An Entropic Relevance Measure for Stochastic Conformance Checking in Process Mining

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

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

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

***Abstract*—Given an event log as a collection of recorded real-world process traces, process mining aims to automatically construct a process model that is both simple and provides a useful explanation of the traces. Conformance checking techniques are then employed to characterize and quantify commonalities and discrepancies between the log's traces and the candidate models. Recent approaches to conformance checking acknowledge that the elements being compared are inherently stochastic – for example, some traces occur frequently and others infrequently – and seek to incorporate this knowledge in their analyses.**

Here we present an *entropic relevance* measure for stochastic conformance checking, computed as the average number of bits required to compress each of the log's traces, based on the structure and information about relative likelihoods provided by the model. The measure penalizes traces from the event log not captured by the model and traces described by the model but absent in the event log, thus addressing both precision and recall quality criteria at the same time. We further show that entropic relevance is computable in time linear in the size of the log, and provide evaluation outcomes that demonstrate the feasibility of using the new approach in industrial settings.

I. Introduction

Process mining studies tools, methods, and techniques for improving real-world processes based on event data generated by historical process executions [1]. The core problem in process mining is that of automatically discovering a process model from a given *event log*, where an event log is a collection of traces, each capturing a sequence of observed process events. Such discovered models should faithfully encode the process behavior captured in the log and, hence, satisfy a range of criteria. Specifically: (1) a discovered model should describe as many as possible of the traces recorded in the log (good *recall*, or *fitness*); (2) should allow as few traces as possible that are not present in the log (good *precision*); and (3) should be as "*simple*" as is consistent with the other two goals (good *simplicity*). In tension with these three is a further objective: (4) the model should allow traces that may stem from the same process but are not present in the sample (good *generalization*). Conformance checking is a subarea of process mining that addresses the problem of measuring and characterizing the four quality criteria when using a discovered process model to explain the corresponding event log.

Classical process mining techniques often consider frequencies of traces in the logs and/or frequencies of events in the traces. However, the implications of such considerations usually stay hidden from the consumers of the artifacts that are produced. For instance, most existing discovery algorithms strive to construct process models that fit, i.e., can replay, frequent traces from the input event logs. However, the vast majority of the discovery techniques construct non-deterministic models in which routing decisions are equiprobable, significantly limiting the ability of the models to explain the behaviors of the true processes sampled via the event logs. Indeed, if traces in a log suggest that failure was observed in nine out of ten traces, a model that says that (only) every second trace is erroneous is of reduced utility, and potentially harmful to subsequent decision-making activities. By accumulating event data over a long time, the log approaches the true event and trace frequencies. If this information about the true process were able to be reflected in the discovered model, the model would generate better process simulations, and hence better predictions of future processes.

We refer to process mining endeavors that process and produce artifacts with explicit information on the likelihood of process events and traces collectively as *stochastic*, or *statistical*, *process mining*. For example, a discovery algorithm could construct a model with annotations of relative likelihoods of making routing decisions such that a large collection of traces induced by the model (by following the encoded stochastic decisions) would result in a probability distribution over traces that closely matches the distribution of log traces. We call such a model a *stochastic process model*. Before designing algorithms for discovering stochastic process models from event logs, we seek to understand which stochastic models can be considered to be "good" explanations of a given log.

Hence this paper, in which we present a novel technique for stochastic conformance checking called *entropic relevance*, or *relevance*. Given a log, the entropic relevance of a stochastic process model is the average number of bits used to compress [2] a trace from the log using the relative likelihoods induced by the model. The fewer bits are used, the better the model "explains" the event log, i.e., the closer the relative likelihoods of traces derived from the model to those obtained from the event log. Log traces that do not fit the model are penalized by encoding them using a more expensive background model. Entropic relevance also penalizes traces that fit the model but are not present in the log, as these reduce the likelihoods of the log traces that fit the model, and increase the length (in terms of bits) of their compressed forms. Hence, an entropic relevance measurement reflects a compromise between the

Postprint, August 2020

precision and recall quality criteria. Such duality is indeed expected in the case of stochastic conformance checking. Finally, given a model and log, relevance is computable in time linear in the size of the log, making it – as we demonstrate below – useful when evaluating the quality of the process discovery algorithms commonly used in the industry.

The remainder of the paper proceeds as follows. The next section presents an overview of the compression methodology we employ, and describes how an event log can be encoded via a stochastic process model. Section III discusses formal stochastic models used in process mining. Based on these models, Section IV presents the notion of entropic relevance. Section V discusses the results of our evaluation of the entropic relevance measure, while Section VI summarizes related work.

II. Motivation and Overview

We employ a minimum description length compression-based framework to measure the quality of process models. Two key observations make this possible: 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 signal benefit of this *entropic relevance* approach is that not only is the model *structure* an influence on its measured usefulness, but also the *probability* of each individual trace having emerged from the model. A relationship fundamental to information theory is critical to understanding the new approach: if a symbol e of probability $p(e)$ occurs, then the information conveyed by that occurrence is $-\log_2 p(e)$ bits, and hence that is also the minimum number of bits required to describe any instance of e. For example, if e has probability $p(e) = 0.8$, then each e that occurs in a stream of such symbols has a cost of 0.3219 bits attributable to it. Practical compression systems operate very close to these entropy-based limits, see, for example, Moffat and Turpin [3, Chapter 5]; and the information-theoretic relationship between probabilities and bits is an achievable one.

Fig. 1 provides an overview of our proposal. In the figure it is supposed that an event log has been provided, sampled from an underlying "true" (but unknown) process; and that two process models are being considered as alternative explanations for that set of observations. If each of the two models assigns a calculable probability to every possible sequence of process states that might occur, then computing $\sum_t -\log_2 p(t \mid M)$ values, where the summation is over the traces t in the log and $p(t \mid M)$ is the probability assigned to trace t by model M, results in a value that encapsulates the information content of the whole log, given M. The smaller that value, the shorter the compressed output would be were it to be generated, and the better the model matches the log. (That is, while Fig. 1 suggests that actual compression occurs, it is the *size* of the output that is of interest, and not the actual bits that arise.)

Fig. 2 provides more details of the compression process, considering each trace in the log. The stochastic process model

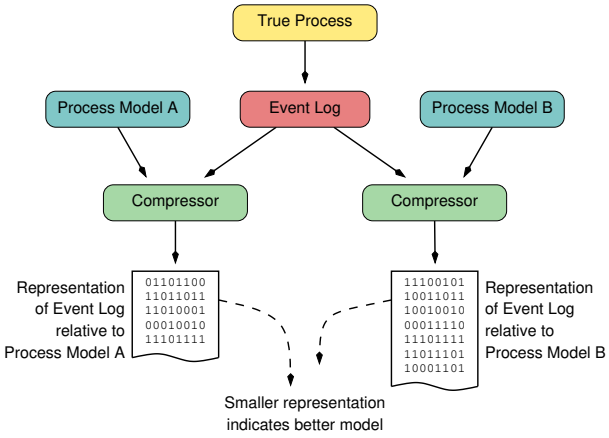

Fig. 1: To measure the entropic relevance of a process model to a collection of traces, the model's structure and probabilities are used to compute the cost in bits of losslessly representing the traces relative to the model model. Better models lead to a shorter compressed forms.

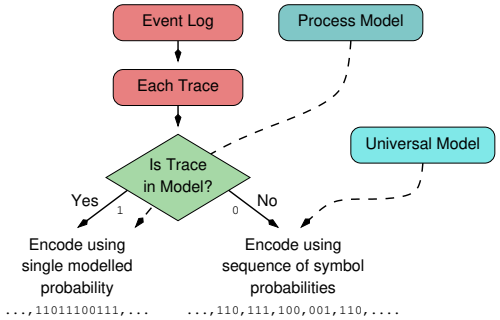

Fig. 2: The two possible options when encoding a set of traces with respect to a probabilistic model: each trace either has a non-zero probability in the model, which can be used to derive a bitstring; or it is spelled-out as symbol-by-symbol codes via a universal model.

(defined in detail in Section III) assigns calculable non-zero probabilities to a finite or infinite subset of the universe of possible traces, rather than to every possible member of that universe; and hence assigns a probability of zero to each of the infinite number of possible state sequences in the complement subset. If some particular trace in the log fits the model, it can be coded as single entity, using its corresponding end-to-end probability in the model. On the other hand, the traces with a probability of zero according to the model must be "spelled out" on a symbol-by-symbol basis, using a background (or *universal*) model in which every possible state always has a non-zero probability, and hence in which every possible sequence of states can always be coded. To choose between these two cases, the output associated with every trace is prefixed by a code – a (biased) 0 or 1 bit – that indicates which option applies. Given such an encoding and the stochastic and background models, one can always reconstruct the original log by applying the reverse procedure.

Fig. 3 steps back from the detail and provides a high-level view of the proposed mechanism. 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

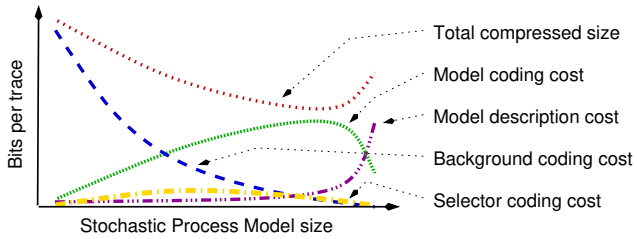

Fig. 3: Schematic showing the total compressed size of a collection of traces relative to a model as the sum of four components: the cost of describing the model and its parameters; the cost of entropy coding the traces that fit the model; the cost of entropy coding the traces that do not fit the model,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

end of the figure, if the process model is small and easily described, it likely fits only a small fraction of the log's traces. The majority of traces, the ones that do not fit the model, are coded using the more expensive background model, and it dominates to total cost. As the process model becomes larger and more sophisticated, it fits a greater fraction of the traces, and the balance shifts from the background model to the more economical stochastic process model. The total compressed size decreases 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 over-fitted to the traces in the particular log. 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 will have increased, because of the complexity and detail required in the description of the process model.

In terms of Fig. 3, we define (see Section IV for full details) entropic relevance to be 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) as a second dimension, shown as 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.

Entropic relevance (again, in anticipation of Section IV) 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, it 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 at the beginning of Section I 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. 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 increased net bit costs.

III. Models of Stochastic Languages

This section introduces the notion of a stochastic language and several models, theoretical and those used in practice, that aim at encoding stochastic languages. The notion and models are used in the subsequent formal discussions and explanations of the conducted empirical evaluations.

A. Stochastic Languages

A *language* is a, possibly infinite, collection of finite sequences of *symbols*. These sequences are often referred to as *words*. In this work, we use words to encode observed processes. Hence, we refer to words as (process) *traces* composed of *actions* rather than symbols. Let Λ be a universe of *actions*. Then, Λ^* is the set of all traces over Λ. By ϵ, $\epsilon \in \Lambda^*$, we denote the empty trace. For example, set $X = \{\epsilon, \mathtt{a}, \mathtt{ab}, \mathtt{abc}, \mathtt{abcd}, \mathtt{abcdd}, \mathtt{abcde}\}$ defines a language of seven traces; we write $\mathtt{abc}$ to denote sequence $\langle \mathtt{a}, \mathtt{b}, \mathtt{c} \rangle$ when the context is clear.

A *stochastic language* is an assignment of probabilities to traces so that the assigned probabilities sum up to one, i.e., a stochastic language is a probability density function over traces. For example, 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. A stochastic language is defined as follows.

Definition III.1 (Stochastic language):
A *stochastic language* L is a function $L : \Lambda^* \to [0, 1]$ for which it holds that:

$$\sum_{\sigma \in \Lambda^*} L(\sigma) = 1.0.$$

⌟

For example, $L_1 = \{(\epsilon, 0.5), (\mathtt{a}, 0.25), (\mathtt{ab}, 0.125), (\mathtt{abc}, 1/16), (\mathtt{abcd}, 1/32), (\mathtt{abcdd}, 1/64), (\mathtt{abcde}, 1/64)\} \cup \bigcup_{t \in \Lambda^* \setminus X} \{(t, 0.0)\}$, where set X is specified above. Given a trace $t \in \Lambda^*$ and a stochastic language L, $L(t)$ specifies the relative likelihood of a randomly drawn trace to be equal to t.

By $\hat{L}$, we denote the set of all traces possible according to L, i.e., $\hat{L} := \{t \in \Lambda^* \mid L(t) > 0.0\}$. We say that L is *finite* if and only if $\hat{L}$ is finite; otherwise L is infinite. It holds that $\hat{L}_1 = X$, i.e., the traces in X are all the possible traces according to L_1, and thus L_1 is finite.

B. Stochastic Deterministic Finite Automata

A stochastic deterministic finite automaton (SDFA) can be used to encode a stochastic language; here we adopt the definition of an SDFA from Carrasco [4].

Definition III.2 (Stochastic deterministic finite automaton): A *stochastic deterministic finite automaton* (SDFA) is a tuple $(S, \Delta, \delta, p, s_0)$, where S is a finite set of *states*, $\Delta \subseteq \Lambda$ is a set of *actions*, $\delta : S \times \Delta \to S$ is a *transition function*, $p : S \times \Delta \to [0, 1]$ is a *transition probability function*, $s_0 \in S$ is the *initial state*, and for each state $s \in S$ it holds that $\sum_{\lambda \in \Delta} p(s, \lambda) \leq 1.0$. ⌟

By $\mathcal{A}$, we denote the universe of SDFAs.

Fig. 4 shows an SDFA using graphical notation. In this notation, the states and transition function are visualized as circles and arcs, respectively. For instance, the SDFA shown in Fig. 4 has seven states $s_0 \ldots s_6$, and its transition function is defined by $\{(s_0, \mathtt{a}, s_1), (s_1, \mathtt{b}, s_2), (s_2, \mathtt{c}, s_3), (s_3, \mathtt{d}, s_4), (s_4, \mathtt{d}, s_5), (s_4, \mathtt{e}, s_6)\}$. Arcs are labeled by actions and transition probabilities. Hence, the arc from state s_0 to state s_1 with label "$\mathtt{a}(1/2)$" specifies that $(s_0, \mathtt{a}, s_1) \in \delta$ and $(s_0, \mathtt{a}, 0.5) \in p$. Consequently, the transition probability function is defined by $\{(s_0, \mathtt{a}, 0.5), (s_1, \mathtt{b}, 0.5), (s_2, \mathtt{c}, 0.5), (s_3, \mathtt{d}, 0.5), (s_4, \mathtt{d}, 0.25), (s_4, \mathtt{e}, 0.25)\}$. State s_0 is the initial state and, hence, is denoted by an arrow leading to it.

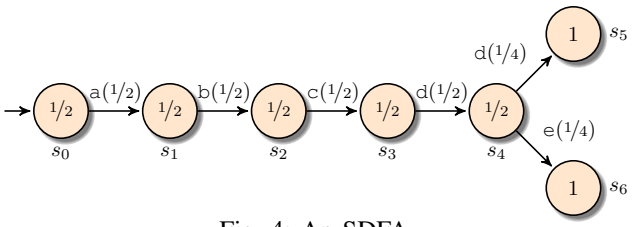

Fig. 4: An SDFA.

Given two traces $t_1, t_2 \in \Lambda^*$, by $t_1 \circ t_2$, we denote their concatenation, i.e., the trace obtained by joining t_1 and t_2 end-to-end. For example, it holds that $\mathtt{tr} \circ \mathtt{ace} = \mathtt{trace}$.

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

$$\pi_A(s, \epsilon) := 1.0 - \sum_{\lambda \in \Delta} p(s, \lambda), \text{and}$$
$$\pi_A(s, \lambda \circ t') := p(s, \lambda)\, \pi_A(\delta(s, \lambda), t'), \lambda \in \Lambda, t = \lambda \circ t'.$$

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 SFDA A from Fig. 4, it holds that $\pi_A(s_i, \epsilon) = 0.5$, $i \in [0\,..\,4]$, and $\pi_A(s_j, \epsilon) = 1.0$, $j = 5$ or $j = 6$.

If L is a stochastic language encoded by some SDFA, we say that L is a *regular stochastic language*. The SDFA in Fig. 4 encodes stochastic language L_1 discussed in Section III-A and, thus, L_1 is regular.

C. Event Logs

An *event log* is a finite collection of events that relate to a process and are distinguished by their attributes and attribute values. Usually, events in an event log encode information about actions executed by software systems that support a business process of an organization. In general, an event can have arbitrary attributes, but three attributes are common in process mining. These are the *case identifier*, *timestamp*, and the *action identifier* attribute. The value of the case identifier attribute of an event relates this event to a case, or instance, of the process; i.e., all events with the same case identifier stem from the same instance of the business process. The time of occurrence of an event is stored in its timestamp attribute. Finally, the action identifier attribute is used to store information about an action that induced the event.

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 with the same case identifier as a trace of corresponding actions (obtained via the action identifier attribute) arranged in the ascending order of the event timestamps. Finally, as there can be several case identifiers that induce the same trace (indeed, several business 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 III.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, $E_1 = [\epsilon^{32}, \mathtt{a}^{16}, \mathtt{ab}^{8}, \mathtt{abc}^{4}, \mathtt{abcd}^{2}, \mathtt{abcdd}^{1}, \mathtt{abcde}^{1}]$ and $E_2 = [\epsilon^{250}, \mathtt{ab}^{250}, \mathtt{abc}^{250}, \mathtt{abcd}^{50}, \mathtt{abce}^{50}, \mathtt{abcde}^{50}, \mathtt{abced}^{50}, \mathtt{abcdde}^{50}]$ are two logs, i.e., $E_1, E_2 \in \mathcal{E}$. Trace $\mathtt{abc}$ occurs in E_1 four times, while in E_2 it is recorded 250 times.

An event log is inherently stochastic. By accumulating a large number of traces, and perhaps over an extended period of time, an event log aims to approach their true underlying probability distribution. Let X be a random variable and let O be a multiset of observations. By $P(X = x \mid O)$, or $P(x \mid O)$ when the context is clear, we denote the estimate based on O of the probability of observing X to be equal to element x. 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 $t \in \Lambda^*$. In this work, we use the maximum likelihood estimation, i.e., $P(t \mid E) := {}^{m_E(t)}/_{|E|}$, where $m_E(t)$ denotes the multiplicity of element t in multiset E. Therefore, L_1 from Section III-A is the stochastic language of event log E_1 from above. Note that the stochastic language L_2 of event log E_2 is given my the function with these non-zero values $\{(\epsilon, 0.25), (\mathtt{ab}, 0.25), (\mathtt{abc}, 0.25), (\mathtt{abcd}, 0.05), (\mathtt{abce}, 0.05), (\mathtt{abcde}, 0.05), (\mathtt{abced}, 0.05), (\mathtt{abcdde}, 0.05)\}$.

D. Frequency Directed Action Graphs

A common approach for representing event logs for consumption and decision making by practitioners is by encoding them into Directly-Follows Graphs (DFGs) [1]. 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 $\mathtt{a}$ to action $\mathtt{b}$ iff E contains a trace $t_1 \circ \mathtt{ab} \circ t_2$ where $t_1, t_2 \in \Lambda^*$ [5].

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 inform the decision-making practices. To describe such filtered graphs mathematically, we introduce the notion of a *frequency directed action graph*.

Definition III.4 (Frequency directed action graph): A *frequency directed action graph* (FDAG) is a tuple $(\Phi, \Psi, \phi, \psi, i, o)$, where $\Phi \subseteq \Lambda$ 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 $i \notin \Lambda$ and $o \notin \Lambda$ are the input and the output of the graph, respectively. ⌟

By $\mathcal{G}$, we denote the universe of FDAGs.

Fig. 5 shows an example FDAG. In the figure, boxes with rounded corners represent actions, whereas 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 encrypted next to the respective actions and arcs.

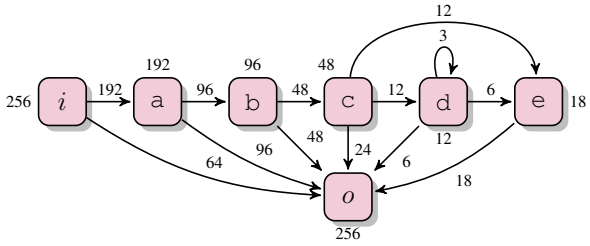

Fig. 5: An FDAG.

Van der Aalst [1] 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. He then discusses several pitfalls this phenomenon can lead to in practice. We agree with those observations, and, for our purpose, fix one such possible interpretation. To avoid ambiguities, next, we define our interpretation rigorously as a mapping from a given FDAG to the corresponding SDFA.

Definition III.5 (SDFA of FDAG): Let $G := (\Phi, \Psi, \phi, \psi, i, o)$ be an FDAG. Then, $(S, \Delta, \delta, p, s_0)$, where $S = \Phi \cup \{i\}$, $\Delta = \Phi$, $\delta = \{(s,t,t) \in (\{i\} \cup \Phi) \times \Phi \times \Phi \,|\, (s,t) \in \Psi\}$, $p = \{(s,t,x) \in (\{i\} \cup \Phi) \times \Phi \times [0,1] \,|\, (s,t) \in \Psi \wedge x = \psi(s,t) / \sum_{(s,u) \in \Psi} \psi(s,u)\}$, and $s_o = i$, is the SDFA of G, denoted by $SDFA(G)$. ⌟

Thus, if $A = SDFA(G)$, then, according to our interpretation, G encodes the possible traces of L_A, i.e., traces $\hat{L}_A$, and the relative frequency of each trace $t \in \hat{L}_A$ is given by $L_A(t)$.

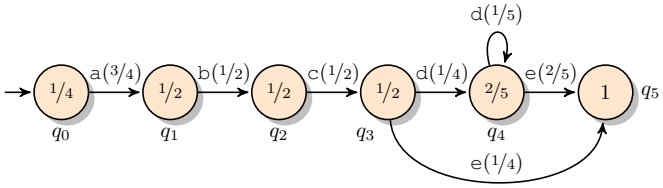

Fig. 6: An SDFA.

For example, Fig. 6 shows the SDFA of the FDAG from Fig. 5 that encodes infinite stochastic language L_2 which, among others, assigns these non-zero values to traces $\{(\epsilon, 1/4), (\texttt{a}, 3/8), (\texttt{ab}, 3/16), (\texttt{abc}, 3/32), (\texttt{abcd}, 3/160), (\texttt{abce}, 3/64), (\texttt{abcdd}, 3/800), (\texttt{abcde}, 3/160), (\texttt{abcddd}, 3/4000), (\texttt{abcdde}, 3/800)\}$; all the other possible according to L_2 traces have the cumulative relative frequency of $9/8000$.

IV. Entropic Relevance

This section gives a precise definition of *entropic relevance*. Given an event log $E \in \mathcal{E}$ and an SDFA A, trace $t \in E$ is either a possible trace according to the stochastic language of A, or is not. This distinction determines how the number of bits required to encode t is computed. If t is possible, then it is presumed to be encoded using the probability of t according to A, i.e., $L_A(t)$. Otherwise, t is presumed to be encoded using background knowledge about the log. These two modes are captured in this next definition.

Definition IV.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, and let $bits : \Lambda^* \times \mathcal{E} \times \mathcal{A} \to \mathbb{R}^+$ be a function that maps any and every sequence $t \in \Lambda^*$ to the number of bits required to uniquely encode t, given a viable encoding that assumes knowledge of E and A. The *trace compression cost of t in the presence of E and A* is defined:

$$cost_{bits}(t, E, A) := \begin{cases} -\log_2(L_A(t)) & t \in \hat{L_A} \\ bits(t, E, A) & \text{otherwise.} \end{cases}$$

⌟

For simplicity, in this first presentation we use a straightforward background model to measure $bits(t, E, A)$, ignoring both E and A, and trivially encoding t by taking each individual action as an equi-probable symbol over the underlying alphabet augmented by an "end of string" symbol, and with each trace terminated by that special symbol: $bits(t, E, A) := (1 + |t|) \log_2(1 + |\Lambda|)$. (We anticipate future exploration of more sophisticated encoding schemes that explore partial embeddings of traces into finite-context automata, and/or make use of the relative frequencies of the actions present in E.)

Given that definition, let A_1 and A_2 be SDFAs shown in Fig. 4 and Fig. 6, respectively. Then, based on logs E_1 and E_2 from Section III-C, the compression costs of traces `abcd` and `abce` are (in bits):

1) $cost_{bits}(\texttt{abcd}, E_1, A_1) = -\log_2(L_{A_1}(\texttt{abcd})) = -\log_2(1/32) = 5.00$;
2) $cost_{bits}(\texttt{abcd}, E_1, A_2) = -\log_2(L_{A_2}(\texttt{abcd})) = -\log_2(3/160) = 5.74$;
3) $cost_{bits}(\texttt{abce}, E_2, A_1) = (1+|\texttt{abce}|)\log_2(1+|\{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e}\}|) = 12.93$;

4) $cost_{bits}(\texttt{abce}, E_2, A_2) = -\log_2\left(L_{A_2}(\texttt{abce})\right) = -\log_2(3/64) = 4.42$.

Hence, A_1 compresses trace `abcd` better than does A_2, while A_2 compresses `abce` much better than A_1. In particular, `abce` is impossible according to A_1, and is encoded using the $bits(\cdot,\cdot,\cdot)$ function (case 3). That encoding needs five symbols: four actions and an "end of string" symbol, each taking $\log_2 6$ bits because the alphabet in E_2 contains five symbols, and an "end of string" symbol must also be included.

Also needed in the nominal compression cost is the "selector" associated with the diamond decision box in Fig. 2, and illustrated by the yellow line in Fig. 3. If the probability associated with a two-choice event is p, the expected cost of coding a stream of such choices is given by $H_0(p) := -p\log_2(p) - (1-p)\log_2(1-p)$; with, by definition, $H_0(0.0) := H_0(1.0) := 0.0$. These considerations lead directly to our main definition.

Definition IV.2 (Entropic relevance):
Let $E \in \mathcal{E}$ be an event log and let $A \in \mathcal{A}$ be an SDFA. Let $\rho(E, A)$ be the overall probability that a trace in E is possible in the stochastic language of A, $\rho(E, A) = \sum_{t \in \hat{L_A}} P(t \mid E)$. Then, the *entropic relevance of A to E*, or *relevance of A to E*, is denoted by $rel(E, A)$ and defined as:

$$rel(E, A) := H_0(\rho(E, A)) + \frac{1}{|E|}\sum_{t \in E} cost_{bits}(t, E, A)\,.$$

⌟

Fig. 2, presented earlier, explains Definition IV.1 and Definition IV.2. During the nominal encoding process, each trace t in the log is considered in turn. If t has a non-zero probability in the model, 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 IV.1). If the probability of t is zero according to the model (the right branch in Fig. 2), the opposite selector code is needed, and then $bits(t, E, A)$ bits are used to represent t on a symbol-by-symbol basis in the universal background model (the second option in Definition IV.1). The selector codes associated with the diamond decision box, needed to differentiate between the two alternatives on a per-trace basis, add an average of $H_0(\rho(E, A))$ bits per trace.

Again, consider automata A_1 and A_2 from Fig. 4 and Fig. 6, respectively, and event logs E_1 and E_2 from Section III-C. Table I summarizes the constituent costs and the resulting entropic relevance values for the four combinations. In the first row, SDFA A_1 explains event log E_1 perfectly, as the traces in the log all fit the automaton, and the relative likelihoods of observing the traces in the log and in the automaton are identical. No other automaton can achieve lower relevance for E_1, and A_1 is *optimal*.

Automaton A_2 explains E_1 reasonably well, but with an increased model coding cost because of mis-matched probabilities for the empty trace and for the one-action trace `a` ($1/2$ vs $1/4$ for the empty trace, and $1/4$ vs $3/8$ for `a`).

The probability $\rho(E_2, A_1)$ that a trace from E_2 is possible according to L_{A_1} is 0.85, and hence $H_0(\rho(E, A)) = 0.61$ bits.

Autom.	Log	ρ	Select.	Bckgrd.	MdlCst.	Relevance
A_1	E_1	1.00	0.00	0.00	1.97	1.97
A_2	E_1	1.00	0.00	0.00	2.26	2.26
A_1	E_2	0.85	0.61	2.33	2.55	5.49
A_2	E_2	0.95	0.29	0.78	3.15	4.22

TABLE I: Entropic relevance (in bits) and its constituents for two example automata and two event logs. The columns list the probability of traces from the log being possible according to the automata (ρ); and the average (over all that log's traces) selector coding cost (Select.), background coding cost (Bckgrd.), and model coding cost (MdlCst.). The final column sums those three to get the entropic relevance.

That is, the choice between the background model and A_1 adds $-\log_2(\rho) = 0.23$ bits to the traces in E_2 that are possible according to L_{A_1}; adds $-\log_2(1-\rho) = 2.74$ bits for traces that are not possible; and averages at 0.61 bits per trace. The relevance of A_1 to E_2 is then obtained by adding in the arithmetic mean of the trace compression costs of the traces in E_2. The smallest compression cost arises for the empty trace (2.59 bits, part of the model coding cost), while the highest cost is associated with `abcdde` (18.09 bits, part of the background coding cost). Overall, A_1 can be used to compress E_2 using, on average, 5.49 bits per trace.

In the last row of the table, A_2 "explains" E_2 using 4.22 bits per trace, and hence A_2 is a better model for E_2 than is A_1. First, note that A_2 fits more of E_2's traces than does A_1, and with $\rho(E_2, A_1) = 0.85$ and $\rho(E_2, A_2) = 0.95$, the average cost of selecting between the process model and background model is less for A_2 than for A_1. In direct correspondence, the share of relevance stemming from the background coding cost is also less for A_2 than for A_1. Finally, despite the fact that more traces from E_2 fit A_2 than A_1, which pushes A_2's model coding cost higher, the overall cost of using A_2 is less.

V. Evaluation

To study the usefulness of entropic relevance, we used event logs from real-world IT-systems made publicly available by the IEEE Task Force on Process Mining.[1] For each log, directly follows models (DFMs, also known as DFGs) were constructed using the approach of Leemans et al. [6], with trace removal thresholds of $1/100$, $2/100$, …, $100/100$. Fig. 7 plots relevance values (and constituents) for three logs: Road Traffic Fine Management [7], Sepsis Cases [8], and BPI Challenge 2012 [9]. The *size* of each model (the horizontal axis) is taken to be the number of states plus the number of edges.

For the first two logs, the curves correspond to the anticipation provided by Fig. 3. In both cases 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) is small over most of the range. Entropic relevance settles at around 2.5 bits per trace for Traffic Fines, indicating that the log is highly regular and hence highly compressible; and at around 30 bits per trace for Sepsis.

[1] https://data.4tu.nl/repository/collection:event_logs_real.

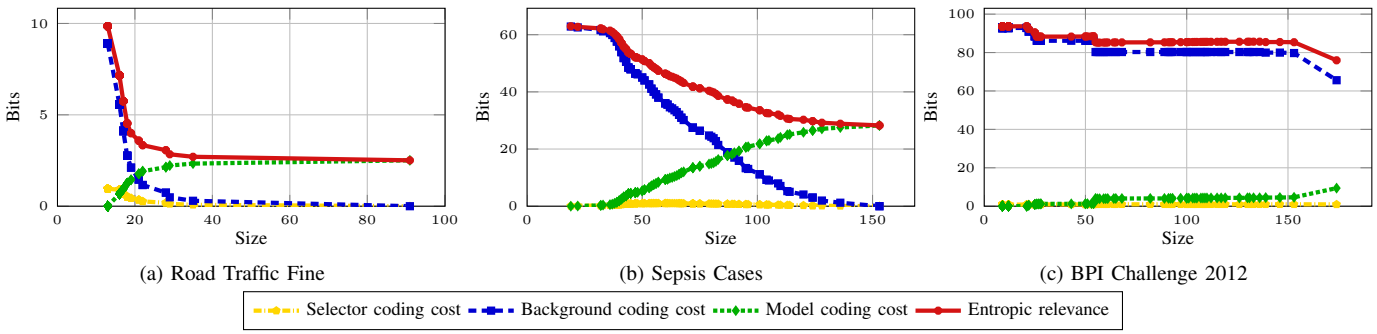

Fig. 7: Entropic relevance and its constituents, plotted as a function of model size measured as states plus edges.

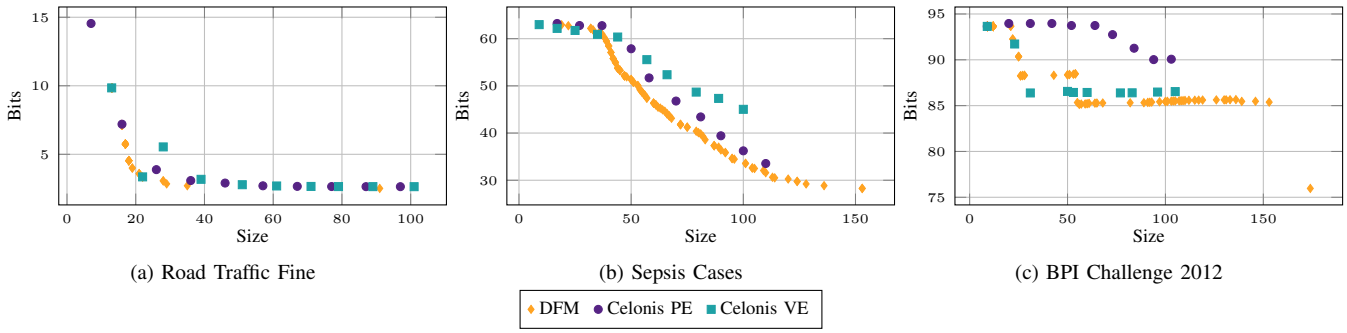

Fig. 8: Entropic relevance, again plotted as a function of model size, for three different techniques.

In the third log in Fig. 7, a different picture emerges. Now the background model dominates the entropic cost and relevance values are high, a consequence of long traces that tend not to repeat in the log. Changing to a more nuanced background model – based on action occurrence frequencies and a zero-order model, for example – would lower the relevance values, but not decrease the model's reliance on them. In the context of the framework introduced here, the traces in this log are relatively inconsistent, and hence incompressible.

Celonis SE (https://www.celonis.com) granted us access to DFGs they constructed using "Celonis Snap", the free version of their enterprise-grade product. For each of the logs in Fig. 7, they generated twenty DFGs using two different techniques (PE and VE) and ten configurations for each technique. These DFGs were transferred to us on July 6, 2020. Fig. 8 plots relevance values in bits for these DFGs, again as a function of model size, plus the DFMs already used in Fig. 7. Now models, and hence methods, can be compared by considering the Pareto frontier of the points plotted in each graph. For these logs the relevance approach identifies the methods of Celonis SE as being preferable when the goal is to have small- to moderate-sized DFGs. However, note that the numeric relevance values are subject to the approach used to map DFGs to SDFAs and to the choice of background coding model. Detailed analysis of these interactions and of the DFGs 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 for computing relevance using our implementation on a commodity laptop computer for the largest analyzed log (BPI Challenge 2018 – Payment application; 43,809 traces and 984,613 events) over the 100 constructed DFMs (size ranged 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 the model description costs (see Fig. 3) were small compared to the entropic relevance scores. Our tool [10] and dataset [11] used to conduct the experiments are publicly available.

VI. Related Work

A plethora of non-stochastic process discovery and conformance checking techniques have been proposed over the last two decades, including the Genetic Mining [12], Heuristic Mining [13], Inductive Mining [14], and Split Mining [15] algorithms, all of which have been well-received by the process mining community. Non-stochastic conformance checking techniques can be broadly classified into *quantitative*, those that summarize conformance diagnostics into a single number, and *qualitative*, those that construct detailed analytics of commonalities and discrepancies between model and log traces. Carmona et al. [16] provide a useful overview.

Recently, Van der Aalst et al. [17], [18] initiated discussion of desired properties for 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 [19], including the strict monotonicity properties.

The selection of currently available stochastic process mining techniques is rather scarce. To the best of our knowledge,

there are two stochastic discovery techniques proposed by academia. The technique presented by Rogge-Solti et al. [20] discovers stochastic Petri nets, while that of Leemans et al. [6] can be used to discover DFGs, and was employed in Section V. There are many commercial tools for discovering DFGs, or FDAGs, from event logs, but these are all closed source.

Two stochastic conformance checking techniques have been proposed. Leemans et al. [21] 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 [22] is inspired by entropy-based conformance checking [19]. Leemans and Polyvyanyy [22] 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 defined here, reflects the compromise between precision and recall in a single value with meaningful units. Exploration of the relationships and correlations between these two previous measures and ours, and understanding of their differences, is an area for future work.

Finally, note that our proposal is an application of the *minimum description length* principle [23], [24]; 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 [2]. Kolmogorov complexity formalizes the notion widely known as "Occam's Razor" [25], 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.

VII. Conclusion

We have presented an entropic relevance measure for stochastic conformance checking. The new measure is grounded in a minimum description length compression-based framework that assesses how accurately a stochastic process model describes an event log by computing the length of an encoding of the log traces relative to the stochastic language expressed by the model. The relevance of a model to a given log reflects a compromise between the precision and recall quality criteria, and is computable in time linear in the size of the log. Relevance is measured in bits, with values being directly interpretable, and with small scores being preferable.

Future work will investigate the effects of using different background models for calculating entropic relevance, with the aim of identifying models that lead to useful relevance measurements; noting that background model cost is one of the three components of the relevance calculation, and in some cases is dominant. We also plan to explore new techniques for discovering stochastic process models in a direct response to entropic relevance. Now that we have defined entropic relevance as a useful quantity, an important next step is to explicitly design models that seek to minimize it.

Acknowledgment. Artem Polyvyanyy was in part supported by the Australian Research Council project DP180102839. Hanan Alkhammash and Thomas Vogelgesang provided assistance with the preparation of the datasets used in Section V.

References

[1] W. M. P. van der Aalst, "A practitioner's guide to process mining: Limitations of the directly-follows graph," *Procedia Computer Science*, vol. 164, 2019.
[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.
[3] A. Moffat and A. Turpin, *Compression and Coding Algorithms*. Boston, MA: Kluwer Academic Publishers, Feb. 2002.
[4] R. C. Carrasco, "Accurate computation of the relative entropy between stochastic regular grammars," *ITA*, vol. 31, no. 5, 1997.
[5] W. M. P. van der Aalst, *Process Mining—Data Science in Action*, 2nd ed. Springer Berlin Heidelberg, 2016.
[6] S. J. J. Leemans, E. Poppe, and M. T. Wynn, "Directly follows-based process mining: Exploration & a case study," in *ICPM*. IEEE, 2019.
[7] M. De Leoni and F. Mannhardt, "Road traffic fine management process," 2015.
[8] F. Mannhardt, "Sepsis cases – event log," 2016.
[9] B. B. F. Van Dongen, "BPI challenge 2012," 2012.
[10] A. Polyvyanyy, H. Alkhammash, C. Di Ciccio, García-Bañuelos, A. Kalenkova, S. J. J. Leemans, J. Mendling, A. Moffat, and M. Weidlich, "Entropia: A Family of Entropy-Based Conformance Checking Measures for Process Mining," in *CoRR*, vol. abs/2008.09558, 2020.
[11] H. Alkhammash, A. Polyvyanyy, A. Moffat, and García-Bañuelos, "Discovered Process Models 2020-08," 8 2020. [Online]. Available: https://doi.org/10.26188/12814535
[12] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic process mining: An experimental evaluation," *Data Min. Knowl. Discov.*, vol. 14, no. 2, 2007.
[13] A. J. M. M. Weijters and W. M. P. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," *Integrated Computer-Aided Engineering*, vol. 10, no. 2, 2003.
[14] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs - A constructive approach," in *Petri Nets 2013*, ser. LNCS, vol. 7927. Springer, 2013.
[15] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, no. 2, 2018.
[16] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking—Relating Processes and Models*. Springer, 2018.
[17] W. M. P. van der Aalst, "Relating process models and event logs—21 conformance propositions," in *ATAED*, ser. CEUR Workshop Proceedings, vol. 2115. CEUR-WS.org, 2018.
[18] A. F. Syring, N. Tax, and W. M. P. van der Aalst, "Evaluating conformance measures in process mining using conformance propositions," *Trans. Petri Nets Other Model. Concurr.*, vol. 14, 2019.
[19] A. Polyvyanyy, A. Solti, M. Weidlich, C. D. Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *ACM TOSEM*, vol. 29, no. 3, 2020.
[20] A. Rogge-Solti, W. M. P. van der Aalst, and M. Weske, "Discovering stochastic petri nets with arbitrary delay distributions from event logs," in *BPM Workshops*, ser. LNBIP, vol. 171, 2013.
[21] S. J. J. Leemans, A. F. Syring, and W. M. P. van der Aalst, "Earth movers' stochastic conformance checking," in *BPM Forum*, ser. LNBIP, vol. 360. Springer, 2019.
[22] S. J. J. Leemans and A. Polyvyanyy, "Stochastic-aware conformance checking: An entropy-based approach," in *CAiSE*, ser. LNCS, vol. 12127. Springer, 2020.
[23] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, 1998.
[24] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," *J. Am. Stat. Assoc.*, vol. 96, no. 454, 2001.
[25] S. C. Tornay, *Ockham: Studies and Selections*. La Salle, Ill., The Open Court Publishing Company, 1938.