac{1}{X}$ for transition a, and the fitted logistic function $\frac{1}{1+e^{\ (\ 0.716+0.356X)}}$ on it; this is the best-fitting logistic function, however it may be possible to fit other functions as well.

6 Conformance Checking

In this section, we introduce a technique to check the conformance of an SLDPN and an event log. If the SLDPN was discovered from an event log, preferably, a test log that has not been used in the discovery of the SLDPN should be used for conformance checking. To evaluate the agreement between an SLDPN and a log, we need to compare their respective probability distributions: whereas a trace has a certain probability in a log, an SLDPN expresses a trace having a probability *for a particular data sequence*. In this section, we first derive conditional probabilities for SLDPNs, then for logs, and we finish with a conformance measure.

6.1 Conditional Probabilities in SLDPNs

Given an SLDPN $M = (P, T, F, \lambda, S_0, \mathtt{w})$ in a marking S, the probability of an enabled transition t *2* $E(S)$ to fire can be determined from the weights of all

enabled transitions given a data state d following Definition 2, i.e.,

$$p_M(t \mid (S,d)) = \frac{\mathfrak{w}(t,d)}{\sum_{t' \in E(S)} \mathfrak{w}(t',d)}.$$

Given a data sequence $D = \langle d_0, \ldots, d_n \rangle$ and a path $P = \langle t_1, \ldots, t_k \rangle$ of M where $k \quad n$, the probability of that path is

$$p_M(\langle t_1, \ldots, t_k \rangle \mid (\langle S_0, \ldots, S_k \rangle, \langle d_0, \ldots, d_k \rangle)) = \prod_{i=1}^{k} p_M(t_i \mid (S_{i \ 1}, d_{i \ 1}))$$

Given a path and the initial marking S_0, the sequence of markings is deterministic (see Equation (2)). Thus, we may omit the sequence of markings.

However, in conformance checking we need to compare activity sequences rather than paths of transitions. Given an activity sequence A, the conditional probability $p_M(A \mid D)$ of the activity sequence given the data sequence D equals the sum of the probabilities of all paths P such that $P\# \ = A$. However, there may be infinitely many corresponding paths for a given activity sequence A, due to duplicate labels, silent transitions and loops. We use the same technique as in [20] to compute the conditional trace probability $p_M(A \mid D)$, which – for bounded SLDPNs – explicitly constructs a state space of the cross product of A and M under assumption of D, and then computes the probability of reaching a deadlock state using standard Markov techniques. Note that the computation requires the data sequence to be at least as long as the longest path taken into consideration, which is easily guaranteed by replicating the last data state in D a sufficient number of times.

Example. From our running example (Figure 1 and Table 1), consider again the data trace $\sigma_e = (\langle a, d \rangle, \langle fX = 5.381523 \rangle, fX = 5.381523, Y = l\rangle, fX = 5.381523, Y = l\rangle)$. As $\langle a, d \rangle$ is the only path in our SLDPN that corresponds to σ_e, we could directly compute $p_M(\sigma_e \quad \mid \sigma_e \quad)$:

$$\begin{aligned} p_M(\langle a, d \rangle \mid (\langle [p_0], [p_1], [p_2] \rangle, \sigma_e \quad)) &= p_M(a \mid ([p_0], fX = 5.381523)) \\ &\quad p_M(d \mid ([p_1], fX = 5.381523, Y = l)) \\ &= \frac{1 \quad \frac{1}{X}}{\frac{1}{X} + 1 \quad \frac{1}{X}} \\ &\quad \frac{0.2 + (0.6 \text{ if } Y = l)}{0.2 + (0.6 \text{ if } Y = k) + 0.2 + (0.6 \text{ if } Y = l)} \\ &= 0.651 \end{aligned}$$

To compute this probability when multiple paths would be present, we compute the cross product of the SLDPN and σ_e, which is shown in Figure 4. The probability of reaching the end state $[p_3]$ from the initial state $[p_0]$ is $0.814 \quad 0.8 = 0.651$. Thus, the conditional probability $p_M(\sigma_e \quad \mid \sigma_e \quad)$ is 0.651.

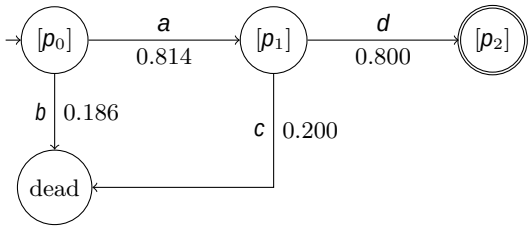


Fig. 4: Cross product of the likelihood of σ_e in our running example.

6.2 Conditional Probabilities in Logs

A log can be seen as a multiset of pairs of an activity sequence A and a data sequence D:

$$L = [(A_0, D_0)^{x_0}, \ldots, (A_n, D_n)^{x_n}].$$

From such a multiset, the probabilities we derive directly are conjunctive. That is, each pair (A, D) is observed a number of times, and the corresponding joint probability concerns both A and D:

$$p_L(A \wedge D) = \frac{L((A, D))}{|L|}$$

The probability of a data sequence is therefore:

$$p_L(D) = \sum_{A \in L_\Sigma} p_L(A \wedge D) = \frac{|[D' \mid (A, D) \in L]|}{|L|}$$

Their ratio is the conditional probability of a trace σ given a data sequence D:

$$p_L(A \mid D) = \frac{p_L(A \wedge D)}{p_L(D)}$$

Notice, however, that if D is unique in L, then $p_L(A \mid D) = 1$ for any A, which makes direct comparisons with an SLDPN challenging.

Example. From our running example (Figure 1 and Table 1), consider again the data trace $\sigma_e = (\langle a, d\rangle, \{X = 5.381523\}, \{X = 5.381523, Y = 'g'\}, \{X = 5.381523, Y = 'gi'\})$. As X is continuous, the data sequence σ is unique in our example log. Then:

$$\begin{aligned} p_L(\sigma_e \wedge \sigma_e) &= 1/10\,000 \\ p_L(\sigma_e) &= 1/10\,000 \\ p_L(\sigma_e \mid \sigma_e) &= 1 \end{aligned}$$

6.3 A Conformance Measure

In this section, we adapt the uEMSC (Equation (1)) stochastic similarity measure to compare an event log L to an SLDPN M.

Since we need to account for the data sequences as well, the uEMSC measure has to be extended to cope with the data-awareness of our approach. By adding the data perspective as an additional dimension to the probability distributions in the uEMSC measure, we directly obtain:

$$duEMSC(L, M) = 1 \quad \sum_{D 2 L_{\Delta}} \sum_{A 2 \widetilde{L_{\Sigma}}} \max(p_L(A \wedge D) \quad p_M(A \wedge D), 0)$$

We can rewrite the joint probabilities using conditional probabilities:

$$p_M(A \wedge D) = p_M(A \mid D) p_M(D)$$

In absence of a data distribution in M, $p_M(D)$ is not defined. However, intuitively, we compare the likelihood of the activity sequences in L ($\widetilde{L}$) with the likelihoods of those activity sequences in M, under the *same* data distribution. Henceforth, we can assume the data distribution of L (L) for M, and thus $p_M(D) = p_L(D)$. Then, the duEMSC measure results to

$$duEMSC(L, M) = 1 \quad \sum_{D 2 L_{\Delta}} \sum_{A 2 \widetilde{L_{\Sigma}}} \max(p_L(A \wedge D) \quad p_M(A \mid D) p_L(D), 0) \quad (4)$$

Notice that if all data sequences in the log are equal, then $duEMSC$ is equal to $uEMSC$, and as such, $duEMSC$ is a proper generalisation of $uEMSC$, and can be used interchangeably.

Example. For our running example (Figure 1 and Table 1), the overall value of duEMSC is 0.997. This value is not precisely 1, which, given the large sample size of the log (10 000), indicates that a logistic formula is not able to capture the distributions in the log perfectly.

7 Evaluation

In this section, we validate our approach threefold: we show its feasibility using an implementation, we compare the discovered models with existing stochastic process discovery techniques, and we illustrate the new types of insights that can be obtained using SLDPNs.

7.1 Implementation

We implemented discovery and conformance checking methods for SLDPNs as plug-ins of the ProM framework[5], in the `StochasticLabelledDataPetriNet`

[5] Available in the nightly builds at `https://promtools.org/`

package. Further functionality for SLDPNs provided by the package are plug-ins to import, export, and visualise and interact with SLDPNs (see Section 7.2). The source code is available at `http://svn.win.tue.nl/repos/prom/Packages/StochasticLabelledDataPetriNet/Trunk`.

The discovery plug-in first uses the alignments provided by ProM [1] to obtain the observation instances, after which the logistic regression implementation provided by Weka 3.8 [32] based on ridge regression [11] is leveraged for inferring the weight function. A parameter adjusts the one-hot encoding for categorical event log attributes: it sets a maximum on the number of categories that are considered for one-hot encoding for a single variable. Another parameter avoids using one-hot encoding altogether and only considers numerical variables. This is useful to avoid attempting to create a model with a very large number of variables which poses the risk of over-fitting and excessive run times.

The conformance checking plug-in implements $duEMSC$, by extending the `EarthMoversStochasticConformance` [20] implementation.

7.2 Insights

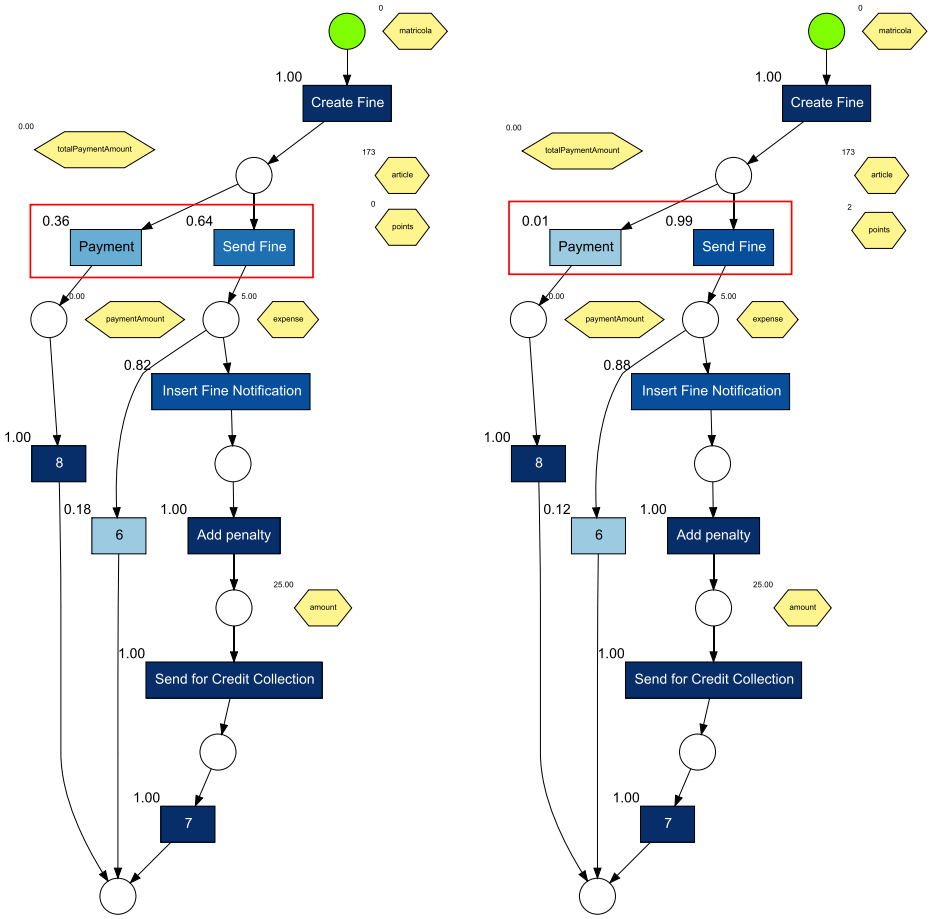
We illustrate the kind of insights provided by the data-dependant stochastic perspective by presenting an example of a discovered SLDPN on a real-life event log indicating a road fines handling process that is known to contain process relevant data attributes [23]. Using the Directly Follows Model Miner (DFM), an SLDPN was discovered using our ProM plug-in using only numeric attributes.

In the interactive visualisation of our ProM Package the discovered SLDPN can explore influence of data variables on the likelihood of transitions. Figure 5a shows the stochastic perspective for the variable `points` being 0, while Figure 5b shows the stochastic perspective for the variable `points` being 2 with all other variables unchanged. This variable indicates the number of penalty points deducted from the driving license. In total a driver has 20 points and a new driving exam needs to be taken if all points are lost.

One can observe the difference in probability in the highlighted choice between `Payment` and `Send Fine`. Here the occurrence of `Send Fine` indicates that the fine was not directly paid [23]. In the SLDPN, we can observe that if a fine corresponds to 2 penalty points deducted from the license, then it is much less likely that the fine is paid on the spot without being sent out (1%) vs. if the fine does not correspond to any points (36%). These types of insights can be obtained with neither common process mining techniques, nor stochastic process mining techniques, nor data-aware process mining techniques.

7.3 Quantitative

In this experiment, we compare the models of our technique with existing stochastic discovery techniques. Figure 6 shows the set-up of this experiment: from several of real-life logs, we first discover control-flow models. Second, on a random 50% trace-based sample, we apply stochastic discovery techniques, including ours. These stochastic process models are then measured with respect to the



(a) Scenario 1: A fine with 0 points. (b) Scenario 2: A fine with 2 points.

Fig. 5: An SLDPN discovered by DFM from the road fines event log visualised interactively in ProM. Variables are shown as yellow hexagon shaped nodes with their assignment next to them. The assignment can be changed to investigate the impact of a data state on the transition weights. Transitions are coloured according to their weights. Note that to give a quick overview the marking is not considered. Nodes have been repositioned for better legibility.

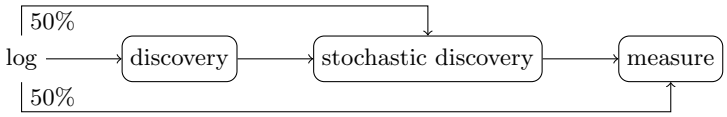


Fig. 6: Set-up of a single quantitative experiment.

Table 3: Details of the quantitative experiment's set-up.

(a) Logs.

Log	traces	events	activities
bpic12-a	6562	30 541	10
bpic13-incidents	3786	32 825	13
bpic13-open problems	412	1 179	5
bpic13-closed problems	748	3 361	7
bpic17-offer log	21 455	96 752	8
bpic20-domestic declarations	5 228	28 108	16
bpic20-international declarations	3 204	35 815	34
bpic20-prepaid travel cost	1 052	9 164	29
bpic20-request for payment	3 447	18 458	18
sepsis	526	7 7615	16
road fines	75 167	280 779	11

(b) Discovery techniques.

Directly Follows Model Miner [19]	DFM
Inductive Miner - infrequent [17] (0.8)	IMf
Flower model: a model that allows for any behaviour of the observed alphabet	FM

(c) Stochastic discovery techniques.

Baseline: uniform choices	BUC
Alignment-based estimator	ABE
Frequency-based estimator	FBE
Data-based stochastic discovery without one-hot encoding (Section 5)	DSDwe
Data-based stochastic discovery (Section 5)	DSD

(d) Measures.

Number of transitions	transitions
Number of transitions with non-1 weights	weights
Number of transitions with data-dependent weights	data weights
unit Eerth Movers' Stochastic Conformance [20]	uEMSC
Data-aware uEMSC (Section 6)	duEMSC

remaining 50% of the log. The entire procedure is repeated 10 times to nullify random effects. Table 3 shows the details of the set-up.

To study the impact of using more variables we not only use our technique (DSD), but also include a variant (DSDwe) that does not use one-hot-encoding. The stochastic discovery was bounded by a timeout of 6 hours, which was never reached. The experiments were conducted on an AMD EPIC 2GHz CPU with 100GB RAM available; the logs were taken from `https://data.4tu.nl/search?q=:keyword:%20%22real%20life%20event%20logs%22`. We archived the code and the full results at Zenodo[6].

Results. Table 4 summarises the full results that are available in the Zenodo archive. The values obtained by uEMSC for BUC, ABE and FBE were equivalent to the values obtained by duEMSC for these stochastic discovery techniques, as shown in Section 6. Therefore, uEMSC is not shown or further discussed.

From these summarised results, it is clear that the data-aware stochastic process discovery techniques can compete with existing stochastic discovery techniques on model quality. In particular, they are – in most cases – able to better represent the behaviour in real-life event logs than existing stochastic discov-

[6] `https://dx.doi.org/10.5281/zenodo.7578655`

Table 4: Summary of our quantitative results for 36 experimental runs.

Stochastic algorithm	Fastest	Highest duEMSC
BUC	36	6
FBE	0	9
ABE	0	12
DSDwe	0	17
DSD	0	19

ery techniques. Out of 36 experiments DSD achieves most often the highest duEMSC with 19 runs. Comparing to DSDwe it seems that considering categorical attributes is useful in two cases but has, overall, a limited impact. This motivates future research on using categorical attributes. A potential pitfall is that by adding more variables, or using other regression functions, the likelihood of over-fitting increases, which would lead to lower scores in this experiment. Unsurprisingly the state-of-the-art on non-data-aware discovery ABE is the second best algorithm with several times achieving the same score. Note that in contrast to typical application scenarios we did not investigate or manually select particular attributes for their relevance. Neither did we select event logs for the suitability to data-aware techniques. Thus, it is expected that DSD cannot always achieve better results.

We discuss the Sepsis log in a bit more detail. For IMf, the model contains quite some concurrency, which involves many potential traces, especially with local loops within concurrent blocks. As alignment-based stochastic discovery techniques are not sensitive to concurrent behaviour – they only consider how *often* transitions are executed, not *when* –, all tested stochastic discovery techniques obtain low duEMSC scores. For the DFM miner, the poor performance may be explained by the repeated blood, leucocytes, lactic acid and CRP measurements are taken regularly throughout the process, which makes control-flow without concurrency challenging. Furthermore, they are performed regularly, that is, they are not dependent on data. For the flower model – in theory – any activity that is executed based on data rather than other activities (control flow), should contribute to the stochastic perspective. Hence, the low duEMSC score for all stochastic models shows that the sepsis log describes a structured process.

Figure 7 shows the distribution of stochastic discovery run times in the experiment. We observe that it takes more time to discover an SLDPN compared to the the non-data-aware approaches BUC, FBE and ABE. BUC does not consider the log at all and simply assigns a weight of 1 to each transition, which takes very little time. FBE traverses the log, and ABE creates an alignment. Thus, DSD is expected to take at least as long as ABE. Still, all the SLDPNs could be discovered within a maximum of 14 seconds, which is highly feasible. Please note that for some logs, such as bpic11 and bpic15, alignments are hard to compute, which keeps these logs out of reach for ABE and DSD.

Figure 8 shows the distribution of the the stochastic conformance checking run times in our experiment. In the worst case the conformance checking took 573 798 milliseconds for the bpic20-international declarations event log and dis-

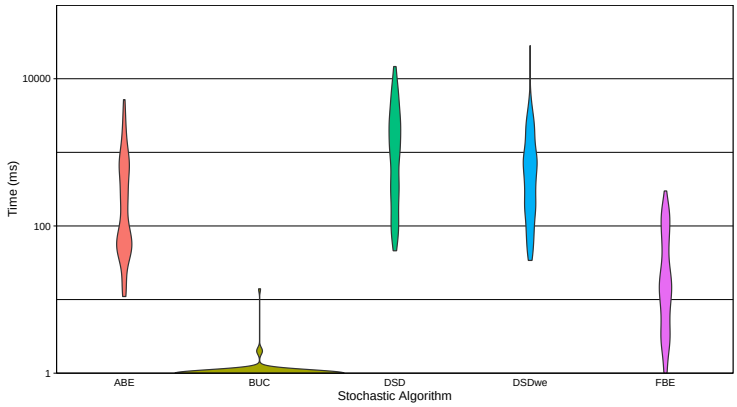


Fig. 7: Run times of the stochastic process discovery.

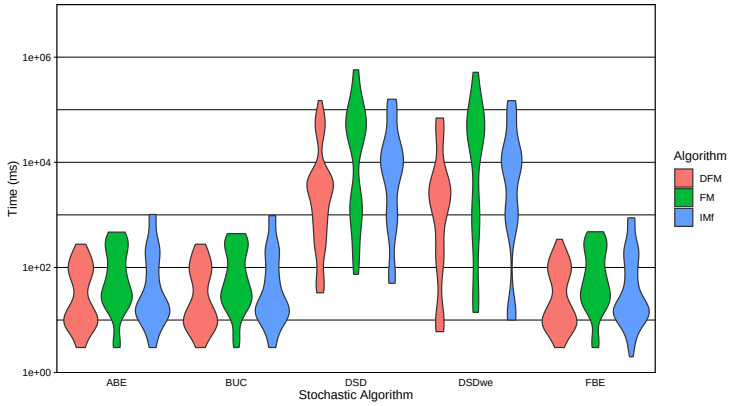


Fig. 8: Run times of stochastic conformance checking for different algorithms.

covered SLDPN, which took into account 24 variables. Overall, conformance checking of the models discovered by FM takes consistently much longer than their respective IMf and DFM counterparts. However, this difference can also be observed in the non-data-aware approaches. With the exception of bpic20-international declarations, the run times stay in most cases within a limit of 1 to 2 minutes. Notably, up to 80GB of RAM was required for these computations.

8 Conclusion

Process models that are typically used in business process management and mining do not incorporate stochasticity: when multiple activities are enabled, no information is incorporated into the model that defines the likelihood of each activity to fire. As a consequence, each activity has the same probability to fire. This is oftentimes not realistic: some activities are more probable than others.

This paper is centered around stochastic process mining, and provides a twofold contribution. On the one hand, it puts forward a technique to discover stochastic models that incorporate a characterization of the probability of each enabled activity to fire. On the other hand, it defines stochastic conformance checking, which do not only aim to verify the compliance of each execution with respect to a model, but also considers whether the distribution of traces in the event log is consistent with the probability distribution of model executions. Conformance checking thus requires to consider the whole event log together, and cannot analyse each event-log trace in isolation.

Some research also exists in stochastic process mining (cf. Sections 1 and 2), and aims to discover and check the conformance of stochastic models that correlate the activity occurrence probability to the activities performed beforehand. This is often limiting, because this probability might be influenced by the current values of the data variables of which process executions change the values.

This paper overcomes this limitation and incorporates the data variables into stochastic process models. In particular, this paper introduces the notion of SLDPN, which is conceptually simple but yet fully equipped to model the process' behavior, in terms of activities and manipulation of data variables, and transition firing probabilities. The paper contributes techniques for discovering and conformance checking of SLDPNs. About discovery, the experiments shows that by including relevant data variables into the computation of the firing probability of SLDPN's transitions can yield a more accurate characterization of transition firing probabilities. In conformance checking, the technique follows the intuition of stochastic conformance that computes metrics at event-log level, rather than considering single traces in isolation.

SLDPNs are very suited to model business simulation models [26]. Business Process Simulation enables to generate an arbitrarily large number of potential process executions. It also allows process analysts to implement various process' modifications with the aim to assess their correlation with process performance. By trying several process modifications without putting them in real production, analysts can determine those that improve the process' performance with little or no consequences. As future work, we plan to exploit the technique to discover transition firing probabilities to mine more accurate and realistic simulation models, compared with the state of the art (cf., e.g., [10]). Indeed, more accurate firing probabilities allow analysts to better model the run-time characterisation of business simulation models.

The discovery of the transition firing probabilities builds on logistic regression as an oracle to find the transition's weights and consequently the transition's firing probabilities. This has shown to be beneficial to better compute weights. Logistic regression also has the advantage to naturally explain how weights are computed in each and every case. However, generally it is not the best regression technique in several settings, especially when the variables are correlated. Here, we intend to evaluate alternative regression techniques, including those based on neural networks, with the goal to improve the weight accuracy.

References

1. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining Knowl. Discov. **2**(2), 182–192 (2012)
2. Acampora, G., Vitiello, A., Stefano, B.N.D., van der Aalst, W.M.P., Günther, C.W., Verbeek, E.: IEEE 1849: The XES standard: The second IEEE standard sponsored by IEEE computational intelligence society [society briefs]. IEEE Comput. Intell. Mag. **12**(2), 4–8 (2017), https://doi.org/10.1109/MCI.2017.2670420
3. Alman, A., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic declarative process mining. Inf. Syst. **109**, 102033 (2022)
4. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
5. Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M.: Extracting decision logic from process models. In: CAiSE 2015, LNCS, vol. 9097, pp. 349–366. Springer (2015). https://doi.org/10.1007/978-3-319-19069-3_22
6. Bazhenova, E., Bülow, S., Weske, M.: Discovering decision models from event logs. In: BIS 2016. LNBIP, vol. 255, pp. 237–251. Springer (2016). https://doi.org/10.1007/978-3-319-39426-8_19
7. Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic trace alignment. In: ICPM. pp. 9–16. IEEE (2021)
8. Burke, A., Leemans, S.J.J., Wynn, M.T.: Stochastic process discovery by weight estimation. In: Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 406, pp. 260–272. Springer (2020), https://doi.org/10.1007/978-3-030-72693-5_20
9. Burke, A., Leemans, S.J.J., Wynn, M.T.: Discovering stochastic process models by reduction and abstraction. In: Petri Nets. Lecture Notes in Computer Science, vol. 12734, pp. 312–336. Springer (2021)
10. Camargo, M., Dumas, M., González-Rojas, O.: Automated discovery of business process simulation models from event logs. Decision Support Systems **134**, 113284 (2020), https://www.sciencedirect.com/science/article/pii/S0167923620300397
11. le Cessie, S., van Houwelingen, J.: Ridge estimators in logistic regression. Applied Statistics **41**(1), 191–201 (1992)
12. Di Francescomarino, C., Ghidini, C.: Predictive process monitoring. In: Process Mining Handbook, pp. 320–346. Springer International Publishing, Cham (2022), https://doi.org/10.1007/978-3-031-08848-3_10
13. Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.: Cocomot: Conformance checking of multi-perspective processes via smt. In: Business Process Management. pp. 217–234. Springer International Publishing, Cham (2021)
14. Gibbs, A.L., Su, F.E.: On choosing and bounding probability metrics. International Statistical Review **70**(3), 419–435 (2002), https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2002.tb00178.x
15. Hensel, C., Junges, S., Katoen, J., Quatmann, T., Volk, M.: The probabilistic model checker storm. Int. J. Softw. Tools Technol. Transf. **24**(4), 589–610 (2022), https://doi.org/10.1007/s10009-021-00633-z
16. Leemans, S.J.J., van der Aalst, W.M.P., Brockhoff, T., Polyvyanyy, A.: Stochastic process mining: Earth movers' stochastic conformance. Inf. Syst. **102**, 101724 (2021), https://doi.org/10.1016/j.is.2021.101724

17. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. Lecture Notes in Business Information Processing, vol. 171, pp. 66–78. Springer (2013), https://doi.org/10.1007/978-3-319-06257-0_6
18. Leemans, S.J.J., Maggi, F.M., Montali, M.: Reasoning on labelled petri nets and their dynamics in a stochastic setting. In: Business Process Management - 20th International Conference, BPM 2022, Münster, Germany, September 11-16, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13420, pp. 324–342. Springer (2022), https://doi.org/10.1007/978-3-031-16103-2_22
19. Leemans, S.J.J., Poppe, E., Wynn, M.T.: Directly follows-based process mining: Exploration & a case study. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019. pp. 25–32. IEEE (2019), https://doi.org/10.1109/ICPM.2019.00015
20. Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 360, pp. 127–143. Springer (2019), https://doi.org/10.1007/978-3-030-26643-1_8
21. Leemans, S.J.J., Tax, N.: Causal reasoning over control-flow decisions in process models. In: Advanced Information Systems Engineering - 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6-10, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13295, pp. 183–200. Springer (2022), https://doi.org/10.1007/978-3-031-07472-1_11
22. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016), https://doi.org/10.1007/s00607-015-0441-1
23. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Decision mining revisited - discovering overlapping rules. In: Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9694, pp. 377–392. Springer (2016), https://doi.org/10.1007/978-3-319-39696-5_23
24. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Data-driven process discovery - revealing conditional infrequent behavior from event logs. In: Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10253, pp. 545–560. Springer (2017), https://doi.org/10.1007/978-3-319-59536-8_34
25. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A.: Predictive monitoring of business processes: A survey. IEEE Transaction on Services Computing **11**(6), 962–977 (2018)
26. Melão, N., Pidd, M.: Use of business process simulation: A survey of practitioners. The Journal of the Operational Research Society **54**(1), 2–10 (2003)
27. Park, G., Song, M.: Prediction-based resource allocation using lstm and minimum cost and maximum flow algorithm. In: International Conference on Process Mining (ICPM). pp. 121–128 (2019)
28. Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: ICPM. pp. 97–104. IEEE (2020)

29. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. Lecture Notes in Business Information Processing, vol. 171, pp. 15–27. Springer (2013), https://doi.org/10.1007/978-3-319-06257-0_2
30. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. **54**, 1–14 (2015), https://doi.org/10.1016/j.is.2015.04.004
31. Rozinat, A., van der Aalst, W.M.P.: Decision mining in ProM. In: BPM 2006, LNCS, vol. 4102, pp. 420–425. Springer (2006). https://doi.org/10.1007/11841760_33
32. Witten, I.H., Frank, E., Hall, M.A.: Data mining: practical machine learning tools and techniques, 3rd Edition. Morgan Kaufmann, Elsevier (2011)