Noname manuscript No.
(will be inserted by the editor)

Entropy as a Measure of Log Variability

Christoffer Olling Back · Søren Debois · Tijs Slaats

Received: date / Accepted: date

Abstract Process mining algorithms fall in two classes: imperative miners output flow-diagrams, showing all possible paths, whereas declarative miners output constraints, showing the rules governing a process. But given a log, how do we know which of the two to apply? Assuming that logs exhibiting a large degree of *variability* are more suited for declarative miners, we can attempt to answer this question by defining a suitable measure of the variability of the log. This paper reports on an exploratory study into the use of *entropy measures* as metrics of variability. We survey notions of entropy used, e.g., in physics; we propose variant notions likely more suitable for the field of process mining; we provide an implementation of every entropy notion discussed; and we report entropy measures for a collection of both synthetic and real-life logs. Finally, based on anecdotal indications of which logs are better suited for declarative/imperative mining, we identify the most promising measures for future studies. For estimating overall entropy, global-block and k-nearest neighbour estimators of entropy appear most promising and excel at identifying noise in logs. For estimating entropy *rate* we identify Lempel-Ziv and certain variants of k-block estimators performing well, and note that the former is more stable, but sensitive to noise, while the latter is less stable, being sensitive to cutoff constraints determining block size.

This work is supported by the Hybrid Business Process Management Technologies project (DFF-6111-00337) funded by the Danish Council for Independent Research.

Christoffer Olling Back
University of Copenhagen, Department of Computer Science
Universitetsparken 1
DK-2100 Copenhagen Ø
E-mail: back@di.ku.dk

Søren Debois
IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
E-mail: debois@itu.dk

Tijs Slaats
University of Copenhagen, Department of Computer Science
Universitetsparken 1
DK-2100 Copenhagen Ø
E-mail: slaats@di.ku.dk

Changes This paper is an extension of earlier work presented at the First Workshop on Business Process Innovation with Artificial Intelligence [7]. The present submission expands substantially on its predecessor by including:

1. A discussion of assumptions regarding ergodicity and stationarity in the Business Process Management (BPM) context (Section 3.4).
2. A treatment of edit-distance based entropy approximation (Section 3.6).
3. A discussion of the use of entropy (H) vs. entropy rate (h) (Section 4).
4. A treatment of the Lempel-Ziv entropy rate approximation (Section 4.2).
5. An open source implementation of these new measures (Section 5).
6. Experimental evaluation of both old and the above new measures on a selection of publicly available real-life logs (Section 5).
7. Experimental evaluation of all measures on artificial logs to test the effect of different modelling paradigms, concurrency and noise (Section 5).
8. A discussion of the experiments leading to a tenuous recommendation of potentially helpful measures. (Section 6)

Keywords Process Mining · Hybrid Models · Process Variability · Process Flexibility · Information Theory · Entropy · Knowledge Work

1 Introduction

Two opposing lines of thought can be identified in the literature on process modelling notations. The *imperative* paradigm, including notations such as Petri nets [2] and BPMN [32], focuses on describing the flow of a process and is considered to be well-suited to structured processes with little variation. The *declarative paradigm*, including notations such as Declare [33], DCR Graphs [14], and GSM [24] focuses on describing the rules of a process and is considered to be well-suited to unstructured processes with large degrees of variation.

For processes with great variability, declarative miners are often at an advantage: the many possible paths through such a process may be succinctly represented by a small number of constraints, whereas an imperative miner must produce an impossible to read "spaghetti model" explicitly showing all these many paths. Conversely, for processes with great regularity, imperative miners are often at an advantage: a small number of explicit paths describes the process concisely, whereas a large and obtuse set of declarative constraints is necessary to capture that exact set of paths.

In this paper, we are motivated by the following question, first identified in [15]: *can we, based on an* a priori *analysis of the input log, determine whether it is better suited to imperative or declarative mining?*

Such a measure is potentially important for *hybrid mining* [34, 40], i.e., process mining where the output model is a combination of declarative and imperative models [29, 35, 41]. Here, our proposed measure could be combined with existing partitioning techniques [9, 20, 30, 42] to determine for each partition if it is more suited for imperative or declarative mining. Moreover, it is potentially useful for the development of novel partitioning techniques that specifically aim to separate structured and unstructured behaviour in a log.

We propose basing such a measure on the notion of *entropy* from the field of information theory. Introduced by Claude Shannon in his seminal 1948 paper [37], entropy measures the information content of a random variable. Intuitively, we can think of entropy as the "degree of surprise" we will experience when obtaining additional information about the state of a system [8].

We propose that the entropy of an event log can serve as a predictor of whether the generating process is structured or unstructured and accordingly, whether it is best modelled using declarative or imperative models. Highly structured processes should generate low entropy logs, whereas more flexible processes should generate high entropy logs. While information theoretic tools have been previously applied to predictive modelling [10], our application to discriminating mining techniques is novel.

The key contribution of the present paper is to study *exactly how* to measure the entropy of a given log. We study various potential measures based on both entropy and entropy rate, ranging from the near-trivial (trace), over language-inspired ones (prefix, k-block, global block), to methods from the study of dynamic systems and molecular structural analysis (nearest neighbours, Lempel-Ziv, block-based entropy rate). We follow our theoretical study with an empirical study, taking the various measures on both synthetic and real-life logs.

In the absence of an existing classification of available real-life logs into those suitable for declarative respectively imperative mining, we are unfortunately unable to objectively determine whether our entropy measures correctly distinguish declarative and imperative logs. However, we can approximate such a classification from the known properties of synthetic logs on the one hand and the common community understanding of select real-life logs on the other, and as such, qualitatively identify the most promising measures for further study.

Altogether, the present paper contributes (1) a survey of several entropy measures; (2) an implementation of these measures; (3) an experimental evaluation on both synthetic and real-life logs; and (4) based on this evaluation, a selection of promising measures, with a discussion of their strengths and shortcomings.

Overview. We first recall basic terminology and introduce four running examples logs in Section 2. We recall Shannon entropy and study ways to apply it in Section 3. In particular, we propose naive measures (trace- and prefix-entropy) and measures from the literature (block entropy and edit distance measures). We proceed to consider entropy *rate* measures in Section 4, studying both block-based estimators and Lempel-Ziv estimators. We report on implementation and experimental results in Section 5, and discuss extensively in Section 6. Finally, in Section 7, we conclude.

2 Process Logs

We recall the standard definitions of events, traces and process logs [3]: an event is an occurrence of an activity in a particular process instance, a trace is a sequence of events of the same such instance, and a log is a multiset of such traces.

Definition 1 (Events, Traces and Logs) Let Σ be an alphabet of activities, $s \in \Sigma$.

A trace $\sigma = \langle e_1, \ldots, e_n \rangle$ is a finite, nonempty sequence of an activities, i.e. a mapping $\sigma : \{1, \ldots, n\} \mapsto \Sigma$. An *event*, denoted e_i, is a specific occurrence of an activity in a trace, i.e. $e_i = \sigma(i)$.

We write $\sigma' \sqsubseteq \sigma$ to indicate that σ' is a prefix of σ. Finally, a *log* is a multiset $[\sigma_1^{w_1}, \ldots, \sigma_n^{w_n}]$ with $w_i \in \mathbb{N}$ denoting the multiplicity of trace σ_i.

In the sequel, we will refer extensively to the following running example.

Example 1 In Figure 1 we present the following three logs:

- L_1 is a very structured log, for which we can easily find a compact imperative model: for example the Petri net shown in Figure 2.
- L_2 is the same log as L_1, except some traces are now more frequent than others.
- L_3 is a much less structured log, with many variations in the ways activities can be ordered.

Figure 3 shows a mined Petri net for the log L_3, whereas figure Figure 4 shows a Declare [33] model. The constraints of the Declare model mean that:

1. a and b can not occur in the same trace,
2. after an a we always eventually see an h,
3. we must have seen at least one a before we can see a c,
4. we must have seen at least one d before we can see a c,
5. we must have seen at least one d before we can see an e,
6. after an e we always eventually see an f,
7. we must have seen at least one f before we can see a g,
8. after an f we will immediately see a g.

Having a closer look at the Petri net will show that the Declare model gives a much more *precise* representation, meaning that it allows less behaviour for which there is no evidence in the log. In particular, in the Petri net the activity d can occur at any point in the process, the precedence relations to c and e are lost. Similarly, the lower branch of the Petri net allows almost any interleaving of activities, with the exception that g should always be preceded by f, and c should always be preceded by a or h. While there exists no conclusive research on measures of relative understandability of these different notations, and this remains a hot topic of debate, it is worth noting that the Petri net employs a significantly larger number of graphical elements than the Declare model.

We will need to disregard the multiplicities of traces, in effect *flattening* the log. This is a common operation in the literature, see, e.g., [4].

Definition 2 Let $\mathsf{flatten}$ be the function on logs which given a log L produces the corresponding set, i.e.,

$$\mathsf{flatten}([\sigma_1^{w_1}, \ldots, \sigma_n^{w_n}]) = [\sigma_1^1, \ldots, \sigma_n^1] = \{\sigma_1, \ldots, \sigma_n\}$$

Example 2 The logs L_1 and L_2 differ only in non-zero multiplicities of traces, so $\mathsf{flatten}(L_1) = \mathsf{flatten}(L_2)$.

Flattening logs is particularly useful when one is interested in finding deterministic models that approximate as accurately as possible the language exhibited by a log. Most state-of-the-art process mining algorithms generate models in deterministic notations (e.g. Declare, Petri nets or BPMN). If the intention is for the model to support all possible behaviour, and not just the most common behaviour, then it makes sense to treat each variation as equal when measuring entropy. In this way we avoid treating logs differently based solely on the statistical distribution of the traces (which will not be represented in the mined models), instead of the inherent entropy of the language that the models should represent.

3 Entropy and Process Logs

Entropy is a measure of the information required, e.g. the average number of bits, to represent an outcome of a stochastic variable, intuitively indicating the "degree of surprise" upon learning a particular outcome [8]. In this paper we focus on Shannon's formulation of entropy [37], fundamental to the field of information theory.

Definition 3 (Shannon entropy) Given a discrete random variable, X, taking on n possible values with associated probabilities $p_1, p_2, \ldots, p_n$, (Shannon) entropy, denoted as H, is given by the expected value of the information content of X:

$$H = H(X) = -\sum_{i=1}^{n} p_i \log_b p_i \tag{3.1}$$

The base of the logarithm corresponds to the choice of coding scheme (i.e., for binary $b = 2$ and for decimal $b = 10$). We shall use the binary logarithm in the sequel.

Shannon justified this choice of measure with the fact that it is (1) continuous w.r.t. p_i (2) monotonically increasing w.r.t. n under uniform distributions and (3) additive under decomposition of choices, i.e.,

$$H(p_1, p_2, p_3) = H(p_1, (p_2 + p_3)) + (p_2 + p_3)H(p_2, p_3)$$

Entropy can be seen as a measure of the structure or predictability of messages coming from some information source. This has important implications for encoding and compression schemes, since homogeneous (i.e., highly redundant) messages can be significantly compressed by assigning shorter codes to likely outcomes at the expense of using longer codes for very rare outcomes. In fact, Shannon's noiseless coding theorem [26] proves that H is a lower bound on the average number of measurement units (bits) needed for lossless compression.

Estimating the entropy of sequences of symbols, including natural languages, is an active field of research and has implications in areas such as bioinformatics, molecular analysis, and chaotic dynamical systems which can be analysed

L_1	L_2	L_3
$\langle a,b,c,d,f,g,h\rangle^5$	$\langle a,b,c,d,f,g,h\rangle^{15}$	$\langle h,a,h,d,c\rangle^5$
$\langle a,b,c,e,f,g,h\rangle^5$	$\langle a,b,c,e,f,g,h\rangle^8$	$\langle a,d,a,c,a,c,e,h,h,f,g\rangle^5$
$\langle a,b,c,d,f,g\rangle^5$	$\langle a,b,c,d,f,g\rangle^5$	$\langle d,e,h,f,g,e,f,g\rangle^5$
$\langle a,b,c,e,f,g\rangle^5$	$\langle a,b,c,e,f,g\rangle^2$	$\langle h,b,b,h,h\rangle^5$
$\langle a,b,b,c,d,f,g,h\rangle^5$	$\langle a,b,b,c,d,f,g,h\rangle^3$	$\langle b,h,d,b,e,h,e,d,f,g\rangle^5$
$\langle a,b,b,c,e,f,g,h\rangle^5$	$\langle a,b,b,c,e,f,g,h\rangle^4$	$\langle a,d,a,d,h\rangle^5$
$\langle a,b,b,c,d,f,g\rangle^5$	$\langle a,b,b,c,d,f,g\rangle^1$	$\langle b,f,g,d\rangle^5$
$\langle a,b,b,c,e,f,g\rangle^5$	$\langle a,b,b,c,e,f,g\rangle^2$	$\langle f,g,h,f,g,h,h,h\rangle^5$

Fig. 1: Example logs. L_1, L_2 are structured logs, differing only in number of occurrences of complete traces. L_3 is an unstructured log.

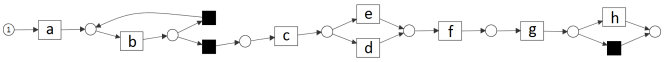

Fig. 2: Petri net for log L_1

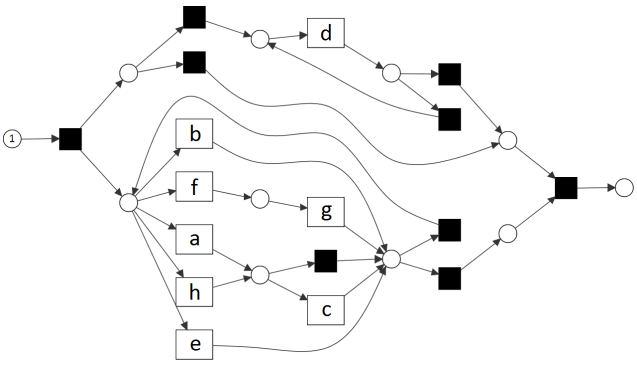

Fig. 3: Petri net for log L_3

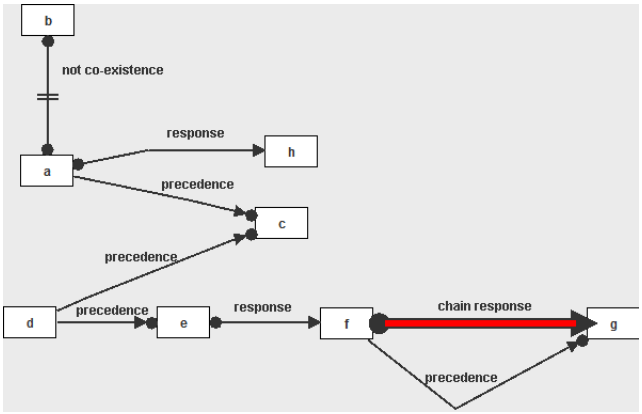

Fig. 4: Declare model for log L_3

using symbolic dynamics [36]. This problem is highly non-trivial, especially when only smaller samples are available and long term correlations are present.

We are interested in applying entropy as a measure with which to predict the suitability of a log for imperative or declarative mining; specifically, we expect higher entropy (less structure, less predictive, more bits required for optimal coding) to indicate suitability of declarative models, and lower entropy that of imperative models. In this setting, noise is a source of variability, and noisy logs will tend to have a large degree of entropy. The primary challenge is to distinguish between unintentional variability (noise) and intentional variability. One approach could be to first filter the log for noise using existing techniques, and then measure its entropy afterwards, accepting the risk of accidentally removing interesting behaviour from the log. Alternatively one could assume that the log contains no noise, measure its entropy, mine the log imperatively, declaratively, or hybridly based on the measure, and then analyse the resulting model for unintended flexibility.

In either case, we will insist that our measure of entropy respects language equivalence.

Definition 4 An *entropy measure for logs* is a function from logs to the reals. An entropy measure e *respects language equivalence* iff for any two language equivalent logs L, L', we have $e(L) = e(L')$.

We note that any entropy measure can be forced to respect language equivalence simply by flattening the log before feeding it to the measure:

Lemma 1 *Let e be an entropy measure. Then the function $e^f(L) = e(\mathsf{flatten}(L)) = e \circ \mathsf{flatten}(L)$ is an entropy measure that respects language equivalence.*

Proof Clearly e^f is a function from logs to reals. Now suppose logs L, L' are language equivalent. Then $\mathsf{flatten}(L) = \mathsf{flatten}(L')$ and it follows that $e^f(\mathsf{flatten}(L)) = e^f(\mathsf{flatten}(L'))$.

The key question in using entropy as a measure of log complexity is: *what is the random variable under consideration in the context of an event log?*

3.1 Trace Entropy

One very simple answer to this question is to take the underlying random variable as ranging over entire traces, with probabilities exactly the frequencies observed in the log. This idea gives rise to the notion *trace entropy*.

Definition 5 (Trace entropy) Let $L = [\sigma_1^{w_1}, \ldots, \sigma_n^{w_n}]$ be a log. The *trace entropy* of L, written $\mathsf{entropy}^{tr}(L)$, is the entropy of the random variable that ranges over the traces in L:

$$\mathsf{entropy}^{tr}(L) = -\sum_{\sigma_i \in L} p_L(\sigma_i) \log p_L(\sigma_i)$$

We distinguish, here and later, between an entropy measure defined on the "true" probability distribution p (as in Definition 5 above) on the one hand, and an estimate $\hat{p}$ of that distribution (Definition 6 below) on the other. We restrict ourselves to simple likelihood (frequency based) estimators, but more sophisticated estimators exist [36]. For example, Bayesian estimators could incorporate prior probabilities based on domain knowledge.

Definition 6 (Trace likelihood estimator) Let L be a log, $L = [\sigma_1^{w_1}, \ldots, \sigma_n^{w_n}]$ be a log. The likelihood estimator for trace σ_i, used to compute $\mathsf{entropy}^{tr}(L)$, is given by

$$\hat{p}_L(\sigma_i) = \frac{w_i}{\sum_{i=1}^{m} w_i}$$

Example 3 Even though the traces of L_1 and L_3 internally have radically different structure, they have the same number of occurrences of distinct traces, and so the same trace entropy:

$$\mathsf{entropy}^{tr}(L_1) = \mathsf{entropy}^{tr}(L_3) = -8 \times \tfrac{5}{40} \log_2 \tfrac{5}{40} = 3$$

Computing the trace entropy of L_2, we find

$$\mathsf{entropy}^{tr}(L_2) = -\left(\tfrac{15}{40} \log_2 \tfrac{15}{40} + \ldots + \tfrac{2}{40} \log_2 \tfrac{2}{40}\right) = 2.55$$

This example demonstrates that trace-entropy is likely *not* a good measure for determining if a model should be modelled imperatively or declaratively: L_1 and L_2 intuitively should map to the same model, but have distinct trace-entropy. On the other hand, L_3 has much more variable behaviour than L_1, yet has the same trace entropy.

Example 4 The logs L_1, L_2 of Example 3 are language equivalent. However, they have different trace entropy measures. It follows that trace entropy does not respect language equivalence.

There is on an intuitive level also a second reason that trace entropy is unhelpful: it does not consider the behaviour exhibited *within* the traces. We saw this in Example 3, where $\mathsf{entropy}^{tr}(L_1) = \mathsf{entropy}^{tr}(L_3)$; that is, trace entropy cannot distinguish internal structure of traces. To consider the full behaviour of a log, we need to determine the entropy on the level of individual events.

3.2 Prefix Entropy

We look for a suitable notion of random variable that generates the traces we observe in the log, while at the same time characterising the internal structure of the individual traces.

Recall that a trace is the execution of a single process instance, taking the form of a sequence of activity executions. At each point in a process execution, we have a prefix of a completed trace. Presumably, the distribution of these prefixes reflect the structure of the process.

Definition 7 (Prefix entropy) Let L be a log. We write $\mathsf{entropy}^{pr}(L)$ for the *prefix entropy* of L, defined as the entropy of the random variable which ranges over all prefixes of traces in L.

Formally, we write $p(\langle e_1, \ldots, e_n \rangle)$ to denote the probability that the outcome of the random variable will be the prefix $\langle e_1, \ldots, e_n \rangle$. Following directly from Equation (3.1), prefix entropy is given by

$$\mathsf{entropy}^{pr}(L) = -\sum_{\langle e_1,\ldots,e_n\rangle \sqsubseteq \Sigma^\star} p_L(\langle e_1, \ldots, e_n \rangle) \log p_L(\langle e_1, \ldots, e_n \rangle)$$

For $n \in \mathbb{N}$.

To estimate the probability distribution on prefixes, we again use a simple likelihood estimator. That is, for each prefix $\langle e_1, \ldots, e_n \rangle \sqsubseteq \sigma$ of a trace σ observed in a log L, we assign as its probability the frequency of that prefix among all occurrences of prefixes in L.

Definition 8 (Prefix likelihood estimator) Let L be a log. The likelihood estimator for prefix $\langle e_1, \ldots, e_n \rangle$, used to compute $\mathsf{entropy}^{pr}(L)$, is given by

$$\hat{p}_L(\langle e_1, \ldots, e_n \rangle) = \frac{\sum_{\langle e_1,\ldots,e_n\rangle \sqsubseteq \sigma \in L} 1}{\sum_{\sigma \in L} |\sigma|} \tag{8.1}$$

Where the sum in the denominator gives the total number of prefixes across the log.

Example 5 In the log L_2, the prefix $\langle a, b, c, d \rangle$ occurs in 20 traces; the log contains a total of $15 \times 7 + 8 \times 7 + \ldots + 2 \times 7 = 280$ prefix occurrences, for a probability of $1/14$.

However, this notion of prefix entropy *does not* respect language equivalence: logs differing only in the number of occurrences of a particular trace may also differ in the set of occurrences of prefixes, we therefore define *flattened prefix entropy* as the function composition of first flattening a log and then determining its prefix entropy.

$$
\begin{array}{c}
L_4 \\
\langle a, x, y, z\rangle^5 \\
\langle a, x, z, y\rangle^5 \\
\langle b, x, y, z\rangle^5 \\
\langle b, x, z, y\rangle^5 \\
\langle c, x, y, z\rangle^5 \\
\langle c, x, z, y\rangle^5 \\
\langle d, x, y, z\rangle^5 \\
\langle d, x, z, y\rangle^5 \\
\langle e, x, y, z\rangle^5 \\
\langle e, x, z, y\rangle^5
\end{array}
$$

Fig. 5: Log L_4 (highly structured).

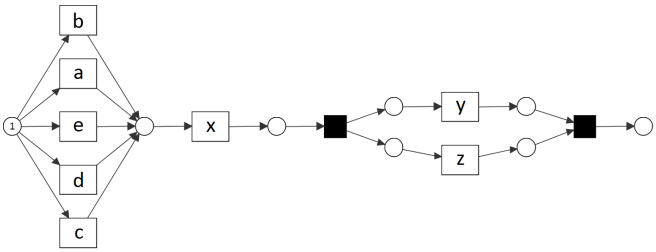

Fig. 6: Petri net for log L_4

Definition 9 (Flattened prefix entropy)

$$\mathsf{entropy}^{fpr} = \mathsf{entropy}^{pr} \circ \mathsf{flatten}$$

Example 6 In the log $\mathsf{flatten}(L_2)$, the prefix $\langle a, b, c, d\rangle$ occurs just twice, among a total of only 56 prefix occurrences, for a probability of $1/26$. Computing the flattened prefix entropy of the example logs of Example 3, we find:

$$\mathsf{entropy}^{fpr}(L_1) \approx \mathsf{entropy}^{fpr}(L_2) \approx 4.09$$
$$\mathsf{entropy}^{fpr}(L_3) \approx 5.63$$

While the notion of flattened prefix entropy may seem promising, there is one caveat: Because it is based on prefixes, it fails to account for common structure appearing after distinct prefixes.

Example 7 Consider the log L_4 in Figure 5. This log is highly structured: it always contains exactly 4 activities; the first is a choice between $\{a, b, c, d, e\}$, the second an x, the third and fourth either x, y or y, x. Figure 6 shows a Petri net admitting exactly this behaviour. However, this log has a trace entropy of $\mathsf{entropy}^{fpr}(L_4) = 4.82$, higher than the apparently *less* structured logs L_1 and L_2.

3.3 Block Entropy

To address this weakness of prefix entropy, we apply ideas from natural language processing [36], where entropy is studied in terms of n-length substrings known as "n-grams" (we will use k instead of n). We consider an individual trace a "word", in which case our log is a multiset of such words, and look at the observed frequencies of arbitrary substrings within the entire log. That is, rather than looking at the frequencies of prefixes, we look at frequencies of substrings. We shall see that while computationally more expensive, this idea alleviates the problems of prefix entropy: observed structure is weighted equally, regardless of where it occurs in a trace.

Definition 10 (k-block entropy) Let L be a log. We define the *k-block entropy* of L, written $\mathsf{entropy}^{bl}_k(L)$, as the entropy of the random variable which ranges over all k-length strings in Σ^*.

Formally, we write $p(\langle s_1, \ldots, s_k\rangle)$ to denote the probability that the outcome of the random variable will be the substring $\langle s_1, \ldots, s_k\rangle$ (in section 3.4 it will become clear why we do not write $\langle e_1, \ldots, e_k\rangle$).

Again, following directly from Equation (3.1), block entropy is given by

$$\mathsf{entropy}^{bl}_k(L) = -\sum_{\langle s_1,\ldots,s_k\rangle \in \Sigma^\star} p_L(\langle s_1, \ldots, s_k\rangle) \log p_L(\langle s_1, \ldots, s_k\rangle)$$

For fixed k.

The k-block entropy measures the amount of information contained in a block of length k. So, if some activities always occur in the same order, little new information is added when they are encountered in that order.

Definition 11 (k-block likelihood estimator) Let L be a log. The likelihood estimator for blocks of length k, used to compute $\mathsf{entropy}^{bl}_k$, is given by

$$\hat{p}_L(\langle s_1, \ldots, s_k\rangle) = \begin{cases} 0 & \text{if } f(L,k) = 0 \\ \dfrac{\sum_{\sigma \in L} n_{s_1,\ldots,s_k}}{f(L,k)} & \text{otherwise} \end{cases} \tag{11.1}$$

Where

$$f(L,k) = \sum_{\sigma \in L} \max\left(0,\ |\sigma| - k + 1\right)$$

Where $n_{s_1,\ldots,s_k}$ is the number of times block $s_1, \ldots, s_k$ occurred in a trace, and $f(L,k)$ gives the total number of k-blocks across the log. The zero condition accounts for all k-blocks longer than the longest trace.

Definition 12 (Flattened k-block entropy)

$$\mathsf{entropy}^{fbl}_k = \mathsf{entropy}^{bl}_k \circ \mathsf{flatten}$$

Example 8 In the flattened version of log L_4 in Figure 5, $\mathsf{flatten}(L_4)$, the 2-blocks $\langle a, x\rangle$, $\langle b, x\rangle$, $\langle c, x\rangle$, $\langle d, x\rangle$ and $\langle e, x\rangle$ all occur 2 times while $\langle x, y\rangle$, $\langle y, x\rangle$, $\langle y, z\rangle$ and $\langle z, y\rangle$ all occur 5 times. The log contains a total of $10 \times 3 = 30$ occurrences of 2-blocks, giving the 2-block entropy for $\mathsf{flatten}(L_4)$:

$$\mathsf{entropy}_k^{fbl}(L_4) = \\ -\left(5 \times \tfrac{2}{30} \log \tfrac{2}{30} + 4 \times \tfrac{5}{30} \log \tfrac{4}{30}\right) \approx 3.03$$

We now define block entropy for all substrings up to the length of the longest trace. That is, instead of restricting the measure to blocks of length k, we include *all* blocks, from length 1 up to the length of the longest trace, in one entropy measure.

Definition 13 (Global block entropy) Let L be a log. The *global block entropy* of L, written $\mathsf{entropy}^{bl}(L)$, is the entropy of the random variable which ranges over all strings in Σ^*.

Again, following directly from Equation (3.1), global block entropy is given by

$$\mathsf{entropy}^{bl}(L) = \\ -\sum_{\langle s_1,\ldots,s_k\rangle \in \Sigma^\star} p_L(\langle s_1, \ldots, s_k\rangle) \log p_L(\langle s_1, \ldots, s_k\rangle)$$

For $k \in \mathbb{N}$.

Definition 14 (Global block likelihood estimator) Let L be a log. The likelihood estimator for blocks of length k, used to compute $\mathsf{entropy}^{bl}(L)$, is given by

$$\hat{p}_L(\langle s_1, \ldots, s_k\rangle) = \frac{\sum_{\sigma \in L} n_{s_1,\ldots,s_k}}{\sum_{\sigma \in L} \frac{|\sigma|(|\sigma|+1)}{2}} \qquad (14.1)$$

Where $n_{s_1,\ldots,s_k}$ is the number times block $\langle s_1, \ldots, s_k\rangle$ occurred in a trace, and the sum in the denominator gives the total number of k-blocks across the log for k ranging from 1 to the length of the longest trace.

Example 9 A trace of length n contributes $1+2+\cdots+(n-1)+n = \Sigma_1^n k = \frac{n(n+1)}{2}$ distinct occurrences of substrings: one of length n; two of length $n-1$ starting at indexes 0 and 1, respectively; three of length $n-2$; and so forth. Summing up the number of occurrences in each trace in the flattened log, we thus get $\Sigma_1^5 k = \frac{5\cdot 6}{2} = 15$ in the first trace, $\Sigma_1^{11} k = \frac{11\cdot 12}{2} = 66$ in the second, and so on, for a total of 248 distinct occurrences.

Counting the number of occurrences of the specific substring (2-block) $\langle a, d\rangle$, we find that it occurs 3 times: once in the second entry, twice in the sixth. Altogether, the probability of $\langle a, d\rangle$ is $3/248 \sim 0.012$.

As for the prefix and k-block entropy, the global block entropy does not respect language equivalence, but its flattening does.

Definition 15 The *flattened global block entropy* $\mathsf{entropy}^{fbl}$ is given by $\mathsf{entropy}^{fbl} = \mathsf{entropy}^{bl} \circ \mathsf{flatten}$.

Example 10 We give the flattened global block entropy for the examples L_1 through L_4.

	L_1	L_2	L_3	L_4
$\mathsf{entropy}^{fbl}(-)$	5.75	5.75	7.04	4.75

Notice how L_3 is still the highest-entropy log, but now L_4 is properly recognised as containing less information than L_1 and L_2.

We conclude this section by noting that while the global block entropy looks promising, it is computationally challenging to apply to large logs. Naively computing the global block entropy of a log requires, in the worst case, tabulating the frequencies of all substrings seen in that log. For a log with n traces, all of length k, the running time is bounded by $\mathcal{O}(n \times k^2)$. This is ameliorated to some degree by using efficient data structures such as a suffix trie for counting k-blocks.

3.4 Stationarity and Ergodicity

In our original definition of $\mathsf{entropy}^{bl}$, we made a notational distinction between the original sequences of events (traces) $\langle e_1, \ldots, e_n\rangle$ and subsequences of activities within those traces $\langle s_1, \ldots, s_n\rangle$, where e denotes a specific event at a specific position in the trace. This distinction stems from an assumption which underlies most of the entropy estimators we consider: that the underlying process generating traces is *stationary* and *ergodic*.

This notion is clearly illustrated in the case of $\mathsf{entropy}_k^{bl}$ where, without making these assumptions, we would denote the probability of seeing a specific sequence of activities $\langle s_1, \ldots, s_k\rangle$ from index t to index $t+k$ in a trace:

$$p_L(\langle e_{t+1} = s_1, \ldots, e_{t+k} = s_k\rangle)$$

That is, the probability that the *specific* event e_{t+1} was an instance of activity s_1, and so on. In order to drop the index t and use a "sliding window" approach, counting any occurrences of $\langle s_1, \ldots, s_k\rangle$ as equivalent regardless of position, we must assume that the position of a k-block in the trace does not influence which k-block is observed and the dynamics of the underlying process do not change over time, i.e. they are *stationary*. Otherwise we would have to consider an occurrence of the same sequence of activities in the beginning and end of a trace as instances of two different outcomes.

More generally stated, *stationarity* assumes that the probability distribution underlying observed outcomes *is not a function of time*. In the current context, outcomes are the observed k-blocks, or individual events in traces for other metrics we will introduce. This is most likely an incorrect assumption for business processes since some events are often associated with the beginning or end of a process, or that some activity always occurs as, for example, the third activity if it occurs. We nonetheless make this assumption under the presumption that much of the structure in activity sequences will still be captured by k-blocks in a sliding window.

Ergodicity implies that this probability distribution can be reconstructed from the observation of a typical sequence (trace), i.e. that the average properties of the process over time (within a trace) is equivalent to the average properties across space (across the traces in a log) which would seem to run contrary to the fact that process outcomes will be determined by the needs of a specific case (e.g. a customer or patient), unless this variation is to be interpreted as statistical *noise*. It is not clear to what degree violation of these assumptions would actually skew the resulting entropy estimation[1]. It may be the case that the proposed estimators nonetheless give reasonable indications of the degree of structure in event logs.

3.5 Block Entropy for Event Logs vs. Natural Language

Beginning with Shannon's original paper [37], research on estimating the entropy of natural languages has demonstrated that studies with human subjects[2] consistently result in a lower entropy estimate than mathematical estimators, suggesting that structural information and long-term correlations are not fully captured [12].

NLP researchers often use text corpora modified by removing punctuation and capitalisation, meaning that sentences are ignored and a single block can span the end of one sentence and the beginning of another. For event logs we want to avoid spurious correlations among events at the end of one trace and the beginning of another, so we keep traces separate.

3.6 Nearest Neighbours Estimators

The measures discussed so far fail to capture aspects of the structuredness of the log: *prefix* and *block entropy* will become slightly skewed in the case where traces differ by just one "missing" activity since they consider any blocks overlapping these activities to be unique, and *trace entropy* cannot guarantee language equivalence for flattened logs since it relies on the distribution of unique traces in the log. Using *edit distance* we can obtain a measure which allows us to compare entire traces while also capturing the internal structure of traces, being tolerant to minor differences.

The distribution of traces in the metric space defined by the edit distance determines the entropy of the log: evenly distributed traces yield a high entropy, whereas similar traces grouped together in a few clusters yield a low entropy. In contrast to the trace entropy of Definition 5, we can meaningfully apply this approach to a flattened log.

In previous applications of trace clustering in a BPM context, authors used a *bag-of-activities* approach, redefining traces as vectors of activity counts: each element representing the occurrences of each activity [13, 21, 42], and using different vector-based distance metrics such as Minkowski distance and Jaccard distance. Researchers in natural language processing commonly use this approach as well [23]. Others propose clustering traces using more structured approaches based on outranking relations theory [17] and distance-graph based representation [22].

Edit distance based clustering was investigated in [9] where the authors propose an algorithm for automatically learning operation costs and use agglomerative hierarchical clustering.

Our approach differs from previous work in two ways. First, we use nearest neighbour distances to investigate entropy, rather than clustering similar traces in order to mine these clusters separately; second, the aim of our approach is to use entropy to choose between mining *paradigms*: imperative versus declarative.

3.6.1 Edit Distance Between Traces

Levenshtein edit distance is the most well known string edit distance metric, and is defined by the number of insertions, deletions and substitutions required to convert one string into another. Levenshtein distance $\mathsf{lev}_{x,y}(|x|,|y|)$ between strings x and y, with non-negative operation costs, fulfils the axioms of a *metric*:

non-negativity
$\mathsf{lev}_{x,y}(|x|,|y|) \geq 0$

identity of indiscernibles a string can be transformed into itself with 0 operations.
$\mathsf{lev}_{x,y}(|x|,|y|) = 0 \implies x = y$

symmetry the same number of operations is required to convert string x to y as to convert y to x.
$\mathsf{lev}_{x,y}(|x|,|y|) = \mathsf{lev}_{y,x}(|y|,|x|) :$

triangle inequality
$\mathsf{lev}_{x,z}(|x|,|z|) \leq \mathsf{lev}_{x,y}(|x|,|y|) + \mathsf{lev}_{y,z}(|y|,|z|)$

[1] See section 4 for the theoretical basis of these assumptions, and an example in which it is impossible to define the entropy of a non-stationary process.

[2] Participants are given prefixes of text and asked to predict the next letter. The entropy *rate* (Section 4) is calculated from the proportion of correct responses.

Edit distance allows us to define a metric space on our log in which the internal structure of the trace, i.e. the ordering of events, is captured. We normalise edit distance by the greatest possible distance between the traces to reflect that a distance of one operation on two very long strings should be considered less significant than on very short strings.

Definition 16 (Normalised Levenshtein distance) Let σ and ς be traces from the same event log. The Levenshtein distance between these two prefixes is given by

$$\mathsf{lev}_{\sigma,\varsigma}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ f(\sigma,\varsigma) & \text{otherwise} \end{cases}$$

Where

$$f(\sigma,\varsigma) = \begin{cases} \min \mathsf{lev}_{\sigma,\varsigma}(i-1,j)+1 \\ \mathsf{lev}_{\sigma,\varsigma}(i,j-1)+1 \\ \mathsf{lev}_{\sigma,\varsigma}(i-1,j-1)+1_{\sigma_i \neq \varsigma_j} \end{cases}$$

Where $1_{\sigma_i \neq \varsigma_j}$ is the indicator function and returns 0 when $\sigma(i) = \varsigma(j)$ and 1 otherwise. That is, in the "otherwise" clause, three types of edits are allowed: insertion, deletion and substitution, which in this case have the same cost, 1. The *normalised Levenshtein distance* is given by

$$\mathsf{lev}_N(\sigma,\varsigma) = \frac{\mathsf{lev}_{\sigma,\varsigma}(|\sigma|,|\varsigma|)}{\max(|\sigma|,|\varsigma|)}$$

Where $\max(|\sigma|,|\varsigma|)$ is an upper bound on $\mathsf{lev}_{\sigma,\varsigma}(|\sigma|,|\varsigma|)$, guaranteeing a range of $[0,1]$ for lev_N. It is straightforward to verify that the normalised Levenshtein distance is in fact a metric.

Example 11 Consider two traces from log L_1 in Figure 5:

$$\langle a,b,c,e,f,g,h\rangle$$
$$\langle a,b,c,d,f,g\rangle$$

Converting the first to the second requires two edits: substituting d for e and and deleting h. The worst case scenario is the length of the longest trace, 7, giving a normalised distance of

$$\mathsf{lev}_N\big(\langle a,b,c,e,f,g,h\rangle, \langle a,b,c,d,f,g\rangle\big) = \tfrac{2}{7} \approx 0.29$$

One approach to computing entropy based on edit distances would be to first cluster traces using a clustering algorithm, and interpreting these clusters as the outcomes of the random variable X, then computing Shannon entropy using the number of traces per cluster as the probability distribution over X. Aside from adding an extra clustering step, this approach adds the challenge of choosing an appropriate clustering algorithm and how many clusters into which to partition the log.

Fortunately, previous research provides nearest neighbour based entropy estimators which estimate entropy *directly* from distances, removing the clustering step.

3.6.2 Distance-based Estimators

Nearest neighbour based entropy estimators are a class of non-parametric estimators widely used in machine learning for goodness-of-fit testing, parameter estimation and even for analysing molecular structure [38, 39], which allow for estimating entropy directly from distances between data points. The nearest neighbour entropy estimator proposed by Kozachenko and Leonenko in 1987 [16] considers only the first nearest neighbour. This measure of entropy is originally formulated on a vector space, and uses the dimension d of that space as a parameter. As we are working merely in a metric space, that d becomes simply a parameter of the measure.

Note that in the following definition ρ ("rho", not p) signifies *distance* and not probability.

Definition 17 (Kozachenko-Leonenko entropy) Let L be a log, let ρ_σ denote the *distance* of trace σ to its nearest neighbour in L, and let d be positive integer. The Kozachenko-Leonenko entropy measure is given by

$$\mathsf{entropy}^{KL}(L) = \frac{d}{|L|}\sum_{\sigma\in L}\log\rho_\sigma + \log\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} + \gamma + \log(|L|-1) \tag{17.1}$$

Where $\Gamma(n)$ is the generalisation of $(n-1)!$ to real (and complex) numbers, $\gamma \approx 0.5772\ldots$ is Euler's constant, and d denotes the dimensionality of the underlying metric space.

Definition 18 The *flattened Kozachenko-Leonenko entropy* is given by $\mathsf{entropy}^{fKL} = \mathsf{entropy}^{KL} \circ \mathsf{flatten}(L)$.

Example 12 Consider the traces from logs L_1 and L_2 in Figure 1. The nearest neighbour calculation for each trace in the flattened logs are shown in Table 1. Using these values and letting $d = 1$, we have

$$\begin{aligned} \mathsf{entropy}^{fKL}(L_1) = {} & \frac{6}{8}\left(\log\frac{1}{8} + \log\frac{\pi^{1/2}}{\Gamma(\frac{3}{2})} + \gamma + \log(7)\right) \\ & + \frac{2}{8}\left(\log\frac{1}{7} + \log\frac{\pi^{1/2}}{\Gamma(\frac{3}{2})} + \gamma + \log(6)\right) \\ \approx {} & 1.17 \end{aligned}$$

The nearest neighbour estimator was expanded upon by Singh et al. in [38] to the k^{th} -nearest neighbour.

Definition 19 (k^{th}-nearest neighbour entropy) Let L be a log. Let $\rho_{\sigma,k}$ denote the distance of trace σ to its k^{th} nearest neighbour in L. The k^{th}-nearest neighbour entropy measure is given by

$$\mathsf{entropy}^{kNN}(L) = \frac{d}{|L|}\sum_{\sigma\in L}\log\rho_{\sigma,k} + \log\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} + \gamma - f(k-1) + \log(|L|)$$

Trace (σ)	Nearest Neighbour (ς)	$\mathsf{lev}(\lvert\sigma\rvert, \lvert\varsigma\rvert)$	$\max(\lvert\sigma\rvert, \lvert\varsigma\rvert)$	$\mathsf{lev}_N(\sigma, \varsigma)$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}, \mathbf{g}, \mathbf{h} \rangle$	$\langle a, b, b, c, d, f, g, h \rangle$	1	8	$1/8 = 0.125$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h} \rangle$	$\langle a, b, b, c, e, f, g, h \rangle$	1	8	$1/8 = 0.125$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}, \mathbf{g} \rangle$	$\langle a, b, c, d, f, g, h \rangle$	1	7	$1/7 \approx 0.143$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{e}, \mathbf{f}, \mathbf{g} \rangle$	$\langle a, b, c, e, f, g, h \rangle$	1	7	$1/7 \approx 0.143$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}, \mathbf{g}, \mathbf{h} \rangle$	$\langle a, b, c, d, f, g, h \rangle$	1	8	$1/8 = 0.125$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{b}, \mathbf{c}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h} \rangle$	$\langle a, b, c, e, f, g, h \rangle$	1	8	$1/8 = 0.125$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}, \mathbf{g} \rangle$	$\langle a, b, b, c, d, f, g, h \rangle$	1	8	$1/8 = 0.125$
$\langle \mathbf{a}, \mathbf{b}, \mathbf{b}, \mathbf{c}, \mathbf{e}, \mathbf{f}, \mathbf{g} \rangle$	$\langle a, b, b, c, e, f, g, h \rangle$	1	8	$1/8 = 0.125$

Table 1: Nearest neighbour calculations for $\mathsf{flatten}(L_1)$ and $\mathsf{flatten}(L_2)$. It happens to be the case for these log that every trace in these logs has a nearest neighbour with an unnormalised edit distance of 1.

Where

$$f(x) = \begin{cases} 0 & \text{if} \quad x = 0 \\ \sum_{y=1}^{x} \frac{1}{y} & \text{if} \quad x \geq 1 \end{cases}$$

Definition 20 (Flattened k^{th}-nearest neighbour entropy)

$$\mathsf{entropy}^{fkNN} = \mathsf{entropy}^{kNN} \circ \mathsf{flatten}(L)$$

Note that we used the *normalised Levenshtein distance* to define ρ_σ in $\mathsf{entropy}^{KL}$ and $\mathsf{entropy}^{kNN}$, but any distance metric fulfilling the axioms in 3.6.1 can be used.

While edit distance allows us to capture local structural differences in traces, allowing us to bin traces together which differ by only a few events, it may fail to capture important similarities among traces. For example, the traces $\langle a, b \rangle$ and $\langle a, b, a, b, a, b \rangle$ will be heavily penalised for being of different length, even though the latter is clearly a repetition of the former. In the next section we discuss measures like Lempel-Ziv which are able to capture this type of structure.

4 Entropy Rate of Stochastic Processes

Up to this point we have explored ways to interpret process logs as the outcome of some stochastic variable X and finding the entropy of this variable. We shall see in Section 5 that logs with longer traces and more activities tend to result in higher entropy measures. It is questionable however whether a larger log by definition is always better suited to declarative modelling.

A more nuanced interpretation of entropy is to consider each *individual event*, rather than the trace as a whole, as the outcome of a separate variable, generated by a stochastic process. This approach is invariant to the number of activities and trace lengths and should better capture what we are interested in: the degree of structure in the underlying process. In the literature, this is referred to as the entropy *rate* of a *process*: the increase in entropy upon considering an additional outcome of the process [43].

We present three estimators of entropy rate in Sections 4.1 and 4.2, but first we will formally define the entropy rate for a process for which the probability distribution is known.

We cover two alternative definitions of entropy rate, which turn out to be equivalent in the limit for stationary processes. First, we will need the definitions of *joint entropy* and *conditional entropy*. See [27] or [43] for details.

Lemma 2 (Joint entropy) *Let $e_1, \ldots, e_n$ be discrete random variables. The* joint entropy *is the Shannon entropy extended to more than one variable. In the present context, the outcomes under consideration are activities $s \in \Sigma$:*

$$H(e_1, \ldots, e_n) = -\sum_{s_1 \in \Sigma} \cdots \sum_{s_n \in \Sigma} p_n \log p_n$$

Where

$$p_n = p(e_1 = s_1, \ldots, e_n = s_n)$$

It is the expected value *of the information content of the variables:*

$$H(e_1, \ldots, e_n) = \mathbb{E}\left[\log p(s_1, \ldots, s_n)\right]$$

Note that this definition is equivalent to our initial definition of Shannon entropy (3.1), if the outcome of variables $e_1, \ldots, e_n$ are instead interpreted as a single vector valued variable.

Example 13 Consider L_4 from Figure 5 where

$$\Sigma = \{a, b, c, d, e, x, y, z\}.$$

If we assume that the log is exactly representative of the outcomes of e_1, e_2, e_3 and e_4, then we have the following marginal probabilities for the individual events,

$$\begin{aligned} p(e_1 = a) &= 1 & p(e_2 = x) &= 1 \\ p(e_1 = b) &= 1 & p(e_3 = y) &= \tfrac{1}{2} \\ p(e_1 = c) &= 1 & p(e_3 = z) &= \tfrac{1}{2} \\ p(e_1 = d) &= 1 & p(e_4 = y) &= \tfrac{1}{2} \\ p(e_1 = e) &= \tfrac{1}{5} & p(e_4 = z) &= \tfrac{1}{2} \end{aligned} \tag{20.1}$$

The probability for all other outcomes (e.g. $e_1 = x, e_2 = a$, etc.) is 0. The joint entropy is given by

$$H(e_1, e_2, e_3, e_4) =$$
$$-\sum_{s_1 \in \Sigma}\sum_{s_2 \in \Sigma}\sum_{s_3 \in \Sigma}\sum_{s_4 \in \Sigma} p(s_1, s_2, s_3, s_4) \log p(s_1, s_2, s_3, s_4)$$
$$= -10 \times \left(\tfrac{1}{5} \times 1 \times \tfrac{1}{2} \times \tfrac{1}{2}\right) \log \left(\tfrac{1}{5} \times 1 \times \tfrac{1}{2} \times \tfrac{1}{2}\right) \approx 2.16$$

Lemma 3 (Conditional entropy) *Let $e_1, \ldots, e_{n+1}$ be discrete random variables. The* conditional entropy *of e_{n+1} given $e_n, \ldots, e_1$ is the amount of uncertainty regarding the outcome of e_{n+1} once $e_n, \ldots, e_1$ have been observed:*

$$H(e_{n+1}|e_n, \ldots, e_1) = -\sum_{s_1 \in \Sigma} \cdots \sum_{s_{n+1} \in \Sigma} p_n \log p_n$$

Where

$$p_n = p(e_1 = s_1, \ldots, e_{n+1} = s_{n+1})$$

It is the expected value *of the conditional probability of e_{n+1} given $e_n, \ldots, e_1$:*

$$H(e_{n+1}|e_n, \ldots, e_1) = \mathbb{E}\left[\log p(s_{n+1}|s_n, \ldots, s_1)\right]$$

By simple rules of probability and logarithms, we can show that conditional entropy is equivalent to the *difference* between the joint entropy of $e_{n+1}, \ldots, e_1$ and $e_n, \ldots, e_1$:

$$H(e_{n+1}|e_n, \ldots, e_1) =$$
$$-\sum_{s_1} \cdots \sum_{s_{n+1}} p(s_1, \ldots, s_{n+1}) \log \frac{p(s_n, \ldots, s_1|s_{n+1})p(s_{n+1})}{p(s_n, \ldots, s_1)}$$
$$= -\sum_{s_1} \cdots \sum_{s_{n+1}} p(s_1, \ldots, s_{n+1}) \begin{pmatrix} \log p(s_{n+1}, \ldots, s_1 \\ -\log p(s_n, \ldots, s_1) \end{pmatrix}$$
$$= H(e_1, \ldots, e_{n+1}) + \sum_{s_1} \cdots \sum_{s_n} p(s_n, \ldots, s_1) \log p(s_n, \ldots, s_1)$$
$$= H(e_1, \ldots, e_{n+1}) - H(e_1, \ldots, e_n)$$

Example 14 For brevity, consider a new log:

$$\{\langle a, b, c\rangle, \langle a, b, d\rangle, \langle a, c, c\rangle\}$$

. We will compute the conditional entropy of e_3, given e_2 and e_1:

$$\begin{aligned} H(e_3|e_2, e_1) = &- \sum_{s_1 \in \Sigma}\sum_{s_2 \in \Sigma}\sum_{s_3 \in \Sigma} p(s_1, s_2, s_3) \log p(s_3|s_2, s_1) \\ = &- p(a, b, c) \log(c|b, a) \\ &- p(a, b, d) \log p(d|b, a) \\ &- p(a, c, c) \log p(c|c, a) \\ = &- \tfrac{1}{3} \log \tfrac{1}{2} - \tfrac{1}{3} \log \tfrac{1}{2} - \tfrac{1}{3} \log 1 = \tfrac{2}{3} \end{aligned}$$

Lemma 4 (Per symbol entropy rate) *Let $e_1, \ldots, e_n$ be a sequence of n variables, representing the outcomes of a stochastic process, then the* per symbol entropy rate *of the* process*, denoted $H(\mathcal{E})$ or h, represents how the* joint entropy *$H(e_1, \ldots, e_n)$ grows with the length n of the sequence:*

$$h = H(\mathcal{E}) = \lim_{n \to \infty} \frac{1}{n} H(e_1, \ldots, e_n) \tag{20.2}$$

In the special case in which $e_1, \ldots, e_n$ are independent and identically distributed (i.i.d.), we have

$$H(\mathcal{E}) = \lim_{n \to \infty} \frac{H(e_1, \ldots, e_n))}{n} = \lim_{n \to \infty} \frac{nH(e_1)}{n} = H(e_1)$$

where $H(e_1, \ldots, e_n) = nH(e_1)$ for i.i.d. variables follows from Shannon's source coding theorem and the asymptotic equipartition principle [27]. Since event logs were generated by some structured process, we cannot assume independence between variables $e_1, \ldots, e_n$. However, we must assume *stationarity*, i.e. that the statistical properties of the process do not change over time, in order to be able to use block-based entropy rate estimators.

The reason for this is illustrated by the case in which we have independent, but not identically distributed random variables: if $p(e_i = s)$ is allowed to be a function of time (i.e. the index i). From the definition of joint entropy and independence we have

$$H(e_1, \ldots, e_n) = \sum_{i=1}^{n} H(e_i)$$

In this case, there exists a sequence of probability distributions across $e_1, \ldots, e_n$ such that

$$H(\mathcal{E}) = \lim_{n \to \infty} \frac{1}{n} \sum H(e_i)$$

does not exist[3]. In such a process, $H(\mathcal{E})$ is undefined [43].

We now define an alternative interpretation of entropy rate, which is equivalent to *per-symbol entropy rate* under the assumption of stationarity.

Lemma 5 (Conditional entropy rate) *Let $e_1, \ldots, e_n$ be a sequence of n variables, representing the outcomes of a stochastic process, then the* conditional entropy rate *of the process, denoted $H'(\mathcal{E})$ or h', represents the* conditional entropy *of the most recently observed variable given the past, in the limit:*

$$h' = H'(\mathcal{E}) = \lim_{n \to \infty} H(e_{n+1}|e_n, \ldots, e_1) \tag{20.3}$$

[3] For example, if the probability of the outcome of $p(e_i = x)$ is a function of the index i, then a probability distribution exists such that the running average of $H(e_i)$ oscillates between 0 and 1.

Theorem 1 *For stationary stochastic processes, limits in $H(\mathcal{E})$ and $H'(\mathcal{E})$ exist and*

$$H(\mathcal{E}) = H'(\mathcal{E})$$

For a proof, see [43].

The entropy rate h represents a lower bound on the compressibility of the sequence of outcomes resulting from the underlying *process*, in contrast to the compressibility of the outcomes of just *one* random variable represented by entropy H (i.e. the compressibility of this variable's probability distribution). This makes entropy rate a more promising guide for selecting a process mining approach.

4.1 Block-based Estimators

We now have a formal definition of the entropy rate of a process, but they are defined as limits where the number of observed events approaches infinity. In reality, we work with event logs of finite length, and must rely on estimates given the available data. From the definitions of *per-symbol* and *conditional* entropy rates, two estimators based on block entropy follow. Namely from

$$\begin{aligned} h = \lim_{k\to\infty} h_k &= \lim_{k\to\infty} \frac{\mathsf{entropy}_k^{bl}}{k} \\ &= \lim_{k\to\infty} \mathsf{entropy}_{k+1}^{bl} - \mathsf{entropy}_k^{bl} \end{aligned}$$

we obtain the following two entropy rate estimators. Each relies on making a particular choice of k, since we cannot in practice let k tend towards infinity; we will see how to choose a value for k in the next subsection (see also [25]).

Definition 21 (Ratio-based k-block entropy rate) Let L be a log. The *ratio-based* k-block entropy rate estimator is given by the ratio of the k-block entropy to the block's length k. The flattened k-block estimator uses the flattened k-block estimator:

$$\mathsf{rate}_k^r(L) = \frac{\mathsf{entropy}_k^{bl}(L)}{k} \qquad \mathsf{rate}_k^{fr}(L) = \frac{\mathsf{entropy}_k^{fbl}(L)}{k} \tag{21.1}$$

Example 15 Consider log L_4 in Figure 5. The flattened 2-block entropy of the log is approximately 3.03, giving

$$\mathsf{rate}_2^{fr}(L_4) = \tfrac{3.03}{2} \approx 1.51$$

Definition 22 (Difference-based k-block entropy rate) Let L be a log. The *difference-based* k-block entropy rate estimator is based on definition (20.3) of entropy rate and is given by the difference between the $k+1$-block entropy and the k block entropy:

$$\mathsf{rate}_k^d(L) = \mathsf{entropy}_{k+1}^{bl}(L) - \mathsf{entropy}_k^{bl}(L)$$

$$\mathsf{rate}_k^{fd}(L) = \mathsf{entropy}_{k+1}^{fbl}(L) - \mathsf{entropy}_k^{fbl}(L) \tag{22.1}$$

Example 16 Consider logs L_1 and L_3 in Figure 1. The k-block entropy and corresponding block-based entropy rates for $1 \leq k \leq 6$ are shown Table 2.

As we see blocks of increasing length, the estimators approach the true entropy rate of the underlying process from above. While they are equal in the limit, at any $k < \infty$, $\mathsf{rate}_k^r \geq \mathsf{rate}_k^d \geq h$ [25].

However, block-based estimators break down at longer block lengths due to the lack of sufficient samples for the frequencies to serve as a valid empirical estimate of the true probability $p(\langle s_1, \ldots, s_k \rangle)$ [25, 36]. This is seen clearly in Figure 7 by the fact that as k grows the entropy rate falls to zero and even goes negative. We now discuss methods for determining a limit to block length that guarantees a sufficient number of samples.

4.1.1 Sufficient Statistics

Entropy rate is formally defined for a sequence as its length approaches infinity, but in practice we must choose a finite value of k for the estimators, (21.1), (22.1). We cannot simply choose the k corresponding to the longest trace due to *under-sampling*: there are not enough examples of k-blocks of this length and all but the longest traces will be omitted. We must therefore find a strategy for choosing a k which is small enough that a sufficient number of k-blocks are observed in the log, yet large enough that the k-blocks reach a length such that the entropy estimate begins to converge to the "true" entropy rate as defined in (20.2) and (20.3). We present five methods for choosing k based on previous research on estimating entropy of very short sequences, drawing heavily upon [25], where detailed derivations of the following constraints can be found.

Interestingly, these constraints in some cases rely on knowing upfront the "true" entropy, which one obviously does not; in practice, one implements these constraints in an iterative fashion, using in each round the previous estimate of entropy as the "true" entropy.

In a scenario of adequate sampling, the length of k-blocks is bounded above by the following function of K, the total length of a sequence, h the entropy rate, and alphabet size $|\Sigma|$:

$$k < \frac{\log K}{h} \qquad \text{if} \qquad h \to \mathcal{O}(1) \tag{22.2}$$

$$k < \frac{Kh}{\log |\Sigma|} \qquad \text{if} \qquad h \to 0 \tag{22.3}$$

Where $\mathcal{O}(1)$ denotes some nonzero constant, meaning that the bound depends on whether the underlying process is fully predictable in the limit.

For "short" sequences (length of less than about 1200, according to [25], as will be common in a BPM context

	Logs	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
$\mathsf{entropy}^{fbl}$	$L_{1,2}$	2.9	3.09	3.32	3.38	3.25	3.0
	L_3	2.85	4.5	5.1	5.0	4.86	4.09
rate_k^{fd}	$L_{1,2}$	2.9	0.19	0.24	0.05	-0.12	-0.25
	L_3	2.85	1.65	0.6	-0.1	-0.42	-0.5
rate_k^{fr}	$L_{1,2}$	2.9	1.54	1.11	0.84	0.65	0.5
	L_3	2.85	2.25	1.7	1.25	0.92	0.68

Table 2: Flattened block-based entropy rate.

where sequences are event traces)[4] we are in a scenario of under-sampling meaning that the bound in (22.2), (22.3) will be too lenient. To determine stricter bounds on k, the authors of (see [25]) draw upon research showing that block based entropy estimates tend to break down when $|\Sigma|^k \approx K$ [36] and the fact that a sequence of length K contains $\frac{K}{k}$ non-overlapping k-blocks to derive the constraint

$$K \geq k|\Sigma|^k \tag{22.4}$$

This formulation assumes sequences of i.i.d. variables, i.e. that correlations between k-blocks result exclusively from their overlap. Dropping this assumption leads to a more stringent constraint based on an adjusted sequence length scaled by the entropy rate, which serves as a measure of correlation not due to block overlap. The upper bound on entropy rate is given by $\log|\Sigma|$, which represents a scenario in which every outcome in Σ is equally probable, regardless of the preceding observations, i.e. $p(s_n|s_{n-1},\ldots,s_1) = p(s_n)$. This leads to a normalised entropy rate $\frac{h}{\log|\Sigma|}$, which can be used to define an "effective" sequence length

$$K_{\text{eff}} \approx K\frac{h}{\log|\Sigma|}$$

Substituting K with K_{eff} in (22.4), we have the new constraint

$$Kh \geq k|\Sigma|^k \log|\Sigma| \tag{22.5}$$

However, taking advantage of the asymptotic equipartition property we can relax this constraint, diminishing the influence of the size of the alphabet. Specifically, for discrete time, finite valued, stationary, ergodic sources the Shannon-McMillan-Breiman theorem, "states that the number of k-blocks of non-negligible probability that actually contribute to entropy is not $|\Sigma|^k$, but 2^{kh}" (see [25] and [43]), giving

$$Kh > k2^{kh}\log|\Sigma| \tag{22.6}$$

Using these constraints, we can find the highest value of k such that the estimators rate_k^r and rate_k^d remain valid, i.e. that enough examples of blocks of length k have been observed to consider their frequencies (11.1) as a reasonable estimate of the true underlying probability distribution.

We stress that these constraints are formally defined for single sequences of length K whereas we are considering sets of sequences, which affects the number of k-blocks we will observe, depending on the distribution of trace lengths in the log: some logs may consist almost entirely of very short traces with only a few long traces, in which case longer k-blocks will still be under-sampled. In our implementation we have used the length of the longest trace as K. Our current results use the constraints defined for single sequences; leaving the generalisation to future work.

In summary, to apply either of the entropy estimators of Definition 21.1 or 22.1, first choose one of the constraints (22.2)–(22.6); then apply the estimator at the maximum k which satisfies that constraint.

Example 17 Consider $\log L_3$ from Figure 1. We will compute rate_k^d up to the highest k which satisfies constraint (22.5). The longest trace is $\langle a,d,a,c,a,c,e,h,h,f,g\rangle$ with a length of 11. The alphabet has size

$$\big|\Sigma\big| = \big|\{a,b,c,d,e,f,g,h\}\big| = 8.$$

The highest value for k which satisfies constraint (22.5) is just 1, this gives $\mathsf{rate}_1^{fd} = \mathsf{entropy}_2^{fbl} - \mathsf{entropy}_1^{fbl} = 4.5 - 2.85 = 1.65$.

k	rate_k^{fd}	$K\hat{h} \geq k\lvert\Sigma\rvert^k \log\lvert\Sigma\rvert$
1	2.85	$11 \times 2.85 \geq 1 \times 8^1 \log 8 \iff 31.35 \geq 24$
2	1.65	$11 \times 1.65 \not\geq 2 \times 8^2 \log 8 \iff 18.15 \not\geq 384$

4.2 Lempel-Ziv Estimators

Lempel-Ziv entropy estimators are based on the properties of the sequence compression algorithms introduced by Abraham Lempel and Jacob Ziv in two papers from 1977 and 1978 [45,46]. These algorithms can be proven to be asymptotically optimal: in the limit they will converge to the entropy rate h when compressing a sequence resulting from a stationary, ergodic source [43].

[4] In the literature on entropy estimation, sequences of this length are considered relatively short since the data under consideration are often large text corpora or outputs from simulations of dynamic systems.

Both versions work by parsing a sequence into unique words (blocks), and are universal coding schemes [43], meaning that they do not rely on the probability distribution of the underlying source. The cost of universality is a higher complexity of the encoder and decoder.

We will use the 1978 version of the compression scheme which moves through a sequence $\langle e_1, \ldots, e_n \rangle$, parsing it into the shortest unique word not yet encountered.

Example 18 Consider the last trace from our running example log L_3 in Figure 1, (a start symbol \$ is included for indexing purposes):

$$\langle \$, f, g, h, f, g, h, h, h \rangle$$

The first, shortest word encountered is f, the second g, and the third h. The fourth element f has already been encountered, so we append the fifth element to the word to arrive at a new word fg. The sixth element h has been seen, so the seventh is appended giving hh. The final element h has already been seen, so no new words are added to the dictionary. This results in the following dictionary in which each word is represented by a tuple containing the index of a prevously encountered word, and a character to append, creating a new word:

$$\langle \$, f, g, h, fg, hh \rangle = \big\langle \langle 0, f \rangle, \langle 0, g \rangle, \langle 0, h \rangle, \langle 1, g \rangle, \langle 3, h \rangle \big\rangle$$

Definition 23 (Lempel-Ziv entropy rate) Let L be a log. Let D_L be the dictionary of subsequences resulting from the parsing of all traces in L using the Lempel-Ziv compression scheme. Then the Lempel-Ziv estimate of the entropy rate of the process generating L is given by

$$\mathsf{rate}^{LZ}(L) = \frac{|D_L| \log \sum_{\sigma \in L} |\sigma|}{\sum_{\sigma \in L} |\sigma|}$$

Definition 24 (F) lattened Lempel-Ziv entropy rate]

$$\mathsf{rate}^{fLZ} = \mathsf{rate}^{LZ} \circ \mathsf{flatten}(L)$$

5 Implementation and Experiments

We present in this section the results of empirically measuring entropy as defined by the various measures in the preceding sections on both synthetic and real-life logs.

5.1 Implementation

To test the various measures we implemented a command line utility for computing the measures introduced in the preceding sections. Using this implementation, we report here the values of the discussed measures.

The implementation is not entirely trivial: in particular, use of both prefix- and suffix tree data structures was necessary to compute the global block entropy in reasonable time on available memory. We note that the Hospital Log and BPI Challenge 2017 were both particularly challenging in this respect. We also used an iterative version of the Levenshtein distance algorithm with a number of heuristics (checking lower bounds and common pre- and suffixes) to avoid unnecessary calculations. While this drastically improves performance, this estimator clearly stands out as the most computationally expensive estimator.

The implementation, along with our results in machine readable format, is available at [5].

5.2 Real-life Logs

We have evaluated a selection of real-life logs available at [1]. We summarise the logs in Table 3.

For those logs which include distinguishing lifecycle transitions for activities such as "*start*", "*pending*" or "*complete*", we include both the log in which these are included (denoted with an asterisk) and those in which they are ignored. When lifecycle transitions are included, activities like "*a + start*" and "*a + pending*" are considered unique events. This is arguably the only defensible approach since lifecycle transitions are included to make a distinction between activities and there is no reason to throw this information away when estimating entropy. However, in the interest of completeness, we report on both versions.

There is not yet a clear agreement in the literature on which of these logs are better suited for imperative or declarative mining. However, the BPI Challenge 2012 log has been used as a use case for hybrid mining algorithms [28], showing a strong advantage over purely imperative miners. We also note that both the Sepsis cases and Hospital log originate from highly flexible and knowledge-intensive processes within a Dutch hospital. A recent investigation involving the BPI Challenge 2013 (incidents) log seemed to indicate that an imperative approach may be the more successful, but draws no concrete conclusions [35]. For every log of Table 3, we report measurements of entropy (Section 3) in Table 4, and measures of entropy rate (Section 4.1) in Table 5. We also present graphs of the block-based entropy rate as a function of block length(k) for two logs, to illustrate how the estimators converge differently.

5.3 Artificial Logs

In order to further illuminate the potential capability of entropy measures to distinguish declarative from imperative logs, we apply in this section the measures to a selection of artificially generated logs. We evaluate the measures on

Log	Act.	Traces	Events	Comment
			DECLARATIVE INDICATIONS	
Hospital Event Log	624	1143	150291	Knowledge-intensive process
Road Traffic Fine	11	150370	561470	Declarative indications in [35]
Sepsis Cases	16	1050	15214	Highly flexible, knowledge-intensive process, declarative indications in [31, 35]
L_3	8	40	280	Declarative by construction
			HYBRID INDICATIONS	
BPI Challenge 2012	24	13087	262200	Loan application process, successfully
BPI Challenge 2012*	36	13087	262200	used for hybrid mining [28]
			IMPERATIVE INDICATIONS	
BPI Challenge 2013	4	7554	65533	Imperative indications in [35]
BPI Challenge 2013*	13	7554	65533	”
NASA CEV split*	94	2566	73638	Imperative miners return succinct models
L_1	8	40	280	Imperative by construction
L_2	8	40	280	Language-equivalent with L_1
L_4	8	50	280	Imperative by construction
			NO INDICATIONS	
BPI Challenge 2017	26	31509	1202267	
BPI Challenge 2017*	66	31509	1202267	
WABO - Receipt	27	1434	8577	
Hospital Billing	18	100000	451359	

Table 3: Overview of real-life logs measured as well as running examples. All logs but L_1 through L_4 from [1]. Logs marked with an asterisk(*) include the lifecycle transition of events.

Log	entropy^{tr}	entropy^{pr}	entropy^{bl}	entropy^{KL}	entropy^{knn} k=1	k=2	k=3	k=4
				DECLARATIVE INDICATIONS				
Hospital Event Log	9.63	16.79	24.71	7.13	7.13	6.22	5.77	5.47
Road Traffic Fine	2.48	6.51	8.73	4.64	4.64	3.76	3.4	3.18
Sepsis Cases	9.33	10.6	14.66	6.16	6.16	5.35	4.96	4.7
L_3	3.0	5.63	7.04	2.78	1.3	1.99	1.56	1.27
				HYBRID INDICATIONS				
BPI Challenge 2012	7.75	12.53	15.99	7.28	7.28	6.47	6.07	5.81
BPI Challenge 2012*	7.75	12.54	16.28	7.34	7.34	6.51	6.11	5.85
				IMPERATIVE INDICATIONS				
BPI Challenge 2013	6.67	11.32	12.23	6.17	6.17	5.28	4.86	4.58
BPI Challenge 2013*	7.48	11.81	14.01	6.95	6.95	6.07	5.64	5.36
NASA CEV split*	11.27	10.12	14.81	6.84	6.84	5.97	5.51	5.27
L_1	3.0	4.09	5.75	1.17	1.3	0.37	-0.09	0.13
L_2	2.55	4.09	5.75	1.17	1.3	0.37	-0.09	0.13
L_4	3.32	4.82	4.75	2.08	2.19	1.19	0.69	0.35
				NO INDICATIONS				
BPI Challenge 2017	11.99	14.42	17.45	8.16	8.16	7.38	7.01	6.76
BPI Challenge 2017*	12.23	14.63	18.24	8.53	8.53	7.71	7.31	7.05
CoSeLog - Receipt	3.21	7.72	10.05	4.43	4.44	3.63	3.26	3.02
Hospital Billing	3.17	9.43	10.49	6.04	6.04	5.22	4.87	4.64

Table 4: Estimates of log entropy (H). Logs marked with an asterisk(*) distinguish events by lifecycle transition.

	Log	rate^{fLZ}	$\mathsf{rate}^{fd}_{k+1} = \mathsf{entropy}^{fbl}_{k+1} - \mathsf{entropy}^{fbl}_{k}$				$k+1$						
			(22.2)	(22.3)	(22.4)	(22.5)	(22.6)	(22.2)	(22.3)	(22.4)	(22.5)	(22.6)	$\lvert\Sigma\rvert$	K
DECLARATIVE INDICATIONS	Hospital Event Log	2.89	≈ 0	0.08	2.2	2.2	1.09	33	18	2	2	5	624	1814
	Road Traffic Fine	1.52	-0.1	0.46	1.55	1.55	1.55	7	5	2	2	2	11	20
	Sepsis Cases	1.89	-0.03	0.22	1.88	1.88	1.74	12	10	2	2	3	16	185
	L_3	2.9	-0.1	0.6	1.65	1.65	2.85	4	3	2	2	1	8	11
HYBRID INDIC.	BPI Challenge 2012	0.96	-0.01	0.37	1.33	1.33	0.46	38	14	2	2	5	24	175
	BPI Challenge 2012*	0.98	-0.01	0.35	1.02	1.02	0.41	38	12	2	2	5	36	175
IMPERATIVE INDICATIONS	BPI Challenge 2013	1.16	0.78	0.18	0.88	0.88	0.87	9	17	3	3	5	4	123
	BPI Challenge 2013*	1.71	1.18	0.33	1.57	1.57	1.49	6	11	2	2	3	13	123
	NASA CEV split*	0.84	6.1	0.5	6.1	6.1	6.1	1	4	1	1	1	94	50
	L_1	2.18	-0.12	0.19	0.19	2.9	2.9	5	2	2	1	1	8	8
	L_2	2.07	-0.12	0.19	0.19	2.9	2.9	5	2	2	1	1	8	8
	L_4	2.13	2.58	0.45	2.58	2.58	2.58	1	2	1	1	1	8	4
NO INDICATIONS	BPI Challenge 2017	0.89	≈ 0	0.39	1.33	1.33	0.52	42	15	2	2	5	26	180
	BPI Challenge 2017*	0.97	-0.01	0.42	0.9	0.9	0.61	40	13	2	2	4	66	180
	WABO - Receipt	2.34	-0.17	0.48	3.77	3.77	1.59	8	5	1	1	2	27	25
	Hospital Billing	1.58	-0.11	0.06	1.42	1.42	0.93	10	9	2	2	4	18	217

$\mathsf{rate}^{fr}_{k} = \mathsf{entropy}^{fbl}_{k}/k$

k

DECLARATIVE INDICATIONS	Hospital Event Log	"	3.3	0.29	5.92	5.92	4.06	3	57	1	1	2	"	"
	Road Traffic Fine	"	3.26	1.2	3.26	3.26	3.26	1	6	1	1	1	"	"
	Sepsis Cases	"	2.28	0.49	3.22	3.22	2.55	3	22	1	1	2	"	"
	L_3	"	2.85	1.25	2.85	2.85	2.85	1	4	1	1	1	"	"
HYBRID INDIC.	BPI Challenge 2012	"	1.05	0.58	3.56	3.56	2.45	7	21	1	1	2	"	"
	BPI Challenge 2012*	"	1.22	0.61	4.56	4.56	4.56	6	20	1	1	1	"	"
IMPERATIVE INDICATIONS	BPI Challenge 2013	"	0.94	0.44	1.11	1.11	1.03	7	26	2	2	3	"	"
	BPI Challenge 2013*	"	1.71	0.62	2.42	2.42	1.99	4	20	1	1	2	"	"
	NASA CEV split*	"	6.1	1.22	6.1	6.1	6.1	1	8	1	1	1	"	"
	L_1	"	2.9	1.54	2.9	2.9	2.9	1	2	1	1	1	"	"
	L_2	"	2.9	1.54	2.9	2.9	2.9	1	2	1	1	1	"	"
	L_4	"	2.58	1.51	2.58	2.58	2.58	1	2	1	1	1	"	"
NO INDICATIONS	BPI Challenge 2017	"	1.41	0.61	3.66	3.66	2.5	5	23	1	1	2	"	"
	BPI Challenge 2017*	"	1.86	0.71	5.27	5.27	5.27	4	20	1	1	1	"	"
	WABO - Receipt	"	3.77	1.38	3.77	3.77	3.77	1	6	1	1	1	"	"
	Hospital Billing	"	1.52	0.35	3.44	3.44	2.43	5	16	1	1	2	"	"

Table 5: Estimates of entropy rate, h, according to the Lempel Ziv estimator, the difference-based of k-block estimator, and the ratio-based k-block entropy estimator. For the latter two, block length k is determined according to five different constraints for identifying the crossover from "good statistics" to "poor statistics" [25] on short sequences. K denotes the length of the longest trace and $|\Sigma|$ the size of the alphabet, i.e. number of activities. Logs marked with an asterisk(*) distinguish events by lifecycle transition.

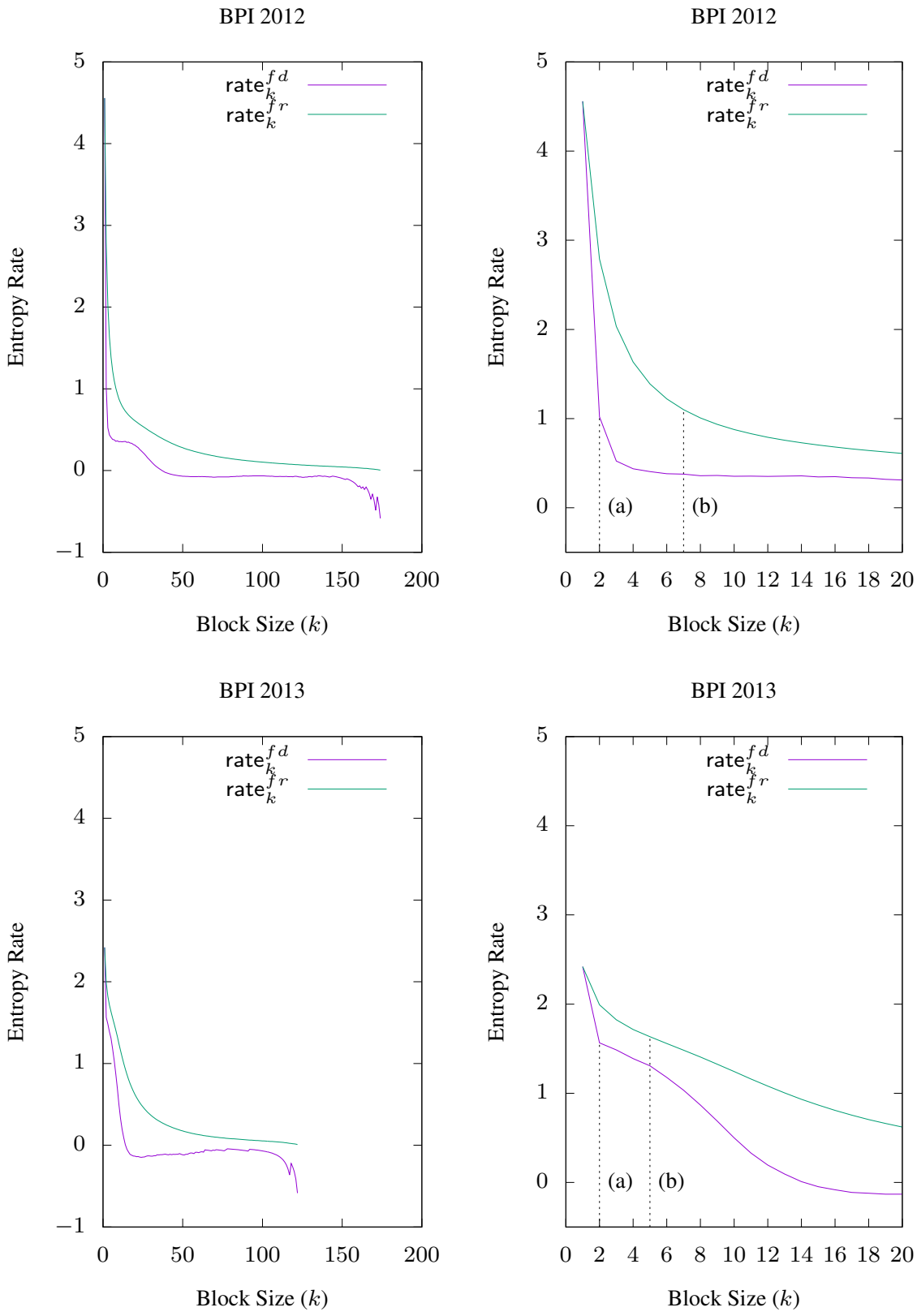

Fig. 7: Block-based estimators of entropy rate (h), as a function of block size (k) for selected logs. The cutoff constraints, $Kh \geq k|\Sigma|^k \log|\Sigma|$ and $k < \frac{\log K}{h}$, are labelled by (a) and (b), respectively. See (22.5) and (22.2)

	Log	Entropy		Entropy Rate		
		$\mathsf{entropy}^{fbl}$	$\mathsf{entropy}^{fKL}$	rate^{LZ}	rate_k^{fd} (22.5)	rate_k^{fr} (22.5)
DECLARATIVE INDICATIONS	Hospital Event Log	24.71	7.25	2.89	2.2	5.92
	Road Traffic Fine	8.73	3.59	1.52	1.55	3.26
	Sepsis Cases	14.65	5.04	1.89	1.88	3.22
	L_3	7.04	2.42	2.9	1.65	2.85
HYBRID INDIC.	BPI Challenge 2012	15.99	7.19	0.96	1.33	3.56
	BPI Challenge 2012*	16.28	7.19	0.98	1.02	4.56
IMPERATIVE INDICATIONS	BPI Challenge 2013	12.22	5.34	1.16	0.88	1.11
	BPI Challenge 2013*	14.01	5.51	1.71	1.56	2.42
	NASA CEV split*	14.81	4.9	0.84	6.1	6.1
	L_1	5.75	1.15	2.18	2.9	2.9
	L_2	5.75	1.15	2.07	2.9	2.9
	L_4	4.75	1.17	2.13	2.58	2.58
NO INDICATIONS	BPI Challenge 2017	17.45	7.4	0.89	1.33	3.66
	BPI Challenge 2017*	18.24	7.57	0.97	0.9	5.27
	WABO - Receipt	10.05	3.74	2.34	3.77	3.77
	Hospital Billing	10.49	5.16	1.58	1.42	3.44

Table 6: Most promising measures across real-life logs and running examples. Logs with declarative, hybrid, and imperative indications should have the highest, middle and lowest values, respectively. No estimators are able to separate logs exactly as desired. Logs marked with an asterisk(*) distinguish events by lifecycle transition.

three sets of logs: one generated from Petri nets with varying degrees of noise, one generated from Declare models with varying degrees of restrictiveness, and a log with varying degrees of concurrency.

5.3.1 Petri net based logs with noise

To evaluate the effect of noise and infrequent behaviour in imperative processes on entropy estimates, we use the set of 120 event logs from [44]. These logs, each containing 1000 traces, are generated from four typical imperative process patterns which can cause difficulty for process mining algorithms. Then, five different forms of noise are introduced in the form of an activity which is added (1) infrequently or very infrequently and (2) locally (in one position), semi-locally (in one of two positions) or globally (anywhere in the process). Along with the noise-free log, this results in six sets of five logs each for each of the four process patterns (the semi-local/very infrequent noise combination is omitted in the original logs).

The four process patterns employed are as follows:

Parallel Five activities which can occur once, in any order.

Skip Three activities, one of which may be skipped.

Duplicates A sequence of activities with one activity occurring twice.

Non-free Choice A choice at the end of the process depends on an earlier choice.

Observations. We report the results of experiments in Figures 8 and 9.

First, we note that entropy measures tend to agree on the relative ordering of the four logs, with the exception of *Skip* and *Duplicates*. Entropy rate measures, on the other hand, tend to rank *Non-free Choice* as having the highest entropy rather than *Parallel*.

Most entropy measures are affected by noise, with the exception of nearest-neighbour approaches on the *Parallel* log. Entropy rate measures perform much more poorly when faced with noise. We note in particular that entropy measures generally preserve the *relative ordering* of the four logs, whereas entropy rate measures exhibit much more "crossover": the addition of noise switches the relative ranking of two or more of the logs.

5.3.2 Declare based logs

We evaluate the output of entropy estimators on declaratively generated logs in terms of model restrictiveness versus expressiveness. Specifically, we measure not only the effect of the number of constraints, but also *types* of constraints, as well as number of activities. Each log consists of 1000 traces with lengths between 5 and 10 events, in order to ensure a comparability with the imperatively generated logs, which are of similar length.

Logs were generated from Declare models using the artificial log generator described in [18]. Insofar as possible, models were built by varying along one dimension at a time,

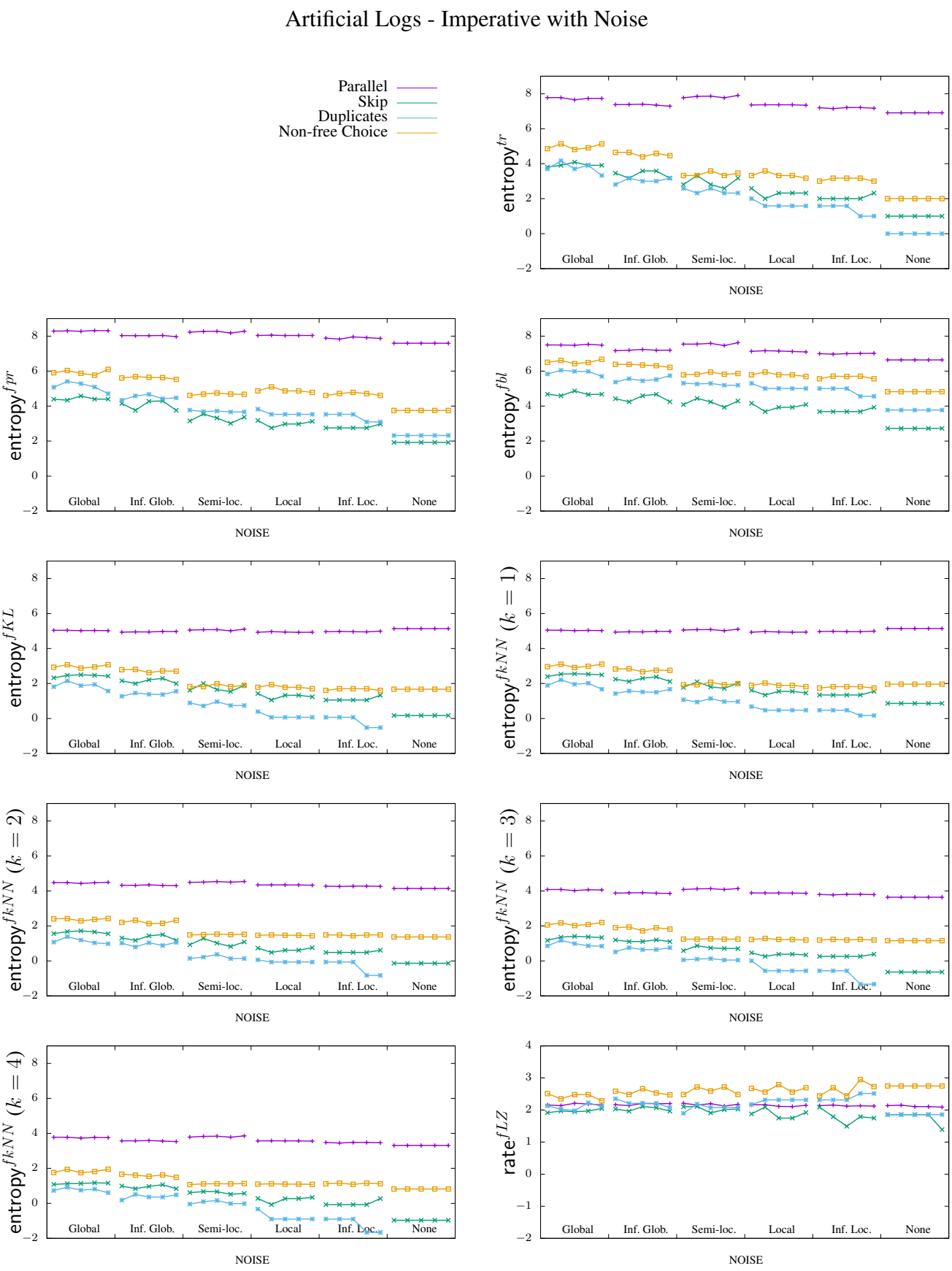

Fig. 8: Estimates of entropy (H) and entropy rate (h) for the "Testing Representational Bias" set of artificial logs. Logs have different degrees and types of noise: infrequent, very infrequent ("inf."); and global, semi-local and local.

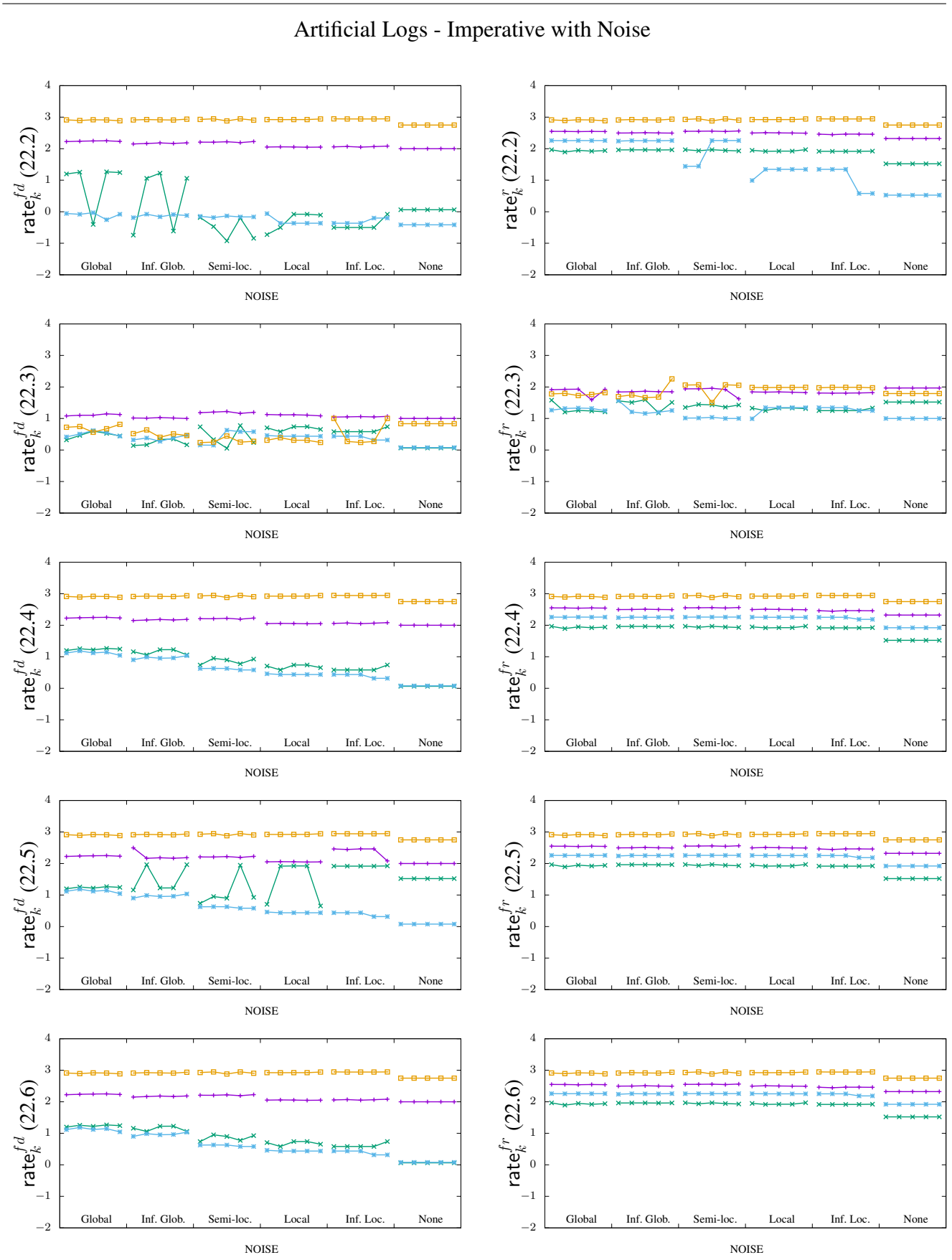

Fig. 9: Estimates of entropy (H) and entropy rate (h) for the "Testing Representational Bias" set of artificial logs. Logs have different degrees and types of noise: infrequent, very infrequent ("Inf."); and global, semi-local and local.

Insensitive to Noise	Correctly Distinguishes Noise	Partly Correct	Skewed by Noise
$\mathsf{rate}_k^{fr}(22.4)$	$\mathsf{entropy}^{tr}$	$\mathsf{entropy}^{fKL}$	$\mathsf{rate}_k^{fd}(22.2)$
$\mathsf{rate}_k^{fr}(22.5)$	$\mathsf{entropy}^{fpr}$	$\mathsf{entropy}^{fkNN}(k=1)$	$\mathsf{rate}_k^{fd}(22.3)$
$\mathsf{rate}_k^{fr}(22.6)$	$\mathsf{entropy}^{fbl}$	$\mathsf{rate}_k^{fd}(22.5)$	$\mathsf{rate}_k^{fr}(22.2)$
	$\mathsf{entropy}^{fkNN}(k=2,3,4)$	$\mathsf{rate}_k^{fr}(22.3)$	rate^{LZ}
	$\mathsf{rate}_k^{fd}(22.4)$		
	$\mathsf{rate}_k^{fd}(22.6)$		

Table 7: Influence of noise on estimators of entropy and entropy rate.

holding others constant. That is, to measure the effect of number of constraints, the type of constraint and number of activities was held constant, and number of constraints incrementally increased. Similarly, to measure the effect of type of constraint, the number of constraints and activities was held constant while the restrictiveness of the constraint type was increased *within its subsumption hierarchy*.

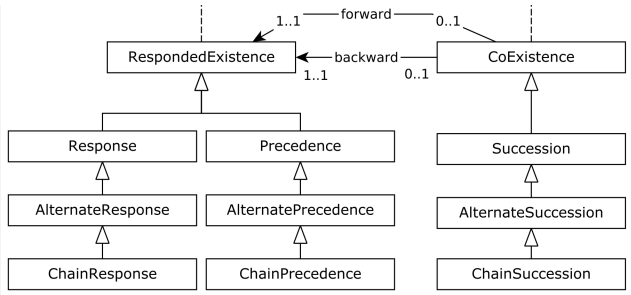

Fig. 10: Portion of the Declare constraint subsumption hierarchy from [18].

Constraint ordering Declare constraints fall into a partial ordering in terms of their restrictiveness [19]. First, they fall into a subsumption hierarchy within the same constraint type e.g. *AlternateSuccession* is more restrive than *Succession*. Second, some constraint types are more restrictive than others e.g. *Succession* is more restrictive than *Response*.

When considering the size of the model space resulting from varying these three dimensions (number of constraints, type of contraints, number of activities), we have favoured systematically adjusting individual parameters over exploring larger, more diverse models. The configuration of constraints adds a potential fourth dimension, which we've chosen to hold constant unless doing so resulted invalid models, which only was the case for some *Chain* constraints.

Monotonicity Finally, we note that the effect of alphabet size in Declare models on entropy measures will always be monotonically increasing when other factors are held constant, since this increases the number of possible outcomes in the sample space, broadening and flattening the probability distribution. Recall also the conjunctive nature of Declare: adding constraints always results in a more restrictive model.

Observations We report the results of experiments along with examples of the Declare models in Figures 11 and 13.

The resulting entropy measures largely fall in line with expectations, with rate_k^{fd} and rate_k^{fr} displaying a pronounced sensitivity to the choice of k-block cutoff constraint.

The most marked trend is the clear effect of *constraint type*. A clear drop in all entropy measures occurs in models with *Alternate* and *Chain* constraint types, with the effect of increased number of constraints being more pronounced for these constraint types as well.

Notably, the number of constraints has a very diminished effect for models consisting of less restrictive constraint types so that, for example, models with five *CoExistence* constraints have a higher entropy across estimators than a model with just one *AlternateSuccession* constraint.

Furthermore, we note that the results mirror the restrictiveness between parallel branches of constraint subsumption hierarchies: *Response* is somewhat more permissive than *Succession*, and this trend is also clear though less pronounced than the effect with subsumption hierarchies. Finally, the effect of adding activities is as expected: models with more activities result in higher entropy estimates.

The effect is clear: once models approach a degree of expressiveness akin to imperative models of moderate complexity, the entropy measure suddenly begins dropping. This matches the intuition that a process resulting from a flexible declarative process will have markedly higher entropy, even if that model consists of many, semantically meaningful constraints.

5.3.3 Concurrent Log

In order to study in detail the effect of concurrency on entropy estimates, we generated a set of logs based on models similar to the *parallel* model above, which vary in the number of activities in the concurrent block and are flanked by tails of sequential activities (see Figure 15). Specifically, we consider traces with a total length ranging from 9 to 26 (with number of unique activities equal to the total trace length). For each total trace length, we generate event logs with blocks of concurrency ranging from 1 to 9 activities. Denoting the number of activities in the concurrent block by j, each event log contains $j!$ unique traces.

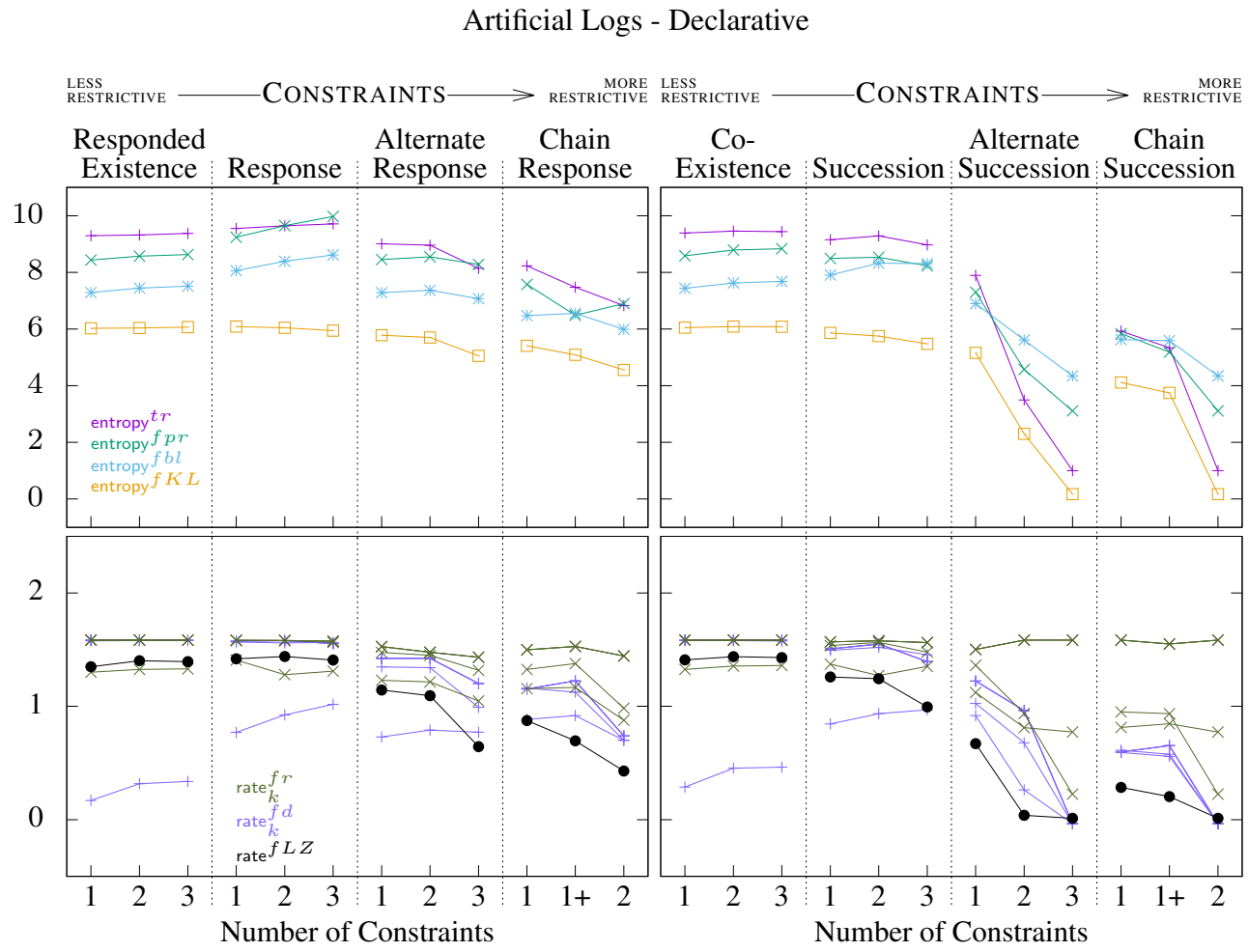

Fig. 11: Entropy measure of artificial event logs generated from Declare models with three activities and different numbers, and types, of constraints. All models consist of the same type of constraint and configuration, excepting *Chain* constraints. Constraints are listed in order of restrictiveness, i.e. their subsumption ordering.

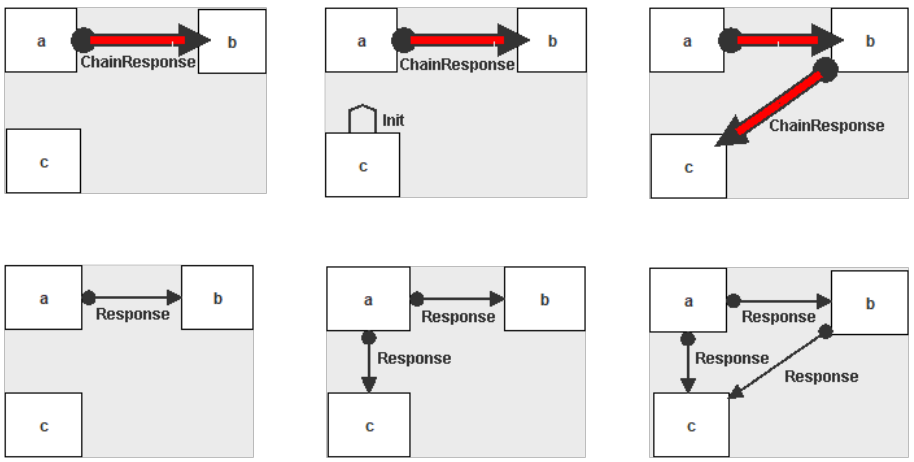

Fig. 12: Examples of the Declare models used to generate artificial logs. All models follow same the configuration as the *Response* models displayed, except for *ChainResponse* since this would lead to an invalid model.In this case,the *Init* constraint has been introduced to create a corressponding number of models with a similarly increasing degree of restrictiveness.

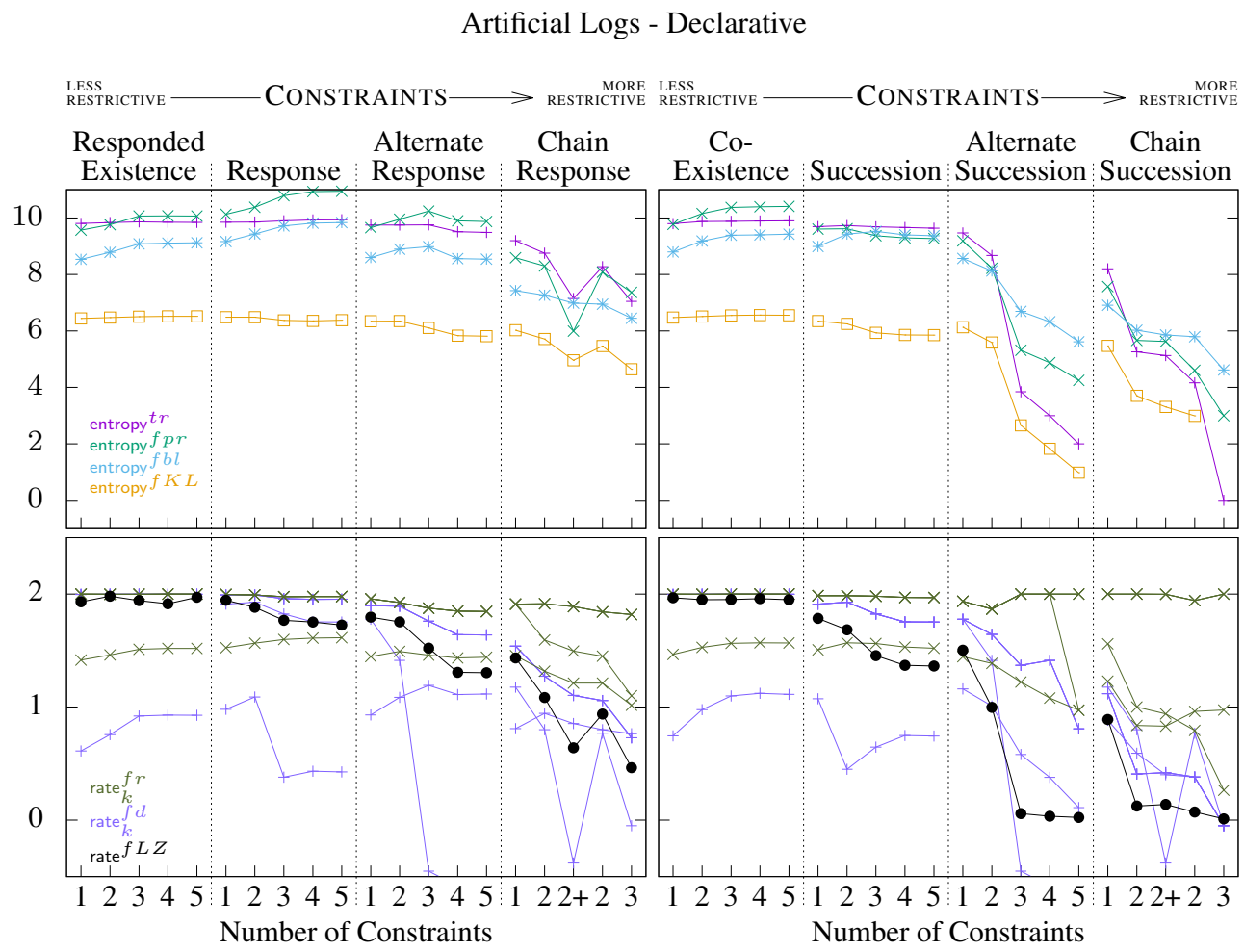

Fig. 13: Entropy measure of artificial event logs generated from Declare models with four activities and different numbers, and types, of constraints. All models consist of the same type of constraint and configuration, excepting *Chain* constraints. Constraints are listed in order of restrictiveness, i.e. their subsumption ordering.

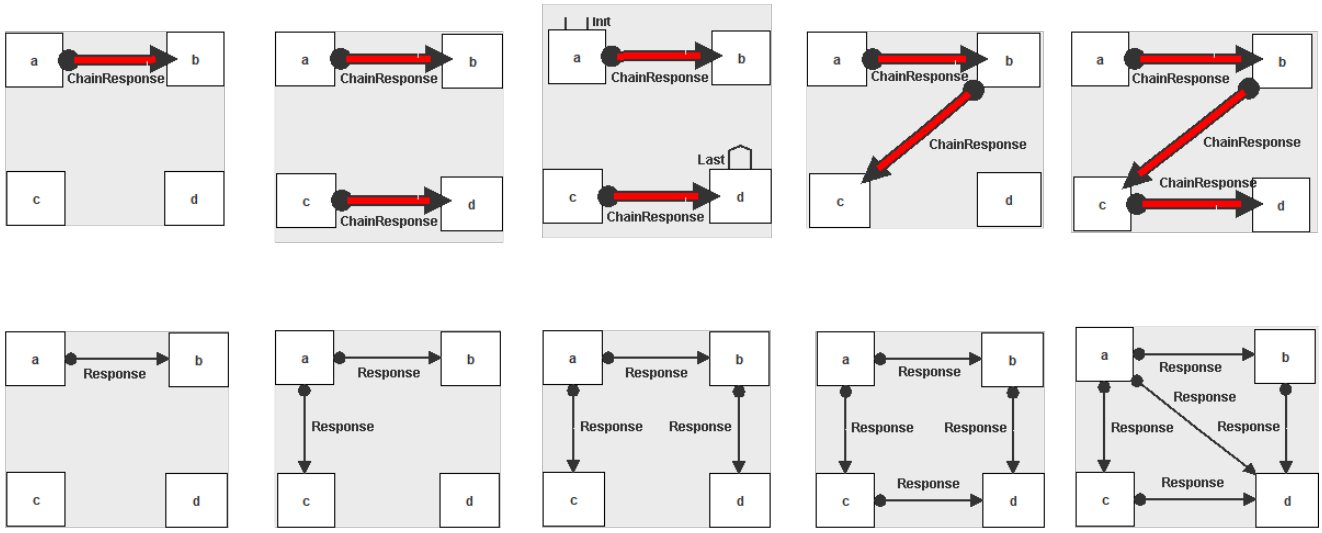

Fig. 14: Examples of the Declare models used to generate artificial logs. All models follow same the configuration as the *Response* models displayed, except for *ChainResponse* since this would lead to an invalid model. In this case,the *Init* and *Last*/*End* constraints have been introduced to create a corresponding number of models with a similarly increasing degree of restrictiveness.

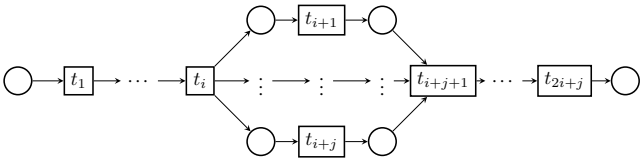

Fig. 15: A Petri net representing a block of concurrency of size j flanked by sequential tails of length i on either end. Activities in the concurrent block can be ordered in $j!$ different permutations.

$\mathbf{j=1}$	$\mathbf{j=2}$
$\langle a,b,c,d,\mathbf{e},f,g,h,i\rangle$	$\langle a,b,c,\mathbf{d},\mathbf{e},f,g,h,i\rangle$
$\langle a,b,c,\mathbf{e},\mathbf{d},f,g,h,i\rangle$	$\langle a,b,c,\mathbf{d},\mathbf{f},\mathbf{e},g,h,i\rangle$
$\mathbf{j=3}$	$\mathbf{j=4}$
$\langle a,b,c,\mathbf{d},\mathbf{e},\mathbf{f},g,h,i\rangle$	$\langle a,b,\mathbf{c},\mathbf{d},\mathbf{e},\mathbf{f},g,h,i\rangle$
$\langle a,b,c,\mathbf{e},\mathbf{d},\mathbf{f},g,h,i\rangle$	$\langle a,b,\mathbf{c},\mathbf{e},\mathbf{d},\mathbf{f},g,h,i\rangle$
$\langle a,b,c,\mathbf{e},\mathbf{f},\mathbf{d},g,h,i\rangle$	$\langle a,b,\mathbf{c},\mathbf{e},\mathbf{f},\mathbf{d},g,h,i\rangle$
$\langle a,b,c,\mathbf{f},\mathbf{d},\mathbf{e},g,h,i\rangle$	$\langle a,b,\mathbf{c},\mathbf{f},\mathbf{d},\mathbf{e},g,h,i\rangle$
$\langle a,b,c,\mathbf{f},\mathbf{e},\mathbf{d},g,h,i\rangle$	$\langle a,b,\mathbf{c},\mathbf{f},\mathbf{e},\mathbf{d},g,h,i\rangle$
	$\langle a,b,\mathbf{d},\mathbf{c},\mathbf{e},\mathbf{f},g,h,i\rangle$
	$\vdots$

Table 8: Traces of total length 9, with blocks of concurrency of size j in the middle.

Observations. We report the results of experiments in Figure 16.

As the block of concurrency grows, we see that estimates of entropy (H) consistently increase. Estimates of entropy rate are less consistent: several block-based estimators level off at a high value (3-5 depending on total trace length) very quickly because the constraint on block length is violated almost immediately. Other block based estimators grow logarithmically, appearing to converge to values between 2-3.

Most striking is the Lempel-Ziv estimator which *falls* with concurrent block size, levelling off at values between 1 and 2.5 depending on total trace length. The high estimates for blocks of length 1, 2 and 3 are most likely due primarily to smaller log sizes. Concurrent logs with many traces will nonetheless contain many shared subsequences between traces, allowing for greater compression. In this sense, Lempel-Ziv in particular and entropy *rate* estimates in general, are not "tricked" by concurrency in the same way that simple entropy estimates are prone to.

The motivation for considering logs with blocks of concurrency logs is the intuition that, although this behaviour can be succinctly captured by a simple Petri net, the large degree of variation between traces would perhaps lead to inappropriately high entropy values. However, a concurrent block can be equally, if not more, succinctly modelled using a declarative model with an *ExactlyOne* constraint on each activity in the block. This is an edge case, and in the circumstance in which a large degree of concurrency exists, but with some constraints, declarative models are capable of capturing this behaviour much more succinctly.

Those logs in which long sequential tails flank a concurrent block may indeed be more succinctly modelled imperatively, but the lower values given by several entropy rate estimators, in particular Lempel-Ziv, do in fact reflect the greater degree of structure in such logs. This is in line with the claim that low entropy logs are better suited to be modelled imperatively. To see this, consider that for logs with a concurrent block of size 9 and no tails, the Lempel-Ziv entropy is high: about 2.5; whereas in the case of the same block flanked by sequential tails of length 9, the entropy falls to about 1.0 (see Figure 16).

6 Discussion

In the previous section we reported entropy measurements generated by a variety of estimators. Most measures were broadly in line with expectations, especially on artificially generated logs, with some proving more robust than others.

To guide our investigation, we designed the synthetic logs L_1, L_2, L_4 to be suitable for imperative mining, and L_3 for declarative mining. Moreover, as indicated in Table 3, the sentiment in the community is that the BPI Challenge 2012, Hospital, and the Sepsis Cases logs are well-suited for declarative mining. Finally, we employed several sets of artificial logs for which the generating model is known. If we take as canonical these indicators, the most promising measures for predicting suitability for imperative and declarative mining appear to be:

$\mathsf{entropy}^{fbl}$
$\mathsf{entropy}^{fKL}$ and $\mathsf{entropy}^{fkNN}$
rate^{fLZ}
rate_k^{fd} and rate_k^{fr} using constraint (22.5)

We summarise these particular measures in Table 6. No one measure is able to classify all logs exactly as desired, which is unsurprising. In practice, several of the strongest estimators might be used in combination, or even as input to a classification algorithm along with other log attributes.

We note that these five estimators perform quite differently when confronted with noise. In particular, rate^{fLZ} is unstable in the presence of noise. The remaining estimators are either able to clearly identify noise, or are insensitive to it, in both cases maintaining a consistent relative ranking of logs. Whether sensitivity to noise is desirable will depend on the task at hand.

Aside from the case of artificial logs, the present evaluation suffers from a significant degree of uncertainty regarding the labelling of logs as "declarative" or "imperative". A

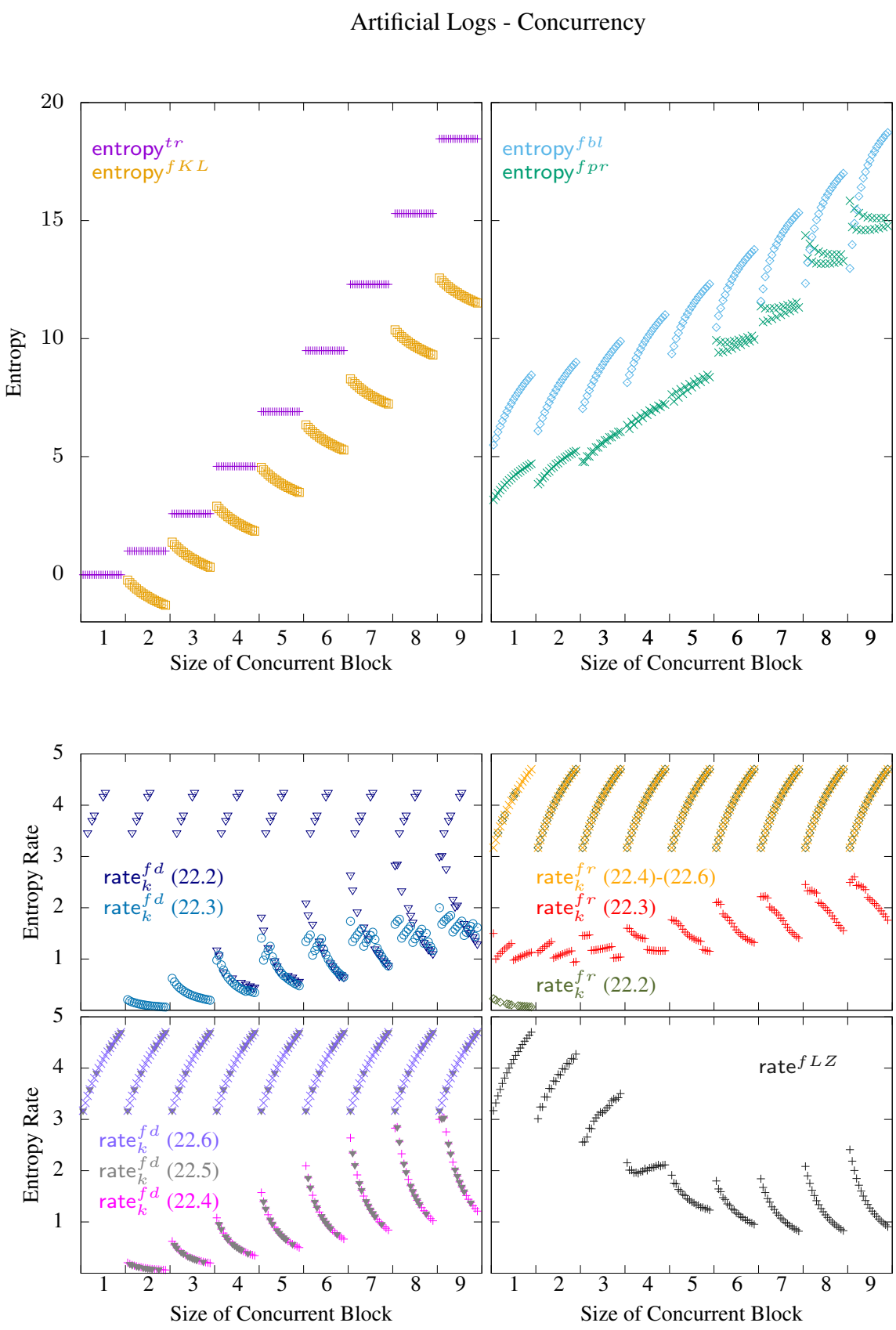

Fig. 16: Estimates of entropy (H) and entropy rate (h) of artificial concurrent logs. Data points are clustered by size of concurrent block (j) with each cluster containing 18 points, representing traces with total length from ranging from 9 to 26, from left to right within each cluster.

more thorough evaluation would involve a *quantitative* analysis of the logs to determine whether they are better suited for imperative or declarative mining according to objective criteria. One way to approach such an analysis could be to mine each log with state-of-the-art imperative and declarative miners and compare the resulting models according to accepted quality criteria which can be applied to models of both paradigms, e.g., [11]. The concrete outlines for such a study have been proposed in [6].

6.1 Entropy measures (H)

We determine that $\mathsf{entropy}^{tr}$ and $\mathsf{entropy}^{fpr}$ have a number of shortcomings, which are partly addressed by $\mathsf{entropy}^{fKL}$, $\mathsf{entropy}^{fkNN}$ and $\mathsf{entropy}^{fbl}$. However, all of these estimators are sensitive to the absolute size of logs, i.e. the length of traces and number of activities. All four seem to clearly misclassify one log in particular: the Road Traffic Fines Management log which receives a low entropy, but is characterised as a declarative process in the literature.

These measures are also strongly affected by concurrency, growing steadily with the size of the concurrent block, regardless of the presence of strictly sequential prefixes and suffixes. One aspect on which these measures perform surprisingly well is in detecting noise, clearly reflecting for most logs in "Testing Representational Bias" event log the degree and type of noise added.

6.1.1 Block Entropy Measures

Global block entropy improves on prefix entropy because it is able to capture differences between traces, even if they share the same prefix. It gives measures which partially reflect our assumptions as to which logs are more or less structured: BPI Challenge 2012, Hospital Event Log and Sepsis Cases have markedly higher global block entropy than more structured logs such as BPI Challenge 2013. However, it clearly gives lower values to smaller logs with L_1 through L_4 all having drastically lower values than the larger logs. This is unsurprising, since logs with longer traces will simply have many more possible k-blocks, flattening the probability distribution over outcomes and increasing the entropy.

6.1.2 Nearest Neighbour Measures

Nearest neighbour based measures give similar results, relatively speaking, to global block entropy on many real logs, assigning higher values to BPI Challenge 2012 and 2017, and Hospital Event Log. However, they characterise Sepsis Cases as having a moderately low entropy.

Despite the nearest neighbour measures being applied to flattened logs, they are in general close to the original trace entropy. For those logs whose nearest neighbour entropy values deviate from trace entropy, the values are significantly lower, suggesting that edit distance really does account for the trace similarity ignored by the original trace entropy measure: the nearest neighbour measures group together very similar traces which are considered distinct by the original trace entropy.

We observe that $\mathsf{entropy}^{kNN}$ tends to decrease with k for real logs, and note that our results confirm Singh, et. al's empirical results showing that $\mathsf{entropy}^{kNN}$ closely matches $\mathsf{entropy}^{KL}$ when $k = 1$.

The nearest neighbor approach is largely unaffected by noise in the parallel artificial log. This is not surprising since the added events will only increase the normalised edit distance to nearest neighbour slightly, and may have no effect on traces with similar noise.

We emphasise that this measure depends crucially on the formulation of *distance* between traces, leaving ample room for improvement by, for example, developing more sophisticated edit operation cost-weightings using domain knowledge or choosing to lessen the penalty to traces of very different lengths.

This approach suffers from one very clear shortcoming: complexity. In the worst case, the distance between every pair of traces must be computed, though unnecessary computations can be avoiding by using a dynamic programming approach, checking lower bounds on edit distance, as well as common prefixes and suffixes. Despite employing these improvements along with a fast iterative implementation for computing lev, we found this measure to be prohibitively time consuming for some of the large artificial logs (concurrent).

6.2 Entropy Rate Measures (h)

Entropy *rate* measures are more resilient to variations in the size of logs and number of activities, as demonstrated by the fact that L_1 through L_4 are assigned entropy rate estimates within the range of the (much larger) real life logs. However, they are in general much more affected by noise: in some cases (see Figure 9), ranking logs by entropy is not always stable under the addition of noise. It is encouraging that Lempel-Ziv and block-based entropy rate estimators using certain cutoff constraints return relatively consistent estimates despite being based on very different approaches. This suggests that they are in fact converging towards something near to the "true" entropy rate.

6.2.1 k-block estimators

We observed that the constraints for "good statistics" for block-based entropy rate estimators were in line with the properties discussed in 4.1.1. Namely, the constraints (22.2)

and (22.3), which are asymptotic upper bounds, allow k to grow much too large. The stricter constraints give much more reasonable entropy rate estimates, but also restrict k to extremely low values.

One important question we leave for future work concerns the appropriateness of the statistical assumptions underlying k-block estimators when applied to *sets* of sequences rather than single sequences. This means we get more samples of k-blocks than would be the case for a single sequence and the exact number of samples depends on the distribution of trace lengths in the log: a log may consist of many very short traces and one very long trace, in which case longer blocks would still be under-sampled. In our implementation we chose to define the sequence length K as the longest trace, but a more principled approach should be considered.

Furthermore, some constraints are a function of the true entropy rate h itself, for which our implementation takes the current running estimate. Another approach may be to use another estimator for this, such as Lempel-Ziv.

In general, k-block entropy rate estimators appear to be rather unstable and very sensitive to the particular combination of constraint on k and log characteristics which is clearly illustrated by the erratic fluctuations in Figure 9. The ratio based formulation rate_k^r converges more slowly (see Figure 7), which is in line with previous research [25, 36]. In this sense, it is more robust than rate_k^d. We note that the unusually high value assigned to some logs, such as NASA CEV, is due to the cutoff constraints restricting the estimator to the 1-block entropy.

Finally, we note that these estimators are surprisingly poor at detecting noise in artificial logs. Only rate_k^d using constraint (22.4) is able to consistently distinguish the degree and type of noise across the "Testing Representational Bias" set of logs. On the concurrent logs these estimators also give somewhat inconsistent results: certain constraints are so strict that the estimate plateaus immediately, with the relationship of concurrent block size to tail size apparently inverted, while other constraints lead to more reasonable estimates, which grow logarithmically with the size of concurrent block.

6.2.2 Lempel-Ziv

The Lempel-Ziv estimator has the advantage over block-based estimators that it is nonparametric in constrast to rate_k^d and rate_k^r which require choosing a cutoff constraint. It also appears to be much more robust, i.e. less erratic.

Wwe observe that for some logs, however, that it returns values contradicting our assumptions regarding modelling paradigms. In particular, BPI Challenge 2012 has a very low rate^{LZ} value, lower than BPI Challenge 2013: the opposite of what we would expect if declarative processes have higher entropy.

An important detail concerns the order in which the logs are parsed. Since the Lempel Ziv algorithm parses sequences based on the order in which symbols are observed, parsing the traces in a different order can result in a different parsing and a slightly different value for rate^{LZ}. In our implementation we parsed logs in their original ordering. A sampling based approach for assessing the variability of the estimates could be an avenue for future work.

The Lempel-Ziv estimator clearly outshines on the concurrent artificial log. It is surprisingly effective at capturing the fact that even large blocks of concurrency should have a low entropy rate when flanked by long sequential tails, while a concurrent blocks with short, or no, sequential tails show a slowly growing entropy rate.

Lempel-Ziv performs less impressively when presented with noise and appears unable to distinguish types and degree of noise, with a number of the estimates "crossing over" for different logs in the presence of noise. For example, it assigns the *Duplicates* log a higher or lower entropy than the *Parallel* log depending on the particular type of noise present: a distinction other estimators are able to make.

On the logs generated from Declare models, Lempel-Ziv performs well, showing the expected drop on very restrictive declarative model and again proving more stable than other entropy rate estimators. Finally, this estimator proved to be one of the fastest to compute.

7 Conclusion

We studied entropy as a measure of the variability of a process log, with the intended application of classifying logs as more suitable for declarative or imperative miners. Specifically, we contributed (1) a survey of potential measures of entropy; (2) an implementation of these measures; (3) an experimental investigation of the proposed measures on both synthetic and real-life logs; and (4) based on this investigation and the community understanding of which logs are likely declarative, a qualitative evaluation of the suitability of the measures.

A more rigorous, quantitative evaluation of the proposed measures requires a clear partitioning of logs into "imperative" or "declarative" classes. More precisely, we need clear ways of evaluating whether mining a log imperatively or declaratively produces "better" models: an open research question in itself. With a clearly defined error measure in hand, one or more entropy estimators could, for example, serve as input features to a classification algorithm to determine which mining approach to apply to a log, or in the case of hybrid mining, applied to partitions of the log in order to determine the mix of mining approaches.

Acknowledgements. We would like to thank Jakob Grue Simonsen for valuable discussions.

References

1. Real life event logs. https://data.4tu.nl/repository/collection:event_logs_real. 4TU Centre for Research Data. Accessed: 2018-23-01
2. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems and Computers **08**, 21–66 (1998). DOI 10.1142/S0218126698000043
3. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011). DOI 10.1007/978-3-642-19345-3
4. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery **2**(2), 182–192 (2012). DOI 10.1002/widm.1045. URL http://dx.doi.org/10.1002/widm.1045
5. Back, C.O.: Eventropy - entropy estimation tool and CLI for XES event logs and other sequential data. https://github.com/backco/eventropy
6. Back, C.O., Debois, S., Slaats, T.: Towards an empirical evaluation of imperative and declarative process mining. In: Accepted for the First International Workshop on Empirical Methods in Conceptual Modeling (EmpER'18) (2018)
7. Back, C.O., Debois, S., Slaats, T.: Towards an Entropy-Based Analysis of Log Variability. In: E. Teniente, M. Weidlich (eds.) Business Process Management Workshops, *Lecture Notes in Business Information Processing*, vol. 308, pp. 53–70. Springer International Publishing, Cham (2018)
8. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
9. Bose, R.J.C., van der Aalst, W.M.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 401–412. SIAM (2009)
10. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Quarterly **40**(4) (2016)
11. Buijs, J., Dongen, B., Aalst, W.: On the role of fitness, precision, generalization and simplicity in process discovery. In: On the Move to Meaningful Internet Systems: OTM 2012, vol. 7565, pp. 305–322. Springer Berlin Heidelberg (2012). DOI http://dx.doi.org/10.1007/978-3-642-33606-5_19. URL http://wwwis.win.tue.nl/~wvdaalst/publications/p688.pdf
12. Cover, T., King, R.: A convergent gambling estimate of the entropy of english. Information Theory, IEEE Transactions on **24**(4), 413–421 (1978)
13. De Medeiros, A., Guzzo, A., Greco, G., Van Der Aalst, W., Weijters, A., Van Dongen, B., SaccÃ , D.: Process mining based on clustering: A quest for precision. pp. 17–29 (2008)
14. Debois, S., Hildebrandt, T., Slaats, T.: Safety, liveness and runtime refinement for modular process-aware information systems with dynamic sub processes. In: International Symposium on Formal Methods, pp. 143–160. Springer International Publishing (2015)
15. Debois, S., Slaats, T.: The analysis of a real life declarative process. In: CIDM 2015, pp. 1374–1382 (2015)
16. Delattre, S., Fournier, N.: On the kozachenko–leonenko entropy estimator.(report). Journal of Statistical Planning and Inference **185** (2017)
17. Delias, P., Doumpos, M., Grigoroudis, E., Matsatsinis, N.: A non-compensatory approach for trace clustering. International Transactions in Operational Research (2017)
18. Di Ciccio, C., Bernardi, M.L., Cimitile, M., Maggi, F.M.: Generating event logs through the simulation of declare models. In: Workshop on Enterprise and Organizational Modeling and Simulation, pp. 20–36. Springer (2015)
19. Di Ciccio, C., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Trans. Manage. Inf. Syst. **5**(4), 24:1–24:37 (2015). DOI 10.1145/2629447
20. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1010–1027 (2006). DOI 10.1109/TKDE.2006.123
21. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1010–1027 (2006)
22. Ha, Q.T., Bui, H.N., Nguyen, T.T.: A trace clustering solution based on using the distance graph model. In: International Conference on Computational Collective Intelligence, pp. 313–322. Springer (2016)
23. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pp. 289–296. Morgan Kaufmann Publishers Inc. (1999)
24. Hull, R., Damaggio, E., Masellis, R.D., Fournier, F., Gupta, M., Heath, F., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Noi Sukaviriya, P., Vaculín, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: DEBS 2011, pp. 51–62 (2011)
25. Lesne, A., Blanc, J.L., Pezard, L.: Entropy estimation of very short symbolic sequences. Physical Review E **79**(4), 046208 (2009)
26. Li, M.: An introduction to Kolmogorov complexity and its applications, 3. ed. edn. Texts in computer science. Springer, New York (2008)
27. MacKay, D.J.C.: Information theory, inference and learning algorithms, 6. print. edn. Cambridge University Press, Cambridge (2003)
28. Maggi, F., Slaats, T., Reijers, H.: The automated discovery of hybrid processes. In: BPM, pp. 392–399 (2014)
29. Maggi, F.M., Slaats, T., Reijers, H.A.: The automated discovery of hybrid processes. In: Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings, pp. 392–399 (2014)
30. Makanju, A.A., Zincir-Heywood, A.N., Milios, E.E.: Clustering event logs using iterative partitioning. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 1255–1264. ACM, New York, NY, USA (2009). DOI 10.1145/1557019.1557154. URL http://doi.acm.org/10.1145/1557019.1557154
31. Mannhardt, F., Blinde, D.: Analyzing the trajectories of patients with sepsis using process mining. RADAR+ EMISA **1859**, 72–80 (2017)
32. Object Management Group: Business Process Modeling Notation Version 2.0. Tech. rep., Object Management Group Final Adopted Specification (2011)
33. Pesic, M., Schonenberg, H., van der Aalst, W.: Declare: Full support for loosely-structured processes. In: EDOC 2007, pp. 287–300 (2007)
34. Reijers, H., Slaats, T., Stahl, C.: Declarative modeling-an academic dream or the future for bpm? In: BPM 2013, pp. 307–322 (2013)
35. Schunselaar, D.M.M., Slaats, T., Maggi, F.M., Reijers, H.A., van der Aalst, W.M.P.: Mining hybrid business process models: A quest for better precision. In: W. Abramowicz, A. Paschke (eds.) Business Information Systems, pp. 190–205. Springer International Publishing, Cham (2018)
36. Schürmann, T., Grassberger, P.: Entropy estimation of symbol sequences. Chaos: An Interdisciplinary Journal of Nonlinear Science **6**(3), 414–427 (1996)
37. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (1948). DOI 10.1002/j.1538-7305.1948.tb01338.x
38. Singh, H., Misra, N., Hnizdo, V., Fedorowicz, A., Demchuk, E.: Nearest neighbor estimates of entropy. American Journal of Mathematical and Management Sciences **23**(3-4), 301–321 (2003)

39. Singh, S., Poczos, B.: Analysis of k-nearest neighbor distances with application to entropy estimation (2016)
40. Slaats, T., Schunselaar, D.M.M., Maggi, F.M., Reijers, H.A.: The semantics of hybrid process models. In: CoopIS, pp. 531–551 (2016)
41. Smedt, J.D., Weerdt, J.D., Vanthienen, J.: Fusion miner: Process discovery for mixed-paradigm models. Decision Support Systems **77**, 123–136 (2015)
42. Song, M., Günther, C.W., Aalst, W.M.: Trace clustering in process mining. In: Business Process Management Workshops, pp. 109–120. Springer (2009)
43. Thomas, J.A., Cover, T.M.: Elements of information theory. John Wiley and Sons (2006)
44. Van Der Aalst, W.W.: Testing representational biases (2017). DOI 10.4121/uuid:25d6eef5-c427-42b5-ab38-5e512cca08a9
45. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. Information Theory, IEEE Transactions on **23**(3), 337–343 (1977)
46. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. Information Theory, IEEE Transactions on **24**(5), 530–536 (1978)