Author/s:
Alkhammash, H;Polyvyanyy, A;Moffat, A;García-Bañuelos, L

Title:
Entropic relevance: A mechanism for measuring stochastic process models discovered from event data

Date:
2022

Citation:
Alkhammash, H., Polyvyanyy, A., Moffat, A. & García-Bañuelos, L. (2022). Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. Information Systems, 107, https://doi.org/10.1016/j.is.2021.101922.

Persistent Link:
https://hdl.handle.net/11343/292014

Entropic Relevance: A Mechanism for Measuring Stochastic Process Models Discovered From Event Data

Hanan Alkhammash
The University of Melbourne
halkhammash@student.unimelb.edu.au

Artem Polyvyanyy
The University of Melbourne
artem.polyvyanyy@unimelb.edu.au

Alistair Moffat
The University of Melbourne
ammoffat@unimelb.edu.au

Luciano García-Bañuelos
Tecnológico de Monterrey
luciano.garcia@tec.mx

Friday 19th November, 2021

Abstract

There are many fields of computing in which having access to large volumes of data allows very precise models to be developed. For example, machine learning employs a range of algorithms that deliver important insights based on analysis of data resources. Similarly, process mining develops algorithms that use event data induced by real-world processes to support the modeling of – and hence understanding and long-term improvement of – those processes.

In process mining, the quality of the learned process models is assessed using conformance checking techniques, which measure how well the models represent and generalize the data. This article presents the *entropic relevance* measure for conformance checking of stochastic process models, which are models that also provide information in regard to the likelihood of observing each sequence of observed events. Accurate stochastic conformance measurement allows identification of models that describe the data better, including the captured sequences of process events and their frequencies, with information about the likelihood of the described processes being an essential step toward simulating and forecasting future processes.

Entropic relevance represents a blend between the traditional precision and recall quality criteria in conformance checking, in that it both penalizes observed processes that the model does not describe, and also penalizes processes that are permitted by the model yet were not observed. Entropic relevance can be computed in time linear in the size of the input data; and measures a fundamentally different phenomenon than other existing measures. Our evaluation over industrial datasets confirms the feasibility of using the measure in practice.

Keywords: Process mining, conformance checking, stochastic conformance checking, model inference, model quality, process learning, explainable AI

Postprint, November 2021

1 Introduction

The research discipline of *process mining* studies methods, techniques, and tools of inference from data to tackle problems associated with the discovery, understanding, improvement, and automation of real-world processes [35]. To achieve that goal, process mining makes use of techniques from data mining and machine learning, and combines them with process modeling and analytical expertise from business process management. One of the main problems investigated in process mining is *process discovery*, which studies algorithms for automatically constructing *process models* from *event data* generated by IT-systems. Such event data is often recorded in an *event log* – a collection of *traces*, each capturing a sequence of observed process *events*. Each single event in a trace usually incorporates information about a *case identifier* that allows amalgamation of the events that relate to the same process instance to form that trace; the time of the event's occurrence (the *timestamp*) so that event order can be known both within and between traces; and a description or label of the process *action* that induced the event occurrence. There might also be additional payload fields that further characterize the event.

A process model automatically discovered from an event log should faithfully encode the traces contained in the log. Four fundamental *quality criteria* have been proposed as ways of assessing this requirement.

(1) A discovered model should describe as many as possible of the log's traces (that is, have good *recall*); this criterion is also known as *fitness* in process mining.

(2) A discovered model should allow as few traces as possible that are not present in the log (that is, have good *precision*).

(3) A discovered model should possess "*simplicity*" in some quantifiable sense.

(4) Finally, a discovered model should also permit traces that may stem from the same process but are not present in the particular log used to construct the model (that is, have good *generalization*).

That is, a learned model of processes in process mining aims to explain the actions and decision points in the process. The subarea of *conformance checking* in process mining studies the problem of measuring and characterizing various quality criteria when using an automatically discovered or manually created process model to explain the corresponding event log [7, 35].

Inferring a model that describes and predicts a set of traces is, in essence, the task of understanding and extracting the log's frequent patterns. As frequent patterns can be used to compress data, a model that can be used to compress the data into a shorter lossless representation captures more of its frequent patterns and, thus, has learned the data better, and can make more accurate predictions. In particular, the minimum description length (MDL) model selection principle states that the model that produces the shortest representation of some data is

the model that provides the best description of the data. Note, however, that the Kolmogorov complexity of a data (i.e., its theoretic minimum description length) cannot, in general, be computed [9].

At a certain abstraction level, events can be identified based on the actions they were induced by and the ordering of those actions, while ignoring their exact timestamps, case identifiers, and payload. This allows an event log to be interpreted as a multiset of traces. Two occurrences of the same trace in such a multiset denote two process instances induced by the same sequence of action occurrences. A multiset of traces, in turn, can be characterized by a *random variable* (a function that maps traces to the real numbers between zero and one), each estimating the likelihood of observing the corresponding trace in the log. Such a random variable is often referred to as a *stochastic language* in linguistics, mathematics, and computer science [13].

This article presents a *relevance* measure that employs the MDL principle to assign a numeric quantity to each possible stochastic process model that might be used to describe a given event log. Those numeric values can then be used as the basis for choosing between alternative stochastic process models that are regarded as being competitors for describing the log.

Several approaches for estimating the stochastic perspective of automatically discovered process models have been proposed [29, 5]. In addition, stochastic process models are designed manually by organizations, for instance, to estimate the future workforce or to enable mechanisms to manage the complexity of designed models [27, 26]. Finally, process models discovered using commercial tools convey information about frequencies of action occurrences and, thus, can be interpreted as stochastic process models [35]. It is important to measure not only how well a model represents the various traces from the event log, but also the frequencies of their occurrences. This ability is essential, for instance, for simulating and predicting future processes [33, 23, 16].

The MDL of a random variable is equal to its entropy [9], and a stochastic process model that both assigns to each described trace the probability with which it appears in the log and also assigns a zero probability to every trace not recorded in the log, is the best-fitting model for that log. But, in this case, the model might be complex and over-fitted relative to the data. It is thus of practical interest to be able to develop compromise models that are simpler, yet provide behavior that is nearly as good. In such models, the valid traces and the probabilities of the traces deviate from those captured in the log. In addition, some traces from the log might not be able to be "replayed" by the model (i.e., their likelihood according to the model is zero). To address that difficulty, our approach also includes a *background coding model* that allows representing log traces impossible according to the model. The *relevance of a stochastic process model (with respect) to an event log* is then the average length of a lossless description of the log's traces, either by using the process model (if that trace is described by the model), or by using the background coding model (if it is not), plus an allowance for an indicator for each trace specifying which of the two alternatives is being used. Relevance is measured in bits per trace; and since it is grounded in the entropy of a stochastic language, we refer to it as *entropic*

relevance.

This article is an extended version of our conference paper [25]. The conference paper made these contributions:

- It applied the MDL model selection principle to solve the problem of *stochastic conformance checking* (checking how well a stochastic process model explains an event log);
- It introduced the notion of *entropic relevance* for stochastic conformance checking that implements the MDL model selection principle; and
- It demonstrated the feasibility of measuring entropic relevance of models discovered from industrial logs.

This article makes these extensions to the original conference paper:

- It uses empirical evidence across a broad range of available datasets to demonstrate that entropic relevance is fundamentally different from other current measures for stochastic conformance checking (i.e., entropic relevance measures a fundamentally different phenomenon than the existing measures);
- It argues that entropic relevance implements a compromise between the precision and recall quality criteria; and
- It explores alternative background coding models for representing event logs and introduces two such models that may result in more compact entropic relevance representations of event logs while aiming to approach their theoretic minimum description length.

Therefore, this article strengthens the theoretical foundations of the entropic relevance measure for stochastic conformance checking.

The remainder of the article proceeds as follows. The next section presents an overview of the approach for measuring the entropic relevance of a stochastic process model to an event log. Section 3 discusses formal models of stochastic languages used in process mining. Based on these models, Section 4 presents the notion of entropic relevance and three alternative background coding models for log traces. Section 5 presents our evaluation results, which include evidence that entropic relevance is different from all current measures for stochastic conformance checking. Section 6 discusses the broader context of our contributions, including the argument that entropic relevance incorporates elements of both the precision and the recall quality criteria, while Section 7 summarizes related work and describes the overall framework to which our mechanism contributes. Finally, Section 8 presents conclusions and lists areas for future work.

2 Motivation and Overview

As anticipated in the Introduction, we employ a minimum description length (MDL) compression-based framework to measure the quality of process models. The two key observations that make this possible are that a good process model is one which accurately describes an observed set of traces, taken as a sample from an underlying universe of traces; and that the "describes" operation can be precisely quantified by assessing the cost of compressing the set of traces relative

to the stochastic language expressed by the model. A singular benefit of this *entropic relevance* approach is that not only is the model *structure* an influence on its measured usefulness, but so too is the *probability* of each individual trace having emerged from the model.

2.1 Measuring Information Content

A relationship fundamental to information theory is critical to understanding the new approach. Suppose, in some context, that a universe of *symbols* is known to be possible, and that associated with each is a probability of occurrence, denoted by $p(\cdot)$, with the probabilities over the set of possible symbols summing to one (that is, we are certain that *something* will occur next, but not *what*). Suppose further that it turns out that the particular symbol that does occur is e, one member of the universe of possibilities, and that its probability is thus $p(e)$. Then information theory tells us that the *information* conveyed by that occurrence of e in that context is $-\log_2 p(e)$ bits, and, hence, that is also the number of bits that should, in an ideal code, be used to describe this instance of e, and any recurrence of e that might also arise – prior to or subsequent to this one – in that same underlying context.

For example, if e has probability $p(e) = 0.8$ within a stream of symbols and all symbols occur independently of each other (that is, all of the symbols in the stream are regarded as having the same context), then each e that occurs has a cost of 0.3219 bits attributable to it. Conversely, the collective set of "not e" symbols, which have a combined probability of 0.2, one quarter as large, should be indicated by a distinguishing code of $-\log_2 0.2 = 2.3219$ bits, plus whatever the cost is of isolating one of those symbols in the reduced context, employing a set of scaled conditional probabilities in which it is known that the next symbol is definitely not an e. Practical compression systems operate very close to these ideal entropy-based limits, see, for example, Moffat and Turpin [22, Chapter 5]; and within small additive constants, the information-theoretic relationship between probabilities and bits is an achievable one.

2.2 Models for Compression

Fig. 1 further motivates the proposed system of measurement. In the figure it is supposed that an event log has been provided, sampled from an underlying "true" (but unknown) process, and that two stochastic process models A and B are being considered as alternative explanations for that set of observations. Further, suppose that stochastic process model M assigns a probability of $p(t \mid M)$ to each trace t that appears in the log, where $p(t \mid M)$ is the overall probability estimate that emerges from M when the sequence t is traced, and is typically computed as a product of the probabilities of the steps and transitions through the model M that resulted in t being matched.

To compare stochastic process models A and B, we first take M to be A, and (in the left side of Fig. 1), compute the summation $\sum_t -\log_2 p(t \mid \mathsf{A})$, with the summation being over all traces t in the log. That summation yields the net

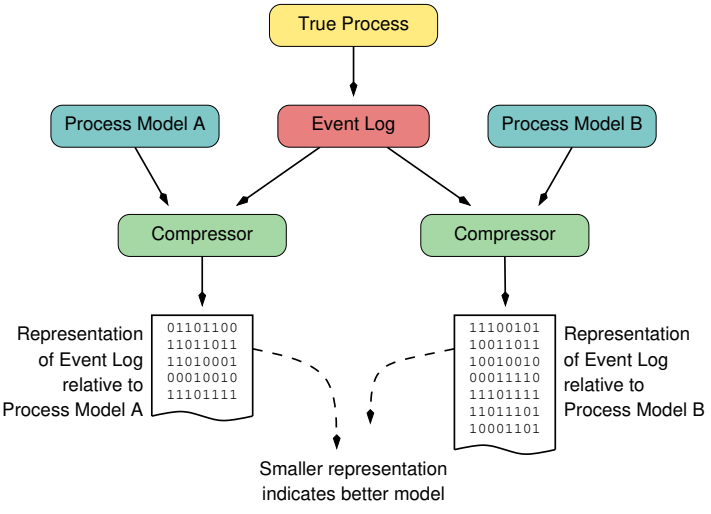


Figure 1: To measure the entropic relevance of a process model to a collection of traces, the model's structure and probabilities are used to compute the cost in bits of losslessly representing the traces relative to the model. Better models lead to shorter compressed forms.

compression cost of representing the complete set of traces in the log using the trace probabilities embedded in stochastic model A. Repeating the calculation, but now using trace probabilities that are derived from stochastic model B (the right path in Fig. 1) and computing $\sum_t -\log_2 p(t \mid \mathsf{B})$ will similarly result in a compression cost being assigned to the log, derived from the trace probabilities expressed by stochastic model B.

The smaller the compression cost, the shorter the compressed output would be were it to be generated, and the better the corresponding model matches the log. For example, in Fig. 1, it is intimated that stochastic model A gives rise to a smaller information-theoretic cost of a compressed representation of the log, suggesting that model A better describes the process underlying that particular log than does model B. Note that while Fig. 1 draws output bits and suggests that actual compression occurs, it is the *size* of the output that is of interest and not the precise bits that arise. That size can be computed by summing logarithms of model-conditioned trace probabilities, that is, by assuming that an ideal entropy coder is available without generating – and then measuring the length of, and then discarding – an actual bitstream.

2.3 Background Model

Fig. 2 provides more details of the simulated compression process, considering the traces in the log one by one. The stochastic process model (defined in detail in Section 3) assigns calculable non-zero probabilities to a finite or infinite subset

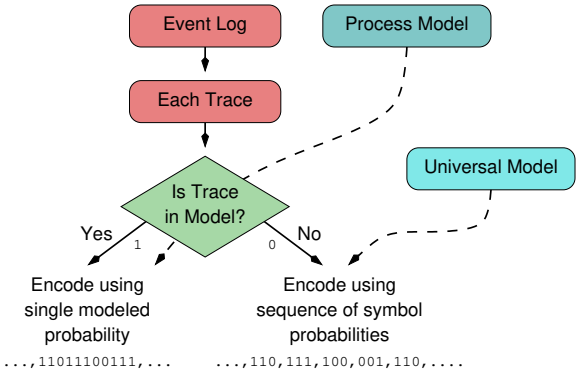


Figure 2: The two possible options when encoding a set of traces with respect to a probabilistic model: a trace either has a non-zero probability in the model, which can be used by an ideal entropy coder to derive a bitstring, or it is spelled-out as symbol-by-symbol codes via a universal background model. The choice between those two alternatives must also be accounted for as part of the compression cost, as a single (but possibly biased) additional bit.

of the universe of possible traces. That is, the stochastic model M assigns a probability $p(t \mid M) > 0$ to some subset of the log's traces, but might not be able to describe all of the log's traces – indeed, in an extreme case, it might not represent *any* of the log's traces. Should such cases arise, we will have $p(t \mid M) = 0$ for at least one trace in the log, and the information-theoretic relationship between probabilities and compression costs breaks down. The branching point in Fig. 2 shows the two distinct pathways that we provide to handle this dilemma.

If some particular trace in the log fits the model, it can be represented as single entity, using its corresponding end-to-end probability in the model. That option is shown by the left-hand path in Fig. 2. The cost incurred includes a preliminary *selector* to convey, in essence, that "what is coming next represents a code for a trace that has a non-zero probability in the model"; that selector bit is shown in the figure as a "1" (see next to the branching diamond), and its cost is in addition to the information-theoretic cost of coding the whole trace derived from the model's assessment of that trace's probability. Note that the selector might be coded as a whole bit, as is illustrated in the figure. But if the selector probability is imbalanced – if, say, the great majority of traces in the log, in fact, *are* matched to non-zero probabilities by the model – then the selector might take only a fraction of a bit, because its corresponding probability might be close to one.

The right-hand path in Fig. 2 shows what happens if a trace is given a zero probability by the stochastic model M. In this case, the selector bit is shown as a "0", but again might be less than or more than one bit, depending on the probability of that branch being followed when measured across all of the traces in the log. Traces that have $p(t \mid M) = 0$ are then represented as a sequence of

elemental symbols using a universal *background* model in which every possible symbol in the alphabet from which traces are composed always has a non-zero probability. That is, the background model provides a "fallback" facility in which every possible sequence of states can always be coded.

As already noted, to choose between these two cases, the output associated with every trace is prefixed by a code – for instance, a biased "0" or "1" bit – that indicates which option applies. In the corresponding decoder, the selector is decoded first and used to determine which path must be followed next: whether to decode a single code relative to the set of trace probabilities relative to the model M or whether to decode a trace from the bitstream relative to the rules of the background model. Either way, an ideal entropy decoder, if it were to be provided with the same stochastic and background models as were used by the hypothesized encoder, charged with decoding the bitstream emitted by the encoder, would be able to exactly (losslessly) reconstruct the original log by applying the reverse procedure. We reiterate that no bits are actually generated in the measurement of entropic relevance. The important point is that the "compression" process, while hypothetical, could nevertheless be realized into a decodeable bitstream only slightly longer than the indicated length if, for some reason, it were to become desirable to do so. In particular, entropic relevance is computed from the *length* of that compressed representation of the log, and is not dependent on the exact bitstream that might be generated by a particular coding process.

Finally, note that it is not the case that $p(t \mid M) > 0$ means that $-\log_2 p(t \mid M)$ must be less than the cost of using the background model. For example, some trace t might have been assigned a probability of 10^{-100} by the model and, hence, correspond to a coded length (the left path in Fig. 2) of 332 bits. But when coded using the background model (the right path), that same trace might (say) only require 54 bits, perhaps a selector needing four bits, followed by ten symbols each of five bits. Note, however, that should such a mismatch arise, it reflects poorly on the quality of the model and its aptness in terms of representing the traces contained in the log. Hence, it may be appropriate for the computed entropic relevance score to be high. That is, a background model is required to exist, but it is not required to be the most expensive way of representing any particular trace; it is simply an alternative way that bypasses the structure defined by the stochastic model, and handles cases for which the model's structure indicates that some trace is impossible.

2.4 The Sum of the Parts

Fig. 3 steps back from the detail and provides a high-level view of the proposed mechanism, the exact details of which will be presented in Section 4. There are four components that collectively sum to the compressed size, and that vary in different ways as the process model changes. At the left end of Fig. 3, if the stochastic process model M is small and easily described, it likely fits only a small fraction of the log's traces. If that is indeed what happens, the majority of traces do not fit the model, and hence are represented using the (probably,

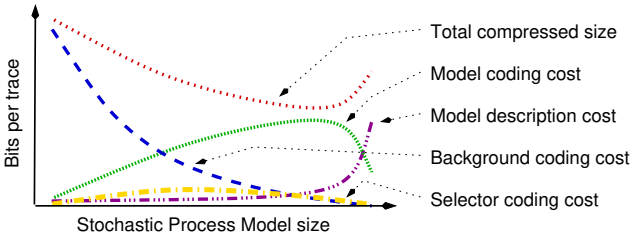


Figure 3: A schematic plot showing the total compressed size of a collection of traces relative to a model as the sum of four components: the cost of describing the model and its parameters (model description cost); the cost of entropy coding the traces that fit the model (model coding cost); the cost of entropy coding the traces that do not fit the model, using a catch-all background technique (background coding cost); and the cost of selecting, for each trace, which approach is used to code it (selector coding cost).

but not certainly) more expensive background model, and may dominate the total output size (that is, cost). Moving across the horizontal axis, as the process model becomes larger and presumably more sophisticated, it is likely to fit a greater fraction of the traces, and the balance shifts from the background model to (what it is hoped is) the more economical stochastic process model. The total compressed size can be expected to decrease as this transition takes place. Throughout this normal operating range, the contributions of the other two factors – the binary per-trace selector flag, and the description of the process model – are typically very small overheads.

In the limit, at the right of Fig. 3, the process model becomes large and is likely to be over-fitted to the traces in the particular log, with a unique pathway specified for every distinct trace. The cost of using an over-fitted model is low, since each pathway through it is unique; the cost of the background model is also low, since no sequences need to be processed via it; and there is no cost involved in selecting between the stochastic model and the background model. However, the total compressed size may be increased because of the complexity and detail required in the description of the process model itself.

In terms of Fig. 3, entropic relevance is calculated as the sum of the selector coding cost, the background coding cost, and the model coding cost. It is useful to retain the model size (measured in some conventional manner, rather than as the length of a compressed representation) as a second dimension, as is shown by the horizontal axis in the plot. Furthermore, since process models are discrete objects (rather than a continuous phenomena), the separation between entropic relevance and model size allows definition of a Pareto frontier, i.e., the set of models that are either smaller in size or superior in terms of entropic relevance to other possible models. Section 4 provides precise definitions of all of these ideas. Our purpose in this section is primarily to give an overall motivation for and perspective across the new approach, rather than immediately presenting the full details.

2.5 Process Mining Goals

The majority of process model discovery algorithms attempt to minimize some subset of a defined range of quality criteria, often either or both of precision and recall. That is, they construct their models relative to, and targeting, the problem they were designed to address, perhaps via some algorithmic parameter that determines a balance between model size and trace coverage in the log. The mechanism used to discover a process model similarly exerts influence on the compressed size in the way we are computing it here. For example, an algorithm could construct a process model of a large size, high generalization, but low precision, leading to an increased compressed size for the log. Similarly, a small-sized model might have low generalization and high precision. The stylized trajectories presented in Fig. 3 show that such a model might also have high entropic relevance, and that there might be a strategic balance-point to be identified across the range of algorithmic parameters that minimizes entropic relevance. Also to be noted in connection with Fig. 3 is that throughout our discussion we assume that a "reasonable" (rather than pathological in some way) process discovery algorithm is being used to construct the models.

Entropic relevance (again, in anticipation of Section 4) is measured in "bits per trace", with small values being preferable to large ones. The numeric range is open-ended, and it is neither desirable nor possible to normalize the measurement in any way to obtain a "0 to 1" range. Instead, entropic relevance has meaningful units that clearly indicate the complexity of the process that is being represented by the model. With that understanding established, the process mining desiderata listed in the Introduction can be considered: (1) traces not covered by the model must be coded using the background predictions, increasing the net bit cost; (2) processes permitted by the process model but not present in the log cause the imputed probabilities of traces that do occur to decrease, again increasing the net bit cost; and (3) simple models have smaller model description costs, decreasing the net bit cost. Finally, objective (4) can also be accounted for by noting that the background model is always available, so hitherto unseen traces can be accommodated, albeit with likely increased net bit costs.

Section 4 also provides details of how the background models might be constructed so that every possible trace can be accommodated. The simplest background model assigns every possible symbol a uniform probability that is the reciprocal of the number of distinct symbols (a "flower" model); that is sufficient to allow every possible sequence. More sophisticated background models might also take into account symbol probabilities, albeit with the added burden of needing to describe (and hence "pay for") those probabilities. However, in the limit, if the background model is made too sophisticated, there is a risk of it no longer being genuinely universal, and there is a sense of "decreasing returns" that must also be allowed for. Section 4 describes three approaches to constructing useful background models.

3 Models of Stochastic Languages

This section introduces the ideas and notation that we employ to describe models of stochastic languages.

3.1 Sequences and Multisets

A *sequence* is an ordered list of elements in which repetitions of elements are allowed. Let Λ be a finite non-empty set of elements. Then, $\sigma \in \Lambda^*$ is a finite sequence over Λ, where Λ^* is the set of all finite sequences over Λ including the empty sequence. For example, $\sigma_1 = \langle \mathtt{a},\mathtt{d},\mathtt{d}\rangle$, $\sigma_2 = \langle \mathtt{e},\mathtt{d}\rangle$, and $\sigma_3 = \langle \mathtt{b},\mathtt{b},\mathtt{e},\mathtt{d},\mathtt{a}\rangle$ are sequences over set $\Lambda = \{\mathtt{a}, \mathtt{d}, \mathtt{b}, \mathtt{f}, \mathtt{e}\}$. When the context is clear, we write sequences as strings, i.e., angle brackets and commas are omitted. For instance, σ_1 is written as $\mathtt{add}$. We write $\epsilon \in \Lambda^*$, to denote the empty sequence.

The length of sequence σ is denoted by $|\sigma|$. Sequence σ_3 contains five elements, i.e., $|\sigma_3| = 5$, whereas the length of the empty sequence is equal to zero, i.e., $|\epsilon| = 0$. By $\sigma(i)$, $i \in [1\,..\,|\sigma|]$, we denote the element at position i in sequence σ. For instance, $\sigma_1(3) = \mathtt{d}$, $\sigma_2(1) = \mathtt{e}$, and $\sigma_3(5) = \mathtt{a}$. By $u(\sigma)$, we denote the set of all the elements that occur in σ, i.e., $u(\sigma) = \{\lambda \in \Lambda \mid \exists\, i \in [1\,..\,|\sigma|] : \sigma(i) = \lambda\}$. For example, it holds that $u(\sigma_1) = \{\mathtt{a}, \mathtt{d}\}$. Given two sequences σ_i, $\sigma_j \in \Lambda^*$, the *concatenation* of σ_i and σ_j, denoted by $\sigma_i \circ \sigma_j$, is obtained by appending σ_j to the end of σ_i. For example, it holds that $\sigma_1 \circ \sigma_2 = \mathtt{added}$.

A *multiset* is a collection of elements in which an element can appear multiple times. Let B be a multiset of finite sequences over set Λ. We write $|B|$ to denote the cardinality of multiset B. By $n(\sigma, B)$, $\sigma \in \Lambda^*$, we denote the multiplicity of σ in B. Consider multiset $B_1 = [\mathtt{bbeda}^3, \mathtt{aaa}^4, \mathtt{deae}^2, \mathtt{add}, \mathtt{be}^4, \mathtt{ed}, \mathtt{ddeaae}^6]$. Sequence $\mathtt{ddeaae}$ occurs in B_1 six times, i.e., $n(\mathtt{ddeaae}, B_1) = 6$, while $\mathtt{add}$ occurs once, i.e., $n(\mathtt{add}, B_1) = 1$. Multiset B_1 contains 21 sequences, i.e., $|B_1| = 21$. We use $u(B)$ to denote the set of unique elements in sequences contained in B, that is $u(B) = \bigcup_{\sigma \in B} u(\sigma)$, $u(B) \subseteq \Lambda$. For example, $u(B_1) = \{\mathtt{a},\mathtt{b},\mathtt{d},\mathtt{e}\}$.

We denote the number of occurrences of element $\lambda \in \Lambda$ in sequence $\sigma \in \Lambda^*$ by $n(\lambda, \sigma)$. For instance, $n(\mathtt{d}, \sigma_1) = 2$ and $n(\mathtt{d}, \sigma_2) = 1$. We write $n(\lambda, B)$ to refer to the number of occurrences of element λ in multiset of sequences B, that is, $n(\lambda, B) = \sum_{\sigma \in B} (n(\lambda, \sigma)\, n(\sigma, B))$. Similarly, $n(B)$ denotes the number of elements in sequences contained in B, such that $n(B) = \sum_{\lambda \in u(B)} n(\lambda, B)$. For example, element $\mathtt{a}$ occurs 30 times in B_1, that is, $n(\mathtt{a}, B_1) = 30$; element $\mathtt{b}$ appears 10 times in B_1, it holds that $n(\mathtt{b}, B_1) = 10$, while $n(B_1) = 84$.

Consider a random variable X whose values are the outcomes of randomly selecting a sequence from multiset B, such that each occurrence of a sequence in the multiset has the same chance of being selected. Then, the probability of observing X taking on the value equal to sequence σ is denoted by $P(X = \sigma \mid B)$, or $P(\sigma \mid B)$ for short. Let B be non-empty, we use the *maximum likelihood algorithm* [21] for estimating $P(\sigma \mid B)$, such that:

$$P(\sigma \mid B) = \frac{n(\sigma, B)}{|B|}.$$

Similarly, the probability $P(\lambda \mid B)$ of observing element $\lambda \in \Lambda$ in B is estimated to be (assuming B contains at least one non-empty sequence):

$$P(\lambda \mid B) = \frac{n(\lambda, B)}{n(B)}.$$

For example, probability $P(\texttt{be} \mid B_1)$ is estimated to be $4/21$, and the probability of observing element $\texttt{b} \in \Lambda$ in B_1, that is, $P(\texttt{b} \mid B_1)$, is estimated to be $5/42$.

3.2 Stochastic Languages

A *language* is a (possibly infinite) set of *words*. A word is usually formalized as a finite sequence of *symbols*. We use words to encode observed processes. Therefore, we refer to words as (process) *traces* composed of *actions* rather than as words of symbols. Let V be a universe of *actions*. A language W over V is a subset of V^*, i.e., $W \subseteq V^*$. For example, set $W = \{\texttt{ab}, \texttt{ae}, \texttt{abcd}, \texttt{abc}, \texttt{aee}, \texttt{abcde}\}$ defines a language of six traces.

Definition 3.1 (Stochastic language)
A *stochastic language* is a probability density function over traces in V^*, specified as a function $L : V^* \to [0, 1]$, for which it holds that:

$$\sum_{t \in V^*} L(t) = 1.$$

⌟

That is, a stochastic language is an assignment of probabilities to traces so that the assigned probabilities sum up to one. A stochastic language might be used to encode the relative likelihoods of observing words in a book, or encountering traces in an event log of a software system. For example, $L_1 = \{(\texttt{ab}, 0.6), (\texttt{ae}, 0.15), (\texttt{abcd}, 0.11), (\texttt{abc}, 0.05), (\texttt{aee}, 0.05), (\texttt{abcde}, 0.04)\} \cup \bigcup_{t \in V^* \setminus W} \{(t, 0)\}$, where set W is specified above, is a stochastic language.

Given a trace $t \in V^*$ and a stochastic language L, $L(t)$ specifies the relative likelihood of a randomly drawn trace to be equal to t. The support of L, written $\bar{L}$, is the set of all *possible traces* according to L, i.e., $\bar{L} := \{t \in V^* \mid L(t) > 0\}$; $V^* \setminus \bar{L}$ are the *impossible traces* according to L. We say that L is *finite* if and only if $\bar{L}$ is finite; otherwise L is *infinite*. It holds that $\bar{L}_1 = W$, i.e., the traces in W are all the possible traces according to L_1 and, thus, L_1 is finite.

3.3 Stochastic Deterministic Finite Automata

A stochastic deterministic finite automaton (SDFA) can be used to encode a stochastic language; here we adopt the definition given by Carrasco [8].

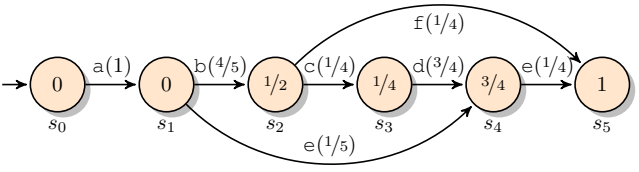


Figure 4: SDFA A_1.

Definition 3.2 (Stochastic deterministic finite automaton) A *stochastic deterministic finite automaton* (SDFA) $A = (S, Q, \delta, p, s_0)$ consists of a finite set of *states* S, with $s_0 \in S$ the *initial state*, a set of *actions* $Q \subseteq V$, a *transition function* $\delta : S \times Q \to S$, and a *transition probability function* $p : S \times Q \to [0, 1]$ such that for each state $s \in S$ holds that $\sum_{q \in Q} p(s, q) \leq 1$. ⌟

We use $\mathcal{A}$ to denote the universe of SDFAs.

Fig. 4 shows an SDFA using graphical notation. In this representation, the states and transition function are visualized as circles and arcs, respectively. The SDFA in the figure has six states s_0 through s_5, and its transition function is defined by $\{(s_0, \mathtt{a}, s_1), (s_1, \mathtt{b}, s_2), (s_1, \mathtt{e}, s_4), (s_2, \mathtt{c}, s_3), (s_2, \mathtt{f}, s_5), (s_3, \mathtt{d}, s_4), (s_4, \mathtt{e}, s_5)\}$. Arcs are labeled by actions and transition probabilities. For example, the arc from state s_0 to state s_1 with label "$\mathtt{a}(1)$" specifies that $(s_0, \mathtt{a}, s_1) \in \delta$ and $(s_0, \mathtt{a}, 1) \in p$. The transition probability function is defined by $\{(s_0, \mathtt{a}, 1), (s_1, \mathtt{b}, 4/5), (s_1, \mathtt{e}, 1/5), (s_2, \mathtt{c}, 1/4), (s_2, \mathtt{f}, 1/4), (s_3, \mathtt{d}, 3/4), (s_4, \mathtt{e}, 1/4)\}$. State s_0 is the initial state and is denoted by an arrow leading to it.

An SDFA $A = (S, Q, \delta, p, s_0)$ encodes stochastic language L_A defined using recursive function $\pi_A : S \times V^* \to [0, 1]$, i.e., $L_A(t) = \pi_A(s_0, t)$, $t \in V^*$, where:

$$\pi_A(s, \epsilon) = 1 - \sum_{q \in Q} p(s, q), \text{ and}$$
$$\pi_A(s, q \circ t') = p(s, q) \times \pi_A(\delta(s, q), t'), \text{ for } q \in V \text{ and } t = q \circ t'.$$

Note that $\pi_A(s, \epsilon)$ denotes the probability of terminating a trace in state s of A. Such probabilities are shown diagrammatically as labels inside of the corresponding states. For example, for SDFA A from Fig. 4, the set of terminating states is $\{s_2, s_3, s_4, s_5\}$, where $\pi_A(s_2, \epsilon) = 1/2$, $\pi_A(s_3, \epsilon) = 1/4$, $\pi_A(s_4, \epsilon) = 3/4$, and $\pi_A(s_5, \epsilon) = 1$. If L is a stochastic language encoded by some SDFA, we say that L is *regular*. The SDFA in Fig. 4 encodes stochastic language that assigns non-zero probabilities to seven traces: $\{(\mathtt{ab}, 2/5), (\mathtt{abc}, 1/20), (\mathtt{abcd}, 9/80), (\mathtt{abcde}, 3/80), (\mathtt{abf}, 1/5), (\mathtt{ae}, 3/20), (\mathtt{aee}, 1/20)\}$.

3.4 Event Logs

An *event log* is a finite collection of traces. A *trace* represents a finite sequence of *events*, each describing execution of an action from a particular process. Events are distinguished by their attributes and attribute values. In this work,

we are neither interested in the exact times of event occurrences (but only in their orderings), nor in the distinctions between events and actions. Thus, we encode all the events that refer to the same case identifier as a trace of actions (obtained via the action identifier attribute), where the actions are arranged in the ascending order of timestamps of the corresponding events. As there can be several case identifiers that induce the same trace (indeed, several processes can induce the same sequence of actions with different timestamps), for our needs, it is convenient to represent an event log as a multiset of traces.

Definition 3.3 (Event log)
An *event log*, or *log*, is a finite multiset of traces. ⌟

By $\mathcal{E}$, we denote the universe of logs. For example, $E_1 = [\mathtt{ab}^{1200}, \mathtt{ae}^{300}, \mathtt{abcd}^{220}, \mathtt{abc}^{100}, \mathtt{aee}^{100}, \mathtt{abcde}^{80}]$ and $E_2 = [\epsilon^{50}, \mathtt{ab}^{50}, \mathtt{abc}^{50}, \mathtt{aeaae}^{40}, \mathtt{aee}^{20}, \mathtt{abcd}^{10}, \mathtt{abcde}^{10}, \mathtt{abee}^{10}, \mathtt{abcff}^{10}]$ are two logs defined over $V = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$; $E_1, E_2 \in \mathcal{E}$. Trace abcd occurs 220 times in E_1, while it is recorded ten times in E_2. Let # be the *end-of-trace* symbol. Given trace t, by $\hat{t} \in \hat{V}^*$, $\hat{V} = V \cup \{\#\}$, $\# \notin V$, we refer to the #-*terminated* version of trace t, i.e., $\hat{t} = t \circ \langle \# \rangle$. We write $\hat{E}$ to denote the multiset of #-terminated traces obtained from event log E. For example, $\hat{E}_1 = [\mathtt{ab\#}^{1200}, \mathtt{ae\#}^{300}, \mathtt{abcd\#}^{220}, \mathtt{abc\#}^{100}, \mathtt{aee\#}^{100}, \mathtt{abcde\#}^{80}]$.

If a trace appears multiple times in a log, it was observed multiple times in the real-world. Hence, the log captures information about the relative frequencies with which the traces were observed. Given an event log $E \in \mathcal{E}$, we define the *stochastic language* L *of* E by assigning probability $L(t) = P(t \mid E)$ to each trace $t \in V^*$. Therefore, L_1 from Section 3.2 is the stochastic language of event log E_1. In turn, stochastic language L_2 of event log E_2 defines non-zero probabilities to traces that are specified in the set $\{(\epsilon, 1/5), (\mathtt{ab}, 1/5), (\mathtt{abc}, 1/5), (\mathtt{aeaae}, 4/25), (\mathtt{aee}, 2/25), (\mathtt{abcd}, 1/25), (\mathtt{abcde}, 1/25), (\mathtt{abee}, 1/25), (\mathtt{abcff}, 1/25)\}$.

3.5 Frequency Directed Action Graphs

A common representation of event logs for consumption and decision making by practitioners is to encode them as Directly-Follows Graphs (DFGs) [35]. A DFG of an event log E is a digraph in which vertices are actions encountered in the traces of E and edges encode the directly-follows relation over the actions, i.e., the DFG contains an edge directed from action a to action b if and only if E contains a trace $t_1 \circ \mathtt{ab} \circ t_2$, where $t_1, t_2 \in V^*$ [32].

As DFGs of industrial event logs are immense, they are often post-processed by filtering out vertices and edges that correspond, respectively, to infrequent actions and pairs of subsequent actions in the log. The vertices and edges of the filtered graphs are then annotated with numbers that reflect the frequencies of observing the corresponding concepts in the log. The frequencies aim to reflect the stochastic nature of the processes encoded in the corresponding log. To describe such filtered graphs mathematically, we introduce the notion of a *frequency directed action graph*.

Definition 3.4 (Frequency directed action graph)
A *frequency directed action graph* (FDAG) is a tuple $(\Phi, \Psi, \phi, \psi, i, o)$, where $\Phi \subseteq V$ is a set of *actions*, $\Psi \subseteq ((\Phi \times \Phi) \cup (\{i\} \times \Phi) \cup (\Phi \times \{o\}))$ is a *directly-follows relation*, $\phi : \Phi \cup \{i,o\} \to \mathbb{N}_0$ is an *action frequency function*, $\psi : \Psi \to \mathbb{N}_0$ is an *arc frequency function*, and $i \notin V$ and $o \notin V$ are the *input* and the *output* of the graph, respectively. ⌟

We use $\mathcal{G}$ to denote the universe of FDAGs.

Fig. 5 shows an example FDAG. In the figure, boxes with rounded corners represent actions, and arcs encode the directly-follows relation. The input node and the output node are denoted by i and o, respectively. The input node has no incoming arcs, while the output node has no outgoing arcs. Finally, the action and arc frequencies assigned by the corresponding frequency functions are associated adjacent to the corresponding objects.

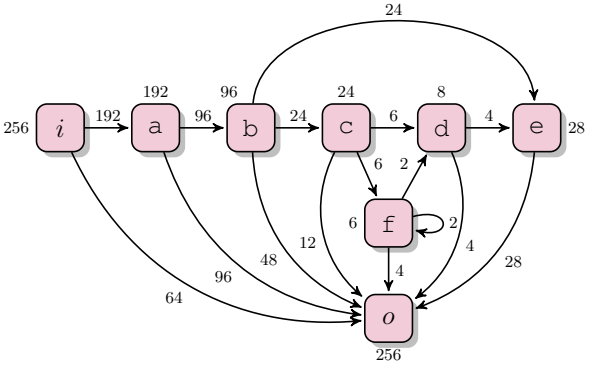


Figure 5: An FDAG.

Van der Aalst [35] observes that practitioners can interpret FDAGs in different ways. Because of the filtering step, it is possible to associate a given FDAG with different collections of traces. Van der Aalst then discusses several pitfalls that can happen in practice due to this pluralism of possible interpretations. For our purpose, we fix one such possible interpretation. To avoid ambiguity, we define this interpretation as a mapping from an FDAG to the corresponding SDFA.

Definition 3.5 (SDFA of FDAG)
Let $G := (\Phi, \Psi, \phi, \psi, i, o)$ be an FDAG. Then, (S, Q, δ, p, s_0), where $S = \Phi \cup \{i\}$, $Q = \Phi$, $\delta = \{(s,t,t) \in (\{i\} \cup \Phi) \times \Phi \times \Phi \mid (s,t) \in \Psi\}$, $p = \{(s,t,x) \in (\{i\} \cup \Phi) \times \Phi \times [0,1] \mid (s,t) \in \Psi \wedge x = \psi(s,t) / \sum_{\{(s,u) \in \Psi \mid u \in \Phi\}} \psi(s,u)\}$, and $s_o = i$, is the *SDFA of* G, denoted by *SDFA*(G). ⌟

According to this interpretation, if $A =$ *SDFA*(G), then G encodes the *possible traces* of L_A, i.e., traces $\bar{L}_A$, and the relative probability of a trace $t \in \bar{L}_A$ is

given by $L_A(t)$. For example, Fig. 6 shows the SDFA of the FDAG from Fig. 5 that encodes infinite stochastic language which, among others, assigns these non-zero values to traces $\{(\epsilon, 1/4), (\mathtt{a}, 3/8), (\mathtt{ab}, 3/16), (\mathtt{abc}, 3/64), (\mathtt{abcf}, 3/256), (\mathtt{abcff}, 3/1024), (\mathtt{abe}, 3/32), (\mathtt{abcfd}, 3/1024), (\mathtt{abcd}, 3/256), (\mathtt{abcde}, 3/256), (\mathtt{abcfde}, 3/1024)\}$; all the other possible traces of this SDFA have, in aggregate, a combined probability of $3/1024$.

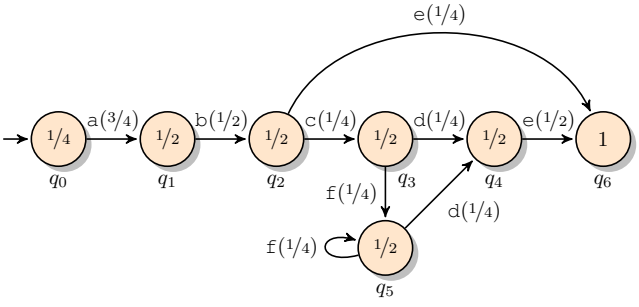


Figure 6: SDFA A_2.

4 Entropic Relevance

This section presents the notion of *entropic relevance* for stochastic conformance checking. As anticipated in the Introduction, entropic relevance specifies the average number of bits per trace required to losslessly represent the log relative to some candidate stochastic process model. A model that leads to a more compact encoding and a lower entropic relevance is regarded as being a better description of the log, see Fig. 1. To provide the necessary details, Section 4.1 gives a definition of entropic relevance, parameterized by a background coding model for assigning costs to traces not possible according to the model. Then, Sections 4.2 to 4.4 present three instantiations of entropic relevance, differing only in their choice of background coding model, and hence their use of differing levels of knowledge about the compared log and model. Finally, Section 4.5 provides examples, and discusses the implications of the different background coding models.

4.1 Definition

An entropic relevance measurement consists of three components. The first component is the *trace compression cost*, namely, the cost of describing a compressed trace. The second component is the *prelude cost*, accounting for the cost of a compressed representation of the additional knowledge essential to encoding impossible traces. The *selector cost*, the third component, is the cost of an economically encoded binary flag that determines whether or not the stochastic model is utilized to encode each trace. Taken together, these three components

quantify the amount of information required to describe a losslessly encoded trace.

The trace compression cost is the number of bits required to encode a trace, using either a stochastic or background coding model. Given an event log $E \in \mathcal{E}$ and an SDFA A of FDAG $G \in \mathcal{G}$, trace $t \in E$ is either a possible trace according to the stochastic language of A, or it is not. If t is a possible trace, it is encoded using the probability of t according to A, given by $L_A(t)$. Otherwise, trace t is encoded using a universal background coding scheme. These two modes are captured in this next definition.

Definition 4.1 (Trace compression cost)
Let $t \in E$ be a trace in an event log $E \in \mathcal{E}$, let $A \in \mathcal{A}$ be an SDFA A of FDAG $G \in \mathcal{G}$, and let $bits_{\mathrm{X}} : V^* \times \mathcal{E} \times \mathcal{A} \to \mathbb{R}^+$ be a function that maps any and every trace $t \in V^*$ to the number of bits required to uniquely encode t, given a viable encoding that assumes knowledge of E and A. Then $cost_{\mathrm{X}}(t, E, A)$ is the *trace compression cost of t in the presence of E and A* and is defined by:

$$cost_{\mathrm{X}}(t, E, A) = \begin{cases} -\log_2(L_A(t)) & t \in \bar{L}_A \\ bits_{\mathrm{X}}(t, E, A) & \text{otherwise.} \end{cases}$$

⌟

The role of $bits_{\mathrm{X}}(\cdot, \cdot, \cdot)$ is to implement the background coding scheme that assigns a coding cost to any and every trace that is not permitted by the automaton. The subscript X in $bits_{\mathrm{X}}$, and similarly in $cost_{\mathrm{X}}$, refers to the concrete background coding model used to compute the cost of trace compression, with the coding cost of impossible traces computed using probability estimates associated with their constituent actions. In Sections 4.2 to 4.4, we define the *uniform background model*, *zero-order background model*, and the *restricted zero-order background model* as three options that vary only in terms of the probability estimates that they apply. Since relevance is grounded in the MDL principle, models that provide more accurate probabilistic estimations of actions, and result in overall shorter bitstreams, are to be preferred. The accuracy of such probability estimates relies heavily on the available detailed knowledge about the log and model.

The uniform background model assigns equal probabilities to all actions that might appear in the log. In contrast, the zero-order background model reduces – except for pathological cases – the trace compression cost by counting the occurrence frequency of actions in the event log, and using those counts as the basis for probability estimates. Any encountered non-uniformities in the symbol frequencies allow more accurately targeted probabilities to be employed, reducing the size of the compressed trace representations. Finally, to even more precisely estimate probabilities, the restricted zero-order background model counts only the frequencies of actions across the impossible traces rather than across all traces. As the background model encodes impossible, as per the model, traces, the restricted zero-order model allows – again except for pathological situations

– for even more targeted probability estimates and, consequently, more compact trace encodings.

One advantage of using the uniform background model is that it does not rely on additional information about the model or log. The other two models require that symbol probabilities are known so that suitable entropy codes can be – or at least, could be if required – deployed, an overhead that might, conceivably, make them more costly than the uniform approach. Nevertheless, if the observed symbol probabilities are non-uniform, and there is some moderate (of the order of hundreds, for example) of symbols to be coded, then it is likely that the two zero-order approaches will usually result in smaller bit lengths than the uniform one, and can never be much worse.

Let A_1 and A_2 be SDFAs shown in Fig. 4 and Fig. 6, respectively. Then, the compression costs (in bits) of traces `abcd` and `aee` from logs E_1 and E_2 in Section 3.4 are as follows:

- $cost_{\mathrm{X}}(\texttt{abcd}, E_1, A_1) = cost_{\mathrm{X}}(\texttt{abcd}, E_2, A_1) = -\log_2\left(L_{A_1}(\texttt{abcd})\right) = -\log_2(9/80) = 3.15$;
- $cost_{\mathrm{X}}(\texttt{abcd}, E_1, A_2) = cost_{\mathrm{X}}(\texttt{abcd}, E_2, A_2) = -\log_2\left(L_{A_2}(\texttt{abcd})\right) = -\log_2(3/256) = 6.42$;
- $cost_{\mathrm{X}}(\texttt{aee}, E_1, A_1) = cost_{\mathrm{X}}(\texttt{aee}, E_2, A_1) = -\log_2\left(L_{A_1}(\texttt{aee})\right) = -\log_2(1/20) = 4.32$.

The example traces are encoded using their probabilities in the two stochastic process models. Frequent traces, like trace `abcd` in A_1, are encoded using fewer bits than infrequent traces, like trace `abcd` in A_2. Whenever a trace is possible according to the process model, its cost is independent of the concrete background model X. However, trace `aee` is impossible according to the stochastic language of A_2, and cannot be handled the same way as it is in A_1. Its encoding cost is considered after the various background coding models have been defined (Sections 4.2 to 4.4).

When a background coding model relies on additional knowledge, such as the probability distribution of elements in the log or parts of the log induced by the stochastic process model, it must be described in a *prelude* to the trace encodings. Since entropic relevance is grounded on the lossless encoding of traces, the prelude should be "transmitted" (by having its bits counted) prior to the other code elements. For example, this can be an encoding of the list of elements' occurrence frequencies from which symbol probabilities can be estimated. Consequently, the prelude should be declared as part of the nominal "compressed" package that is being measured. The prelude cost, denoted by $prelude_{\mathrm{X}}(E, A)$, is the number of bits required to encode the prelude. Sections 4.2 to 4.4 define the prelude costs for the three background coding models. To reflect that the prelude cost is associated with each particular background coding model, it is parameterized by it and, hence, the subscript X.

The role of the selector code – the third component – is to indicate whether a trace is encoded using the stochastic process model or the background model (the left and right branches out of the decision diamond in Fig. 2). That is, each trace in the compressed event log is preceded by a binary flag that indicates whether or not the trace is possible in the stochastic model. If the probability associated with one branch is q, the expected average cost of describing one code in a stream of selector codes is given by $H_0(q) = -q\log_2(q) - (1-q)\log_2(1-q)$,

taking $H_0(0) = H_0(1) = 0$ by definition. This probability q can be the overall probability that a trace in the event log E is possible in the stochastic language of model A, given as $\rho(E, A) = \sum_{t \in \bar{L}_A} P(t \mid E)$.

The cost associated with the selectors is thus the information content of the probability q derived from the likelihood of the trace fitting the process model, that is, $H_0(\rho(E, A))$. The largest value of $H_0(\rho(E, A))$ is 1 bit per selector when $\rho(E, A) = 0.5$ and only half the traces can be derived from the stochastic process model. For example, consider the two SDFAs A_1, A_2, and event log E_1 discussed above. Model A_1 can replay every trace in log E_1, that is, $\rho(E_1, A_1) = 1$. The cost of selector $H_0(\rho(E_1, A_1))$ is thus 0, indicating the certainty of choosing A_1 as the coding scheme. That certainty is not possible for E_1 and A_2, as the overall probability of possible trace is $\rho(E_1, A_2) = 0.80$. Now the selector cost is, on average, 0.722 bits per trace.

When summed, the trace compression cost, prelude cost, and selector cost lead directly to the definition of entropic relevance.

Definition 4.2 (Parameterized entropic relevance from [25])
Let $E \in \mathcal{E}$ be an event log and let $A \in \mathcal{A}$ be an SDFA. Let $\mathrm{X} \in \{\mathrm{U}, \mathrm{Z}, \mathrm{R}\}$ be a parameter that defines a background coding model. Then, the *entropic relevance, or relevance, of A to E* (parameterized with X) is denoted by $rel_{\mathrm{X}}(E, A)$ and is defined by:

$$rel_{\mathrm{X}}(E, A) = H_0(\rho(E, A)) + \frac{1}{|E|} \sum_{t \in E} \Big(n(t, E) \cdot cost_{\mathrm{X}}(t, E, A) \Big) + \frac{prelude_{\mathrm{X}}(E, A)}{|E|} .$$

⌟

As the definition of relevance relies on the concrete chosen trace compression costing scheme and the corresponding prelude costing approach, it is also parameterized by the background coding model X. Note that while $rel_{\mathrm{X}}(E, A)$ derives its value via the measurement of a whole log E relative to the given automaton A, Definition 4.2 expresses entropic relevance in terms of "bits per trace"; that is, as being averaged over the whole of E. Note also that we regard the structure and probabilities associated with A to be "given", and not part of what is charged back to the relevance score; but that the cost of the background model is charged. This subtle distinction is in accordance with our desire to measure the complexity of A via human-centered approaches, refer to Section 2.

Fig. 2 helps explain Definition 4.1 and Definition 4.2. During the nominal encoding process, each trace t in the log is considered in turn. If trace t is possible in the stochastic process model, it is assigned a non-zero probability according to the stochastic language of the model, and the corresponding selector code is generated (the left branch out of the decision diamond in Fig. 2), and then that probability is used to encode t relative to the model (the first option in Definition 4.1). If the probability of t is zero according to the model (the right branch in Fig. 2), the opposite selector code is generated, and the

background model uses $bits_{\text{x}}(t, E, A)$ bits to represent t (the second option in Definition 4.1). That is, the selector codes associated with the diamond decision box in Fig. 2, and illustrated by the yellow line in Fig. 3, are needed to differentiate between the two alternatives on a per-trace basis, and add an average of $H_0(\rho(E, A))$ bits per trace. Finally, prior to the encoding of any traces via the background model, $prelude_{\text{x}}(E, A)$ bits must be spent to ensure the ability to losslessly decode those impossible traces.

Note also the comment that was made in Section 2.3 in regard to the relative costs associated with the background model and the process model for fitting traces. It is entirely possible for the background model to encode some fitting trace t more economically than the stochastic process model A, suggesting that perhaps a "best choice available" mechanism should be employed in our nominal compression codec, to allow a more nuanced selection of technique. In fact, we stay strictly with the $p(t \mid A) > 0$ criteria for fitting traces, and count even low-probability traces as being "handled" by A. The rationale for this decision is that the high bit cost that is then associated with t is actually a direct consequence of the stochastic model A having mis-estimated $t's$ probability, and should be counted against A as we seek to measure A's relative goodness-of-fit to the log as a whole.

We use SDFA A_1 and event log E_1 to exemplify the computation of entropic relevance. All traces in E_1 have non-zero probabilities in A_1, such that:

$$\{(\texttt{ab}, 2/5), (\texttt{ae}, 3/20), (\texttt{abcd}, 9/80), (\texttt{abc}, 1/20), (\texttt{aee}, 1/20), (\texttt{abcde}, 3/80)\}.$$

Then, each log trace is encoded using its probability in the model. Hence, the total encoding cost of all the traces in the log is equal to:

$$(1.32\times1200)+(2.74\times300)+(3.15\times220)+(4.32\times100)+(4.32\times100)+(4.74\times80) =$$

$$= 4342.20 \text{ bits}.$$

Consequently, the average encoding cost of a trace, and thus the entropic relevance of A_1 with respect to log E_1, is $4342.20/2000$, or approximately 2.17 bits. Note that the cost of the selector is $H_0(\rho(E_1, A_1)) = H_0(1) = 0$ bits per trace. Here we assume that the prelude cost is also 0 bits per trace, which is the case when the uniform background model is used. Below, we proceed with the discussion of this background model.

4.2 Uniform Background Model

Ignoring both the event log and the process model, the *uniform* background coding model proposed in our preliminary description [25], denoted here as $bits_{\text{U}}(\cdot, \cdot, \cdot)$, trivially encodes a trace t by taking each individual element in $\hat{t}$, the #-*terminated* version of t, as an equi-probable symbol over the underlying alphabet, i.e., the probabilities of observing actions from $u(\hat{E})$ in the log are considered to be uniformly distributed.

Definition 4.3 (Uniform background coding model [25])
Let $t \in E$ be a trace in an event log $E \in \mathcal{E}$ and let $A \in \mathcal{A}$ be an SDFA. Then, the *uniform background encoding cost of* t *in the presence of* E *and* A, denoted by $bits_{\text{U}}(t, E, A)$, is defined by:

$$bits_{\text{U}}(t, E, A) = |\hat{t}| \log_2 \Big(|u(\hat{E})| \Big).$$

⌟

Since trace aee is impossible according to the stochastic language of A_2, it is encoded using the background model and it thus holds that $cost_{\text{U}}(\texttt{aee}, E_2, A_2) = bits_{\text{U}}(\texttt{aee}, E_2, A_2) = 11.23$ bits, where E_2 is from Section 3.4 and A_2 is shown in Fig. 6. The encoding is based on four symbols, three actions from the trace plus the end-of-trace symbol #, each taking $\log_2(7)$ bits because $u(\hat{E}_2) = \{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e},\texttt{f},\#\}$, yielding $4 \log_2(7) = 11.23$ bits.

The uniform background model makes no assumptions about the distribution of actions in the log, meaning that the corresponding prelude cost is zero, $prelude_{\text{U}}(E, A) = 0$ for any event log E and automaton A.

4.3 Zero-Order Background Model

The actions in an event log may not be uniformly distributed. If they are not, the encoding cost of a trace t computed using background model $bits_{\text{U}}$ might be larger than necessary, and hence more pessimistic than is warranted. A second option for the background model is thus to have it reflect the probability distribution of the actions in the event log, by accumulating action probabilities as occurrence frequency counts. Frequent actions can then be coded using fewer bits than infrequent actions. Since such occurrence frequency counts of actions are determined without considering the action ordering with traces, we refer to this type of model as being a *zero-order* background coding model, $bits_{\text{Z}}$, and define it as follows.

Definition 4.4 (Zero-order background coding model)
Let $t \in E$ be a trace in an event log $E \in \mathcal{E}$ and let $A \in \mathcal{A}$ be an SDFA. Then, the *zero-order background encoding cost of* t *in the presence of* E *and* A, denoted by $bits_{\text{Z}}(t, E, A)$, is defined by:

$$bits_{\text{Z}}(t, E, A) = -\sum_{i=1}^{|\hat{t}|} \log_2 \Big(P(\hat{t}_i \mid \hat{E}) \Big).$$

⌟

It holds that $cost_{\text{Z}}(\texttt{aee}, E_2, A_2) = bits_{\text{Z}}(\texttt{aee}, E_2, A_2) = 8.95$ bits, where, again, E_2 is from Section 3.4 and A_2 is shown in Fig. 6. The encoding is based on the estimates of the probabilities of a, e, and # occurring in $\hat{E}_2$. These probabilities

are 0.298, 0.160, and 0.266, respectively. Therefore, actions a and e, and the end-of-trace # symbol get coded using 1.75, 2.65 and 1.91 bits, respectively; and then (presenting all values rounded to two decimals) $1.75 + 2.65 + 2.65 + 1.91 = 8.95$ bits.

In this arrangement the coding cost of the trace can only be achieved given knowledge of the frequency distribution of actions occurring in the log, information that must be accounted for. One possible way of computing a prelude cost is to assume the use of Elias codes [12]. Let $x > 0$ be an integer number. We write $C_\gamma(x)$ to refer to the encoding cost of integer number x using Elias' gamma code, and note that the gamma code represents arbitrary positive integers x is $C_\gamma(x) = 2\lfloor \log_2(x) \rfloor + 1$ bits. We favor the use of the Elias gamma code here, as it is at most a constant factor worse than optimal for non-increasing distributions over the positive integers, consuming $O(\log x)$ bits, and does not require that an upper bound be placed on x to achieve that coding rate. Note, however, that other infinite codes with different cost profiles could also be employed, including the Elias δ code, or a Golomb code [22, Chapter 3]. Thus, the prelude cost in case of the zero-order background model is given by $\mathit{prelude}_{\text{Z}}(E, A) = \sum_{\lambda \in V} \big(C_\gamma(n(\lambda, E) + 1)\big) + C_\gamma(|E| + 1)$.

As an example, consider the previous example of E_2 (see Section 3.4). In that event log we have $n(\texttt{a}, E_2) = 280$, and so action a contributes 17 bits to $\mathit{prelude}_{\text{Z}}(E_2, A)$. Each of the other symbols also has a cost associated with its frequency. In total the prelude costs $17+15+13+9+15+9+15 = 93$ bits across actions a, b, c, d, e, f, and #, respectively. That total represents a relatively small overhead in comparison to the 250 traces in E_2, but, nevertheless, is a cost that should be accounted for.

4.4 Restricted Zero-Order Background Model

To further refine the probability estimates, the zero-order background cost model can be restricted to consider the occurrence frequency of actions in only the impossible traces. This refinement in probability estimates of actions aims to obtain compact codes for traces impossible according to the stochastic language of the process model. Following is the definition of the *restricted zero-order* background coding model.

Definition 4.5 (Restricted zero-order background coding model) Let $t \in E$ be a trace in an event log $E \in \mathcal{E}$ and let $A \in \mathcal{A}$ be an SDFA. Then, the *restricted zero-order background encoding cost of* t *in the presence of* E *and* A, denoted by $\mathit{bits}_{\text{R}}(t, E, A)$, is defined by:

$$\mathit{bits}_{\text{R}}(t, E, A) = \begin{cases} \mathit{bits}_{\text{Z}}(t, E_{\neg A}, A) & \text{if } t \in E_{\neg A}, \\ +\infty & \text{otherwise} \end{cases}$$

where by $E_{\neg A}$ we denote the multiset of all the traces from E that are not possible according to the stochastic language of A. ⌟

For event log E_2 from Section 3.4 and automaton A_2 from Fig. 6 it holds that $cost_{\text{R}}(\texttt{aee}, E_2, A_2) = bits_{\text{R}}(\texttt{aee}, E_2, A_2) = 6.51$ bits. This is correct because $E_{2\neg A_2} = [\texttt{aeaae}^{40}, \texttt{aee}^{20}, \texttt{abee}^{10}]$ and, the coding is thus performed using the estimates of probabilities of observing actions a and e, and the end-of-trace symbol # of 0.405, 0.378, and 0.189, respectively. Consequently, actions a and e, and the end-of-trace # symbol get coded using 1.30, 1.40 and 2.40 bits, and making the total background cost of trace aee $1.30 + 1.40 + 1.40 + 2.40 = 6.51$ (again, rounding values to two decimal places). Note that the restricted zero-order background coding model now uses detailed information about the stochastic process model to compute the background coding cost – it needs to be informed which traces are and which traces are not possible according to the model, and the prelude cost is affected by both log and model.

We obtain the prelude cost for the restricted zero-order background model as the prelude cost for the zero-order background model computed over the impossible traces of the process model, i.e., $prelude_{\text{R}}(E, A) = prelude_{\text{Z}}(E_{\neg A}, A)$, again making use of the Elias gamma code. Note that $n(\texttt{a}, E_{\neg A}) \leq n(\texttt{a}, E)$, for all symbols a, and hence that $prelude_{\text{Z}}(E_{\neg A}, A) \leq prelude_{\text{Z}}(E, A)$.

4.5 Examples

With the required definitions in place, we now calculate the entropic relevance for automata A_1 and A_2 from Fig. 4 and Fig. 6, respectively, with respect to the event logs E_1 and E_2 from Section 3.4. Table 1 summarizes the constituent costs and the entropic relevance values for the four combinations of log and automata.

Table 1: Entropic relevance (in bits) and its constituents for two example automata A_1 and A_2 from Fig. 4 and Fig. 6, respectively, and event logs E_1 and E_2 from Section 3.4. The columns list the probability of traces from the log being possible according to the automata (ρ); the average (over all that log's traces) selector coding cost (Select.); the model coding cost (MdlCst.); the average per-trace coding cost (Bkgnd.) of the Uniform, Zero-order, and Restricted zero-order background coding models; the associated average prelude coding cost (*prelude*); and the entropic relevance value (*rel*).

Autom.	Log	ρ	Select.	MdlCst.	BkgndCstMdl.	Bkgnd.	*prelude*	*rel*
A_1	E_1	1.00	0.00	2.17	Uniform	0.0	0.0	2.17
					Zero-order	0.0	0.06	2.23
					Restricted zero-order	0.0	0.003	2.18
A_2	E_1	0.80	0.72	2.63	Uniform	1.68	0.00	5.03
					Zero-order	1.61	0.06	5.02
					Restricted zero-order	1.02	0.03	4.41
A_1	E_2	0.56	0.99	1.79	Uniform	4.49	0.00	7.27
					Zero-order	3.68	0.37	6.84
					Restricted zero-order	3.37	0.27	6.42
A_2	E_2	0.72	0.86	2.62	Uniform	4.15	0.00	7.63
					Zero-order	3.18	0.37	7.02
					Restricted zero-order	2.45	0.21	6.13

In the first row of the table, SDFA A_1 explains event log E_1 well, as the traces in the log all fit the automaton. The relative likelihoods of observing the

traces in the log and in the automaton differ for trace `ab`, `abcd`, `abcde`, and `abf`, which suggests the possibility that other automata can achieve even lower relevance for E_1. The values of relevance when using $bits_{\text{Z}}$ and $bits_{\text{R}}$ are greater than the relevance value for $bits_{\text{U}}$ because of the modest, but non-zero, prelude coding cost associated with the latter two approaches.

In the second row, automaton A_2 provides a less effective description of E_1, primarily because one fifth of the traces cannot be represented in the primary model, and must be handled by the background model. As a result, all three cost components are higher, and a clear difference emerges between A_1 and A_2, with A_1 preferred. In A_2, the restricted frequency counts associated with $bits_{\text{R}}$ give that method an advantage over $bits_{\text{U}}$ and $bits_{\text{Z}}$, but that gain is not enough to overcome the innate superiority of A_1.

Automaton A_1 compresses E_2 (row three) better than automaton A_2 (row four) when background costing models $bits_{\text{U}}$ and $bits_{\text{Z}}$ are used; but the restricted background model $bits_{\text{R}}$ reverses that relationship, and model A_2 with $bits_{\text{R}}$ is (by a narrow margin) the best explanation of E_2 when all cost components are summed, even though A_1 (column "MdlCst.") provides more accurate encodings of the (albeit smaller) subset of traces that can be accommodated by the model. The restricted background model $bits_{\text{R}}$ comes into its own in this example, adjusting its probabilities to the particular nature of the non-fitting traces. This second example highlights the importance of including all of the three cost components, and illustrates the balances that can affect the overall entropic relevance scores.

The restricted zero-order background model $bits_{\text{R}}$ is superior to the other two background models, the expected relationship when at least a moderate number (dozens or hundreds) of symbols are encoded as part of the non-fitting traces, and where the observed frequency distribution across the symbols is non-uniform. When the number of non-fitting traces is very small, or the frequency distribution is very close to uniform, the cost of the prelude required may make $bits_{\text{Z}}$ and $bits_{\text{R}}$ more costly than $bits_{\text{U}}$, but the difference is likely to be small. Because the frequencies of symbols associated with $bits_{\text{R}}$ can never exceed those that pertain to $bits_{\text{U}}$, and since $bits_{\text{R}}$ is the correct probability distribution for the symbols in the non-fitting traces and they are the (only) ones coded using the background model, $bits_{\text{R}}$ can never be larger than $bits_{\text{Z}}$.

5 Evaluation

The entropic relevance measure described in Section 4, including the range of background coding models, has been implemented and integrated into the *Entropia* tool [24], and experimentally assessed using a range of synthetic and real-world datasets, all of which are publicly available [2]. The experiments address two main research questions:

RQ1: Is measuring of entropic relevance feasible for industrial datasets?
RQ2: Is entropic relevance fundamentally different from other measures for (stochastic) conformance checking?

To answer RQ1, we computed entropic relevance for thirty real-world event logs and corresponding process models discovered from these logs. The results and execution times are reported in Section 5.1.

To answer RQ2, we compare entropic relevance with other conformance checking measures in current use. Section 5.2 begins by describing the methodology used to perform the comparison. For each comparator measure, we then present evidence that it quantifies a different phenomenon than does entropic relevance. Section 5.3 then considers the stability of the measurements that are reported.

Table 2: Conformance checking measures.

Short label	**Full name and reference**
Precision	Exact-matching entropy-based precision [28]
Recall	Exact-matching entropy-based recall [28]
Stoc.Precision	Stochastic entropy-based precision [16]
Stoc.Recall	Stochastic entropy-based recall [16]
EMSC	Earth movers' stochastic conformance [18]

Table 3: Process discovery techniques.

Short label	**Full name and reference**
Apromore	Apromore (Academic Edition) [1]
Celonis PE	Celonis SE (Process Explorer) [2]
Celonis VE	Celonis SE (Variant Explorer) [2]
DFvM	Directly Follows visual Miner [17]
Disco	Fluxicon Disco (Academic Edition) [3]
Minit	Minit (Academic Edition) [4]
Split Miner	Split Miner [3]

Table 2 and Table 3 list, respectively, the conformance checking measures and process discovery techniques used in our experiments. The conformance checking measures from Table 2 are discussed in more details in Section 7.

5.1 Feasibility

To study the feasibility of using entropic relevance in industrial settings, we experimented with thirty real-world event logs recording transactions of IT-systems executing business processes, as made publicly available by the IEEE Task Force on Process Mining.[5] Using the discovery techniques listed in Table 3, FDAGs (also known as DFGs) were constructed, with each pair of log and

[1] https://apromore.org
[2] https://www.celonis.com
[3] https://fluxicon.com/disco/
[4] https://www.minit.io
[5] https://data.4tu.nl/repository/collection:event_logs_real.

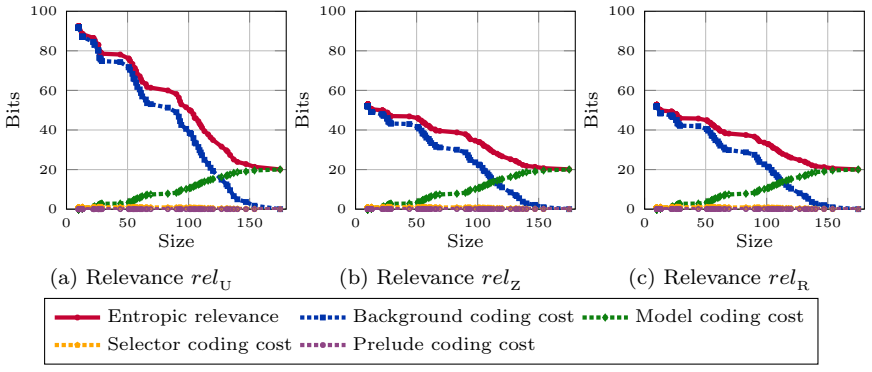


Figure 7: Three entropic relevance measurements (rel_{U}, rel_{Z}, and rel_{R}) and their constituents, plotted as a function of model size measured as the sum of the numbers of actions (vertices) and directly-follows arcs in the model; models discovered using DFvM from BPIC 2012 event log. These three graphs are concrete realizations of the arrangement sketched in Fig. 3.

discovery technique leading to 100 FDAGs for various parameterizations of the technique, chosen so as to create models of different sizes. For example, for the DFvM technique, trace removal thresholds were set at $^{1}/_{100}$, $^{2}/_{100}$, ..., $^{100}/_{100}$ to discover the 100 models.

Fig. 7 plots the values of the three versions of relevance (rel_{U}, rel_{Z}, and rel_{R}, that is, based on three different background models), and their constituents for FDAGs generated by DFvM from BPI Challenge (BPIC) 2012 event log [37].[6] The *size* of each model (the horizontal axis) is taken to be the number of action nodes plus the number of edges. In the three plots, the curves correspond to the anticipation provided by Fig. 3. As the models become more complex, an increasing fraction of traces fit the model (green line); a decreasing fraction are coded using the background model (blue line); and the selector coding cost (yellow line) and the prelude coding cost (purple line) are small over most of the range. All three versions of entropic relevance settle at around 20 bits per trace, by which point the background model is not being used. That is, while a background model must always be specified, and must always be available, if all of the traces in the log fit the main process model, then all of rel_{U}, rel_{Z}, and rel_{R} will give behavior very close to each other.

Note that in our initial conference presentation, and also when discussing Fig. 3, we presumed $bits_{U}$ as a background model and hence did not need to account for a background model prelude. But we have included it as a visible component in Fig. 7 (albeit, a very minor one) so that $bits_{Z}$ and $bits_{R}$ are properly accounted for. Compared to the size of the event log, the alphabet and the frequency of its actions are small sets, making the prelude cost a minor

[6] An error in the corresponding plot for Fig. 7(a) in the original conference paper [25, Figure 7(c)] has been found and corrected in this article.

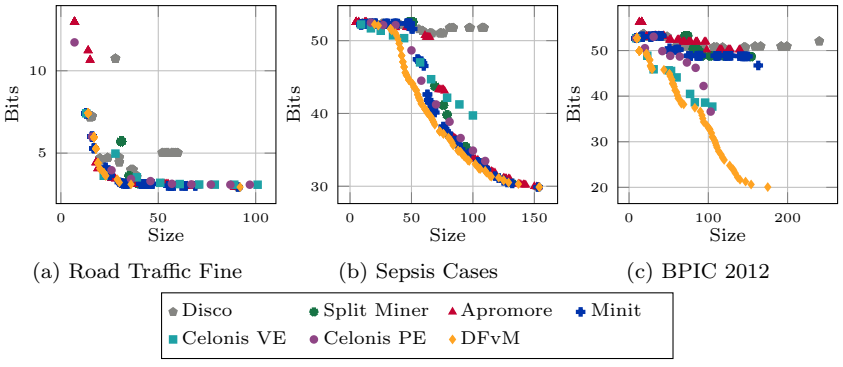


Figure 8: Entropic relevance (rel_{R}) of FDAGs discovered from three real-world logs using the process discovery techniques listed in Table 3, plotted as a function of model size.

component. For $bits_{\mathrm{R}}$, the prelude cost gradually decreases, with the non-fitting symbol frequencies becoming smaller as the process model grows in size and accommodates more traces.

Fig. 8 plots relevance values in bits for the sets of FDAGs constructed using the seven discovery techniques listed in Table 3 from three logs: Road Traffic Fine Management [10], Sepsis Cases [20], and BPIC 2012 [37], again as a function of model size. Now models, and hence discovery techniques, can be compared by considering the Pareto frontier of the points plotted in each graph. For all logs, in general, as the models become more complex, the fraction of traces that fit the FDAGs increases and relevance values decrease. Fig. 9 shows enlarged regions from the corresponding plots in Fig. 8, spanning the discovered FDAGs of size less than approximately 50, the range of small- to medium-sized models which can be appreciated visually and are of interest to business analysts. Note that the numeric relevance values are subject to the approach used to map FDAGs to SDFAs, and to the choice of background coding model. Detailed analysis of these interactions, and of FDAGs discovered using other techniques, is future work.

The computation of entropic relevance requires straightforward data structures and execution loops. With a hash-map used to implement the set of edges at each state in the model, computation time is linear in the total volume of log data processed (number of traces times average length). The average CPU time to compute entropic relevance using our implementation and a commodity laptop computer for the largest log analyzed (BPIC 2018 – Payment application; 43,809 traces and 984,613 events) over the 100 DFMs that were constructed (with sizes ranging from 25 to 238) was 0.47 sec.

Note also that none of the models plotted in Figs. 7 and 8 were over-fitted to the data, and that the model description costs (see Fig. 3), which are excluded from our definition of entropic relevance, were small compared to the entropic relevance scores.

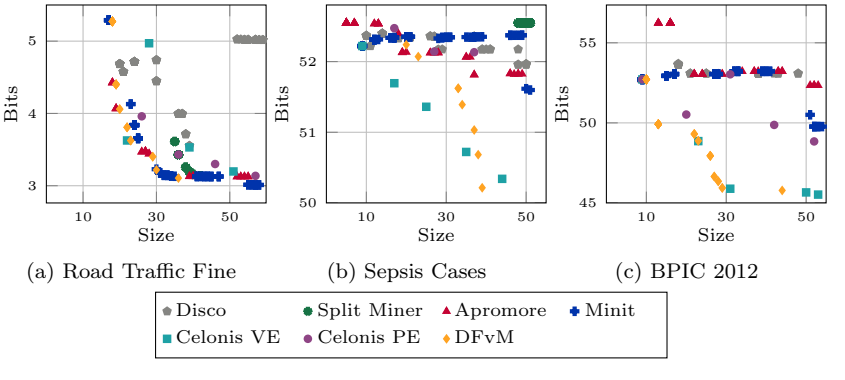
5.2 Comparison With Other Measures

Section 4 introduced entropic relevance as a stochastic conformance checking measure that assesses how accurately a stochastic process model describes an event log. This section discusses and compares entropic relevance with a range of previous conformance checking techniques (see Table 2) using a range of real-world and synthetic datasets. In addition to the five techniques listed in Table 2, we compare entropic relevance against *coverage*, defined as the number of traces from the log that are described by the model (including counting duplicate traces) divided by the total number of traces in the log. That is, a total of six measures are used as reference points and compared to entropic relevance.

To carry out each of those comparisons, we demonstrate that we can find two sets of models, with scores within one set positively correlated between the measure and entropic relevance, and scores in the other set negatively correlated. Whenever both of those sets exist they jointly provide clear evidence that relevance is not blindly mirroring that other measure.

For simplicity, we fix on one relevance, rel_{U}, to decrease the number of pairs of measures, noting that all of rel_{U}, rel_{Z}, and rel_{R} have relative ordering in terms of their behavior. A similar process of generating models to that described in Section 5.1 was followed to construct the models. For each log and corresponding models, we computed scores for all seven measures. The strength and the direction of the association between the values of entropic relevance and each of the other six measures was then calculated using Spearman's correlation test with 95% confidence intervals, with Spearman's coefficient preferred to Pearson's correlation coefficient to avoid any requirement that the measurements follow a linear relationship or comply with a normal distribution.

To obtain some of the required correlations, we constructed a synthetic log E over the alphabet $V = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}, \texttt{g}, \texttt{h}\}$, containing 20,000 instances of $t = \texttt{abcdefgh}$, plus one instance of each other permutation of the alphabet (another $8! - 1 = 40{,}319$ traces), plus another 20,320 traces generated randomly with replacement from the same set of 40,319 permutations (that is, with an



(a) Road Traffic Fine (b) Sepsis Cases (c) BPIC 2012

Figure 9: Enlarged regions of interest from the plots in Fig. 8.

average additional frequency per permutation of approximately 0.5, making an average of 1.5 occurrences in total). In total, E thus contained 80,639 traces.

Fig. 10 shows how entropic relevance calculated using rel_{U} relates to the other conformance checking measures, for each measure picking out instances of logs for which positive (top half) and negative (bottom half) correlations were observed. That is, the first two rows show six positive correlations between the entropic relevance measurements; then the lower two rows show exactly the same set of six measures, but now with negative observed correlations, illustrated via a different set of logs. The models used to obtain the measurements were discovered using DFvM, except plot Fig. 10(i), where Split Miner was used. Three real-world event logs were used (Sepsis, BPIC 2012, and BPIC 2018 Entitlement Application [38]), plus the synthetic log E described above.

Fig. 10 presents an unambiguous picture: for each of the other six measures it is possible to identify a situation (involving a log and a discovery mechanism) in which entropic relevance is positively correlated with that measure, and also another situation in which it is negatively correlated. That is, Fig. 10 provides compelling evidence that entropic relevance is demonstrably independent of previous conformance measures, and while unnecessary proliferation of entities is to be eschewed (the Occam's razor principle), in this case, the new approach clearly introduces a perspective that adds to our ability to measure process models, rather than merely replicates a previous technique.

5.3 Stability

The models constructed by a process discovery algorithm from statistically similar logs may differ considerably, i.e., discovery algorithms may be unstable [36]. To quantify the potential risk of this phenomenon on our conclusions about entropic relevance, we tested the stability of the discovery algorithms when constructing the models used in the correlation analysis, adopting a standard bootstrapping approach. A total of 100 bootstrap replicates were generated from each log, using sampling with replacement to create logs of the same size as the original. For each replicate, the same procedure of generating models as used in the correlation analysis was followed, except that only ten instead of 100 different models were constructed from each sample log, using ten different parameterizations of the model construction process. The 1000 data points were then scatter-plotted with transparency, and Spearman's correlations were computed between relevance and the other measures. A sample of the results obtained is shown in Fig. 11.

Fig. 11 shows several effects. The most noticeable of these is that the overall shapes of the three regression curves closely match the corresponding lines for the original logs, shown in Fig. 10(c), Fig. 10(f), and Fig. 10(h). This is as expected, since each bootstrap replicate should possess the same statistical properties as the corresponding original log from which it was generated. The computed R values are also in agreement. But within that overall agreement, there are also interesting differences. In particular, the "point clouds" that are visible in Fig. 11 are a result of the randomness inherent in the bootstrap

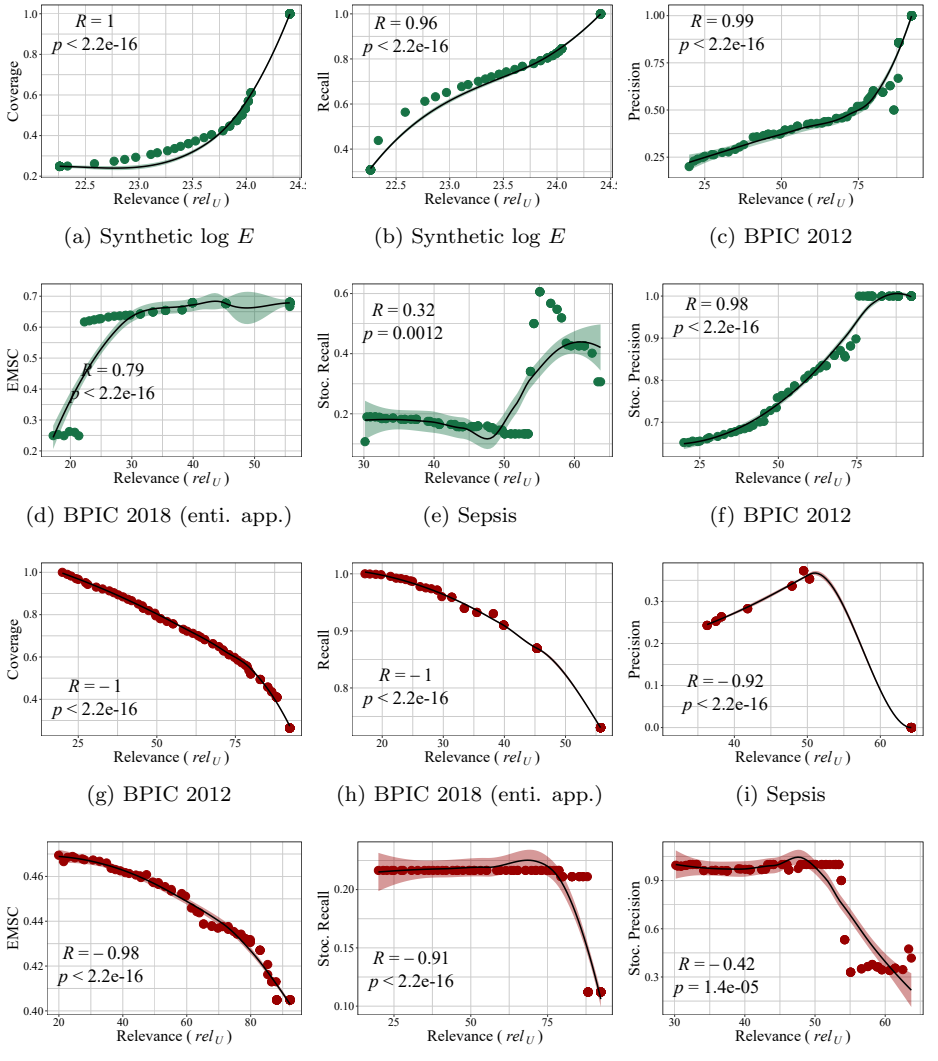
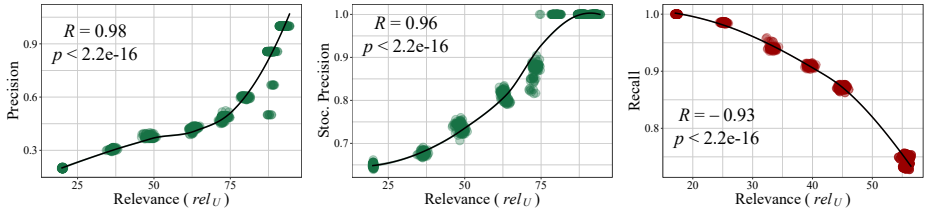


Figure 10: Entropic relevance (rel_{U}) plotted against coverage, exact-matching entropy-based recall and precision, earth movers' stochastic conformance, and stochastic entropy-based recall and precision, using a variety of process logs. Each point represents a process model; smooth black lines represent the LOESS regression functions for the points with bandwidth equal to 75% of the data; shaded regions depict 95% confidence intervals; R values represent Spearman's correlation coefficients; and p values refer to the sign of the correlation coefficients.



(a) BPIC 2012 (b) BPIC 2012 (c) BPIC 2018 (enti. app.)

Figure 11: Entropic relevance (rel_{U}) plotted against exact-matching entropy-based precision and recall, and stochastic entropy-based precision. Each individual point represents the process model derived from one of 100 bootstrapped replicates of the corresponding original log, for each of 10 different parameter settings. Smooth black lines represent the LOESS regression functions for the 1000 points with bandwidth equal to 75% of the data; the shaded regions (almost identical with those lines) depict 95% confidence intervals; R values represent Spearman's correlation coefficients; and p values refer to the sign of the correlation coefficients. The three plots here can be compared with Fig. 10(c), Fig. 10(f), and Fig. 10(h), respectively.

process; and the fact that some parameter settings led to multiple clouds is an illustration of the discrete nature of process inference. Small (essentially random) changes in the log can lead to different models being generated, with notably different quality scores then emerging as a consequence of the non-continuous nature of the "model" space.

Similar testing was performed for each correlation plot in Fig. 10, with the same range of outcomes observed.

6 Discussion

Relevance, the way it is defined in Section 4, provides a compromise between the precision and recall quality criteria of models automatically discovered from event logs. This claim is justified by the case-analysis provided in Table 4.

Entropic relevance is the average cost of encoding a trace from the log using a model discovered from this log (or indeed, discovered from any other log over the same set of elements A, or via any other mechanism including human analysis). Table 4 shows how different traces contribute to this average cost. A trace is either in or not in the log, and is either described or not described by the model. These possible options are captured in the first two columns of the table. In addition, a trace can be frequent or infrequent in the log, and/or frequent or infrequent in the model. These options are listed in columns three and four of the table, with "n/a" entries denoting combinations that are not plausible. For example, traces that are absent from the log or not described by the model are neither frequent nor infrequent. Columns five and six classify the model and background coding costs induced by the trace characterized in the first four columns. Five cost classes are introduced. A trace that incurs no cost

Table 4: The classification of the magnitudes of coding costs of various traces when measuring entropic relevance. The first four columns characterize a trace as Present in the Log/Model and Frequent in the Log/Model (*yes*: present/frequent; *no*: absent/infrequent; *n/a*: not applicable). The last two columns specify if the corresponding trace characterized in the first four columns of the table should (*yes*) or should not (*no*) be penalized by Precision and Recall quality criteria. The Model and Background coding cost columns specify the magnitude of the corresponding coding costs of the trace (*zero*: zero cost; *entropy*: cost of the contribution of the trace to the entropy of the stochastic language; *low*: low cost; *medium*: medium cost; *high*: high cost); the costs shown in parenthesis are induced indirect costs of other traces.

Trace				**Model coding cost**	**Background coding cost**	**Precision**	**Recall**
Present		**Frequent**					
Log	**Model**	**Log**	**Model**				
yes	yes	yes	yes	entropy	zero	no	no
yes	yes	yes	no	high	zero	no	yes
yes	yes	no	yes	low (high)	zero	yes	no
yes	yes	no	no	entropy	zero	no	no
yes	no	yes	n/a	zero	high	no	yes
yes	no	no	n/a	zero	medium	no	yes
no	yes	n/a	yes	zero (high)	zero	yes	no
no	yes	n/a	no	zero (medium)	zero	yes	no
no	no	n/a	n/a	zero	zero	no	no

is denoted by "zero" in the table. Suppose a trace makes the same (or a very similar cost) contribution to relevance as to the entropy of the corresponding stochastic language (i.e., $-p \log_2(p)$, where p is the probability of observing the trace). In that case, it is denoted by "entropy" in the table. Finally, "low", "medium", and "high" entries suggest, respectively, the low, medium, and high coding cost of the corresponding trace. In parentheses, we give the indirect cost of the trace stemming from the impact on the costs of the other traces.

For example, a trace that is frequent in both log and model induces no background coding cost, as it is coded using the probability of that trace in the model. As this probability is the same as or very similar to the probability of observing the trace in the log, the model coding cost of this trace is (if not precisely) very similar to $-p \log_2(p)$, where p is the probability of observing the trace either in the log or model. This cost roughly corresponds to the contribution of the trace to the entropy of the stochastic language – a theoretically minimum contribution to the description of the stochastic language. As another example, if a trace appears frequently in the log with probability q, but the model specifies that this trace is infrequent and assigns probability $p \ll q$, then the contribution to relevance is $-q \log_2(p)$, and is larger than would be ideal. In contrast, a trace that is infrequent in the log but frequent in the model will incur a low cost. However, if such traces exist, they inevitably lead to the existence of frequent log traces that are infrequent according to the model. As already noted, such traces incur high model coding costs; hence the "low (high)" entry in the table suggesting low direct costs but high indirect costs for such traces.

The precision and recall quality criteria of discovered models penalize traces denoted by "yes" in columns seven and eight in Table 4. Interestingly, all the corresponding traces are also penalized, to various degrees, by the entropic relevance measure. Note that the entropy coding cost is the minimum required cost to encode the trace losslessly. Hence, it is not considered a penalty. All three cases of the recall penalties correspond to direct trace costs, while the cases of precision penalties stem predominantly from the indirect trace costs.

The coverage measure used in our evaluations penalizes traces that do not fit the model. Unfitting traces are also penalized by the entropic relevance measure; refer to the rows in Table 4 marked with "yes" in the first column and "no" in the second column. However, there are several fundamental differences between the coverage measure and the relevance measure.

First, if a model cannot replay any of the traces in the log, the corresponding relevance value (via the background model) still reflects the complexity of the log. Note that the corresponding coverage value of zero does not provide such information, which makes the situations with fully unfitting logs indistinguishable. For example, the entropic relevance measurements of the small models in Fig. 9(a) are smaller than 15 bits, while the entropic relevance measurements of the small models in Fig. 9(b) are larger than 50 bits. These small models hardly fit any traces from the corresponding logs and have coverage of (close to) zero. Consequently, we can conclude that the traces from the Sepsis Cases log are more complex than traces from the Road Traffic Fine Management log; that is, they are less compressible relative to the background model.

Second, similar to the coverage value of zero, the coverage value of one, which denotes a situation when the stochastic languages induced by the model and log are the same, does not provide any additional information. The corresponding relevance value, however, conveys information about the underlying distribution of traces. For example, the relevance score for a situation in which a small set of traces are frequent in the log and model will be smaller than when all the traces are equi-probable. For instance, the entropic relevance of large models constructed from the Road Traffic Fine Management log is small, approximately two bits; refer to Fig. 9(a). At the same time, large models constructed from the Sepsis Cases log have an entropic relevance of approximately 30 bits; see Fig. 9(b). Note that these large models fit most of the log traces and, hence, have coverage of (close to) one. Consequently, we can conclude that the Road Traffic Fine Management log has a small number of relatively frequent traces and a long tail of rare traces. In contrast, the distribution of traces in the Sepsis Cases log is more even.

Finally, while a log trace has the same effect on coverage irrespective of its likelihood according to the model, relevance takes both probabilities into account. Entropic relevance allows for a nuanced analysis of discrepancies between the trace likelihoods in logs and models.

7 Related Work

A wide range of non-stochastic process discovery techniques have been proposed over the last two decades, including the Genetic Mining [11], Heuristic Mining [39], Inductive Mining [15], and Split Mining [3] algorithms, all of which have been well-received by the process mining community. Non-stochastic conformance checking techniques can be broadly classified into *quantification* techniques, those that summarize conformance diagnostics into a single number, and *characterization* techniques, those that construct detailed analytics of commonalities and discrepancies between model and log traces; Carmona et al. [7] provide a useful overview.

Recently, Van der Aalst et al. [34, 30] initiated discussion of desired properties for non-stochastic conformance checking techniques. Entropy-based measures are the only quantitative conformance checking techniques that are known to satisfy all the properties for precision and recall that have been proposed to date, including the strict monotonicity properties [28].

Our proposal here is an application of the MDL principle [4, 14]; which, in turn, is related to the 1965 definition by Kolmogorov that the intrinsic descriptive complexity of an object is the length of the shortest binary computer program that describes it [9]. Kolmogorov complexity formalizes the notion widely known as "Occam's Razor" [31], a problem-solving principle attributed to William of Ockham which suggests that the simplest (that is, shortest) sufficient explanation of a phenomenon is the best.

We are not the first to consider using MDL principles to measure the relative fit of a log and a process model. Notably, Calders et al. [6] have also investigated the possibility of coding traces against a model. In their proposal the "coding" is provided on a per-trace basis as a sequence of integer values that indicate which – of a set of possibilities that evolves as each symbol in the trace is accounted for – transition in a Petri net must fire next in order to generate the trace. For example, if the configuration of the Petri net at some point in time is such that there are three possible "next" moves, then an integer between 1 and 3 will be associated with each of those moves, and the bit-cost of coding that integer is associated with the corresponding symbol in the trace. Like our background model, Calders et al. also provide an "escape" mechanism to handle traces that for some reason are not permitted outputs from the Petri net. In their case the integer 0 is permanently associated with a "state jump" in the Petri net, to shift it to another more amenable configuration in which the next symbol is a legal one. In the same example, if there were three legal transitions available, there would thus be four integers in the "possible" set, and hence a two-bit integer generated (followed by further bits in the case of that integer being 0, to specify the transition assumed to have fired even when not enabled from amongst the complete set of all possible transitions). Since the state jump is retained as an option at every transition, the minimum cost associated with each transition in the Petri net is one bit. That is, a described trace that is completely deterministic and unambiguously generated by the Petri net will nevertheless be costed as requiring one bit per transition. In our proposed

approach here, the escape to the background model is on a whole-of-trace basis, and never adds more than one bit to the measured cost of the whole trace.

As well as their use of binary codes and interleaved escape signals, the proposal of Calders et al. [6] differs from our work here in a number of other key ways. In our description and experiments we choose to retain the model cost as a separate value, and express it in terms of conventional dimensions such as nodes and edges. This allows entropic relevance to be viewed as a function of model complexity, the latter being an important human criteria. In contrast, Calders et al. include the cost of representing the model, using static integer codes such as the Elias C_γ code. This difference means that entropic relevance is not, strictly speaking, a "pure" MDL mechanism, but does bring the benefit of what might be termed "scale constancy". By that we mean that if a log of traces is "multiplied by k", so that each trace appears k times rather than once, the entropic relevance with respect to any given model remains constant. In contrast, the Calders et al. MDL cost (when converted from the total value that is computed by their formulation into a per trace value) will decrease towards an asymptotic value as the log is multiplied.

As a further distinction, there are no entropy codes employed by Calders et al. [6] – their proposal uses only binary and static integer codes, and there is no use of symbol or trace probabilities to compute its conformance values. Hence, while the cost assigned to each trace is a direct consequence of the elements in the trace, and of their sequencing, there is no sense in which common traces become advantaged relative to rare traces, and every repetition of any given trace incurs the same computed cost as the first instance of that trace.

At a broader level, and following on from that last observation, note that the selection of currently available stochastic process mining techniques – the output of which is assumed as the required input for our work here – is rather limited. To the best of our knowledge, there are two stochastic discovery techniques proposed by academia. The technique presented by Rogge-Solti et al. [29] discovers stochastic Petri nets, while that of Leemans et al. [17] can be used to discover DFGs, and was employed in Section 5. There are many commercial tools for discovering DFGs, or FDAGs, from event logs, but these are all closed source.

Two further quantification techniques for stochastic conformance checking have been proposed. Leemans et al. [18] base their approach on the "earth movers' distance", and measure the effort to transform the distribution of log traces into the distribution of model traces, seen as two piles of dirt that need to be aligned with minimal effort. The technique is computationally demanding and suggests practical trade-offs between accuracy, run time, and memory usage. The approach of Leemans and Polyvyanyy [16] is inspired by entropy-based conformance checking [28]. Leemans and Polyvyanyy [16] also suggest a range of desired properties for stochastic precision and recall and show that their measures indeed possess these properties. The calculation of the measures requires (in the worst case) a quadratic number of steps in the size of the corresponding SDFAs, while entropic relevance runs in linear time in the size of the log. In addition, relevance, as discussed in Section 6, reflects the compromise between

precision and recall in a single value with meaningful units.

The basic characterization approach for capturing non-stochastic conformance diagnostic between a trace and a process model is to construct an optimal alignment between the two [1]. An *alignment* is a sequence of moves. A synchronous move is a pair in which one element refers to an event in the trace, and the other element refers to an action in the model that triggered the event. If one element in a move is a special skip symbol, the move is said to be asynchronous. When one traverses moves in the order they appear in the alignment and mentions all the non-skip symbols that are log events, one identifies the trace used to construct the alignment. In turn, when non-skip symbols that are model actions are mentioned, a valid execution of the model is identified. Asynchronous moves describe discrepancies between the log trace and the model and, thus, if possible, must be avoided in alignments. Hence, an *optimal alignment* between the trace and the model is an alignment with the lowest cost, assuming some non-negative costs of asynchronous moves are used.

Non-stochastic alignments have been recently extended to *stochastic alignments* that account for the frequencies of traces in logs and stochastic languages induced by stochastic process models [19]. Instead of identifying moves as synchronous and asynchronous, stochastic alignments compute the likelihood for moves to be synchronous or asynchronous.

Alignments can be used to define techniques that quantify conformance between logs and models. For example, the coverage measure we used in the evaluations can be implemented by computing optimal alignments between the log traces and the model to identify log traces described by the model as those that induce optimal alignments of zero cost. Otherwise, alignments are not directly comparable to entropic relevance and serve a different purpose: to diagnose, rather than measure, conformance.

8 Conclusion

This article refines and provides a more detailed justification for the recently introduced notion of entropic relevance for stochastic conformance checking. In particular, we have demonstrated both via argument and via experimental results that entropic relevance can be used to assess how closely a process model learned from event data predicts the data that it is intended to represent.

The entropic relevance measure is motivated by and approximates (but only approximates, because we do not include the description cost of the primary model in our accounting) the MDL principle, which in essence asserts that the most compact description of some set of data is the best explanation of it. The entropic relevance measure adheres to this principle, and allows principled numeric comparison between alternative stochastic models of process action, and hence accurate comparison. In addition, the experimental results we have presented provide strong justification that entropic relevance is fundamentally different to other conformance checking techniques, in that it measures phenomena that are demonstrably neither directly dependent on nor predictably

correlated with those other measures.

We have also provided evidence that the entropic relevance of a log with respect to a given model reflects a compromise between the classical precision and recall quality criteria for assessing discovered models. Entropic relevance is measured in bits, with values being directly interpretable, and with small scores being preferable. Moreover, it is computable in time linear in the size of the log. As the next step in our research, we plan to explore new techniques for discovering stochastic process models from event data that are guided by – and perhaps even able to optimize for – the notion of entropic relevance.

Acknowledgment. Artem Polyvyanyy was in part supported by the Australian Research Council project DP180102839.

References

[1] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011, Helsinki, Finland, August 29 – September 2, 2011*, pages 55–64. IEEE Computer Society, 2011.

[2] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Discovered process models 2020-08, 2020. Available at https://doi.org/10.26188/12814535.

[3] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Split miner: Automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems*, 59(2):251–284, 2019.

[4] Andrew R. Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[5] Adam Burke, Sander Leemans, and Moe Thandar Wynn. Stochastic process discovery by weight estimation. In *International Workshop on Process Querying, Manipulation, and Intelligence, PQMI 2020, Padua, Italy, October 5, 2020*, 2020.

[6] Toon Calders, Christian W. Günther, Mykola Pechenizkiy, and Anne Rozinat. Using minimum description length for process mining. In *Proceedings of ACM Symposium on Applied Computing, ACM/SAC 2009, Honolulu, Hawaii, USA, March 9–12, 2009*, pages 1451–1455. ACM, 2009.

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

[8] Rafael C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *RAIRO Theoretical Informatics and Applications*, 31(5):437–444, 1997.

[9] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2006.

[10] Massimiliano De Leoni and Felix Mannhardt. Road traffic fine management process – event log, 2015. Available at https://doi.org/10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5.

[11] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: An experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[12] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[13] King-Sun Fu and T. Huang. Stochastic grammars and languages. *International Journal of Parallel Programming*, 1(2):135–170, 1972.

[14] Mark H. Hansen and Bin Yu. Model selection and the principle of minimum description length. *American Statistical Association*, 96(454):746–774, 2001.

[15] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs – A constructive approach. In *Proceedings of Application and Theory of Petri Nets and Concurrency, PETRI NETS 2013, Milan, Italy, June 24–28, 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2013.

[16] Sander J. J. Leemans and Artem Polyvyanyy. Stochastic-aware conformance checking: An entropy-based approach. In *Proceedings of Advanced Information Systems Engineering, CAiSE 2020, Grenoble, France, June 8–12, 2020*, volume 12127 of *Lecture Notes in Computer Science*, pages 217–233. Springer, 2020.

[17] Sander J. J. Leemans, Erik Poppe, and Moe Thandar Wynn. Directly follows-based process mining: Exploration & A case study. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24–26, 2019*, pages 25–32. IEEE, 2019.

[18] Sander J. J. Leemans, Anja F. Syring, and Wil M. P. van der Aalst. Earth movers' stochastic conformance checking. In *Proceedings of Business Process Management Forum, BPM 2019, Vienna, Austria, September 1–6, 2019*, volume 360 of *Lecture Notes in Business Information Processing*, pages 127–143. Springer, 2019.

[19] Sander J. J. Leemans, Wil M. P. van der Aalst, Tobias Brockhoff, and Artem Polyvyanyy. Stochastic process mining: Earth movers' stochastic conformance. *Information Systems*, 102, 2021.

[20] Felix Mannhardt. Sepsis cases – event log, 2016. Available at https://doi.org/10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460.

[21] Tom M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[22] Alistair Moffat and Andrew Turpin. *Compression and Coding Algorithms*. Kluwer Academic Publishers, 2002.

[23] Rouven Poll, Artem Polyvyanyy, Michael Rosemann, Maximilian Röglinger, and Lea Rupprecht. Process forecasting: Towards proactive business process management. In *Proceedings of Business Process Management, BPM 2018, Sydney, NSW, Australia, September 9–14, 2018*, volume 11080 of *Lecture Notes in Computer Science*, pages 496–512. Springer, 2018.

[24] Artem Polyvyanyy, Hanan Alkhammash, Claudio Di Ciccio, Luciano García-Bañuelos, Anna A. Kalenkova, Sander J. J. Leemans, Jan Mendling, Alistair Moffat, and Matthias Weidlich. Entropia: A family of entropy-based conformance checking measures for process mining. In *Proceedings of International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4–9, 2020*, volume 2703 of *CEUR Workshop Proceedings*, pages 39–42. CEUR-WS.org, 2020.

[25] 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.

[26] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Process model abstraction: A slider approach. In *International IEEE Enterprise Distributed Object Computing Conference, EDOC 2008, Munich, Germany, September 15–19, 2008*, pages 325–331. IEEE Computer Society, 2008.

[27] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing complexity of large EPCs. In *Modellierung betrieblicher Informationssysteme - Modellierung zwischen SOA und Compliance Management, MobIS 2008, Saarbrücken, Germany, November 27–28, 2008*, volume P-141 of *LNI*, pages 195–207. GI, 2008.

[28] 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 Transactions on Software Engineering and Methodology*, 29(3):1–41, 2020.

[29] Andreas Rogge-Solti, Wil M. P. van der Aalst, and Mathias Weske. Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In *Business Process Management Workshops, BPM 2013, Beijing, China, August 26, 2013, Revised Papers*, volume 171 of *Lecture Notes in Business Information Processing*, pages 15–27. Springer, 2013.

[30] Anja F. Syring, Niek Tax, and Wil M. P. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Transactions on Petri Nets and Other Models of Concurrency*, 14:192–221, 2019.

[31] Stephen Chak Tornay. *Ockham: Studies and Selections*. La Salle, Ill., The Open Court Publishing Company, 1938.

[32] Wil M. P. van der Aalst. *Process Mining – Data Science in Action*. Springer, 2016.

[33] Wil M. P. van der Aalst. Process mining and simulation: A match made in heaven! In *Proceedings of Computer Simulation Conference, SummerSim 2018, Bordeaux, France, July 09–12, 2018*, pages 1–12. ACM, 2018.

[34] Wil M. P. van der Aalst. Relating process models and event logs – 21 conformance propositions. In *Proceedings of International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2018, Bratislava, Slovakia, June 25, 2018*, volume 2115 of *CEUR Workshop Proceedings*, pages 56–74. CEUR-WS.org, 2018.

[35] Wil M. P. van der Aalst. A practitioner's guide to process mining: Limitations of the directly-follows graph. *Procedia Computer Science*, 164:321–328, 2019.

[36] Jan Martijn E. M. van der Werf, Artem Polyvyanyy, Bart R. van Wensveen, Matthieu Brinkhuis, and Hajo A. Reijers. All that glitters is not gold – towards process discovery techniques with guarantees. In *Proceedings of Advanced Information Systems Engineering, CAiSE 2021, Melbourne, VIC, Australia, June 28–July 2, 2021*, volume 12751 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2021.

[37] Boudewijn B. F. Van Dongen. BPI challenge 2012 – event log, 2012. Available at https://doi.org/10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F.

[38] Boudewijn B. F. Van Dongen and Florian Borchert. BPI challenge 2018 – event log, 2018. Available at https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972.

[39] A. J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.