Stochastic-Aware Conformance Checking: An Entropy-Based Approach

Sander J.J. Leemans[1(✉)] and Artem Polyvyanyy[2]

[1] Queensland University of Technology, Brisbane, QLD 4000, Australia
s.leemans@qut.edu.au
[2] The University of Melbourne, Parkville, VIC 3010, Australia
artem.polyvyanyy@unimelb.edu.au

Abstract. Business process management (BPM) aims to support changes and innovations in organizations' processes. Process mining complements BPM with methods, techniques, and tools that provide insights based on observed executions of business processes recorded in event logs of information systems. State-of-the-art discovery and conformance techniques completely ignore or only implicitly consider the information about the likelihood of processes, which is readily available in event logs, even though such stochastic information is necessary for simulation, prediction and recommendation in models. Furthermore, stochastic information can provide business analysts with further actionable insights on frequent and rare conformance issues.

In this paper, we propose precision and recall conformance measures based on the notion of entropy of stochastic automata that are capable of quantifying, and thus differentiating, frequent and rare deviations between an event log and a process model. The feasibility of using the proposed precision and recall measures in industrial settings is demonstrated by an evaluation over several real-world datasets supported by our open-source implementation.

Keywords: Process mining · Information theory · Stochastic conformance checking · Entropy · Precision · Recall · Fitness

1 Introduction

A business process is an orchestration of activities and resources in an organisation, aiming to achieve a business objective. Business process management (BPM) is an interdisciplinary field that studies concepts and methods that support and improve the way business processes are designed, performed, and analyzed in organizations, with the ultimate goal of reducing their costs, execution times, and failure rates through incremental changes and radical innovations [1,2]. Research in BPM has resulted in a range of methods, tools and techniques for identifying, designing, enacting, monitoring and innovating operational business processes [3,4].

S. Dustdar et al. (Eds.): CAiSE 2020, LNCS 12127, pp. 217–233, 2020.
https://doi.org/10.1007/978-3-030-49435-3_14

Process mining aims to discover, monitor and improve real-world processes using the knowledge accumulated in event logs produced by modern information systems [5], where an event log is a collection of traces, each representing executed events of a customer, order, claim, etc. traversing the business process. As multiple traces might share the same sequence of steps through the process, event logs are inherently stochastic: by accumulating information about business process executions observed over extended periods of time, event logs encode the true likelihood of executing the various sequences of steps through the process. This knowledge about the frequencies attached to real-world processes is invaluable for business process redesign and analysis practices [6], as it can inform flexible performance management [7] and generation of novel business models and processes, both incremental [8] and radical [9,10]. For instance, consider the following event logs, each consisting of 2 distinct traces, with 1,000 traces in total:

$$L_1 = [\langle \text{x-ray}, \text{treat} \rangle^{999}, \langle \text{MRI}, \text{treat} \rangle^{1}] \quad \text{vs} \quad L_2 = [\langle \text{x-ray}, \text{treat} \rangle^{1}, \langle \text{MRI}, \text{treat} \rangle^{999}]$$

Even though these event logs consist of the same distinct traces, they are very different. In L_1, the $\langle \text{MRI}, \text{treat} \rangle$ trace is the exception, while in L_2 it is the rule, which likely will influence optimisation strategies.

Some examples of advanced uses of process mining are prediction, recommendation and simulation. In a running trace, using a process model, prediction techniques aim to estimate certain properties of the trace's future steps towards completion, for instance its outcome, its risk of being delayed, its cost, etc. Based on these predictions, recommendation techniques automatically suggest mitigation or optimisation steps for the future of the trace. As different paths through the process model might lead to different properties, prediction and recommendation techniques inherently need to be aware of the stochastic perspective of the process model.

In process optimisation projects, simulation can be used to measure the impact of proposed process changes before they are implemented, and thus before the implementation costs are incurred. That is, several process models with proposed changes are simulated and key performance indicators (for instance, throughput, trace duration characteristics, etc.) are measured, such that a favourable model can be chosen. Key performance indicators such as throughput and trace duration largely depend on the paths taken through the model and, hence, the outcome of the simulations depends on the stochastic perspective of the model.

Even though simulation, prediction and recommendation depend heavily on the stochastic perspective of process models (*stochastic process models*), few techniques have been proposed to construct such models automatically (*stochastic process discovery* techniques) [11]. Typically, the stochastic perspective is constructed by hand as an extension of an existing process model.

However, to truly treat the stochastic perspective of process models as a first-class citizen, it is also necessary to evaluate it. That is, the stochastic perspectives,

as modelled manually or discovered by stochastic discovery techniques, might differ substantially from the stochastic perspective of the event log. Thus, stochastic process models risk not being true representations of the actual real-life business process and predictions, recommendations and simulations might return misleading results [12]. Few techniques have been proposed that can be used to verify or assess the quality of stochastic process models with respect to event logs, that is, to perform *stochastic conformance checking* [13], however with the limitation of not supporting loops.

In classical (non-stochastic) conformance checking, typically four dimensions are considered to compare a log to a (non-stochastic) process model: (1) fitness, which expresses the part of behaviour of the event log that is supported by the model, (2) precision, which expresses the part of the model's behaviour that is also in the event log, (3) generalisation, which expresses the likelihood that future behaviour is captured in the model, and (4) simplicity, which expresses whether the model expresses its behaviour in a clear and concise way [14,15]. However, in these existing measures the stochastic perspective of models is not taken into account, and thus they are not suitable to fully evaluate models for, e.g., prediction, recommendation and simulation.

For instance, in [16], we reported on a project with a major German health insurance company that aimed to analyse and simplify about 4,000 of their stochastic process models captured using the EPC notation annotated with probabilities of taking various decisions. The insurer relied on these stochastic models to estimate the number of employees to hire to enact all the operational processes in a calendar year. Given logs of executed processes at the end of the year, the measures proposed in this paper can be used to assess the correctness of the estimates. In Sect. 4, we further illustrate the applicability of our measures in this scenario.

In this paper, we lift two quality measures used in process mining, namely fitness and precision, to consider the stochastic perspectives of event logs (which are inherently stochastic due to the multiplicities of traces occurring) and stochastic process models. That is, we propose two stochastic conformance checking measures, which compare an event log to a stochastic process model. The measures consider both log and model as stochastic automata, and compare the entropy [17] of these automata with the entropy of a third automaton that represents the conjunctive stochastic behaviour of the log and the model. While the measures support any stochastic process model whose behaviour can be represented in a finite stochastic deterministic automaton (see Sect. 2), we illustrate and implemented the measures for Stochastic Petri nets (see Sect. 2). Concretely, this paper contributes:

- Stochastic-aware recall and precision conformance measures for event logs and process models grounded in the entropy of stochastic languages [17,18];
- Eight properties for stochastic-aware conformance measures that aim at establishing the usefulness of measures that satisfy them;
- A publicly available implementation of the proposed conformance measures; and

- An evaluation that demonstrates the applicability and feasibility of the measures in real-life industrial settings.

The remainder of the paper is structured as follows: The next section introduces notions used to support subsequent discussions. Section 3 presents our stochastic-aware precision and recall measures. After that, the measures are evaluated in Sect. 4, and related work is discussed in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Stochastic Languages, Petri Nets and Automata

This section introduces notions used in the discussions in the subsequent sections.

Let Σ be an alphabet of activities, then Σ^* is the set of all possible sequences of activities (*traces*) over Σ. Let ϵ denote the empty trace. A *language* $\subseteq \Sigma^*$ is a, possibly infinite, set of traces.

Definition 1 (Stochastic language). *A* stochastic language L *is a function* $L\colon \Sigma^* \to [0,1]$*, denoting a probability for each trace, such that* $\sum_{t\in\Sigma^*} L(t) = 1$.

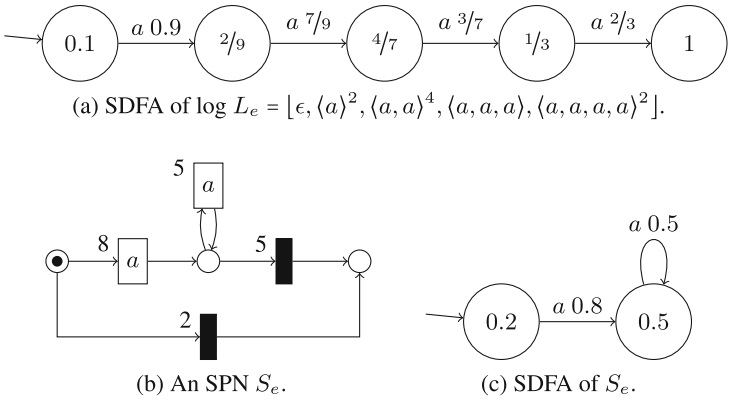
An *event log* is a multiset of traces. For instance, the event log $L_e = [\epsilon, \langle a\rangle^2, \langle a,a\rangle^4, \langle a,a,a\rangle, \langle a,a,a,a\rangle^2]$ consists of 10 traces. Its corresponding stochastic language is $[\epsilon^{0.1}, \langle a\rangle^{0.2}, \langle a,a\rangle^{0.4}, \langle a,a,a\rangle^{0.1}, \langle a,a,a,a\rangle^{0.2}]$ and its corresponding language is $\{\epsilon, \langle a\rangle, \langle a,a\rangle, \langle a,a,a\rangle, \langle a,a,a,a\rangle\}$.

Definition 2 (Stochastic deterministic finite automaton, adapted from [18]). *A* stochastic deterministic finite automaton *(SDFA) is a tuple* $(S, \Sigma, \delta, p, s_0)$*, where* S *is a set of states,* Σ *is an alphabet of activities,* $\delta : S \times \Sigma \to S$ *is a transition function,* $p : S \times \Sigma \to [0,1]$ *is a probability function, and* $s_0 \in S$ *is the initial state.*

The probability to terminate in a particular state s is denoted by $p(s,\lambda)$, $\lambda \notin \Sigma$, and is equal to $1 - \sum_{a\in\Sigma} p(s,a)$. Consequently, for each state, the probabilities of leaving the state or terminating at it should sum to 1, i.e., $\forall s \in S : p(s,\lambda) + \sum_{a\in\Sigma} p(s,a) = 1$.

The stochastic languages that can be represented by SDFAs are called *stochastic deterministic regular languages* [18]. For instance, all event logs can be represented by SDFAs (we included a translation in an accompanying technical report [19]). Figure 1a shows the SDFA of our example event log L_e. Notice that SDFAs do not inherit all the properties of deterministic finite automata. For instance, SDFAs are not closed under union, that is, the union of two stochastic languages represented by SDFAs is not necessarily expressible by an SDFA [20]. Therefore, we did not attempt to find valid reduction strategies for SDFAs, but leave this as future work.

Definition 3 (Petri net). *A* Petri net *(PN) is a tuple* (P, T, A, M_0, l) *in which* P *is a set of places,* T *is a set of transitions* $(T \cap P = \varnothing)$*,* $A \subseteq (P \times T) \cup (T \times P)$ *is an arc relation,* M_0 *(multiset over* P*) is the initial marking and* $l\colon T \to \Sigma$ *is a partial labelling function.*



(a) SDFA of log $L_e = \lfloor \epsilon, \langle a \rangle^2, \langle a, a \rangle^4, \langle a, a, a \rangle, \langle a, a, a, a \rangle^2 \rfloor$.

(b) An SPN S_e.

(c) SDFA of S_e.

Fig. 1. Examples of an event log and a Stochastic Petri net, and their corresponding stochastic deterministic finite automata. For convenience, the numbers in the states denote the probability of termination.

A *marking* is a multiset over P, capturing the state of the net by indicating tokens on the places in P. A transition $t \in T$ is *enabled* in a marking M if for each place p' such that $(p', t) \in A$ it holds that $p' \in M$. If t fires, then all these places p' are removed from M, and to each p'' such that $(t, p'') \in A$ a token is added to the new marking, and if $l(t)$ exists, it indicates this activity $l(t)$ being executed. A *path* in a Petri net is an alternating sequence of markings and transitions such that the markings can be traversed by firing the immediately preceding transitions, and such that in the last marking no transition is enabled. The trace corresponding to a path is the sequence of transitions projected to activities using l, excluding transitions that are not mapped by l. The language of the net is the set of all possible traces for which there exist corresponding paths in the net.

A *stochastic Petri net* (SPN) is a Petri net that expresses a stochastic language. Several ways to enrich a Petri net with stochastic information have been proposed (refer to [21] for an overview). The techniques presented in this paper apply to any type of SPN that can be translated to an SDFA. Nevertheless, for illustrative purposes, we consider a type of SPN in which transitions are annotated with weights:

Definition 4 (Stochastic Petri net). *A* Stochastic Petri net *(SPN) is a tuple* (P, T, A, M_0, l, w) *such that* (P, T, A, M_0, l) *is a Petri net and* $w \colon T \to \mathbb{R}^+$ *is a function that assigns weights to transitions.*

Given a marking M, the probability that an enabled transition t fires in M, denoted by $p(M, t)$, is proportional to t's weight compared to the weight of all enabled transitions: $p(M, t) = w(t) / \sum_{t' \text{ enabled in } M} w(t')$. Then, the probability of a path consisting of transitions $t_1 \dots t_n$ and markings $M_0 \dots M_n$ in an SPN is the product of the transitions' probabilities: $\Pi_{1 \leq i \leq n} p(M_i, t_i)$. The probability of a trace in an SPN is the sum of the probabilities over all paths that induce the

trace, and the stochastic language of an SPN is the collection of all the traces induced by all the paths in the SPN (and all other traces having probability 0). Figure 1b shows an example of an SPN S_e.

If an SPN can be translated to an SDFA, then the SPN must have a finite state space (which still might include loops), and its stochastic perspective needs to be describable by an SDFA. For instance, Fig. 1c shows the SDFA of SPN S_e in Fig. 1b. We characterise the class of SPNs that express stochastic regular languages and discuss some particularities that arise when translating SPNs to SDFAs in an accompanying technical report [19].

3 Stochastic-Aware Conformance Checking

This section presents our new technique for stochastic-aware precision and recall measures, which computes these measures by considering the SDFAs of an event log and a stochastic process model. It first creates a projection of both SDFAs to obtain the behaviour that is common to both. Then, precision and recall are obtained by considering the entropy of the SDFAs and their projections.

Our technique can be applied to any stochastic process modelling formalism, as long as the stochastic language of a model can be expressed as an SDFA. We first introduce the projection, second we describe how we compute entropy, and finally we explain how we compute precision and recall. We then discuss practical considerations of our implementation of the measures, and introduce desirable properties for stochastic conformance checking measures.

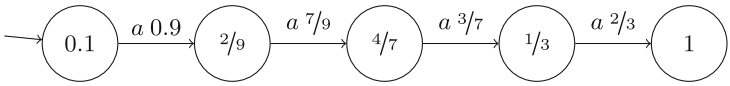
Projection. A *projection* of two SDFAs L and M, denoted by $\mathcal{P}(L, M)$, is an SDFA that contains the behaviour that is present in both L and M. For non-stochastic deterministic finite automata, there are well-known algorithms to establish a projection [22].[1] These algorithms typically construct synchronous walks in both automata, taking a step only when it is allowed in both L and M. We use a similar strategy: whenever both automata are able to take a step, this step is added to the projection. The probability of such a step is the probability of the corresponding step in L.

For instance, consider the two SDFAs shown in Figs. 1a and 1c. Their projections are shown in Figs. 2a and 2b. Notice that if from a particular state an outgoing edge is removed, then the probability of this edge is added to the termination probability at that state.

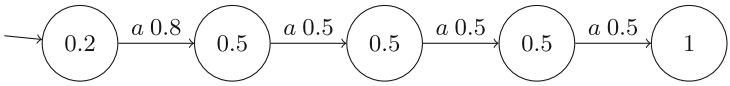
Entropy. Intuitively, the *entropy* of an SDFA describes the number of yes/no questions (bits) that would on average be required to guess an unknown random trace supported by the SDFA. For any stochastic language L, the entropy H can be defined as follows, using a convention that $0 \log 0 = 0$, cf. [17]:

$$H(L) = -\sum_{t \in \Sigma^\star} p(t \in L) \log_2 p(t \in L). \tag{1}$$

[1] For non-stochastic DFAs, a projection is often called a conjunction. We do not use this term here to avoid confusion with the "stochastic" conjunction of two SDFAs, as this "stochastic" conjunction may not necessarily yield an SDFA again.



(a) Projection $\mathcal{P}(L_e, S_e)$ using the probabilities of the log.



(b) Projection $\mathcal{P}(S_e, L_e)$ using the probabilities of the model.

Fig. 2. Projections of the SDFAs shown in Fig. 1.

As Σ^* is infinite, H cannot be computed by iterating over Σ^*. Therefore, we compute the entropy using a procedure adapted from [18]. Given an SDFA $A = (S, \Sigma, \delta, p, s_0)$ that describes a stochastic language, the entropy of the stochastic language of A is:

$$H(A) = -\sum_{\delta(s,a)} c_s p(s,a) \log_2 p(s,a) - \sum_{s \in S} c_s p(s,\lambda) \log_2 p(s,\lambda), \tag{2}$$

where each state $s \in S$ uses a constant c_s, which can be obtained iteratively [18]:

$$c_s^0 = 0 \tag{3}$$

$$c_s^{t+1} = \left(\sum_{\delta(s',a)=s} c_{s'}^t \cdot p(s',a) \right) + \begin{cases} 1 & s = s_0 \\ 0 & s \neq s_0 \end{cases} \tag{4}$$

For instance, for the automaton shown in Fig. 1c, the iterative steps are as follows: $c^0 = [0, 0]$, $c^1 = [1, 0]$, $c^2 = [1, c_0^1 \cdot 0.8 + c_1^1 \cdot 0.5] = [1, 0.8]$, $c^3 = [1, c_0^2 \cdot 0.8 + c_1^2 \cdot 0.5] = [1, 1.2]$, $c^4 = [1, 1.4]$, $c^5 = [1, 1.5]$, $c^6 = [1, 1.55]$, $c^7 = [1, 1.575]$, $\ldots$ $c = [1, 1.6]$ and $H = -(c_0 0.8 \log_2 0.8 + c_1 0.5 \log_2 0.5) \approx 1.05$. This method converges deterministically to the correct value [18].

Computing Precision & Recall. Finally, to compute precision and recall for a log L and a model M (both translated to SDFAs), our technique uses the entropy of the projection $\mathcal{P}$ and compares it to the entropy of L and M:

$$recall(L, M) = \frac{H(\mathcal{P}(L, M))}{H(L)} \qquad precision(L, M) = \frac{H(\mathcal{P}(M, L))}{H(M)} \tag{5}$$

For these measures to work, the entropy of the log and the model cannot be 0. Furthermore, in an accompanying technical report [19] we show that $H(\mathcal{P})$ is always lower than both $H(L)$ and $H(M)$, thus our measures return values between 0 and 1.

For our example log L_e and model S_e (Fig. 1), recall is 1 and precision is 0.914.

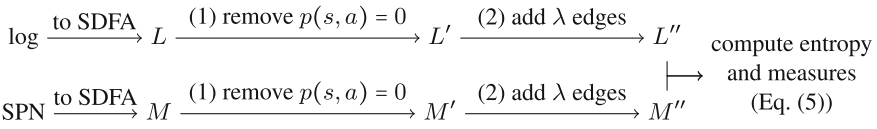


Fig. 3. Overview of the steps taken to increase the applicability of our measures.

Practical Considerations. Next, we discuss some practical considerations that accompany our new measures, and additional steps to increase their applicability, using the overview shown in Fig. 3.

Step (1): Equation (2) requires that every edge in the two input SDFAs has a non-zero probability, as $\log 0$ is undefined (i.e., if $\delta(s,a) = b$ then $p(s,a) > 0$). This is easily ensured using a pre-processing step on the SDFAs, which filters out these edges.

Step (2): Model and log cannot have zero entropies, i.e., they must contain more than one trace with non-zero probability (be deterministic). In our implementation, we pre-process each SDFA before projecting and measuring entropy: from each terminating state s, we add one step out of s with a small probability λ towards a fresh state. This transition has a fresh label, and this label is reused for the pre-processing of both SDFAs. This influences entropy in both SDFAs, but only by $0 \sim 0.15$ entropy.

In [18], it is shown that Eq. (4) converges for SDFAs as long as from each state it is possible to eventually terminate. This corresponds with our definition of stochastic languages (Definition 1), which requires that the sum of probabilities over all traces should be 1. In case an SDFA has a livelock which can be reached with non-zero probability, the probabilities of its traces do not sum to 1 and hence such an SDFA has no stochastic language. This is inherently satisfied by event logs, and ensured with a check in our implementation of the translation of SPNs to SDFAs.

Empty event logs or stochastic process models that do not support any traces do not describe stochastic languages and are hence not supported by our technique. This is a common restriction in process mining: sound workflow nets and process trees have the same limitation and cannot express the empty language either.

Implementation. The proposed measures have been implemented as a plug-in of the ProM framework [23]: "Compute relative entropy of a log and a stochastic Petri net". The measures themselves are deterministic. However, due to the order in which transitions are read from a Petri net and double-precision arithmetic, small differences might occur between runs.

Properties of Stochastic Precision and Recall. A measure that is not known to satisfy any property can be considered to return "magic" numbers. In [12,14,15,24], several properties for classical conformance measures are proposed. Next, we adapt some existing properties to the realm of stochastic-aware

measures, introduce new stochastic-specific properties, and justify that our measures indeed possess these properties.

P1 A stochastic-aware conformance measure should be deterministic;
P2 A stochastic-aware conformance measure should depend on the stochastic languages of logs and models and not on their representations;

Properties P1 and P2 hold for our conformance-aware precision and recall measures, as both the projection and the entropy are computed using deterministic procedures with only stochastic languages as inputs.

P3 Stochastic-aware conformance measures should return values greater than or equal to 0 and less than or equal to 1;

Our precision and recall measures satisfy Property P3: as shown in an accompanying technical report [19]. A conformance value of 1 signifies a perfect conformance, which for the stochastic-aware measures can be instantiated as follows:

P4 If an event log and a model express the same stochastic language, then they should have a perfect stochastic-aware precision, i.e., a precision of 1;
P5 If an event log and a model express the same stochastic language, then they should have a perfect stochastic-aware recall, i.e., a recall of 1;

Properties P4 and P5 hold for our precision and recall measures, because if the log and model express the same stochastic language, then the projection will have the same stochastic language as well. Then, the entropy of all three stochastic languages is obviously equal, hence the numerator and denominator in Eq. (5) are equal.

P6 If a log L_1 assigns to each trace from a model M a higher probability than another log L_2, then the precision of L_1 should be higher than of L_2:
If $\forall_{t \in \Sigma^*} M(t) > 0 \Rightarrow (L_1(t) \geq L_2(t))$ then $precision(L_1, M) \geq precision(L_2, M)$;
Furthermore, if there is a trace of M in L_1 and not in L_2, then the precision of L_1 should be strictly higher than of L_2:
If $\forall_{t \in \Sigma^*} M(t) > 0 \Rightarrow L_1(t) \geq L_2(t)$ and $\exists_{t \in \Sigma^*} M(t) > 0 \wedge L_1(t) > 0 \wedge L_2(t) = 0$, then $precision(L_1, M) > precision(L_2, M)$;
P7 If a model M_1 assigns to each trace from an event log L a higher probability than another model M_2, then the recall of M_1 should be higher than of M_2:
If $\forall_{t \in \Sigma^*} L(t) > 0 \Rightarrow M_1(t) \geq M_2(t)$ then $recall(L, M_1) \geq recall(L, M_2)$;
Furthermore, if there is a trace of L in M_1 and not in M_2, then the recall of M_1 should be strictly higher than of M_2:
If $\forall_{t \in \Sigma^*} L(t) > 0 \Rightarrow M_1(t) \geq M_2(t)$ and $\exists_{t \in \Sigma^*} L(t) > 0 \wedge M_1(t) > 0 \wedge M_2(t) = 0$, then $recall(L, M_1) > recall(L, M_2)$;

The first parts of P6 and P7 hold for our measures: for recall (resp. precision), the projection $P(L, M_1)$ is a super-graph of the projection $P(L, M_2)$, and as for recall (resp. precision) all the probabilities are derived from L (resp M),

the probabilities on the edges common to these SDFAs are equivalent. Then, the properties follow using reasoning similar to P3. The second part of the properties then holds by extension.

Finally, similar to the precision and recall measures in information retrieval, we argue that stochastic-aware precision should be equal to recall with the arguments flipped:

P8 Given two stochastic languages A and B and stochastic-aware precision (*precision*) and recall (*recall*) measures, it should hold that $precision(A, B) = recall(B, A)$.

Property P8 holds for our measures by definition.

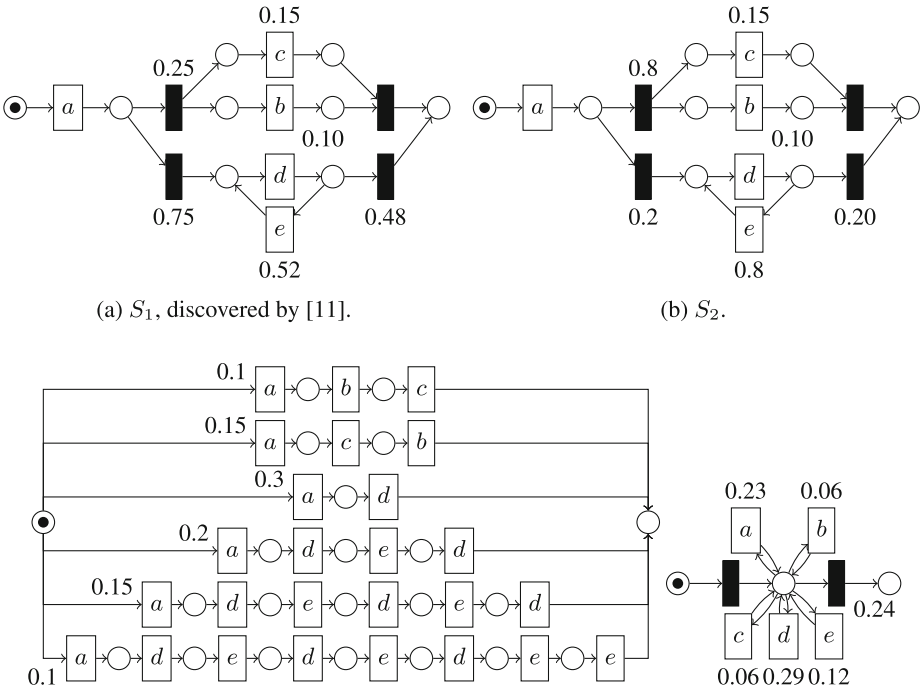
4 Evaluation

In this section, we evaluate the measures introduced in this paper. First, we investigate whether the measures are true reflections of differences in stochastic languages. Second, we show that the measures are feasible to compute on real-life event logs and stochastic models. Third, we illustrate the practical relevance of our measures on a repository of real-life industrial stochastic process models.

Real Reflections of Differences: Ranking of Synthetic Models. Consider an event log L containing 6 distinct traces: $[\langle a, b, c\rangle^{10}, \langle a, c, b\rangle^{15}, \langle a, d\rangle^{30}, \langle a, d, e, d\rangle^{20}, \langle a, d, e, d, e, d\rangle^{15}, \langle a, d, e, d, e, d, e, d\rangle^{10}]$. In this example, we consider four different stochastic process models (SPNs, see Fig. 4) that a user might consider to represent this event log and use to gain insights about the process that generated the event log. Model S_1 was discovered by a stochastic process discovery technique [11] from L. Model S_2 is a manually created SPN that is similar to S_1 but has different probabilities. That is, the stochastic perspective differs. Model S_3 enumerates L's traces having corresponding probabilities: a *trace model*. Model S_4 represents all behaviour and is a *flower model*, with probabilities derived from L based on the frequencies of the activities. Table 1 shows (fragments of) the stochastic languages of these models.

We applied the measures presented in this paper (S), the Earth Movers' (EMSC) [13] measure, as well as the non-stochastic alignment-based (A) [25] and projected (P) [26] precision measures. The results are shown in Table 2 (recall of S is 1 for all models).

All measures, corresponding to intuition, consider the trace model S_3 to be perfectly representing the event log, and agree on the flower model S_4 having the lowest precision. Second, intuitively, the probabilities that S_1 attaches to the traces in L are *closer* to those in L than the probabilities that S_2 attaches to these traces. Thus, we would argue that S_1 represents L better than S_2, which both stochastic conformance checking measures confirm (S, EMSC). A and P do not see any difference between these models. Finally, it is remarkable that EMSC's values for S_2 and S_4 are very close, which may be due to EMSC having to unfold the loop in the flower model, which is bounded and brings the compared



(a) S_1, discovered by [11]. (b) S_2.

(c) A trace model S_3. (d) A flower model S_4.

Fig. 4. Four Stochastic Petri nets that could represent our event log L.

Table 1. Stochastic languages of L and the SPNs in Fig. 4 (p is probability).

Trace	p in L	p in S_1	p in S_2	p in S_3	p in S_4
$\langle a, b, c \rangle$	0.1	0.1	0.32	0.1	0.000199
$\langle a, c, b \rangle$	0.15	0.15	0.48	0.15	0.000199
$\langle a, d \rangle$	0.3	0.36	0.05	0.3	0.016008
$\langle a, d, e, d \rangle$	0.2	0.19	0.03	0.2	0.000557
$\langle a, d, e, d, e, d \rangle$	0.15	0.1	0.03	0.15	0.000019
$\langle a, d, e, d, e, d, e, d \rangle$	0.1	0.05	0.02	0.1	0.000001
Other traces	0	0.05	0.08	0	0.983017

language falsely closer to L. Our measures support loops, which is reflected in the low precision score for S_4.

This illustrates that conformance techniques that are not stochastic-aware cannot fully grasp the differences between the stochastic perspective in these process models.

Table 2. Stochastic measures compared to regular conformance checking techniques.

Rank	This paper (S)	EMSC [13]	Alignment-based (A) [25]	Projected (P) [26]
1	S_3 (1)	S_3 (1)	S_3 (1)	S_3 (1)
2	S_1 (0.918)	S_1 (0.908)	S_1, S_2 (0.846)	S_1, S_2 (0.934)
3	S_2 (0.834)	S_2 (0.570)		
4	S_4 (0.096)	S_4 (0.509)	S_4 (0.292)	S_4 (0.551)

Practical Feasibility. Next, we report on the feasibility of the proposed stochastic-aware precision and recall measures. To this end, we applied a stochastic discovery technique proposed in [11] to 13 publicly available real-life event logs[2] to obtain stochastic block-structured Petri nets.[3] We then measured the precision and recall values for the event logs and the corresponding discovered nets, as well as the times spent to compute them. The code used to run the evaluation is publicly available.[4] The machine used to compute the measures had an E5-1620 CPU with 32 GB RAM.

Table 3. Precision and recall values and times taken to compute them.

Log	Traces	Events	Activities	Recall	Precision	Time (ms)
BPIC11	1,143	150,291	624	0.000	0.000	6,747,782
BPIC12	13,087	262,200	36	0.770	0.080	147,497
BPIC13-closed	1,487	6,660	7	0.677	0.888	22
BPIC13-open	7,554	65,533	13	1.000	0.605	11
BPIC13-incidents	819	2,351	5	1.000	0.630	13,427
BPIC15-1 to 5	Discovery out of memory					
BPIC17	31,509	1,202,267	66	0.111	0.011	2,919,223
Road Fines	150,370	561,470	11	0.772	0.497	245
Sepsis	1,050	15,214	16	0.939	0.109	3,210

Table 3 summarises the results. Some results could not be obtained: for the BPIC15 logs, the discovery technique ran out of memory, thus our measures could not be applied. For BPIC11, discovery returned a model in which all transitions were silent. Hence, this model expressed the stochastic language $[\epsilon^1]$ and thus recall and precision are both 0, as the log did not have the empty trace. One

[2] The event logs are accessible via https://data.4tu.nl/repository/collection:event_logs_real.

[3] The source code of the discovery technique is accessible via https://svn.win.tue.nl/repos/prom/Packages/StochasticPetriNets/Trunk (svn revision 39823).

[4] The source code used in the evaluation is accessible via https://svn.win.tue.nl/repos/prom/Packages/StochasticAwareConformanceChecking/Trunk (revision 41855).

could argue that our measures are strict as both the traces and their probabilities captured in the log and model should match well for high scores. However, one could also argue that the tested discovery technique is, apparently, unable to discover models that represent the likelihood of traces in the event logs well, indicating the need for further research in such techniques.

The reported computation times show that the computation of the stochastic-aware precision and recall measures is feasible on real-life logs, even on complex event logs like BPIC11, taking at most two hours, but much less time for the other tested event logs. Further analysis showed that for SDFAs with large cycles, Eq. (4) might need a quadratic number of steps (in the size of the state space S) to converge, and that this is indeed the most expensive step of our measures. However, run time was not infeasible in our evaluation: at most two hours for the largest logs of most complex processes we tested, but generally much less. Nevertheless, as future work, this step might be optimised using the SDFA's structure.

Practical Usefulness: German Health Insurance Company. In this section, we demonstrate the practical usefulness of our measures in the context of the case study with the German health insurance company [16]. As the company used the hand-crafted stochastic process models for resource planning, it is important that they do not describe the same traces. Otherwise, there is a risk of double resource allocation and, consequently, financial loss to the company. Due to a high number of models, i.e., approximately 4 000 models, manual analysis is intractable.[5]

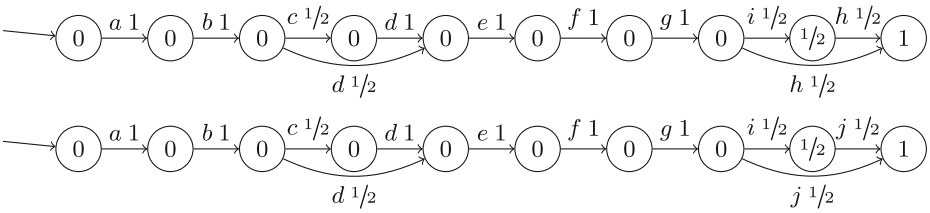


Fig. 5. Two slightly different SDFAs from a German insurer ($h \leftrightarrow j$).

To identify models that describe identical (and frequent) traces, we performed their pairwise comparisons using our stochastic-aware conformance measures. Models that do not describe a proper stochastic language were discarded. Furthermore, only models with a single start node and a single end node were considered. This filtering step resulted in the collection of 3 090 models. The average time of computing a stochastic conformance measure between a pair of models using our tool, either precision or recall, was 69 ms. As a result, we discovered 48 pairs of distinct models that describe, among others, some identical traces. Two anonymised

[5] The models are not publicly available, as per the terms of the agreement with the company.

models from the collection, for which both stochastic recall and precision values are equal to 0.4, are shown in Fig. 5. Business analysts of the insurance company should assess these two models for potential double allocation of resources for support of the corresponding business operations. As these models capture identical frequent traces, the analysts may further consider to combine them into a single model.

5 Discussion and Related Work

A dozen of conformance checking measures have been proposed to date. For a comprehensive overview of the conformance checking field, we refer the reader to [5,14,27]. The vast majority of the existing conformance measures address nondeterministic models and logs. Nondeterminism, as a concept in computer science, was introduced in [28] in the context of nondeterministic finite automata. Nondeterminism, as used in automata theory, states that a choice of the next step in a computation does not necessarily determine its future. This interpretation differs from the one employed in the context of distributed systems, which says that there is no preference among the computations of a system. As such, the latter interpretation provides an abstraction mechanism that allows treating all the computations of a system as being equally good, or equally likely to be induced by the system. Similar to nondeterminism, probabilities can be used to abstract from unimportant or unknown aspects of a system. However, by associating different probabilities with different computations of a system one can encode that certain computations are more likely to be induced by the system than others [20]. In [12], van der Aalst stressed the need to consider probabilities in conformance checking.

Some conformance checking techniques use stochastic elements, however without targeting stochastic models. For instance, Hidden Markov Models (HMMs) have been used to model business processes and to check conformance. In [29], recall and precision are computed by translating Petri nets and event logs to HMMs. However, the stochastic perspective of HMMs is not used, as all the events in a particular state are treated as being equally likely. Another limitation is that parallelism is not supported.

In [14], a precision measure and a recall measure were proposed for process mining founded in the notion of the topological entropy of a regular language. In [14], a framework for conformance checking approaches is proposed, which is instantiated using cardinalities and entropy. The measures proposed in this paper can be seen as extensions of the entropy-based technique for stochastic languages.

Alignments [30] search for a least-cost path through event log and model, thereby being robust to slight deviations between traces. As recall takes the frequency of traces into account, the stochastic perspective of logs is taken into account. However, alignment-based precision measures [25] do not consider the stochastic perspective of the model. Alignment-based precision measures might be extended to support stochastic process models, for instance by searching for a

most-likely path. Projected conformance checking [26] addresses long run times of conformance checking techniques by projecting behavior onto subsets of activities of a certain size. The measures presented in this paper can be extended in a similar fashion. Generalised conformance checking [31] compares an event log and model based on a given trust level for each, derived, for instance, from identified data quality issues [32]. In stochastic conformance checking, one could consider the probability attached to each trace in log and model to be an indication of trust, yielding an alternative, possibly more fine-grained, view on their differences.

To the best of our knowledge, the Earth Movers' Stochastic Conformance checking technique [13] is the only stochastic conformance checking technique proposed today. In this technique, the log and model's stochastic languages are seen as distributions of traces, and the Wasserstein metric is applied. While intuitive, it does not support infinite languages (that is, models with loops), while our measure supports such languages. Furthermore, our work contributes to the ongoing discussion on ideal conformance checking measures by proposing properties that this measure should have [12,14,15,24], by extending these to the stochastic context.

Finally, to compare SDFAs, the Kullback-Leibler (KL) divergence [18] could be used. However, KL-divergence does not exist if one SDFA supports a trace that the other SDFA does not support, making it unsuitable for conformance checking purposes.

6 Conclusion

In process mining, the stochastic perspective of event logs and process models is essential to inform process optimisation efforts and techniques, such as simulation, prediction, recommendation, and to inform staffing decisions: without a stochastic perspective, efforts spent on optimisation are at risk of being spent on rare, exceptional behavior, and lead to misinformed decisions.

In this paper, we contributed to making the stochastic perspective a first-class citizen of process mining techniques, by introducing a stochastic-aware conformance checking technique for two measures: fitness and precision. The proposed precision and recall measures are applicable to an arbitrary event log and a model that describes a finite or infinite number of traces using a finite number of reachable states. Eight desirable properties of stochastic conformance checking measures were identified, and the adherence of our measures to these properties was shown.

An evaluation based on our publicly available implementation confirmed the feasibility of using the measures in real-life industrial settings. We acknowledge that our measures have limitations, which give rise to future work: Various notions of correctness for process models, like boundedness or soundness, classify a process model that can induce an infinite number of states as incorrect. However, as such models can appear in practice due to modelling errors, it is relevant to extend the proposes measures to account for infinite-state models.

Our measures address (to some extent) the problem of partial trace matches [15]: common prefixes of traces are considered and contribute to the measures, however common postfixes are not. Thus, a model and a log that have their first activity different will be considered to be completely disjoint. This limitation can be addressed by considering both the original SDFA and its edge-reversed version during construction of the projection. Finally, our measures consider the stochastic perspective of either log or model, but not both. In future work, this could be addressed.

Acknowledgment. Artem Polyvyanyy was partly supported by the Australian Research Council Discovery Project DP180102839.

References

1. Weske, M.: Business Process Management. Concepts, Languages, Architectures. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28616-2
2. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-56509-4_9
3. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**, 1–37 (2013)
4. Rosemann, M., vom Brocke, J.: The six core elements of business process management. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1. IHIS, pp. 105–122. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-642-45100-3_5
5. van der Aalst, W.M.P.: Process Mining-Data Science in Action, 2nd edn. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
6. Feldman, M.S., Pentland, B.T.: Reconceptualizing organizational routines as a source of flexibility and change. Adm. Sci. Q. **48**(1), 94–118 (2003)
7. LeBaron, C., Christianson, M.K., Garrett, L., Ilan, R.: Coordinating flexible performance during everyday work: an ethnomethodological study of handoff routines. Organ. Sci. **27**(3), 514–534 (2016)
8. Yi, S., Knudsen, T., Becker, M.C.: Inertia in routines: a hidden source of organizational variation. Organ. Sci. **27**(3), 782–800 (2016)
9. Deken, F., Carlile, P.R., Berends, H., Lauche, K.: Generating novelty through interdependent routines: a process model of routine work. Organ. Sci. **27**(3), 659–677 (2016)
10. Sonenshein, S.: Routines and creativity: from dualism to duality. Organ. Sci. **27**(3), 739–758 (2016)
11. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 15–27. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_2
12. van der Aalst, W.M.P.: Relating process models and event logs-21 conformance propositions. In: ATAED, CEUR, vol. 2115, pp. 56–74 (2018)
13. Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNBIP, vol. 360, pp. 127–143. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26643-1_8

14. Polyvyanyy, A., Solti, A., Weidlich, M., Ciccio, C.D., Mendling, J.: Monotone precision and recall measures for comparing executions and specifications of dynamic systems. ACM TOSEM (2020, in press)
15. Polyvyanyy, A., Kalenkova, A.A.: Monotone conformance checking for partially matching designed and observed processes. In: ICPM, pp. 81–88 (2019)
16. Polyvyanyy, A., Smirnov, S., Weske, M.: Reducing complexity of large EPCs. In: MobIS, GfI, vol. 141, pp. 195–207 (2008)
17. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley, Hoboken (2006)
18. Carrasco, R.C.: Accurate computation of the relative entropy between stochastic regular grammars. ITA **31**(5), 437–444 (1997)
19. Leemans, S.J., Polyvyanyy, A.: Proofs with stochastic-aware conformance checking: an entropy-based approach (2019). https://eprints.qut.edu.au/129860/
20. Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., Carrasco, R.C.: Probabilistic finite-state machines-part I. IEEE Trans. Pattern Anal. Mach. Intell. **27**(7), 1013–1025 (2005)
21. Marsan, M.A., et al.: The effect of execution policies on the semantics and analysis of stochastic petri nets. IEEE Trans. Software Eng. **15**(7), 832–846 (1989)
22. Linz, P.: An Introduction to Formal Languages and Automata, 4th edn. JBP, Burlington (2006)
23. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: a new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_25
24. Tax, N., Lu, X., Sidorova, N., Fahland, D., van der Aalst, W.M.P.: The imprecisions of precision measures in process mining. Inf. Process. Lett. **135**, 1–8 (2018)
25. Muñoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15618-2_16
26. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Softw. Syst. Model. **17**(2), 599–631 (2016). https://doi.org/10.1007/s10270-016-0545-x
27. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking-Relating Processes and Models. Springer, Cham (2018)
28. Rabin, M.O., Scott, D.S.: Finite automata and their decision problems. IBM J. Res. Dev. **3**(2), 114–125 (1959)
29. Rozinat, A.: Process mining: conformance and extension. Ph.D. thesis, Eindhoven University of Technology (2010)
30. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. DMKD **2**(2), 182–192 (2012)
31. Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., Gal, A.: In log and model we trust? A generalized conformance checking framework. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 179–196. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_11
32. Suriadi, S., Andrews, R., ter Hofstede, A., Wynn, M.: Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. IS **64**, 132–150 (2017)