Author/s:
Kalenkova, A;Polyvyanyy, A

Title:
A Spectrum of Entropy-Based Precision and Recall Measurements Between Partially Matching Designed and Observed Processes

Date:
2020

Citation:
Kalenkova, A. & Polyvyanyy, A. (2020). A Spectrum of Entropy-Based Precision and Recall Measurements Between Partially Matching Designed and Observed Processes. Kafeza, E (Ed.) Benatallah, B (Ed.) Martinelli, F (Ed.) Hacid, H (Ed.) Bouguettaya, A (Ed.) Motahari, H (Ed.) ICSOC 2020 proceedings, 12571, pp.337-354. Springer. https://doi.org/10.1007/978-3-030-65310-1_24.

A Spectrum of Entropy-Based Precision and Recall Measurements Between Partially Matching Designed and Observed Processes

Anna Kalenkova and Artem Polyvyanyy

School of Computing and Information Systems
The University of Melbourne, Parkville, VIC, 3010, Australia
anna.kalenkova@unimelb.edu.au; artem.polyvyanyy@unimelb.edu.au

Abstract. Modern software systems are often built using service-oriented principles. Atomic components, be that web- or microservices, allow constructing flexible and loosely coupled systems. In such systems, services are building blocks orchestrated by business processes the system supports. Due to the complexity and heterogeneity of industrial software systems, implemented processes may deviate from those initially designed. In this paper, we propose a *spectrum* of conformance measurements. The spectrum results from a generalization of the recently introduced entropy-based approaches for measuring *precision* and *recall* between *observed* process executions and *designed* process models. The new generalized measures of precision and recall inherit the desired for this class of measures properties and provide analysts with flexible control over the sensitivity for identifying commonalities and discrepancies in the compared processes and performance of the techniques. The reported evaluation based on our implementation of the measures over real-world event logs and automatically discovered models confirms the feasibility of using the approach in industrial settings.

1 Introduction

In a service-oriented architecture (SOA), business processes can be implemented as compositions of loosely coupled services that interact to achieve concrete business goals [10]. The historical data on executions of such processes are often recorded in event logs. These logs can be subsequently analyzed to discover, check, and improve service compositions [1] using process mining techniques. Process mining combines studies of inferences from data in data mining and machine learning with process modeling and analysis to tackle the problems of discovering, monitoring, and improving real-world processes [2]. One of the core problems in process mining is *conformance checking* [5], which studies relationships between processes recorded in an event log and described by a process model to characterize and/or measure commonalities and discrepancies between the *observed* real-world and *designed* processes. Two core measures in conformance checking are *precision* and *recall*. A precise process model should not allow for behavior unrelated to what was seen in the event log, while a model with good recall should allow for the behavior seen in the event log.

In our previous work, we devised two approaches for measuring recall and precision between process models and event logs [21,19] founded on the notion of topological entropy [6] of the behaviors, i.e., collections of traces, that they describe. The measures

presented in [21] have been recently recognized in [23] as the *only* recall and precision measures, among the evaluated state-of-the-art measures, that satisfy all the desired properties. For example, they are deterministic, depend only on the underlying behaviors and not on their representations, and are monotone, i.e., the more common behavior the model and log have, the greater the recall and precision values are. These measures can be computed for behaviors that describe arbitrary, including infinite, collections of traces. However, they rely on the exact matching of traces, i.e., two different traces are always treated as totally dissimilar, even if they differ only in a single task. In [19], we extended the measures to account for partially matching traces. Instead of measuring the original collections of traces, the new measures quantify and compare the "diluted" behaviors, where the diluted version of a behavior consists of all the traces obtained from original traces by skipping an arbitrary number of tasks; note that once a task is skipped the order of the remaining tasks in the resulting trace does not change. These new measures inherit the properties of the original measures and enjoy some further properties specific to the partial matching of traces. For example, the more common subtraces the model and log describe, the greater the recall and precision values are.

The two approaches for measuring recall and precision presented in [21,19] address two *extremes*, i.e., no support for partial matching of traces and ability to detect and quantify any partial similarity between traces. Such extreme approaches are doomed for limitations. The former approach may overlook the commonalities in traces, while the latter may miss the discrepancies. This calls for a *compromise* approach.

In this paper, we present a novel technique for measuring precision and recall between two (not necessarily finite) collections of partially matching traces that can be configured as to when two different traces should be considered similar. The configuration consists of two non-negative integers that specify the maximum numbers of tasks that can be skipped in traces in each of the two compared collections to arrive at the same trace and, consequently, accept the compared traces as similar. For example, traces $\langle a, b, c\rangle$ and $\langle b, d, c, e\rangle$ are *dissimilar* if one is allowed to skip only one task in each trace. However, they can be accepted as *similar* if one is allowed to skip two tasks in the latter trace. Indeed, one arrives at the trace $\langle b,c\rangle$ by skipping task a in the former trace and tasks d and e in the latter trace. Hence, two traces are said to be similar if they have a common subtrace that can be constructed by skipping up to the configured number of tasks in each trace. Such common subtrace captures the common behavior of the compared traces. The technique then proceeds by measuring the amount of all common subtraces in both collections as per the supplied configuration.

The new technique results in a *discrete spectrum* of recall and precision measurements induced by all the configurations in the Cartesian square of the natural numbers ($\mathbb{N}_0 \times \mathbb{N}_0$). In this spectrum, the (k, m) configuration suggests that up to k and m skips are allowed in the traces of log and model, respectively. The recall and precision measures for the $(0, 0)$ configuration correspond to the exact matching measures from [21]. Using a formal proof, we show that measures for the (k, m) configurations approach the partial matching measures from [19] when $k, m \to \infty$, allowing for a gradual adjustment of the analysis between the two extremes.

Process analysts can rely on our new technique to (in a flexible way) adjust the "sensitivity" of the measured recall and precision values to mismatches in the compared

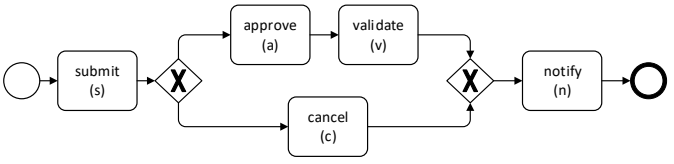


Fig. 1: A BPMN model of a loan application process.

behaviors. Such an adjustment may, for instance, be guided by domain knowledge. An analyst may know that logs are recorded with noise, e.g., a sequence of initialization tasks at the start of each trace, and, thus, adjust the allowed skips in log traces accordingly. In addition, the new measures demonstrate good runtime characteristics for the practically relevant range of configurations. The most computationally demanding step of the partially matching approach from [19] is the construction of a deterministic version of the automaton that encodes the completely diluted version of the original behavior. This step has exponential worst-case time complexity and, unfortunately, close-to-worst cases manifest often for industrial datasets [19]. However, as confirmed through our evaluations, for configurations (x, y), where $x, y \leq 10$, the worst-case complexity manifests only for sub-problems of small sizes, or not at all.

Section 2 motivates the problem addressed in this paper and demonstrates our approach for solving the problem by means of an intuitive example. Next, Section 3 introduces the basic notions used in the discussions of the subsequent sections. Then, Section 4 presents our approach and discusses several its properties. Section 5 presents the results of our evaluations of the new measures. Section 6 positions our work among the state-of-the-art results in conformance checking. Section 7 concludes the paper.

2 Motivating Example

Consider the example loan application process in Fig. 1. First, the client submits an application. Then, the submitted application is reviewed by a bank analyst and either approved or canceled. If the application is approved, it is then validated. In both cases, the client is eventually notified. The BPMN model in Fig. 1 describes the expected process behavior. This behavior can be described by the set of traces $M = \{\langle s, a, v, n\rangle, \langle s, c, n\rangle\}$, which contains the two traces that the model supports.[1]

Suppose that the corresponding event log L contains only one trace $t = \langle s,\ a,\ n\rangle$. Then, the model and log have no common traces, i.e., $M \cap L = \emptyset$. Consequently, precision and recall measures founded on the exact matching of traces, e.g., the entropy-based approach presented in [21], are equal to zero, suggesting no similarity between the behaviors of the model and log. Despite different, the traces in M and L show some similarity. For example, it holds that t and traces in M contain subtrace $\langle s,n\rangle$.

The partial matching approach from [19] addresses the above issue by comparing collections of diluted traces. For example, the diluted versions of L and M are $\{\langle s, a, n\rangle, \langle s, a\rangle, \langle s, n\rangle, \langle a, n\rangle, \langle s\rangle, \langle a\rangle, \langle n\rangle, \langle\rangle\}$ and $\{\langle s, a, v, n\rangle, \langle a, v, n\rangle, \langle s, v, n\rangle, \langle s, a, n\rangle, \langle s, a, v\rangle, \langle s, c, n\rangle, \langle v, n\rangle, \langle a, n\rangle, \langle a, v\rangle, \langle s, n\rangle, \langle s, v\rangle, \langle s, a\rangle, \langle s, c\rangle, \langle c, n\rangle, \langle s\rangle, \langle a\rangle, \langle v\rangle, \langle n\rangle, \langle c\rangle, \langle\rangle\}$, where $\langle\rangle$ is the empty trace. We denote the former set and the latter set as

[1] We use short task names to specify traces, while the corresponding full names are in Fig. 1.

	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
$M^{(0)}$	0.000	0.000	0.000	0.000
$M^{(1)}$	1.000	0.793	0.568	0.464
$M^{(2)}$	1.000	1.000	0.908	0.741
$M^{(3)}$	1.000	1.000	1.000	1.000

(a) Recall

	$L^{(0)}$	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$
$M^{(0)}$	0.000	0.000	0.000	0.000
$M^{(1)}$	0.549	0.670	0.670	0.670
$M^{(2)}$	0.382	0.589	0.745	0.745
$M^{(3)}$	0.299	0.459	0.642	0.785

(b) Precision

Table 1: The spectrum of the entropy-based precision and recall measurements between the model in Fig. 1 that describes set of traces $M = \{\langle s,a,v,n\rangle, \langle s,c,n\rangle\}$ and event log $L = \{\langle s,a,n\rangle\}$ calculated for different numbers of skipped tasks in traces of M and L.

L^{∞} and M^{∞}, respectively. The diluted traces can be used to identify commonalities and discrepancies between the original traces. For example, it holds that $\langle s,n\rangle \in L^{\infty} \cap M^{\infty}$ and $\langle s,c,n\rangle \in M^{\infty} \setminus L^{\infty}$. The partial matching precision and recall measures between a model and log quantify the commonalities and discrepancies in their diluted traces. Since $L^{\infty} \subset M^{\infty}$, such diluted recall is equal to 1.0, suggesting that the model allows for the behavior seen in the log perfectly. As $M^{\infty} \not\subseteq L^{\infty}$, such diluted precision is not perfect. The particular precision value obtained using the technique from [19] is 0.785; note that the absolute value is of less interest here as these are the relations between the measurements that allow comparing different behaviors.[2]

The above examples highlight the limitations of the two extreme approaches mentioned in the Introduction. The approach founded on the exact matching of traces overlooks the existing partial commonalities in traces, while the approach that relies on the arbitrary skips of tasks in traces misses to identify the discrepancies, cf. recall of 1.0.

Consider two sets $L^{(1)}$ and $M^{(1)}$ with all the traces constructed from the traces in L and M, respectively, by skipping at most one task in a trace from the original set, i.e., $L^{(1)} = \{\langle s,\ a,\ n\rangle,\ \langle a,\ n\rangle,\ \langle s,\ n\rangle,\ \langle s,\ a\rangle\}$ and $M^{(1)} = \{\langle s,\ a,\ v,\ n\rangle,\ \langle a,\ v,\ n\rangle,\ \langle s,\ v,\ n\rangle,\ \langle s,\ a,\ n\rangle,\ \langle s,\ a,\ v\rangle,\ \langle s,\ c,\ n\rangle,\ \langle c,\ n\rangle,\ \langle s,\ n\rangle,\ \langle s,\ c\rangle\}$. It holds that $L^{(1)} \cap M^{(1)} = \{\langle s,a,n\rangle, \langle s,n\rangle\}$, $L^{(1)} \setminus M^{(1)} \neq \emptyset$, and $M^{(1)} \setminus L^{(1)} \neq \emptyset$. Hence, sets $L^{(1)}$ and $M^{(1)}$ contain information about commonalities and discrepancies of log and model. The entropy-based recall and precision computed based on the traces in $L^{(1)}$ and $M^{(1)}$ are equal to 0.793 and 0.670, respectively, and, thus, confirm that the behaviors are neither completely different, nor are in the subsumption relation. Again, the absolute values of precision and recall are irrelevant, as they indeed satisfy the monotonicity properties discussed later. Importantly, these measure neither suggest perfect match nor the complete dissimilarity of the compared behaviors.

Tables 1a and 1b demonstrate the spectrum of precision and recall values calculated for the traces in L and M. Given a set of traces X, by $X^{(k)}$, $k \in \mathbb{N}_0$, we denote the set of all traces obtained from the traces in X by skipping up to k arbitrary tasks in the original traces. Thus, it holds that $M^{(0)} = M$ and $L^{(0)} = L$. For this example, it also holds that $M^{(m)} = M^{\infty}$ and $L^{(m)} = M^{\infty}$, where $m \geq 3$.

Note that the recall values do not decrease when more skips are allowed in the model traces, i.e., $recall^{(k,m)}(M,L) \leq recall^{(k+1,m)}(M,L)$, $k,m \in \mathbb{N}_0$; here, k and

[2] In general, precision and recall measure of one suggest perfect conformance, while the values of zero suggest no behavioral similarities between the compared model and log.

m refer to the numbers of allowed skips in M and L, respectively, used to compute the conformance values (refer to Section 4 for details). Indeed, by extending the behavior of the model more, we can use it to cover more of the traces in the log. On the other hand, precision values do not decrease when more skips are allowed in the log traces, i.e., $prec^{(k,m)}(M,L) \leq prec^{(k,m+1)}(M,L)$, $k, m \in \mathbb{N}_0$, as by extending the behavior of the log we can use it to cover more behavior of the model. These properties of precision and recall measures are formally proved in Section 4.3.

3 Basic Notions

This section introduces basic notions and definitions used in the remainder of the paper.

3.1 Sequences, Languages, and Event Logs

Let X be a set of elements. The *power set* of X, denoted as $\mathcal{P}(X)$, is the set of all subsets of X. By $\langle x_1, x_2, \ldots, x_k \rangle$, where $x_1, x_2, \ldots, x_k \in X$, $k \in \mathbb{N}$, we denote a *sequence* of elements from X of length k. The *empty sequence* of zero length is represented by $\langle\rangle$. By X^*, we denote the set of all finite sequences over X. A *concatenation* of two sequences $\langle x_1, x_2, \ldots, x_k \rangle$ and $\langle y_1, y_2, \ldots, y_l \rangle$ is denoted by $\langle x_1, x_2, \ldots, x_k \rangle \cdot \langle y_1, y_2, \ldots, y_l \rangle$ and is the sequence $\langle x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_l \rangle$.

Given a sequence x and a set K, by $x|_K$, we denote a sequence obtained from x by removing all elements of x that are not members of K without changing the order of the remaining elements, e.g., it holds that $\langle a, c, b, a, d, c \rangle|_{\{c,d\}} = \langle c, d, c \rangle$.

An *alphabet* is any nonempty finite set. The elements of an alphabet are its *labels*. A *language* L over an alphabet Σ is a (not necessarily finite) set of sequences, or *words*, of labels from Σ, i.e., $L \subseteq \Sigma^*$. By $C_n(L)$, we denote the set of all words in L of length n. By Ξ, we denote a universe of all possible *observable* labels, while τ, $\tau \notin \Xi$, denotes a special *silent* label. Let L_1 and L_2 be two languages. Then, their concatenation is the language $L = \{l_1 \cdot l_2 \mid l_1 \in L_1, l_2 \in L_2\}$, denoted by $L_1 \circ L_2$. Given a language L, L^* is the language defined by $\bigcup_{n=0}^{\infty} L^n$, where $L^0 = \{\langle\rangle\}$, $L^n = L^{n-1} \circ L$.

Let E be a finite nonempty set of *tasks*, or *events*. A finite language $L \subset E^*$ is an *event log* and its words are called *traces* [2].

3.2 Finite Automata

A *nondeterministic finite automaton* (NFA) is a 5-tuple $(Q, \Lambda, \delta, q_0, A)$, where Q is a finite nonempty set of *states*, $\Lambda \subseteq \Xi$ is a set of *labels*, $\delta : Q \times (\Lambda \cup \{\tau\}) \to \mathcal{P}(Q)$ is the *transition function*, $q_0 \in Q$ is the *start state*, and $A \subseteq Q$ is the *set of accept states*.

An NFA induces a collection of computations. A *computation* of an NFA $B = (Q, \Lambda, \delta, q_0, A)$ is either the empty word or a word $\sigma = \langle a_1, \ldots, a_n \rangle$, $n \in \mathbb{N}$, where $a_i \in \Lambda \cup \{\tau\}$, $i \in [1 \,..\, n]$, and exists a sequence of states $\langle q_0, q_1, \ldots, q_n \rangle$, such that for every $k \in [1 \,..\, n]$ it holds that $q_k \in \delta(q_{k-1}, a_k)$. We say that σ *leads from* q_0 *to* q_n. By convention, the empty word leads to the start state. NFA B *accepts* a word $w \in \Lambda^*$ iff exists a computation $\sigma \in (\Lambda \cup \{\tau\})^*$ that leads to one of its accept states and it holds that $w = \sigma|_\Lambda$. The *language* of B is denoted by $lang(B)$ and is the set of all words B accepts. We also say that B *recognizes* $lang(B)$.

A *deterministic finite automaton* (DFA) is an NFA $(Q, \Lambda, \delta, q_0, A)$, where for every $q \in Q$ it holds that $\delta(q, \tau) = \emptyset$, and for every $q \in Q$ and every $a \in \Lambda$, $|\delta(q, a)| \leq 1$. For a language L recognized by an NFA, exists a DFA that recognizes L [12].

(a) A DFA that recognizes M.

(b) A DFA that recognizes L.

Fig. 2: Two DFAs that recognize languages M and L.

Figs. 2a and 2b present DFAs that recognize, respectively, languages M and L discussed in Section 2. States are shown as circles, start states are marked with incoming arrows, transitions are encoded as arcs, and accept states are shown with double border.

A DFA $(Q, \Lambda, \delta, q_0, A)$ is *ergodic* if its underlying graph is strongly irreducible, i.e., for all $(q, p) \in Q \times Q$, $q \neq p$, there is a sequence of states $\langle q_1, \ldots, q_n \rangle \in Q^*$, $n \in \mathbb{N}$, such that $q_1 = q$, $q_n = p$, and for every $k \in [1\,..\,n-1]$ there exists $\lambda \in \Lambda$ such that $q_{k+1} \in \delta(q_k, \lambda)$. A language L is *regular* iff there exists a DFA that recognizes L. A regular language L is *irreducible* iff it is a language of an ergodic DFA [6].

3.3 Topological Entropy

Let Σ be an alphabet and let $L \subseteq \Sigma^*$ be an irreducible language over Σ. The *topological entropy* of L, which estimates the cardinality of L by measuring the ratio of the number of distinct words in L to the length of these words, is given below [6]:

$$ent(L) = \limsup_{n \to \infty} \frac{\log |C_n(L)|}{n}. \tag{1}$$

The languages recognized by automata and event logs are *regular*; note that an event log can be encoded as a DFA, cf. Fig. 2b. But, not all such languages are *irreducible*. Given a regular language L, in [21], the authors proposed to compute the *short-circuit* topological entropy of L, denoted by $ent\bullet(L)$, as the topological entropy of the irreducible language $(L \circ \{\langle\chi\rangle\})^* \circ L$, $\chi \notin \Sigma$, i.e., $ent\bullet(L) = ent((L \circ \{\langle\chi\rangle\})^* \circ L)$. Note that one can always construct a DFA that recognizes $(L \circ \{\langle\chi\rangle\})^* \circ L$ from a DFA B that recognizes L by adding fresh transitions in B that are labeled with χ and connect the accept states of B with its start state. For example, the short-circuit topological entropy of the language recognized by the automaton in Fig. 2a is equal to the topological entropy of the language recognized by the automaton in Fig. 3.[3]

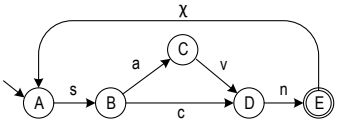


Fig. 3: A DFA that recognizes language $(M \circ \{\langle\chi\rangle\})^* \circ M$.

Finally, this result follows immediately from the definition of the short-circuit topological entropy and Lemma 4.7 in [21]:

Corollary 3.1 (Topological entropy). *Let L_1 and L_2 be two regular languages.*

1. *If $L_1 = L_2$, then $ent\bullet(L_1) = ent\bullet(L_2)$;*
2. *If $L_1 \subset L_2$, then $ent\bullet(L_1) < ent\bullet(L_2)$.*

[3] The topological entropy of an ergodic DFA is given by the logarithm of the Perron-Frobenius eigenvalue of its adjacency matrix [6].

4 Comparing Designed and Observed Processes

This section describes existing entropy-based conformance checking techniques and proposes a new approach that can control the number of skipped tasks.

4.1 Existing Entropy-Based Conformance Checking Techniques

Conformance checking techniques [5] measure the discrepancies and commonalities between the behaviors described in process models and event logs. *Precision* estimates the share of the common model and log behavior with respect to the overall model behavior, while *recall* assesses the share of the log behavior captured by the model.

The *exact matching* conformance checking approach for measuring precision and recall proposed in [21] relates two regular languages of a process model (M) and event log (L) with their intersection; note that the intersection of M and L is also a regular language [12]. The measurements of the behaviors encoded in these languages are carried out using the short-circuit topological entropy. The entropy-based precision ($prec$) and recall ($recall$) values between the model and log are then defined as shown below [21]:

$$prec(M, L) = \frac{ent\bullet(M \cap L)}{ent\bullet(M)}, \qquad recall(M, L) = \frac{ent\bullet(M \cap L)}{ent\bullet(L)}.$$

As follows from Corollary 3.1, for any two regular languages M and L, it holds that $prec(M, L)$ and $recall(M, L)$ values belong to the interval $[0, 1]$. In contrast to other conformance checking techniques, this approach is *trace monotone* [23], i.e., the higher the share of the model traces that are presented in the log, the higher the precision value, and similarly, the higher the share of the log traces that are captured by the model, the higher the recall value. As shown in Section 2, this approach can be too restrictive and in case $M \cap L = \emptyset$, $prec(M, L) = recall(M, L) = 0$.

The *partial matching* precision and recall measures described in [19] compare regular languages that allow for arbitrary skips within the model and log behaviors. First, for regular languages M and L that encode the model behavior and the log behavior, respectively, M^∞ and L^∞ are constructed. NFAs that recognize languages M^∞ and L^∞ for the example behaviors discussed in Section 2 are shown in Fig. 4. Then, these NFAs are converted to equivalent DFAs [12] and, finally, the precision and recall values for these "diluted" languages are calculated:

$$prec^{(\infty,\infty)}(M, L) = \frac{ent\bullet(M^\infty \cap L^\infty)}{ent\bullet(M^\infty)}, \qquad recall^{(\infty,\infty)}(M, L) = \frac{ent\bullet(M^\infty \cap L^\infty)}{ent\bullet(L^\infty)}.$$

Such conformance checking technique assesses the share of the common log and model behavior, including all the shared subtraces. In contrast to the original exact matching approach, the partial matching technique is not restrictive. Moreover, it "dilutes" the initial languages by adding extra behavior that, in some cases, results in too many matches, which hampers analysis. Recall that for the languages discussed in Section 2, it holds that $recall^{(\infty,\infty)}(M, L) = 1.0$, as $L^\infty \subset M^\infty$, while $recall(M, L) = 0.0$, because M and L do not have common traces, i.e., $M \cap L = \emptyset$.

(a) An NFA that recognizes language M^{∞}. (b) An NFA that recognizes language L^{∞}.

Fig. 4: Two NFAs that recognize languages M^{∞} and L^{∞} discussed in Section 2.

4.2 *k*-Skips Conformance Checking

The primary task of the proposed technique is to assess the log and the model similarities assuming that some predefined numbers of steps can be skipped. Suppose the event log contains traces with additional steps not presented within the original model. In that case, this approach can still consider these traces, because a limited number of skips within the log is allowed. Similarly, the behavior of the model with a controlled number of skips can match some of the event log traces that skip the model's tasks.

Let M and L be two regular languages capturing the behavior of a process model and event log, respectively. Suppose that $M^{(l)}$ and $L^{(k)}$, where $l, k \in \mathbb{N}_0$, are languages obtained from M and L by allowing up to l and k skips in the original traces of M and L. Then, we define the precision and recall measures for these allowed skips as follows:

$$prec^{(l,k)}(M, L) = \frac{ent\bullet(M^{(l)} \cap L^{(k)})}{ent\bullet(M^{(l)})}, \qquad recall^{(l,k)}(M, L) = \frac{ent\bullet(M^{(l)} \cap L^{(k)})}{ent\bullet(L^{(k)})}.$$

Again, according to Corollary 3.1, these precision and recall values belong to the interval $[0, 1]$. While the entropy calculation techniques [6,21] and the set operations over regular languages [12] are well-defined, we still need to build $M^{(l)}$ and $L^{(k)}$ languages. Without loss of generality, we consider language $L^{(k)}$ and define it constructively using Algorithm 1 by building a DFA that recognizes $L^{(k)}$.

Firstly, this algorithm constructs an NFA B_{NFA} that recognizes language $L^{(k)}$. To that end, $k + 1$ copies of the DFA that recognizes L and referred to as *layers* are added to the NFA (Lines 1 and 5). The start state of B_{NFA} is the start state of the first layer (Layer 0). The transition function δ is considered as a relation (a set of pairs) in this algorithm. As suggested in Line 10, for each transition (q^{i-1}, a) of Layer $i - 1$, a transition (q^{i-1}, τ) leading to the only state in the set $\delta(q^i, a)$, where q^{i-1} and q^i are copies of the same state at Layers $i - 1$ and i, respectively, is added.

The NFA constructed by Algorithm 1 for the language $M^{(2)}$, where M is the language discussed in Section 2, is presented in Fig. 5a. This automaton contains three layers connected by additional transitions labeled by τ. The layers of B_{NFA} correspond to the number of skips made. For instance, visiting a state at Layer 1 means that one skip has been made in the computation and, hence, it is still possible to make one more skip by visiting a state at Layer 2.

Once the NFA has been constructed, it is converted to an equivalent DFA using the approach from [12] (Line 14). Fig. 5b shows the minimal DFA constructed from the NFA in Fig. 5a. The states of this DFA correspond to sets of the NFA states, i.e., $S_0 = \{A, B', C'', D''\}$, $S_1 = \{D, D', E', E''\}$, $S_2 = \{E, E', E''\}$, $S_3 = \{C', D''\}$, $S_4 = \{D''\}$, $S_5 = \{B, C', D', D'', E''\}$, and $S_6 = \{C, D', E''\}$. NFA states A', A'', and B'' (highlighted in gray in Fig. 5a) are *dead*, because they cannot be reached from the start state A and, thus, are not represented in the resulting DFA.

Algorithm 1: Construct a DFA that recognizes language $L^{(k)}$

Input: A DFA $B^0 = (Q^0, \Lambda, \delta^0, q_0^0, A^0)$ that recognizes language L and $k \in \mathbb{N}$.
Output: A DFA B_{DFA} that recognizes language $L^{(k)}$.

$Q \leftarrow Q^0$; $\delta \leftarrow \delta^0$; $q_0 \leftarrow q_0^0$; $A \leftarrow A^0$;
$B_{NFA} \leftarrow (Q, \Lambda, \delta, q_0, A)$; /* Initialize an NFA B_{NFA} */
for $i \leftarrow 1$ **to** k **do**
  $Q^i \leftarrow Q^0$; $\delta^i \leftarrow \delta^0$; $A^i \leftarrow A^0$; /* Clone states and transitions */
  $Q \leftarrow Q \cup Q^i$; $\delta \leftarrow \delta \cup \delta^i$; $A \leftarrow A \cup A^i$; /* Add next layer */
  /* Connect layers */
  foreach $q^{i-1} \in Q^{i-1}$, $a \in \Lambda$ **do**
    if $\delta(q^{i-1}, a) \neq \emptyset$ **then**
      /* q^{i-1}, q^i are copies of the same state from Q^0 */
      $\delta \leftarrow \delta \cup \{((q^{i-1}, \tau), \delta(q^i, a))\}$;
    end
  end
end
$B_{DFA} \leftarrow Determinize(B_{NFA})$;
return B_{DFA};

4.3 Formal Properties

According to Algorithm 1, for any regular language L, it holds that $L^{(0)} = L$, and for any $k \geq 1$, by construction, it holds that $L^{(k-1)} \subseteq L^{(k)}$. From the monotonicity of $ent\bullet$ measure, refer to Corollary 3.1, it holds that $ent\bullet(L^{(k-1)}) \leq ent\bullet(L^{(k)})$, for any $k \geq 1$. This leads to the following two propositions.

Proposition 4.1. *Let M and L be two regular languages.*
Then, it holds that $prec^{(l,k)}(M, L) \leq prec^{(l,k+1)}(M, L)$, where $l, k \in \mathbb{N}_0$.

Proof. By definition, it holds that $prec^{(l,k)}(M, L) = {ent\bullet(M^{(l)} \cap L^{(k)})}/{ent\bullet(M^{(l)})}$ and $prec^{(l,k+1)}(M, L) = {ent\bullet(M^{(l)} \cap L^{(k+1)})}/{ent\bullet(M^{(l)})}$. Since it holds that $(M^{(l)} \cap L^{(k)}) \subseteq (M^{(l)} \cap L^{(k+1)})$, then ${ent\bullet(M^{(l)} \cap L^{(k)})}/{ent\bullet(M^{(l)})} \leq {ent\bullet(M^{(l)} \cap L^{(k+1)})}/{ent\bullet(M^{(l)})}$. □

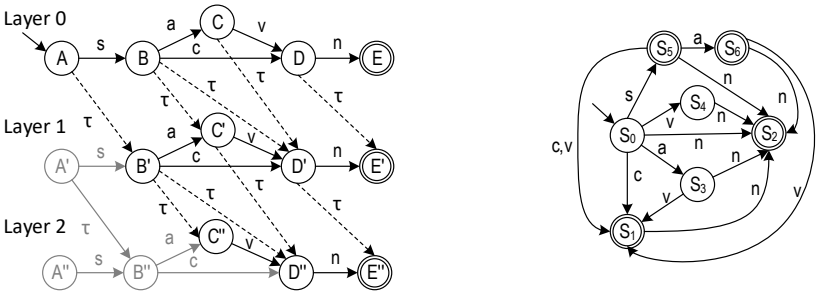
Proposition 4.2. *Let M and L be two regular languages.*
Then, it holds that $recall^{(l,k)}(M, L) \leq recall^{(l+1,k)}(M, L)$, where $l, k \in \mathbb{N}_0$

The proof of Proposition 4.2 is similar to that one of Proposition 4.1.

Propositions 4.1 and 4.2 confirm the monotonicity of the k-skips measures. The next result states that the k-skips measures tend to the partial measure extreme as k approaches the infinity.

Theorem 4.1. *Let L be a regular language over Σ.*
Then, it holds that $\lim_{k \to \infty} ent\bullet(L^{(k)}) = ent\bullet(L^{\infty})$.

Proof. By definition, limit superior of $\{{\log |C_n((L^{\infty} \circ \{\langle\chi\rangle\})^* \circ L^{\infty})|}/{n}\}$, $\chi \notin \Sigma$, when n tends to ∞, is equal to $ent\bullet(L^{\infty})$. Let $x_n = {\log |C_n((L^{\infty} \circ \{\langle\chi\rangle\})^* \circ L^{\infty})|}/{n}$. Suppose that $\{x_{n_l}\}_{l=1}^{\infty}$, where $n_1 < n_2 < \ldots$, is a subsequence, such that $x_{n_l} \to ent\bullet(L^{\infty})$, as



(a) An NFA that recognizes language $M^{(2)}$.

(b) A DFA that recognizes language $M^{(2)}$.

Fig. 5: Two automata that recognize languages $M^{(2)}$ and $L^{(2)}$.

$l \to \infty$. By the definition of limit, $\forall\, \varepsilon > 0\, \exists\, N(\varepsilon)\ \forall l \geq N$, where N is a number that depends on ε, it holds that $|x_{n_l} - ent\bullet(L^{\infty})| < \varepsilon$. Consider a large enough $K(l)$ such that $\forall\, k > K$, $|C_{n_l}((L^{(k)} \circ \{\langle\chi\rangle\})^* \circ L^{(k)})| = |C_{n_l}((L^{\infty} \circ \{\langle\chi\rangle\})^* \circ L^{\infty})|$. Let y_n^k be a sequence $\log |C_n((L^{(k)} \circ \{\langle\chi\rangle\})^* \circ L^{(k)})| / n$. Then, $\forall\, \varepsilon > 0\, \exists N(\varepsilon)\ \forall l \geq N$ holds that $\exists K(l)$, such that $\forall k > K : |y_{n_l}^k - ent\bullet(L^{\infty})| < \varepsilon$. Since $y_{n_l}^k \to ent\bullet(L^{(k)})$, as $l \to \infty$, it holds that $\forall \varepsilon > 0\, \exists K(\varepsilon)$, such that $\forall k > K : |ent\bullet(L^{(k)}) - ent\bullet(L^{\infty})| < \varepsilon$. Therefore, $\lim\limits_{k \to \infty} ent\bullet(L^{(k)}) = ent\bullet(L^{\infty})$. □

Similarly to Theorem 4.1, the following theorem can be formulated and proved.

Theorem 4.2. *Let M and L be two regular languages over Σ. Then, it holds that* $\lim\limits_{k \to \infty} ent\bullet(M^{(k)} \cap L) = ent\bullet(M^{\infty} \cap L)$ *and* $\lim\limits_{k \to \infty} ent\bullet(M^{(k)} \cap L^{(k)}) = ent\bullet(M^{\infty} \cap L^{\infty})$.

The next corollary follows immediately from Theorems 4.1 and 4.2.

Corollary 4.1. *Let M and L be two regular languages over Σ. Then, it holds that:*

- $\lim\limits_{k,l \to \infty} prec^{(l,k)}(M, L) = prec^{(\infty,\infty)}(M, L)$, $\lim\limits_{k,l \to \infty} recall^{(l,k)}(M, L) = recall^{(\infty,\infty)}(M, L)$;
- $\lim\limits_{l \to \infty} prec^{(l,k)}(M, L) = prec^{(\infty,k)}(M, L)$, $\lim\limits_{l \to \infty} recall^{(l,k)}(M, L) = recall^{(\infty,k)}(M, L)$; *and*
- $\lim\limits_{k \to \infty} prec^{(l,k)}(M, L) = prec^{(l,\infty)}(M, L)$, $\lim\limits_{k \to \infty} recall^{(l,k)}(M, L) = recall^{(l,\infty)}(M, L)$.

In practice, the results of Corollary 4.1 allow balancing smoothly between the two extremes of the exact and partial measures. Starting with the exact measurements when $k = 0$ and $l = 0$, we can gradually approach the partial measurements by increasing parameters k and l. As discussed in [12], the number of states can grow exponentially when an NFA is converted to a DFA that recognizes the same language. The following theorem defines a condition under which the number of states is polynomially bounded.

Theorem 4.3. *Let $B = (Q, \Lambda, \delta, q_0, A)$ be a DFA and let $k \in \mathbb{N}$ such that $lang(B) = L$ and for any symbol $a \in \Lambda$, any state $q \in Q$, and any two (possibly empty) sequences of transitions of length less than or equal to k, one leading from q to $q' \in Q$ and the other leading from q to $q'' \in Q$ and enabling a in q' and q'', i.e., $\delta(q', a) \neq \emptyset$ and $\delta(q'', a) \neq \emptyset$, it holds that $q' = q''$. Then, there exists a DFA $B^k = (Q^k, \Lambda, \delta^k, q_0^k, A^k)$ such that $lang(B^k) = L^{(k)}$ and $|Q^k| \leq (k + 1) \cdot |Q|$.*

Proof. Consider NFA B_{NFA} constructed from B at lines 1–13 of Algorithm 1. Let $closure(q)$ denote the set of all states that can be reached from a state q of B_{NFA} via τ-transitions, including q. By induction, we prove that each state of the resulting DFA B^k obtained by determinization [12] of B_{NFA} is a $closure$ of one state from B_{NFA}. Basis of induction: According to the determinization algorithm [12], $q_0^k = closure(q_0)$, and hence, we say q_0^k corresponds to q_0. Step of induction: Let $q^k \in Q^k$ be a $closure$ of a state $\hat{q_1}$, i.e., $q^k = \{\hat{q_1}, \hat{q_2}, \hat{q_3}, \ldots, \hat{q_m}\}$, where $\hat{q_1}, \hat{q_2}, \hat{q_3}, \ldots, \hat{q_m}$ are some states of B_{NFA}. By construction (see Algorithm 1), states $\hat{q_1}, \hat{q_2}, \hat{q_3}, \ldots, \hat{q_m}$ may belong to different layers of the NFA and for each state $\hat{q_i}$, $i \in [1 \,..\, m]$, there exists a (possibly empty) sequence of τ-transitions with a maximum length of k leading from $\hat{q_1}$ to $\hat{q_i}$. Again, by construction, states $\hat{q_1}, \hat{q_2}, \hat{q_3}, \ldots, \hat{q_m}$ correspond to some states $q_1, q_2, q_3, \ldots, q_l$, $l \leq m$, of B and each sequence of τ-transitions with a maximum length of k in the NFA corresponds to some sequence of transitions with a maximum length of k in B. Suppose that for some $\hat{q_j}$, $j \in [1 \,..\, m]$, exists $b \in \Lambda$ such that $\delta(\hat{q_j}, b) \neq \emptyset$; transition $\delta(\hat{q_j}, b)$ corresponds to a DFA transition and can contain not more than one NFA state. Let $\delta(\hat{q_j}, b) = \{\hat{q_*}\}$, where $\hat{q_*}$ is a state of the NFA. Then, for any $\hat{q_p}$, $p \in [1 \,..\, m]$, $p \neq j$, it holds that $\delta(\hat{q}, b) = \emptyset$; otherwise, there is more than one transition labeled by b within corresponding sequences of states belonging to $\{q_1, q_2, q_3, \ldots, q_l\}$ in DFA B, and since the length of these sequences is less or equal to k, we obtain a contradiction to the conditions of the theorem. Hence, according to the determinization algorithm in [12], $\delta^k(q^k, b) = closure(\hat{q_*})$, i.e., the next state of B^k is a $closure$ of a state from B_{NFA}. Thus, each state of B^k corresponds to a $closure$ of some state of B_{NFA}. Since B_{NFA} has at most $(k+1) \cdot |Q|$ states, it holds that $|Q^k| \leq (k+1) \cdot |Q|$. □

This theorem proves that if the original DFA that recognizes language L does not contain occurrences of the same symbol within k-length sequences of transitions, the size of the DFA recognizing language $L^{(k)}$ is bounded linearly by the size of the NFA constructed by Algorithm 1. Real-life logs and models can contain task repetitions and this, as shown in [12], can potentially lead to the state space explosion in DFAs modeling event log languages with skips. However, as shown in the next section, such cases manifest rarely in practice and pose practical limitations only for large values of k.

5 Evaluation

This section presents results of applying our approach to computing the spectrum of entropy-based precision and recall measurements on the real-world event data. All the experiments were carried out using Intel Xeon Gold 6154 CPU @3.00 GHz with 128 GB RAM and can be reproduced with our publicly available tool [18].

To perform the experiments, we used logs of real-world IT-systems made publicly available by the IEEE Task Force on Process Mining.[4] Prior to the analysis, we filtered out infrequent events that appear in less than 80% of traces using the "filter log using simple heuristics" Process Mining (ProM) plug-in [9]. Hence, we used the same logs as in [19]. Table 2 summarizes characteristics of the filtered logs by showing the total number of unique traces (# Traces), size of the alphabet (# Ev. Classes), and the total number of event occurrences (# Events). Next, we applied the *Inductive* miner [13]

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

to automatically construct Petri nets from the logs. For each Petri net, its reachability graph, represented as a DFA, was constructed. The event logs were also encoded as DFAs. Finally, these DFAs and Algorithm 1 were used to compute the precision and recall values presented in Section 4.2 for different parameters.

Table 3 presents the numbers of states in DFAs that encode the behaviors of process models ($M^{(k)}$), event logs ($L^{(k)}$), and their intersections ($M^{(k)} \cap L^{(k)}$) for different values of k, $k \in \{0, 1, 2, 5, 10, 20, \infty\}$, and the times (in milliseconds) taken to construct the DFAs; we used the technique from [19] to construct DFAs with arbitrary skips ($k = \infty$). If no DFA was constructed (using 128Gb of memory), the corresponding values are not provided. The results show that the numbers of states and the times start to grow as k increases (up to k of 5 for event logs 2 and 3, and $k \in \{10, 20\}$ for the other logs), and then drop. The non-linear growth of states with increasing k (see, for example, event log 4 and the corresponding log DFAs for $k = 5$ and $k = 10$) can be explained by the fact that for large k, events are more likely to be repeated in k-length subsequences and, thus, Theorem 4.3 does not apply. The decreases in the numbers of states relate to the cases when allowing too much behavior leads to DFAs with less number of states; indeed, the fully permissive *flower* model that recognizes all possible traces over a given alphabet contains only one state [2]. Note that all the results for parameters $k \leq 10$ were computed and are suitable for practical applications. Indeed, the precision and recall values computed for up to ten skips are sufficient for many practical scenarios. Note also that all the (not shown in the table) eigenvalues of the corresponding adjacency matrices were computed fast, always under two minutes and often within couple of seconds.

Table 2: Characteristics of event logs.

No.	Event log	# Traces	# Ev. Classes	# Events
1	BPIC'12	2,320	18	164,144
2	BPIC'13 closed	111	3	5,179
3	BPIC'13 open	45	3	1,403
4	BPIC'13 incid.	832	4	44,607
5	WABO 1	709	64	25,823
6	WABO 2	449	85	20,420
7	WABO 3	756	56	28,482
8	WABO 4	580	61	21,848
9	WABO 5	704	68	29,513

Table 4 presents (parts of) the corresponding spectrums of the precision and recall values. Using such spectrums, one can smoothly balance between the two extremes of the exact matching ($k = 0$) and the partial matching ($k = \infty$). Note that the values also confirm the result of Theorem 4.2, which states that $prec^{(k,k)}$ and $recall^{(k,k)}$ approach $prec^{(\infty,\infty)}$ and $recall^{(\infty,\infty)}$ when k approaches infinity.

6 Related Work

Over the past decade, a plethora of conformance checking methods [5] have been developed and proven to be effective in analyzing real-world process data. These methods vary in types of process models and event logs being analyzed, as well as in types of results being produced. Conformance checking techniques can produce a single number assessing the behavioral similarities of process models and event logs (*quantitative* conformance checking) or can provide rich diagnostic information highlighting deviations in model and log behaviors (*qualitative* conformance checking). In this paper, we develop and investigate a novel quantitative conformance checking technique.

Table 3: Numbers of states in DFAs and construction times (in milliseconds).

Event log	DFA	# States / Time (in milliseconds)						
		$k=0$	$k=1$	$k=2$	$k=5$	$k=10$	$k=20$	$k=\infty$
1	$L^{(k)}$	9,102 / 2,129	18,283 / 24,208	31,936 / 48,462	98,523 / 125,806	367,203 / 330,671	1,360,759 / 1,636,200	90,557 / 95,370
	$M^{(k)} \cap L^{(k)}$	9,102 / 15	18,283 / 31	31,936 / 51	98,523 / 230	367,203 / 949	1,360,759 / 4,477	90,557 / 215
	$M^{(k)}$	4 / 4	4 / 8	3 / 12	3 / 4	3 / 7	3 / 20	3 / 4
2	$L^{(k)}$	156 / 15	381 / 47	696 / 74	1,185 / 172	1,213 / 399	216 / 1,000	216 / 44
	$M^{(k)} \cap L^{(k)}$	16 / 27	83 / 31	230 / 39	936 / 63	1,274 / 82	216 / 4	216 / 0
	$M^{(k)}$	3 / 4	7 / 4	9 / 8	15 / 8	25 / 8	45 / 15	1 / 8
3	$L^{(k)}$	33 / 4	51 / 7	62 / 12	69 / 19	17 / 35	17 / 63	17 / 7
	$M^{(k)} \cap L^{(k)}$	33 / 0	51 / 0	62 / 0	69 / 4	17 / 4	17 / 0	17 / 0
	$M^{(k)}$	3 / 7	5 / 4	7 / 8	13 / 8	23 / 8	43 / 15	1 / 8
4	$L^{(k)}$	2,032 / 121	7,451 / 1,324	23,733 / 2,906	417,814 / 37,495	11,331,602 / 1,599,223		24,336 / 1,379,026
	$M^{(k)} \cap L^{(k)}$	6 / 137	3,736 / 324	19,332 / 1,199	440,216 / 33,502	11,635,787 / 1,632,042	–	24,336 / 39
	$M^{(k)}$	5 / 8	9 / 4	13 / 4	25 / 11	45 / 12		1 / 8
5	$L^{(k)}$	10,784 / 2,277	25,259 / 33,135	48,686 / 64,729	254,067 / 202,598	2,020,868 / 1,230,863		
	$M^{(k)} \cap L^{(k)}$	10,784 / 19	25,259 / 51	48,686 / 125	254,067 / 1,344	2,020,868 / 18,595	–	–
	$M^{(k)}$	13 / 59	24 / 82	32 / 71	56 / 102	96 / 118		
6	$L^{(k)}$	12,316 / 2,110	26,892 / 44,344	46,613 / 81,855	182,837 / 217,852	1,308,926 / 1,188,256		
	$M^{(k)} \cap L^{(k)}$	6,482 / 1,277	23,576 / 8,098	47,467 / 27,151	218,291 / 174,728	1,308,926 / 11,282	–	–
	$M^{(k)}$	15 / 50	27 / 63	37 / 54	73 / 63	133 / 148		
7	$L^{(k)}$	9,590 / 4,044	23,647 / 31,585	47,422 / 56,458	275,803 / 210,160	2,040,917 / 1,931,814		1,665,113 / 1,370,908
	$M^{(k)} \cap L^{(k)}$	8,140 / 797	23,747 / 12,993	48,265 / 37,546	275,969 / 263,653	2,040,917 / 21,566	–	1,665,113 / 18,392
	$M^{(k)}$	29 / 82	55 / 78	76 / 97	145 / 270	265 / 2,181		5 / 97
8	$L^{(k)}$	9,187 / 2,824	21,579 / 24,970	41,584 / 48,922	233,666 / 197,214	2,243,180 / 1,454,254		
	$M^{(k)} \cap L^{(k)}$	7,981 / 1,207	21,215 / 8,524	43,074 / 28,967	236,664 / 233,901	2,243,180 / 24,525	–	–
	$M^{(k)}$	57 / 164	116 / 211	176 / 312	356 /1,800	656 / 22,498		
9	$L^{(k)}$	12,891 / 8,165	30,325 / 61,564	58,596 / 106,293	326,177 / 359,597	2,994,538 / 2,666,084		
	$M^{(k)} \cap L^{(k)}$	12,891 / 27	30,325 / 23,771	58,621 / 108,520	326,177 / 1,726	2,994,538 / 27,819	–	–
	$M^{(k)}$	13 / 50	23 / 62	28 / 54	49 / 67	89 / 101		

Existing quantitative conformance checking techniques include such methods as *Projected conformance checking* [14], *k-order Markovian abstractions* [4], *Escaping edges* [17], *Set difference* [11], *Negative events* [7], *Anti-alignments* [8], and *Entropy-based exact* [21] and *partial matching* [19]. Several quantitative stochastic conformance

Table 4: Precision and recall values.

Event log	$prec^{(k,k)}$ / $recall^{(k,k)}$						
	$k=0$	$k=1$	$k=2$	$k=5$	$k=10$	$k=20$	$k=\infty$
1	0.147 / 1.000	0.194 / 1.000	0.241 / 1.000	0.386 / 1.000	0.547 / 1.000	0.650 / 1.000	0.709 / 1.000
2	0.918 / 0.797	0.981 / 0.856	0.990 / 0.918	0.959 / 0.997	0.946 / 1.000	0.961 / 1.000	0.960 / 1.000
3	0.903 / 1.000	0.950 / 1.000	0.955 / 1.000	0.974 / 1.000	0.980 / 1.000	0.980 / 1.000	0.980 / 1.000
4	0.575 / 0.824	0.679 / 0.952	0.763 / 0.988	0.936 / 1.000	0.973 / 1.000	–	0.995 / 1.000
5	0.025 / 1.000	0.034 / 1.000	0.046 / 1.000	0.087 / 1.000	0.145 / 1.000	–	–
6	0.016 / 0.991	0.023 / 0.979	0.031 / 0.877	0.072 / 0.830	0.791 / 1.000	–	–
7	0.030 / 1.000	0.043 / 1.000	0.057 / 1.000	0.095 / 1.000	0.137 / 1.000	–	0.393 / 1.000
8	0.027 / 1.000	0.037 / 1.000	0.048 / 1.000	0.090 / 1.000	0.135 / 1.000	–	–
9	0.020 / 1.000	0.025 / 0.861	0.032 / 0.386	0.083 / 1.000	0.859 / 1.000	–	–

checking approaches have been recently proposed [16,15,20]; these account for the relative likelihoods of traces described in models and recorded in logs. Finally, methods that combine quantitative and qualitative conformance checking techniques visualize the conformance diagnostics over the process model and are based on *alignment*, *token replay*, or *footprint* matrices, refer to [3], [22], and [2], respectively.

In [23], the authors propose various properties that precision and recall measures need to fulfill. Among precision and recall measures [3,14,8,17,11,7,2,24,21,22] being analyzed in [23], only the entropy-based exact matching [21] fulfills all the formal properties. The entropy-based exact and partial matching techniques were also compared to other conformance checking techniques [4,3,14,8,17,11,7] during a qualitative analysis provided in [19]. As demonstrated in [19], the entropy-based methods [21,19] prove their applicability to accurately rank models by their precision values in accordance with the share of behavior not present in the analyzed event log. Although the existing entropy-based measures have advantages over other conformance checking techniques, they present two different extreme measures. The exact entropy-based matching technique is too restrictive, while the partial entropy-based technique substantially extends the log and the model behaviors prior to the comparison. This paper presents an approach that gradually balances between these two different measures.

7 Conclusion and Future Work

This paper proposes a spectrum of conformance measurements for finding deviations between designed and observed processes. The new conformance values inherit properties of the recently proposed entropy-based techniques and provide flexible control over the sensitivity for identifying differences in the compared processes. We prove that with the new conformance measures, one can smoothly balance between the two existing extreme entropy-based techniques. Additionally, we analyzed the new methods' performance characteristics and showed their scalability for analyzing real-world event data. In future work, we plan to extend the techniques by providing qualitative information on differences between designed and observed processes, including the identification and visualization of deviations.

Acknowledgments. This work was in part supported by the Australian Research Council project DP180102839.

References

1. van der Aalst, W.: Service mining: Using process mining to discover, check, and improve service behavior. IEEE Transactions on Services Computing **6**(4), 525–535 (2013)
2. van der Aalst, W.: Process Mining: Data Science in Action. Springer (2016)
3. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B., van der Aalst, W.: Measuring precision of modeled behavior. Inf. Syst. and e-Business Managem. **13**(1), 37–67 (2015)
4. Augusto, A., Armas-Cervantes, A., Conforti, R., Dumas, M., La Rosa, M., Reissner, D.: Abstract-and-compare: A family of scalable precision measures for automated process discovery. In: Business Process Management. pp. 158–175. Springer, Cham (2018)
5. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking—Relating Processes and Models. Springer (2018)
6. Ceccherini-Silberstein, T., Machì, A., Scarabotti, F.: On the entropy of regular languages. Theor. Comp. Sci. **307**, 93–102 (2003)
7. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A robust F-measure for evaluating discovered process models. In: CIDM. pp. 148–155. IEEE (2011)
8. van Dongen, B., Carmona, J., Chatain, T.: A unified approach for measuring precision and generalization based on anti-alignments. In: BPM. pp. 39–56. Springer, Cham (2016)
9. van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: The ProM Framework: A new era in process mining tool support. In: ATPN. pp. 444–454 (2015)
10. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, USA (2005)
11. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Trans. on Knowl. and Data Eng. **18**(8), 1010–1027 (2006)
12. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 3rd Edition. Pearson International Edition, Addison-Wesley (2007)
13. Leemans, S., Fahland, D., van der Aalst, W.: Discovering Block-Structured Process Models from Incomplete Event Logs. In: ATPN, LNCS, vol. 8489, pp. 91–110. Springer (2014)
14. Leemans, S., Fahland, D., van der Aalst, W.: Scalable process discovery and conformance checking. Software & Systems Modeling **17**(2), 599–631 (2018)
15. Leemans, S., Polyvyanyy, A.: Stochastic-aware conformance checking: An entropy-based approach. In: CAiSE. LNCS, vol. 12127, pp. 217–233. Springer (2020)
16. Leemans, S., Syring, A., van der Aalst, W.: Earth movers' stochastic conformance checking. In: BPM Forum. LNBIP, vol. 360, pp. 127–143. Springer (2019)
17. Muñoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: Business Process Management. pp. 211–226. Springer, Berlin, Heidelberg (2010)
18. Polyvyanyy, A., Alkhammash, H., Di Ciccio, C., García-Bañuelos, Kalenkova, A., Leemans, S., Mendling, J., Moffat, A., Weidlich, M.: Entropia: A Family of Entropy-Based Conformance Checking Measures for Process Mining. In: CoRR. vol. abs/2008.09558 (2020)
19. Polyvyanyy, A., Kalenkova, A.: Monotone conformance checking for partially matching designed and observed processes. In: ICPM. pp. 81–88. IEEE (2019)
20. Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: ICPM. IEEE (2020), In Press
21. Polyvyanyy, A., Solti, A., Weidlich, M., Di Ciccio, C., Mendling, J.: Monotone precision and recall measures for comparing executions and specifications of dynamic systems. ACM Trans. Softw. Eng. Methodol. **29**(3) (Jun 2020)
22. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008)
23. Syring, A., Tax, N., van der Aalst, W.: Evaluating conformance measures in process mining using conformance propositions. In: ToPNoC XIV. pp. 192–221. Springer (2019)
24. Weijters, A., van der Aalst, W., Alves De Medeiros, A.: Process mining with the heuristics miner algorithm. Tech. rep., TU/e, Eindhoven (2006)