Stochastic-Aware Precision and Recall Measures for Conformance Checking in Process Mining

Sander J.J. Leemans[a,*], Artem Polyvyanyy[b]

[a] *RWTH, Aachen, Germany*
[b] *The University of Melbourne, Australia*

Abstract

Process mining studies ways to improve real-world processes using historical event data generated by IT systems that support business processes of organisations. Given an event log of an IT system, process discovery algorithms construct a process model representing the processes recorded in the log, while conformance checking techniques quantify how well the discovered model achieves this objective. State-of-the-art discovery and conformance techniques either completely ignore or consider but hide from the users information about the likelihood of process behaviour. That is, the vast majority of the existing process discovery algorithms construct non-stochastic aware process models. Consequently, few conformance checking techniques can assess how well such discovered models describe the relative likelihoods of traces recorded in the log or how well they represent the likelihood of future traces generated by the same system. Note that this is necessary to support process simulation, prediction and recommendation. Furthermore, stochastic information can provide business analysts with further actionable insights on frequent and rare conformance issues. This article presents precision and recall measures based on the notion of entropy of stochastic automata, which are capable of quantifying and, hence, differentiating, between frequent and rare deviations of an event log and a process model that is enriched with the information on the relative likelihoods of traces it describes. An evaluation over several real-world datasets that uses our open-source implementation of the measures demonstrates the feasibility of using our precision and recall measures in industrial settings. Finally, we propose a range of intuitive desired properties that stochastic precision and recall measures should possess, and study our and other existing stochastic-aware conformance measures with respect to these properties.

Keywords: Process mining, stochastic-aware conformance checking, precision, recall, fitness, information theory, entropy

*Corresponding author
Email addresses: s.leemans@bpm.rwth-aachen.de (Sander J.J. Leemans), artem.polyvyanyy@unimelb.edu.au (Artem Polyvyanyy)

For convenience of reviewers, added text has been annotated.

1. Introduction

A *business process* is a plan and coordination of activities and resources of an organisation that aims to achieve a business objective. Business process management (BPM) is an interdisciplinary field. It studies concepts and methods that support and improve the way business processes are designed, performed, and analyzed in organisations with the ultimate goal of reducing their costs, execution times, and failure rates through incremental changes and radical innovations [45, 10]. Research in BPM has resulted in a range of methods, tools, and techniques for identifying, designing, enacting, monitoring, and innovating operational business processes [39, 34].

Process mining studies ways to discover, monitor, and improve real-world processes using the knowledge accumulated in event logs produced by information systems of organisations [40]. An *event log*, or *log*, is a collection of recorded traces, where each *trace* is a sequence of timestamped events observed during the execution of some business process case. As different cases of a business process can follow the same sequence of steps, the corresponding event log may contain multiple traces that follow the same sequence of events.

Event logs are inherently stochastic. By accumulating information about business process executions over extended periods, the relative frequency of observing a trace in a log aims to encode the true likelihood of executing the corresponding sequence of steps through the process. This knowledge about the frequencies of real-world processes is invaluable for business analysis and redesign initiatives [11], as it can inform flexible performance management [13] and generation of novel business models and processes, both incremental [46] and radical [9, 36].

For instance, consider the following two event logs, each containing of two 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 have the same distinct traces, they stem from two fundamentally different real-world processes. In event log L_1, the $\langle \text{MRI}, \text{treat} \rangle$ trace is the exception, while in event log L_2, it is the rule. Similarly, the $\langle \text{x-ray}, \text{treat} \rangle$ trace is the rule in log L_1 and the exception in log L_2. Non-stochastic aware techniques would consider these logs as being equal, and are thus potentially oblivous for their large differences. Considering the stochastic perspective makes it possible to consider these differences.

Some examples of advanced uses of process mining are prediction, recommendation, and simulation. Prediction techniques study ways to estimate properties of the future steps of a running trace; for example, the risk of the trace being delayed or the overall cost of the trace. Based on such predictions, recommendation techniques automatically suggest mitigation or optimisation of

the future steps of the trace to meet the envisioned objectives. Prediction and recommendation techniques can benefit from stochastic-aware process models, i.e., process models supplied with the information about the relative likelihoods of decisions that impact the routing of the execution of the model. Indeed, given the current state of the trace in the process model, one can use the stochastic perspective of the model to estimate the likelihoods of the various evolution scenarios of the trace and quantify the consequences. Simulation can be used to measure the impact of process changes before they get implemented, for example, in process redesign projects. It is common in such projects to construct several candidate models with proposed changes and then simulate them to measure key performance indicators, for instance, throughput or cycle time, to identify the candidate model with the desired performance characteristics. The outcomes of the simulations depend on the stochastic perspective of the model and, hence, do the measured key performance indicators based on which the decision are made.

Even though simulation, prediction, and recommendation can benefit from stochastic-aware process models, or *stochastic process models*, few techniques have been proposed to construct such models automatically from event data [33]. We refer to such techniques as *stochastic process discovery* techniques. Typically, the stochastic perspective is constructed by hand as an extension of an existing process model [29].

To treat the stochastic perspective of process models as a first-class citizen, it should be possible to evaluate it. The stochastic perspectives of models, as constructed manually or by stochastic process discovery techniques, may differ substantially from the stochastic perspective of the event log they are constructed from. Hence, stochastic process models risk not being faithful representations of the actual real-life business processes. Consequently, predictions, recommendations, and simulations that rely on such low-quality models may return misleading results [41].

In classical (non-stochastic) conformance checking, typically four dimensions are considered to compare a log to a (non-stochastic) process model [4]: (1) *recall*, also known as fitness, quantifies the part of the behaviour of the event log that is supported by the model; (2) *precision* quantifies the part of the model's behaviour that is also in the event log; (3) *generalisation* measures the likelihood that the future behaviour of the system that induced the log is captured in the model, and (4) *simplicity* measures whether the model is clear and concise. However, the vast majority of existing conformance measures do not take the stochastic perspective of process models into account. 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*. In [18], the authors proposed a technique for stochastic conformance checking grounded in the notion of the *earth movers' distance*. This technique, however, has limitations when it comes to assessing the quality of stochastic process models with loops. The *entropic relevance* measure employs a minimum description length compression-based framework to quantify the quality of stochastic process models with respect to the log they were con-

structed from [28, 2]. The relation of these stochastic conformance measures to the four quality criteria in process mining is a subject of ongoing research. For instance, entropic relevance represents a blend between the traditional precision and recall quality criteria in conformance checking; it penalizes log traces the model does not describe and traces permitted by the model but not recorded in the log [2].

In this article, we lift two classical quality criteria, namely recall and precision, to consider the stochastic perspectives of event logs and process models. The measures treat both log and model as stochastic automata and compare the entropy [8] 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, we illustrate and implemented the measures for Stochastic Petri nets (refer to Section 2).

In summary, this paper investigates how stochastic models and event logs can be compared, by making the following contributions: two new entropy-based sets of stochastic conformance checking measures (in particular supporting loops); a detailed discussion on desirable properties for stochastic conformance checking measures; an implementation of our new measures; and a three-fold evaluation of differences between measures, feasibility and practical usefulness.

In [29], we reported on a project with a major German health insurance company that aimed to analyze and simplify about 4,000 of their hand-crafted 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 the upcoming calendar year. Given logs of executed processes at the end of the year, the measures proposed in this article can be used to assess the correctness of the initial estimates. In Section 5, we further illustrate the applicability of our measures in this scenario.

This article extends [16] with (1) new stochastic-aware recall and precision measures that are also grounded in the entropy of stochastic languages of log and model but, differently, quantify the potential gain due to the use of both log and model when describing the traces they share; (2) new and revised properties for stochastic conformance checking measures, (3) an analysis of the existing stochastic-aware conformance measures with respect to the properties they satisfy, (4) a publicly available implementation of the new conformance measures, and (5) an extended evaluation of the applicability and feasibility of using the measures in industrial settings.

The remainder of the paper is structured as follows: The next section introduces formal notions that are used to support the subsequent discussions. Section 3 presents our stochastic-aware precision and recall measures. Section 4 introduces properties for stochastic-aware conformance measures and evaluates the existing and presented in this article measures against these properties. Subsequently, our measures are evaluated in Section 5, and related work is discussed in Section 6. Finally, Section 7 concludes the paper.

2. Stochastic Languages, Petri nets , Automata & Conformance

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

Stochastic Languages. 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* $L{:}\,\Sigma^* \to [0,1]$*, 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 $\{\epsilon, \langle a \rangle, \langle a,a \rangle, \langle a,a,a \rangle, \langle a,a,a,a \rangle\}$.

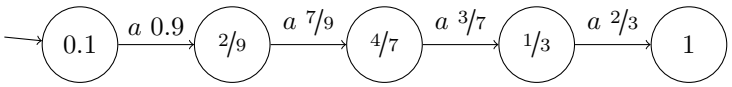
Stochastic Deterministic Finite Automata.

Definition 2 (Stochastic deterministic finite automaton, adapted from [7])**.** *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.*

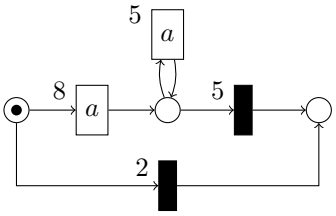
Intuitively, an SDFA starts in state s_0. In a state s, $p(s,a)$ denotes the probability that an activity a is executed. If a is executed in s, then $\delta(s,a)$ denotes the resulting state. We denote termination with λ, such that $\lambda \notin \Sigma$. The probability to terminate in a particular state s is denoted by $p(s,\lambda)$, which is defined as $1 - \sum_{a \in \Sigma} p(s,a)$. Consequently, for each state, the probabilities of leaving the state or terminating at it should sum to 1, i.e. $\forall s \in S : p(s,\lambda) + \sum_{a \in \Sigma} p(s,a) = 1$.

A *trace* in an SDFA is a sequence of transitions $\langle a_1 \dots a_n \rangle$, of which the probability can be found by combining the trace with a sequence of states $s_0 \dots s_n$ such that $\forall_{1 \le i \le n} \delta(s_{i-1}, a_i) = s_i$; the probability is then $\prod_{1 \le i \le n} p(s_{i-1}, a_i) * p(s_n, \lambda)$. If this probability is 0, the SDFA does not *support* the trace. Finally, the *stochastic language* of an SDFA is the set of all supported traces by the SDFA, with their probabilities.

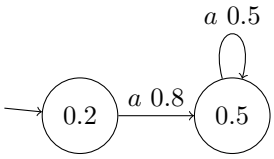
The stochastic languages that can be represented by SDFAs are called *stochastic deterministic regular languages* [7]. For instance, all event logs can be represented by SDFAs (we included a translation in Appendix Appendix A.1). Figure 1a shows the SDFA of our example event log L_e. Notice that SDFAs 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 [44]. Therefore, we did not attempt to find valid reduction strategies for SDFAs, but leave this as future work.



(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.

Figure 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.

(Stochastic) Petri nets.

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* $l{:}T \to \Sigma$ *is a partial labelling function.*

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 [23] 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* $w{:}T \to \mathbb{R}^+$ *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 probablities 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.

We discuss translating SPNs to SDFAs in Appendix A.2. For instance, Figure 1c shows the SDFA of SPN S_e in Figure 1b.

A necessary condition for an SPN to be translatable to an SDFA is that the SPN must have a finite state space, or in other words, the underlying Petri net must be bounded. Notice that this does not exclude loops.

Furthermore, the stochastic perspective of the SPN must be deterministic. A sufficient (but not necessary) condition for this is that for any reachable marking of the SPN, if there exist two non-equal fireable chains of unlabelled transitions such that at the end of each chain a transition with the same label is enabled, then the marking that results from firing these two chains of transitions is equal. That is, if it is possible that the same label is reachable using more than one path of transitions, then the marking resulting from these paths must be equal.

Stochastic conformance checking techniques. Next, we discuss two existing stochastic conformance checking techniques.

The Earth Movers' Stochastic Conformance [19] (EMSC) is a single measure describing the difference between two stochastic languages. Intuitively, one can consider a stochastic language as a distribution of earth: a pile of earth of a particular shape, with a total amount of earth of 1. Then, EMSC is the effort to transform one pile into the other, that is, the distance that earth needs to be moved times the amount of earth that needs to be moved. While conceptually well-defined for any stochastic language (EMSC supports both log-log, log-model and model-model comparisons), implementations only consider finite stochastic languages. Thus, the stochastic language for models with loops or other infinite behaviour needs to be unfolded and thus be truncated. As distance between two traces, current implementations use the normalised Levenshtein distance. [19]

The *entropic relevance* measure for stochastic conformance checking is grounded in a minimum description length compression-based framework [28, 2]. Given a collection of candidate stochastic process models, it can be used to select a model from the collection that represents the given event log, including its stochastic perspective, better. Entropic relevance is computed as the average number of bits required to compress a trace from the log using the structure and information about the relative likelihoods of traces captured in the model. The more traces from the log the model describes and the closer the model reflects the relative likelihoods of traces induced by the log, the better the model can compress the traces. An entropic relevance value is a non-negative number with

meaningful units (*bits* per trace) taken from the open-ended numeric range, with small values being preferable over large values, as smaller values reflect the ability of the model to compress traces better. Given that the probability of a trace in the stochastic language of the model can be computed in the time that is linear in the size of the trace, the computation time of entropic relevance is linear in the size of the event log (number of traces in the log times average length of a trace in the log) [28].

3. Stochastic-Aware Conformance Checking

This section presents our stochastic-aware conformance checking measures. In classical, that is, non-stochastic, conformance checking, precision and recall are often measured as a share of the shared information about traces in the compared log and model in the information about traces described in either the model or log [30]. In stochastic conformance checking, this intuition can be implemented in various ways, and we propose two variants of stochastic-aware precision and recall measures that implement this intuition. The first variant uses SDFA projections to prioritise the stochastic information of either the model or log, while the second variant of the measures – called *gain* – uses the stochastic information from both the model and log equally when computing the measurements. That is, the first variant of the measures relies on a projection of SDFAs that encode process model and event log 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. The second variant of the measures quantifies the potential gain from describing the common behaviour using both model and event log compared to a description that relies either on the model or the log.

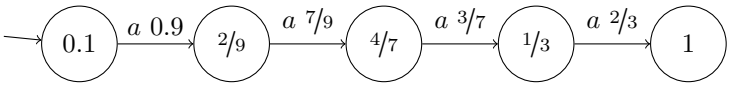
Our measures can be applied to models captured using any process modeling formalism as long as they describe stochastic languages that can be represented by SDFAs. Next, in this section, we introduce the projection operation of SDFAs. Then, we describe a procedure we use to compute the entropy of a stochastic language. Subsequently, we present our stochastic measures of precision and recall between designed and observed processes. We then discuss the practical considerations of our implementation of the measures.

3.1. Projection

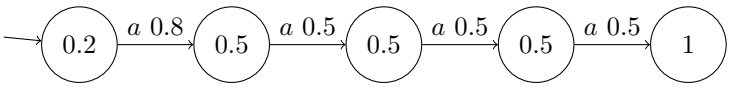
A *projection* of two SDFAs L and M, denoted by $\mathcal{P}(L, M)$, is an SDFA containing the behaviour present in both L and M. For non-stochastic deterministic finite automata, there are well-known algorithms to establish a projection [20].[1] These algorithms typically construct synchronous walks in both automata, taking a step only when it is allowed in both L and M. We use a similar strategy:

[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.

whenever both automata can take a step, this step is added to the projection. The probability of such a step is taken as the probability of the corresponding step in automaton L.



(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.

Figure 2: Projections of the SDFAs shown in Figure 1.

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

3.2. Entropy

The entropy H of a stochastic language L is defined as follows:

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

By convention, we accept that $0 \log 0 = 0$. The entropy of L is a measure of the average uncertainty in the stochastic language, which can be seen as a random variable whose value is a trace randomly selected from the language according to the probabilities assigned to the traces. It is the number of bits that is on average required to describe a trace from the language [8]. Hence, a language with only a few very likely traces has low entropy, while a high entropy characterises a language with many distinct traces that have low chances of occurring.

As Σ^* is infinite, H cannot be computed by iterating over Σ^*. Therefore, we compute the entropy using a procedure adapted from [7]. 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 [7]:

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

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

For instance, for the automaton shown in Figure 1c, the iterative steps are as follows: $c^0 = [0, 0]$, $c^1 = [1, 0]$, $c^2 = [1, c_0^1 \cdot 0.8 + c_1^1 \cdot 0.5] = [1, 0.8]$, $c^3 = [1, c_0^2 \cdot 0.8 + c_1^2 \cdot 0.5] = [1, 1.2]$, $c^4 = [1, 1.4]$, $c^5 = [1, 1.5]$, $c^6 = [1, 1.55]$, $c^7 = [1, 1.575]$, ...$c = [1, 1.6]$ 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 [7].

3.3. Projection-Based Precision & Recall

To compute precision and recall for a log L and a model M (both translated to SDFAs), our first approach 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. For our example log L_e and model S_e (Figure 1), recall is 1 and precision is 0.914.

3.4. Potential Gain Precision & Recall

Let X be a stochastic language. By $\hat{X}$, we denote the set of all *possible traces* of X, i.e., $\hat{X} = \{t \in \Sigma^* \mid X(t) > 0\}$. Let X and Y be two stochastic languages. By $gain(X, Y)$, we denote:

$$gain(X, Y) = \frac{\sum_{t \in \hat{X} \cap \hat{Y}} \min\{-X(t) \log_2 X(t), -Y(t) \log_2 Y(t)\}}{H(Y)} \tag{6}$$

The entropy of a random variable, in our case of a stochastic language, is a lower bound on the average number of bits required to represent the random variable [8]. The numerator of $gain(X, Y)$ is the sum of the contributions of the traces that are possible according to both X and Y to the minimum description lengths of the corresponding languages, where for each trace only the smallest out of the contributions for the two languages is considered in the summation. Thus, intuitively, $gain(X, Y)$ is a measure of how compact the average description of a trace that is possible according to both X and Y can be when both languages are used for the encoding compared to the average description of a trace in Y using Y, that is, an estimate of the *gain* in compression of a trace common to X and Y when both languages are available instead of Y only.

Let M and L be a model and log, respectively. Then, we define the gain-based precision and recall between M and L as follows:

$$gain\ recall(L, M) = gain(M, L) \qquad gain\ precision(L, M) = gain(L, M) \tag{7}$$

As $\hat{L}$ is finite, $\hat{L} \cap \hat{M}$ is also finite, and, thus, the numerator of $gain(L, M)$ (or the numerator of $gain(M, L)$) can be computed in the finite loop that iterates

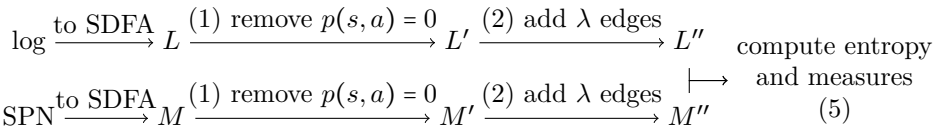


Figure 3: Overview of the steps taken to increase the applicability of our measures.

over all the possible traces of L that are also possible traces of M. Besides, as $H(L)$ and $H(M)$ can always be computed, refer to Section 3.2, the gain-based precision and recall can always be computed using a deterministic procedure.

Similar to the work on *entropic relevance* [28, 2], one can incorporate the selector coding costs in the numerator of $gain(X, Y)$ to account for the sequence of decisions of selecting a stochastic language, either X or Y, that results in a more compact encoding of the trace. The inclusion of the selector coding costs would result in the use of a lossless encoding of the traces in the numerator of $gain(X, Y)$. As such selector coding costs encode binary decisions, they are small, at most one bit per trace. Consequently, as the ability to decode the traces from the codes used to compute $gain(X, Y)$ is not essential for the definition of the conformance measures but complicates their computation, we postpone the analysis of the impact of the selector coding costs on the precision and recall measures to future studies. For future work, we also postpone the study of generalising the precision and recall measures captured in Eq. 7 for two input models. Although definitions of such measures are straightforward, their computation is related to some non-trivial challenges that deserve dedicated discussions.

For our example log L_e and model S_e in Figure 1, gain-based recall and precision are 0.74 and 0.78, respectively. Notice that these values differ from the projection-based recall and precision computed earlier. In Section 4, we study the differences between the two sets in more detail.

3.5. 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 Figure 3.

Step (1): Eq. (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, and obviously this step does not influence stochastic behaviour.

Step (2): Model and log cannot have zero entropies, i.e. they must contain more than one trace with non-zero probability (be determininstic). 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. For the gain measures, we compute the added entropy of this step directly:

$$gain(X,Y) = \frac{\sum_{t \in \hat{X} \cap \hat{Y}} \begin{pmatrix} \min(-(X(t)(1-\lambda)) \log_2(X(t)(1-\lambda)), \\ \quad -(Y(t)(1-\lambda)) \log_2(Y(t)(1-\lambda))) + \\ \min(-(X(t)\lambda) \log_2(X(t)\lambda), \\ \quad -(Y(t)\lambda) \log_2(Y(t)\lambda)) \end{pmatrix}}{H(Y)} \quad (8)$$

This influences entropy in both SDFAs, but only by $0 \sim 0.15$ entropy.

In [7], it is shown that Equation (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.

4. Properties of Stochastic Precision and Recall

In the field of conformance checking, many different measures have been proposed, each based on a conceptual idea. For instance, precision is typically conceptualised as indicating the proportion of behaviour of the model that was observed in an event log. Different measures might operationalise these concepts differently, and recently, a discussion has emerged in the field in which precision, fitness, recall and generalisation have been defined, and properties for measures targeting these concepts have been proposed [38, 41, 30, 27]. A measure that is not known to satisfy any property can be considered to return "magic" numbers. These properties provide general guidance, but are not sufficient – and not intended – to establish comparability within or between measures. That is, even measures satisfying all properties might disagree when applied to the same models and logs, or provide counterintuitive results. In fact, the proposed properties in [38, 41, 30, 27] do not contain strict inequalities at all, thus could be satisfied by trivial non-informative measures.

In this section, we first adapt some existing properties to the realm of stochastic-aware measures and introduce new stochastic-specific propertiesincluding properties having strict inequalities. Second, we establish whether several existing and our new measures satisfy these properties.

Properties establish a conceptual baseline for (stochastic) conformance checking measures, however do not provide guidance on the practical interpretation

of differences in measures (e.g. "how much is 0.1 extra recall?" or "is 0.6 gain-recall higher than 0.6 projection-recall?"). Further research on comparability inter- and intra-measure remains necessary.

4.1. Properties

For readability, the properties have been divided into 5 categories: basic, stochastic-specific, language equality, inequalities and strict inequalities properties.

4.1.1. Basic Properties

A measure should be deterministic: it should always result in the same value for an input log and model:

P1 A stochastic-aware conformance measure should be deterministic;

Property **P1** is mentioned in [38, Axiom A1], [41, DetPro+].

▷ A measure should only depend on the languages of the given log and model rather than on their particular representation and structural differences in SDFAs or Stochastic Petri nets used to encode the languages:

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

Property **P2** implicitly follows from [38, Axiom A1] and is explicitly mentioned in [38, Axiom A4], [41, BehPro+].

In line with all the existing conformance measures, we argue that the values of stochastic-aware conformance measures should fall into the [0,1] interval. Having fixed bounds for all event logs and models allows for easier comparisons between different log and model combinations.

P3 A stochastic-aware conformance measure should return values greater than or equal to 0 and less than or equal to 1;

Property **P3** is implicit in [41], i.e. can be trivially deduced from the claimed properties, and is mentioned as customary in [38]. This property could be generalised to a measure merely having a lower and an upper bound [30], however for simplicity, without loss of generality, we assume these to be 0 and 1 here.

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:

P4 Given two stochastic languages A and B and stochastic-aware precision (*precision*) and recall (*recall*) measures, it should hold that $\mathit{precision}(A, B) = \mathit{recall}(B, A)$.

As [38] considers only precision, no corresponding axiom is included, while [41] considers precision and recall in isolation.

4.1.2. Stochastic-Specific Properties

Inherent to the idea of stochastic conformance checking, a measure should take the stochastic perspective of both log and model into account. That is, each measure should take both the stochastic perspectives of both log and model into account. For instance, $[\langle a\rangle^{0.99}, \langle b\rangle^{0.01}]$ and $[\langle a\rangle^{0.1}, \langle b\rangle^{0.99}]$ should not have a recall or precision of 1.

P5 A stochastic-aware conformance measure should take the stochastic perspective of both log and model into account. For instance, let L_1 and L_2 be stochastic languages that only differ in their stochastic perspective ($\forall_t L_1(t) > 0 \Leftrightarrow L_2(t) > 0$ and $\exists_t L_1(t) \neq L_2(t)$). Then, $\mathit{precision}(L_1, L_2)$ is not 1 and $\mathit{recall}(L_1, L_2)$ is not 1.

Obviously, property **P5** has no equivalent in [38, 41, 30, 27].

Finally, it has been generally accepted that ideally, conformance measures should take the differences within traces into account. That is, traces with small differences should have a smaller impact on the conformance measure than traces with large differences. For instance, comparing $L_1 = [\langle a, b, c, d, e, f, g, h\rangle]$ with $L_2 = [\langle a, b, c, d, e, f, g, z\rangle]$ and $L_3 = [\langle a\rangle]$ and $L_4 = [\langle z\rangle]$, intuitively L_1 and L_2 are closer than L_3 and L_4, even though these pairs of logs have no mutual traces.

P6 A stochastic-aware conformance measure should take differences within traces into account. For instance, let $L_1 \dots L_4$ be equivalent stochastic language, except for four traces $t_1 \dots t_4$ such that $\forall_i (t_i \in L_i \wedge \forall j \neq i t_i \notin L_j)$, such that $L_1(t_1) = L_2(t_2) = L_3(t_3) = L_4(t_4)$. Let δ be a trace distance function, such that $\delta(t_1, t_2) < \delta(t_3, t_4)$. Then, $\mathit{precision}(L_1, L_2) < \mathit{precision}(L_3, L_4)$ and $\mathit{recall}(L_1, L_2) < \mathit{recall}(L_3, L_4)$.

We acknowledge that the intuitive property implies the formalisation, but not the other way around, and we leave a "full" formalisation for future work. Surprisingly, despite the apparent agreement in the community on this idea, only [27] mentions a similar property.

4.1.3. Language-Equality Properties

A conformance value of 1 signifies a perfect conformance, which for the stochastic-aware measures is instantiated as follows:

P7 If an event log and a model express the same stochastic language, then they should have a stochastic-aware precision of 1;

P8 If precision is 1, then an event log and a model should express the same stochastic language;

P9 If an event log and a model express the same stochastic language, then they should have a stochastic-aware recall of 1;

P10 If recall is 1, then an event log and a model should express the same stochastic language;

No equivalent for non-stochastic conformance measures for these properties can be derived from [38], while [41, RecPro5+] correspond to **P9** and [41, PrecPro5+, PrecPro60] correspond to **P9**.

4.1.4. Inequality Properties

In general, different probabilities of traces from an event log and those described by a model should result in non-perfect conformance values. The larger the differences, the smaller the returned conformance values should be:

P11 If for all traces of a model M the difference in probability between M and a log L_1 is lower than for another log L_2, then the precision of L_1 should be higher than of L_2:
If $\forall_{t \in \Sigma^*} M(t) > 0 \Rightarrow |L_1(t) - M(t)| \leq |L_2(t) - M(t)|$ then $precision(L_1, M) \geq precision(L_2, M)$;

P12 If for all traces of a log L the difference in probability between L and a model M_1 is lower than for another model M_2, then the recall of M_1 should be higher than of M_2:
If $\forall_{t \in \Sigma^*} L(t) > 0 \Rightarrow |M_1(t) - L(t)| \leq |M_2(t) - L(t)|$ then $recall(L, M_1) \geq recall(L, M_2)$;

These properties have resemblances to [41, PrecPro1-4], [38, Axiom A2] and [38, Axiom A5].

4.1.5. Strict Inequality Properties

These properties can be strenghtened to strict inequalities by requiring that at least one trace of M is in L_1 and not in L_2 (symmetric for recall):

P13 If for all traces of a model M the difference in probability between M and a log L_1 is lower than for another log L_2 and 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 $\forall_{t \in \Sigma^*} M(t) > 0 \Rightarrow |L_1(t) - M(t)| \leq |L_2(t) - M(t)|$ and $\exists_{t \in \Sigma^*} M(t) > 0 \wedge L_1(t) > 0 \wedge L_2(t) = 0$ then $precision(L_1, M) > precision(L_2, M)$;

P14 If for all traces of a log L the difference in probability between L and a model M_1 is lower than for another model M_2 and 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 $\forall_{t \in \Sigma^*} L(t) > 0 \Rightarrow |M_1(t) - L(t)| \leq |M_2(t) - L(t)|$ and $\exists_{t \in \Sigma^*} L(t) > 0 \wedge M_1(t) > 0 \wedge M_2(t) = 0$, then $recall(L, M_1) > recall(L, M_2)$;

There is no equivalent of these properties in [38, 41], as none of the relevant axioms/propositions involves a strict inequality. In [30] however, similar properties are introduced.

Table 1: Overview of properties of new and existing stochastic conformance measures.

	EMSC [19]	ER [28, 2]	entropy recall & precision (Eq. 5)	gain entropy recall & precision (Eq. 7)
P1	yes (without unfolding)	yes	yes	yes
P2	yes	yes	yes	yes
P3	yes	no*	yes	yes
P4	n/a	n/a	yes	yes
P5	yes	yes	no	yes
P6	yes (without unfolding)	no	prefix only	no
P7	yes	yes*	yes	yes
P8	yes (without unfolding)	yes*	no	yes
P9	yes	yes*	yes	yes
P10	yes (without unfolding)	yes*	no	yes
P11	no	n/a	yes	no
P12	no	no*	yes	no
P13	no	n/a	yes	no
P14	no	no*	yes	no

* refer to the discussion in Section 4.2.2 for details.

4.2. Analysis of Stochastic Conformance Measures

In this section, we show which of our measures possess which of these properties. Furthermore, we also consider EMSC. Table 1 shows an overview, and next we discuss these for EMSC, entropic relevance and our new measures.

4.2.1. Properties of Earth Movers' Stochastic Conformance

The Earth Movers' Stochastic Conformance [19] (EMSC) measure combines recall and precision into one measure, thus **P4** does not apply. All of these types of comparisons are theoretically defined, however for practical applications, for the latter two types, EMSC applies a pre-processing step in which the stochastic language of the model is unfolded, thus truncating loops and concurrent behaviour. This unfolding step is conceptually not deterministic (**P1**).

EMSC, by construction, considers only stochastic languages (**P2**), results in a number between 0 and 1 (**P3**), takes the stochastic perspective of both log and model into account (**P5**), and by use of the Levenshtein trace distance, considers partially matching traces (**P6**). In [19, Lemma 1], it was shown that **P7**, **P8**, **P9** and **P10** hold for the log-log and the theoretical un-truncated, version of EMSC.

A counterexample for **P11** is shown in Table 2. In this counterexample, the probability mass targeted by the assumptions of **P11** is not large enough to counteract the influence of the probability mass not targeted, which we exploit by introducing a trace that is very close to a trace in M but not equivalent. By symmetry, **P12** does not hold, and by similar reasoning, **P13** and **P14** also do not hold for EMSC. This also illustrates that it might be challenging to define stochastic conformance measures that satisfy both **P11**, **P12**, **P13** and **P14** as well as **P6**.

Table 2: A counterexample that **P11** does not hold for EMSC: $EMSC(L_1, M) \approx 0.5$ and $EMSC(L_2, M) \approx 0.8$.

Languages	M	L_1	L_2
$\langle a \rangle$	0.5	0.6	0.7
$\langle a, a, \ldots \rangle$	0.5	0	0
$\langle a, a, \ldots b \rangle$	0	0	0.3
$\langle c \rangle$	0	0.4	0

Distance	$\langle a, a, \ldots \rangle$	$\langle a, a, \ldots b \rangle$	$\langle c \rangle$
$\langle a \rangle$	~1	~1	1
$\langle a, a, \ldots \rangle$	0	~0	1
$\langle a, a, \ldots b \rangle$	~0	0	1

Reall. $M \to L_1$	$\langle a \rangle$	$\langle c \rangle$
$\langle a \rangle$	0.5	0
$\langle a, a, \ldots \rangle$	0.1	0.4

Reall. $M \to L_2$	$\langle a \rangle$	$\langle a, a, \ldots b \rangle$
$\langle a \rangle$	0.5	0
$\langle a, a, \ldots \rangle$	0.2	0.3

4.2.2. Properties of Entropic Relevance

By definition, the entropic relevance measure is deterministic (**P1**), subject to floating-point arithmetic error, and the measured values are determined by the stochastic languages of the compared log and model (**P2**). Properties **P3** and **P7**–**P10** are not directly applicable to entropic relevance. First, it is neither intended nor possible to normalise the measure to fit the "0 to 1" interval (**P3**); note that entropic relevance measurements have meaningful units, *bits* required to describe a trace from the log using the model. Second, entropic relevance reflects the compromise between precision and recall in a single measurement [2]. However, the minimal possible entropic relevance rel_{min} for a given log is the relevance computed for the log and a model that expresses the same stochastic language as the log (**P7** and **P9**); rel_{min} can be computed using the log as the model. Moreover, the entropic relevance measure can be configured to ensure that the measurement of rel_{min} guarantees that the log and the model express the same stochastic language by using the background costing model that assigns high costs to traces not supported by the model (**P8** and **P10**).

The entropic relevance measure implements a compromise between precision and recall, and, thus, **P4** does not apply. Furthermore, computing the measurements takes the stochastic perspectives of both the log and the model into account (**P5**). Finally, entropic relevance is grounded in the exact matching of traces in the compared stochastic languages, and, hence, **P6** is not fulfilled.

As entropic relevance is designed for comparing candidate models for being a better representation of a given log and not for comparing candidate logs for being a better instantiation of the model, **P11** and **P13** do not apply. Properties **P12** and **P14** do not hold in the general case. The entropic relevance supports a nuanced analysis of the trace probabilities. For instance, to avoid model overfitting, it can penalize a model that describes an infrequent log trace more than it would "punish" a model that describes a frequent log trace with a probability that deviates from its probability in the log.

To conclude, we emphasize that entropic relevance was designed to support the use case of choosing a candidate model that represents the log traces better

than some other candidate model. It was not intended as a measure of precision or recall and is, in fact, a blend of the two [2]. Thus, the analysis in this section is provided for informational, rather than comparison, purpose.

4.2.3. Properties of Entropy Recall and Precision

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.

Our precision and recall measures satisfy **P3**:

Theorem 1. *For any log L and model M (given as SDFAs), it holds that $0 \leq \mathit{recall}(L, M) \leq 1$ and $0 \leq \mathit{precision}(L, M) \leq 1$.*

Proof. (Sketch) $\text{precision}(L, M) = H(P)/H(M)$, where P is the projection of L and M that preserves the probabilities of the model. Note that for any stochastic language X, it holds that $H(X) \geq 0$. Hence, it holds that $\text{precision}(L, M) \geq 0$. Next, we show that $H(P) \leq H(M)$. Let P be obtained from M by keeping the structure, that is, the states S, the transition function δ, and the start state s_0, of M unchanged and augmenting the probability function of M (possibly by setting some values of the probability function to 0); note that this does not always hold true, however, other situations can be trivially reduced to this case by unfolding the structure of M and preserving the original probabilities for the unfolded transitions. Let $s \in S$ be a state. Let U_s be the set of all subtraces that can be used to reach s from s_0 in P and let V_s be the set of all subtraces that can be used to reach a termination from s in P. Similarly, let X_s be the set of all subtraces that can be used to reach s from s_0 in M and let Y_s be the set of all subtraces that can be used to reach a termination from s in M. It holds that $U_s \subseteq X_s$ and there exists $Z_s \subseteq V_s$ such that $U_s \circ Z_s$ are the traces of M, where $\circ$ is the concatenation operation. By construction of P, it holds that $p(t \in P) = p(t \in M)$, $t \in U_s \circ Z_s$. Note that the traces in $X_s \setminus U_s$ are the traces of M but not the traces of P. Next, we observe that $P[s] = H(U_s \circ Z_s, P) + H(U_s \circ (V_s \setminus Z_s), P)$ *is less than or equal to* $M[s] = H(U_s \circ Z_s, M) + H((X_s \setminus U_s) \circ Y_s, M) + H(U_s \circ (Y_s \setminus Z_s), M)$, with $H(A, B) = -\sum_{t \in A} p(t \in B) \log_2 p(t \in B)$, where A is a set of traces and B is a stochastic language. Indeed, it holds that $H(U_s \circ Z_s, P) = H(U_s \circ Z_s, M)$. Moreover, it holds that $H((X_s \setminus U_s) \circ Y_s, M) \geq 0$. Finally, it holds that $H(U_s \circ (V_s \setminus Z_s), P) \leq H(U_s \circ (Y_s \setminus Z_s), M)$ because, by construction of P, there exists a bijection β from $U_s \circ (V_s \setminus Z_s)$ to a partition of $U_s \circ (Y_s \setminus Z_s)$ such that for each $t \in U_s \circ (V_s \setminus Z_s)$ it holds that t is a prefix of every trace in $\beta(t)$ and $p(t \in P) = \sum_{x \in \beta(t)} p(x \in M)$. Finally, we observe that $P[s_0] = H(P)$ and $M[s_0] = H(M)$. Thus, it holds that $H(P) \leq H(M)$ and, consequently, $H(P)/H(M) \leq 1$. Because, by definition, $\text{recall}(L, M) = \text{precision}(M, L)$, it holds that $0 \leq \text{recall}(L, M) \leq 1$. □

Properties **P7** and **P9** hold for our precision and recall measures, because if the log and model express the same stochastic language, then the projection will have this same stochastic language as well. Then, the entropy of all three stochastic languages is obviously equal, hence the numerator and denominator

in (5) are equal. However, their reverse (**P8** and **P10**) do not hold: the logs $[\langle a\rangle^{0.99}, \langle b\rangle^{0.01}]$ and $[\langle a\rangle^{0.1}, \langle b\rangle^{0.99}]$ have a recall and precision of 1 but do not have the same stochastic language. This is also a counterexample for **P5**.

Properties **P11** and **P12** 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 properties **P13** and **P14** then hold by extension.

Property **P4** holds for our measures by definition.

Property **P6**, that is, that the measures should be forgiving for partial trace matching, holds for our measures, however only equal prefixes are considered: two traces of $\langle a, b, c, d, e, f, g\rangle$ and $\langle a, b, c, d, e, f, z\rangle$ are considered equal up to their last event (g vs z), and the conjunctive automaton is constructed up to this point. In contrast, the traces $\langle a, b, c, d, e, f, g\rangle$ and $\langle z, b, c, d, e, f, g\rangle$ would be considered completely different by our measures.

Please note that the inclusion of λ edges as described in Section 3.5 has no influence on these arguments, if λ is chosen sufficiently small.

4.2.4. Properties of Gain Entropy Recall and Precision

Properties **P1** and **P2** hold for gain measures, as both the projection and the entropy are computed using deterministic procedures with only stochastic languages as inputs. By construction of Equation 6, both measures are between 0 and 1 (**P3**), symmetric (**P4**), and take the stochastic perspective of both log and model into account (**P5**).

Properties **P7** and **P9** hold for our gain measures, because if the log and model express the same stochastic language, then the nominator of Equation 6 will by construction have the same entropy as both log and model. Similarly, their reverse (**P8** and **P10**) holds by construction.

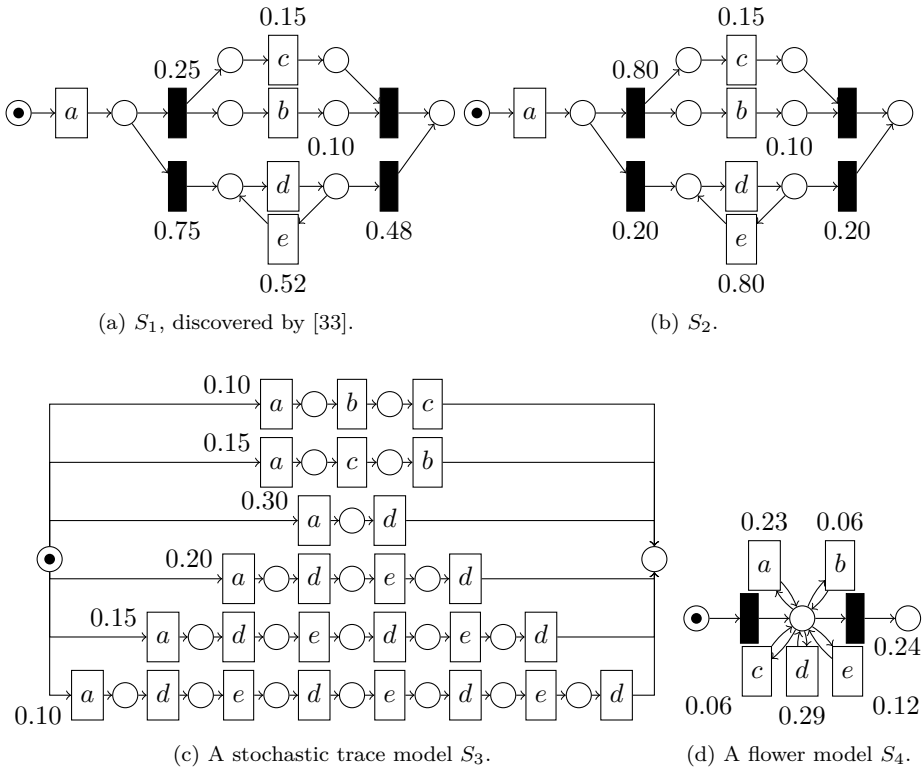
Property **P11** does not hold for the gain measures: the denominator of Eq. 6 does not change as M does not change, and as the nominator takes the minimum entropy between the log and model, there is no guarantee that if the difference between log and model is lowered for a particular trace, the minimum entropy also (non-strictly) decreases. For instance, consider a trace t such that $M(t) = 0.5$, $L_1(t) = 0.6$ and $L_2(t) = 0.3$. Then, $\min(M(t)\log_2 M(t), L_1(t)\log_2 L_1(t)) \approx 0.44$ while $\min(M(t)\log_2 M(t), L_2(t)\log_2 L_2(t)) = 0.5$. Consequently, Properties **P12**, **P13** and **P14** do not hold either.

The gain measures do not consider partial trace equivalences (**P6**), due to the nominator of Equation 6.

Please note that the inclusion of λ edges as described in Section 3.5 has no influence on these arguments, if λ is chosen sufficiently small.

5. 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



(a) S_1, discovered by [33].

(b) S_2.

(c) A stochastic trace model S_3.

(d) A flower model S_4.

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

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.

5.1. Implementation

The proposed measures have been implemented as a plug-in of the ProM framework [43]: "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.

5.2. 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 Figure 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 [33] from L. Model S_2 is a manually created SPN

Table 3: Stochastic languages of L and the SPNs in Figure 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	0.04	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

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 *stochastic 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 3 shows (fragments of) the stochastic languages of these models.

We applied the measures presented in this paper (entropy recall and gain entropy recall), the Earth Movers' (EMSC) [18] measure, entropic relevance (ER) [28], as well as the non-stochastic alignment-based (A) [24] and projected (P) [15] precision measures. The results are shown in Table 4 (recall, where relevant, is 1 for all models).

Intuitively, the trace model S_3 perfectly represents the event log, thus should have a perfect stochastic conformance. Entropy, gain and EMSC express this. ER also ranks S_3 highest. Note that ER does not have a notion of a "perfect" conformance; the lower the relevance value the better. A and P do not consider the stochastis perspective, thus also express S_3 to have a perfect conformance. The flower model S_4 intuitively should have the lowest precision, as it can generate any trace of the alphabet with nonzero precision. All measures agree on S_4 having the lowest conformance, where entropy and gain approach their theoretical limit of 0, which indicates that the stochastic state spaces are almost completely different; EMSC indicates that a bit over half of the probability mass needs to be moved to align the stochastic languages of L and S_4[2]; ER ranks the stochastic perspective of model S_4 to be much different from the stochastic perspective of the log as the value is much greater than for the other three models. For S_1 and S_2, 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 argue that S_1 represents L better than S_2, which all stochastic conformance checking measures confirm (entropy, gain, EMSC and ER). For EMSC, there are large differences between S_1 and S_2, which seems to indicate that the control flow cannot play a large role for this measure. In contrast, for entropy S_1 and S_2 are close, which here would be our preference, but highlights the need for future studies into the small differences of (stochastic) conformance

[2]Please note that EMSC supports loop behaviour by unfolding, thus the measure can get arbitrarily close to 0 by unfolding loop behaviour more, which makes the actual value a bit arbitrary.

Table 4: Stochastic measures compared to regular conformance checking techniques.

rank	entropy	gain	EMSC [18]	ER [28]	A [24]	P [15]
1	S_3 (1)	S_3 (1)	S_3 (1)	S_3 (2.471)	S_3 (1)	S_3 (1)
2	S_1 (0.918)	S_1 (0.900)	S_1 (0.908)	S_1 (2.910)	S_1, S_2 (0.846)	S_1, S_2 (0.934)
3	S_2 (0.834)	S_2 (0.541)	S_2 (0.570)	S_2 (4.473)		
4	S_4 (0.096)	S_4 (0.043)	S_4 (0.509)	S_4 (12.376)	S_4 (0.292)	S_4 (0.551)

checking techniques beyond properties. ER gives a small preference to model S_1 over S_2. Measures A and P do not see any difference between models S_1 and S_2, which highlights their non-stochastic nature. Finally, it is remarkable that EMSC's values for S_2 and S_4 are closer, which may be due to EMSC having to unfold the loop in the flower model, which is bounded and brings the compared language falsely closer to L.

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

5.3. Practical Feasibility

Next, we report on the feasibility of the proposed stochastic-aware conformance measures. To this end,we followed the set-up shown in Figure 5: we used ten publicly available real-life event logs arbitrarily chosen from the logs published by the IEEE Task Force on Process Mining[3]. To these logs, we applied 4 stochastic discovery techniques to obtain stochastic Petri nets: (1) the stochastic miner by Rogge-Solti et al. (ARS) [33], (2) Frequency Estimator [5] using Inductive Miner - infrequent [14] (FEIMf), (3) Right Hand Activity Pair Estimator [5] using Inductive Miner - infrequent [14] (RHEIMf), and (4) a baseline model including the most-occurring trace of the log with probability 1 (MOT).

We then applied 5 stochastic conformance measures to the 40 pairs of event logs and discovered models: (1) EMSC [19], (2) ER [28], (3) our new entropy recall and precision (Equation (5)), (4) our new gain entropy recall and precision (Equation (7)), and (5) alignment-based fitness and precision [42, 1][4] as a non-stochastic baseline.

We also measured the time these measures took, and the size of the models in nodes (places and transitions) and edges. The code used to run the evaluation is publicly available.[5] The machine used to compute the measures had a W-2195 CPU with 64GB RAM running Ubuntu 20.04, however as all time measures were only taken once, they can reveal general trends only. The BPIC13 logs were pre-processed by removing the lifecycle:transition attribute from all events, to ensure appropriate interpretation by the ER implementation.

[3]See `https://data.4tu.nl/repository/collection:event_logs_real`.

[4]Please note that this computation also includes generalisation as a side effect

[5]The source code used in the evaluation is accessible via `https://svn.win.tue.nl/repos/prom/Packages/StochasticAwareConformanceChecking/Trunk` (revision 44718).

Table 5: Precision and recall values and times taken to compute them in ms.

log	algorithm	model size		EMSC [19]		ER [28]		entropy			gain entropy			alignments [42, 1]		
		nodes	edges	EMSC	time	relevance	time	recall	precision	time	recall	precision	time	fitness	precision	time
BPIC13-op	FEIMf	15	18	.496	12	12.281	3	.417	.936	6	.024	.048	6	.781	.841	56
BPIC13-op	RHEIMf	15	18	.497	14	11.595	3	.417	.921	4	.027	.059	6	.781	.841	50
BPIC13-op	ARS	31	38	.494	26	8.016	3	!	!	!	!	!	!	1.000	.581	4721
BPIC13-op	MOT	5	4	.547	10	7.192	2	.258	1.000	275	10^{-9}	.269	5	.627	1.000	13
BPIC17-o	FEIMf	15	18	.890	160	8.429	276	.996	1.000	64	.546	.717	245	.995	.972	1049
BPIC17-o	RHEIMf	15	18	.890	259	8.429	244	.996	1.000	129	.543	.720	150	.995	.972	717
BPIC17-o	ARS	29	34	.912	152	2.840	239	1.000	.995	43	.837	.644	154	1.000	.841	658
BPIC17-o	MOT	9	8	.770	162	12.388	115	.592	1.000	73	10^{-9}	.426	170	.810	1.000	662
BPIC18-4	FEIMf	41	52	.304	10^7	84.153	355	!	!	!	!	!	!	.836	.325	10^6
BPIC18-4	RHEIMf	41	52	.341	10^5	84.501	425	!	!	!	!	!	!	.836	.325	10^6
BPIC18-4	ARS	122	154	!	!	84.153	10^4	0.000	0.000	10^7	0.000	0.000	10^7	0.000	0.000	10^4
BPIC18-4	MOT	9	8	.359	520	81.042	336	.105	1.000	10^7	10^{-10}	.238	10^6	.456	1.000	3667
BPIC11	FEIMf	765	1042	!	!	!	!	!	!	!	!	!	!	!	!	!
BPIC11	RHEIMf	738	1022	!	!	!	!	!	!	!	!	!	!	!	!	!
BPIC11	ARS	error in discovery														
BPIC11	MOT	5	4	.149	235	1229.736	372	.093	1.000	10^7	10^{-10}	.086	10^7	.198	1.000	484
BPIC13-cp	FEIMf	13	14	.230	9158	15.797	5	.533	.039	75	.002	10^{-4}	22	.824	.747	329
BPIC13-cp	RHEIMf	13	14	.231	10^4	14.840	5	.533	.086	146	.007	.003	62	.824	.747	162
BPIC13-cp	ARS	31	38	.440	240	12.735	7	.656	.015	32	0.000	0.000	131	1.000	.503	10^4
BPIC13-cp	MOT	5	4	.619	12	11.328	4	.271	1.000	26	10^{-9}	.379	16	.720	1.000	40
BPIC15-1	FEIMf	1085	1550	.068	10^5	!	!	!	!	!	!	!	!	!	!	!
BPIC15-1	RHEIMf	1073	1536	!	!	!	!	!	!	!	!	!	!	!	!	!
BPIC15-1	ARS	error in discovery														
BPIC15-1	MOT	13	12	.117	164	384.581	180	.110	1.000	10^5	10^{-10}	.054	10^5	.198	1.000	261
Sepsis	FEIMf	57	72	.494	10^8	62.898	60	!	!	!	!	!	!	.903	.434	10^5
Sepsis	RHEIMf	57	72	!	!	60.687	122	!	!	!	!	!	!	.903	.434	10^5
Sepsis	ARS	100	126	!	!	63.313	148	!	!	!	0.000	0.000	4593	!	!	!
Sepsis	MOT	7	6	.284	103	62.979	24	.121	1.000	2698	10^{-10}	.083	2282	.417	1.000	91
Roadfines	FEIMf	42	52	.610	523	16.982	256	!	!	!	!	!	!	.984	.665	3369
Roadfines	RHEIMf	42	52	.581	501	23.168	416	!	!	!	!	!	!	.984	.665	2344
Roadfines	ARS	75	92	.747	10^5	16.970	413	.683	.458	173	10^{-4}	10^{-5}	593	!	!	!
Roadfines	MOT	11	10	.614	388	9.846	248	.820	1.000	142	10^{-9}	.421	429	.673	1.000	2079
BPIC13-i	FEIMf	22	24	!	!	25.277	152	.882	10^{-4}	10^4	10^{-4}	10^{-8}	7129	.920	.654	10^5
BPIC13-i	RHEIMf	22	24	!	!	24.414	37	.882	.002	10^4	.001	10^{-6}	5596	.920	.603	10^4
BPIC13-i	ARS	37	46	.386	10^4	22.465	30	.251	.870	10^4	!	!	!	1.000	.393	10^7
BPIC13-i	MOT	7	6	.530	57	21.091	19	.221	1.000	10^4	10^{-9}	.283	6883	.638	1.000	352
BPIC12	FEIMf	80	102	!	!	96.671	104	!	!	!	!	!	!	!	!	!
BPIC12	RHEIMf	80	102	!	!	98.130	213	!	!	!	!	!	!	!	!	!
BPIC12	ARS	error in discovery														
BPIC12	MOT	7	6	.405	10^8	92.476	137	.107	1.000	10^5	10^{-9}	.312	10^5	.474	1.000	1719

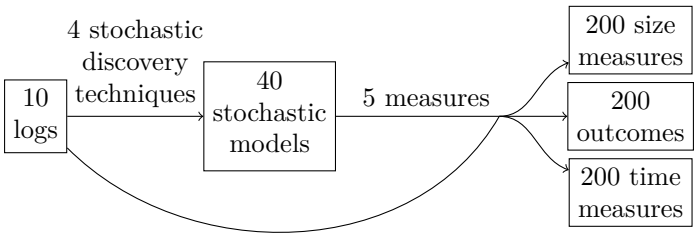


Figure 5: Set-up of our practical feasibility experiment.

Table 5 shows the results. Some results could not be obtained: for BPIC11, BPIC15-1 and BPIC12, ARS did not return a model as it ran out of memory. Where a conformance measure did not return a result, an exclamation mark was annotated. For EMSC, this was due to time: a timeout of a day was applied. For the entropy and gain entropy measures, this was mostly due to models that were not translatable into SDFAs (see Section 3.5). An exception were the measures for BPIC15-1, where the automaton-generation (shared by ER, entropy and gain entropy) step timed out. For all other tested logs and models, ER completed. To improve the entropy-based tools, one can look into computing the trace probabilities directly using SPNs, rather than using the derived SDFAs. However, computation of trace probabilities directly on SPNs is associated with challenges related to the handling of silent transitions [18].

Run time is comparable between entropy and gain entropy measures, while ER seems less influenced by the size of the log or the complexity of the model. EMSC is clearly the most complex of the tested stochastic conformance measures: for MOT it is very fast, while for real models it in general takes considerably longer than the other tested conformance measures. We also included standard alignments [42, 1] to illustrate runtime compared to this non-stochastic aware conformance checking technique. For MOT, alignments can be faster than our new measures, due to our measures constructing the entire state space of a model, while alignment-based precision constructs only the state space covered by the log, which for this baseline miner MOT is a single trace. For the three actual stochastic discovery techniques, compared with entropy and gain entropy, alignments show mixed computing time behaviour: on smaller models (BPIC13*, BPIC17-o, Roadfines), our new measures are faster than alignments. There are some settings in which the entropy and gain entropy measures did not produce a result but alignments did. This can be due to the model being not translatable to an SDFA or the state space being too large. Alignment-based precision only computes the state space that was covered by the log, plus one more step according to the model [1], thus on large state spaces does not cover the entire state space of the model. While both alignments and our new measures contain optimisation steps, we conclude that considering the stochastic perspective does in most cases not inflict a higher run time, if the model is supported by our measures.

Log BPIC17-o for ARS provides an example for the benefit of the gain measures: entropy recall is 1, however as Property **P10** does not hold for this measure, we cannot conclude stochastic language equivalence between the model by ARS and the log. EMSC gives no clarity either, as the model might have been unfolded and truncated. Using the gain entropy recall, for which Property **P9** holds, we can conclude that ARS did not discover a stochastic language-equivalent model.

For EMSC/ER/recall, all measures seem to agree on the models discovered by FEIMf and RHEIMf to be similar: the measured differences are minimal, except for BPIC13-i. This might be due to the control flow perspective of these miners be equivalent, where the stochastic perspective matters less. For EMSC/ER/precision, the differences are slightly larger. It would be interesting to compare the measured values based on intuition, and to develop intuition accordingly, however we leave this for future work.

One 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.

Further analysis showed that for SDFAs with large cycles, Equation (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 both sets of our measures. However, run time was not infeasible in our evaluation: at most two hours for the largest logs of most complex procesess we tested, but generally much less. Nevertheless, as future work, this step might be optimised using the SDFA's structure.

Please note that this experiment did not attempt to study the values of the measures, though it is clear that entropy and gain entropy might give considerably different results for the same inputs. As our discussion on properties for these measures showed (Section 4), both provide different guarantees, and measure differences in stochastic behaviour differently. We leave a detailed study and analysis of what these measures penalise for, in terms of control flow and the stochastic perspective, for future work.

5.4. Practical Usefulness: German Health Insurance Company

In 2008, we conducted a project with a German health insurance company aimed to analyse and improve their resource planning practices [29]. The company had a dedicated department to hand-craft stochastic process models that describe their critical operational processes. Concretely, the processes were described as EPCs augmented with decision probabilities and average times for conducting business activities (modeled as functions in EPCs). The probabilities of making the various decisions in processes were introduced to model the likelihood of observing the corresponding business decisions in the real world. The augmented models were used by the company to plan the workforce in the upcoming calendar year. Given a forecast of the expected number of cases of a particular process, the company relied on the estimated decision probabilities

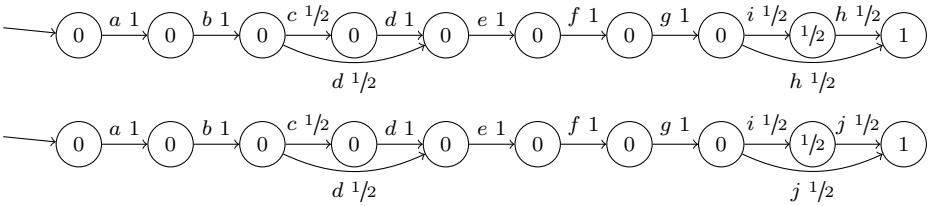


Figure 6: Two slightly different SDFAs from a German insurer. ($h \leftrightarrow j$).

and activity durations to compute how much effort must be invested into the process and, consequently, resources hired to support that process.

To ensure that no double resource allocation occurs, as this results in a financial loss to the company, it is important that no two models describe the same traces. The more similar or identical traces two models describe, the higher the chance they (partly) address the same business case, such as an insurance claim handling or an insurance application process. Due to a high number of models, i.e., approximately 4 000 models were available at the time of the project, their manual analysis was deemed intractable.[6]

Armed with the stochastic conformance measures introduced in this article, to identify models that describe identical (and frequent) traces, we performed their pairwise comparison. 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 in our analysis. This filtering step resulted in the collection of 3 090 models. The average time of computing projection-based conformance, either precision or recall, for a pair of models using our tool was measured to be 69 ms. As a result, we discovered 48 pairs of distinct models that describe some identical traces. Two anonymised models from the collection translated to SDFAs, for which both stochastic recall and precision values are equal to 0.4, are shown in Figure 6. As such models can potentially lead to double allocation of resources by the company, they should be further analysed by business analysts. As these models describe identical frequent traces, to reduce the number of maintained models, the analysts may consider combining them into a single model.

6. 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 [40, 6, 30]. The vast majority of the existing conformance measures address nondeterministic models and logs. Nondeterminism, as a concept in computer

[6]Unfortunately, the models were not made publicly available, as per the terms of the agreement with the company.

science, was introduced in [31] 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 [44]. In [41], van der Aalst stressed the need to consider probabilities in conformance checking.

The stochastic perspective has seen recent attention in process discovery techniques, such as the miner by Rogge-Solti et al. [33], which uses alignments [42] to estimate arbitrary delay distributions of stochastic Petri nets thereby implying a stochastic perspective; and six techniques that take an existing control flow model and estimate probabilities for the transitions to derive an SPN [5]. Further similar approaches include declarative constraints with bounds on the fraction of cases that must satisfy or violate constraints [22, 21] and Bayesian model fitting [12].

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 [35], 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 [30], 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 [30], 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 [42] 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 [24] 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 [15] addresses long run times of conformance checking techniques by projecting behaviour onto subsets of activities of a certain size. The measures presented in this paper can be extended in a similar fashion. Generalised conformance checking [32] compares an event log and model based on a given trust level for each, derived, for instance, from identified data quality issues [37]. 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 (EMSC) [18, 19] and Entropic Relevance (ER) [28, 2] techniques are the only stochastic conformance checking techniques proposed today. In EMSC, 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 measures support such languages. In ER, the log and model are compared to obtain the average number of bits required to compress a trace from the log using the model. The better the model represents the log traces and their relative likelihoods, the better compression can be achieved. In Section 4, we analysed EMSC and ER in detail.

The stochastic perspective has enabled several new types of analyses, such as computing the trace attributes that have the largest impact on the followed stochastic process [17], finding stochastic-based changes in processes (process drift) [3] and anomaly detection without using process models [26, 25].

Furthermore, our work contributes to the ongoing discussion on ideal conformance checking measures by proposing properties that these measures should have [38, 41, 30, 27].

Finally, to compare SDFAs, the Kullback-Leibler (KL) divergence [7] 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.

7. 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 two sets of stochastic-aware conformance checking techniques with two measures each: fitness and precision. The proposed sets of 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. Fourteen desirable properties of stochastic conformance checking measures were identified and the adherence of our measures to these properties was analysed.

An evaluation based on our publicly available implementation confirmed the feasibility of using the measures in real-life industrial settings. **For models with finite and deterministic stochastic state spaces. ARTEM: This is not a sentence** 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 entropy measures address (to some extent) the problem of partial trace matches [27]: 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, the entropy measures consider the stochastic perspective of either log or model, but not both. In future work, this could be addressed. The gain entropy measures are the opposite: they do not consider partial trace matches, but consider the stochastic perspective of both log and model.

As future work, it might be interesting to characterise the class of stochastic Petri nets that can be represented by deterministic stochastic languages and, by extension, SDFAs. Furthermore, we encourage more research on stochastic conformance measures that satisfy the properties identified in this paper.

Acknowledgments. Sander Leemans was partly supported by QUT's Centre for Data Science. Artem Polyvyanyy was in part supported by the Australian Research Council project DP180102839.

References

[1] Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Alignment based precision checking. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business Information Processing*, pages 137–149. Springer, 2012.

[2] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. *Information Systems*, page 101922, 2021.

[3] Tobias Brockhoff, Merih Seran Uysal, and Wil M. P. van der Aalst. Time-aware concept drift detection using the earth mover's distance. In Boudewijn F. van Dongen, Marco Montali, and Moe Thandar Wynn, editors, *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*, pages 33–40. IEEE, 2020.

[4] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In *CoopIS*, 2012.

[5] Adam Burke, Sander J. J. Leemans, and Moe Thandar Wynn. Stochastic process discovery by weight estimation. In Sander J. J. Leemans and Henrik Leopold, editors, *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*, volume 406 of *Lecture Notes in Business Information Processing*, pages 260–272. Springer, 2020.

[6] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking—Relating Processes and Models*. Springer, 2018.

[7] Rafael C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *ITA*, 31(5):437–444, 1997.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of information theory, 2nd Ed.* Wiley, 2006.

[9] Fleur Deken, Paul R. Carlile, Hans Berends, and Kristina Lauche. Generating novelty through interdependent routines: A process model of routine work. *Organization Science*, 27(3):659–677, 2016.

[10] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[11] Martha S. Feldman and Brian T. Pentland. Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, 48(1):94–118, 2003.

[12] Gert Janssenswillen, Benoît Depaire, and Christel Faes. Enhancing discovered process models using bayesian inference and MCMC. In Adela del-Río-Ortega, Henrik Leopold, and Flávia Maria Santoro, editors, *Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers*, volume 397 of *Lecture Notes in Business Information Processing*, pages 295–307. Springer, 2020.

[13] Curtis LeBaron, Marlys K. Christianson, Lyndon Garrett, and Roy Ilan. Coordinating flexible performance during everyday work: An ethnomethodological study of handoff routines. *Organization Science*, 27(3), 2016.

[14] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In Niels Lohmann, Minseok Song, and Petia Wohed, editors, *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer, 2013.

[15] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Scalable process discovery and conformance checking. *Software and System Modeling*, 17(2):599–631, 2018.

[16] Sander J. J. Leemans and Artem Polyvyanyy. Stochastic-aware conformance checking: An entropy-based approach. In *CAiSE*, volume 12127 of *Lecture Notes in Computer Science*, pages 217–233. Springer, 2020.

[17] Sander J. J. Leemans, Shiva Shabaninejad, Kanika Goel, Hassan Khosravi, Shazia W. Sadiq, and Moe Thandar Wynn. Identifying cohorts: Recommending drill-downs based on differences in behaviour for process mining. In Gillian Dobbie, Ulrich Frank, Gerti Kappel, Stephen W. Liddle, and Heinrich C. Mayr, editors, *Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings*, volume 12400 of *Lecture Notes in Computer Science*, pages 92–102. Springer, 2020.

[18] Sander J. J. Leemans, Anja F. Syring, and Wil M. P. van der Aalst. Earth movers' stochastic conformance checking. In *BPM*, pages 127–143, 2019.

[19] Sander J. J. Leemans, Wil MP van der Aalst, Tobias Brockhoff, and Artem Polyvyanyy. Stochastic process mining: Earth movers' stochastic conformance. *Information Systems*, page 101724, 2021.

[20] Peter Linz. *An introduction to formal languages and automata, 4th Ed.* JBP, 2006.

[21] Fabrizio Maria Maggi, Marco Montali, and Rafael Peñaloza. Probabilistic conformance checking based on declarative process models. In Nicolas Herbaut and Marcello La Rosa, editors, *Advanced Information Systems Engineering - CAiSE Forum 2020, Grenoble, France, June 8-12, 2020, Proceedings*, volume 386 of *Lecture Notes in Business Information Processing*, pages 86–99. Springer, 2020.

[22] Fabrizio Maria Maggi, Marco Montali, and Rafael Peñaloza. Temporal logics over finite traces with uncertainty. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 10218–10225. AAAI Press, 2020.

[23] Marco Ajmone Marsan 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.

[24] Jorge Munoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In *BPM 2010*, pages 211–226, 2010.

[25] JC Mussche, Ir RM Remco Dijkman-TU, and FPR Floris Olsthoorn-ASML. Detecting process deviations and studying their mutual relation and impact on business performances.

[26] Timo Nolle. Process learning for autonomous process anomaly correction. 2020.

[27] Artem Polyvyanyy and Anna A. Kalenkova. Monotone conformance checking for partially matching designed and observed processes. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24–26, 2019*, pages 81–88. IEEE, 2019.

[28] Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. An entropic relevance measure for stochastic conformance checking in process mining. In *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4–9, 2020*, pages 97–104. IEEE, 2020.

[29] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing complexity of large EPCs. In *MobIS*, volume 141, pages 195–207. GfI, 2008.

[30] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. Monotone precision and recall measures for comparing executions and specifications of dynamic systems. *ACM Trans. Softw. Eng. Methodol.*, 29(3):17:1–17:41, 2020.

[31] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[32] Andreas Rogge-Solti, Arik Senderovich, Matthias Weidlich, Jan Mendling, and Avigdor Gal. In log and model we trust? A generalized conformance checking framework. In *BPM 2016*, pages 179–196, 2016.

[33] Andreas Rogge-Solti, Wil M. P. van der Aalst, and Mathias Weske. Discovering stochastic petri nets with arbitrary delay distributions from event logs. In *BPM workshops 2013*, pages 15–27, 2013.

[34] Michael Rosemann and Jan vom Brocke. The six core elements of business process management. In *Handbook on Business Process Management 1, 2nd Ed.*, pages 105–122. Springer, 2015.

[35] Anne Rozinat. *Process mining: conformance and extension*. PhD thesis, Eindhoven University of Technology, 2010.

[36] Scott Sonenshein. Routines and creativity: From dualism to duality. *Organization Science*, 27(3):739–758, 2016.

[37] Suriadi Suriadi, Robert Andrews, Arthur ter Hofstede, and Moe Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *IS*, 64:132–150, 2017.

[38] Niek Tax, Xixi Lu, Natalia Sidorova, Dirk Fahland, and Wil M. P. van der Aalst. The imprecisions of precision measures in process mining. *Information Processing Letters*, 135:1–8, 2018.

[39] Wil M. P. van der Aalst. Business process management: A comprehensive survey. *ISRN Software Engineering*, pages 1–37, 2013.

[40] Wil M. P. van der Aalst. *Process Mining—Data Science in Action, 2nd Ed.* Springer, 2016.

[41] Wil M. P. van der Aalst. Relating process models and event logs-21 conformance propositions. In *ATAED*, volume 2115, pages 56–74. CEUR, 2018.

[42] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *DMKD*, 2(2):182–192, 2012.

[43] Boudewijn van Dongen, Ana Karla A. de Medeiros, H. Verbeek, A. Weijters, and Wil van der Aalst. The ProM framework: A new era in process mining tool support. In *Petri Nets*, 2005.

[44] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines-part I. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7), 2005.

[45] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 3nd edition*. Springer, 2019.

[46] Sangyoon Yi, Thorbjørn Knudsen, and Markus C. Becker. Inertia in routines: A hidden source of organizational variation. *Organization Science*, 27(3):782–800, 2016.

Appendix A. Translation

Appendix A.1. Translating Event Logs to SDFAs

In the implementation used for this paper, we translate an event log into an SDFA by incrementally constructing a deterministic finite automaton. For each trace, we start in the initial state of the SDFA and for each event in the trace, we follow the existing corresponding transition of the SDFA, or add a new transition and state if no transition existed yet. Thus, the SDFA will exhibit a tree structure. While constructing the automaton, for each state, we count the number of times each transition in the automaton is used in the event log, and we count the number of times a trace ended. As a last step, we normalise these counters to obtain the probabilities to obtain an SDFA.

There are no restrictions on event logs, that is, each event log can be translated into a corresponding SDFA.

Appendix A.2. Translating Stochastic Petri Nets to SDFAs

Appendix A.2.1. Termination

To establish the language of a stochastic Petri net, it is firstly necessary to define a notion of termination.

SPNs do not possess an explicit notion of probabilistic termination, that is, they can only deterministically terminate. However, a final marking M can be added to any SPN by adding a silent transition with the desired weight/waiting time properties, such that this transition takes tokens from all places in M, has inhibitor arcs to all other places, and produces a token in a newly-added place, causing the SPN to deadlock.

Thus, in this paper, we assume that an SPN can only terminate if no more transitions are enabled (deadlock).

Appendix A.2.2. Translation

Transitions in the SPN might have a label attached, and firing a transition will execute the activity corresponding to the label. For labelled transitions, we assume the standard semantics of SPNs, which straightforwardly lead to SDFAs, where each state of the SDFA represents a particular marking in the SPN.

For unlabelled transitions, that is, transitions without a label attached, *silent transitions*, a more involved translation is necessary. That is, from each reachable marking M, any silent transitions are exhaustively followed, while keeping track of the chains of silent transitions that lead to either termination or labelled transitions. These paths then determine the probability of each labelled transition or termination, that is, the probabilities of all paths leading to a certain transition are added. However, as an SDFA must be deterministic, only the shortest path is considered for marking purposes.

Transition labels are not necessarily unique, that is, several transitions might have the same label. The translation described is possible if at any reachable state of the SPN, there do not exist two non-equal fireable chains of unlabelled transitions such that at the end of each chain a transition with the same label is enabled. Furthermore, in some cases, choices between silent transitions might lead to non-deterministic behaviour.