Monotone Conformance Checking for Partially Matching Designed and Observed Processes

Artem Polyvyanyy
School of Computing and Information Systems
The University of Melbourne
Email: artem.polyvyanyy@unimelb.edu.au

Anna Kalenkova
Laboratory of Process-Aware Information Systems
Higher School of Economics
Email: akalenkova@hse.ru

***Abstract*—Conformance checking is a subarea of process mining that studies relations between designed processes, also called process models, and records of observed processes, also called event logs. In the last decade, research in conformance checking has proposed a plethora of techniques for characterizing the discrepancies between process models and event logs. Often, these techniques are also applied to measure the quality of process models automatically discovered from event logs. Recently, the process mining community has initiated a discussion on the desired properties of such measures. This discussion witnesses the lack of measures with the desired properties and the lack of properties intended for measures that support partially matching processes, i.e., processes that are not identical but differ in some steps. The paper at hand addresses these limitations. Firstly, it extends the recently introduced precision and recall conformance measures between process models and event logs that possess the desired property of monotonicity with the support of partially matching processes. Secondly, it introduces new intuitively desired properties of conformance measures that support partially matching processes and shows that our measures indeed possess them. The new measures have been implemented in a publicly available tool. The reported qualitative and quantitative evaluations based on our implementation demonstrate the feasibility of using the proposed measures in industrial settings.**

Keywords: Process mining, conformance checking, entropy, partial matching, monotonicity.

I. Introduction

Process mining aims to discover, monitor, and improve processes *observed* in the real world using the knowledge accumulated in event logs produced by executions of *designed* processes implemented in modern information systems [1], where an event log is a collection of recorded *traces* each capturing an instance of an executed business process. Process discovery and conformance checking are two central subareas in process mining. Process discovery studies methods, techniques, and tools to automatically construct a "good" process model from an event log to aggregate and analyze the knowledge about business processes as they were observed in the real world [2]–[4]. Conformance checking studies methods, techniques, and tools to characterize and quantify differences between an event log and process model [5]–[8], e.g., to measure and explain the "goodness" of the model (constructed or discovered) that encodes the traces from the event log.

Two basic measures in conformance checking are *precision* and *recall* (a.k.a. *fitness*). Precision aims to characterize the relation between the amount of process information shared by the traces of an event log and process model and the amount of process information encoded in the traces of the model. Recall, in turn, strives to characterize the relation between the shared information and the information about processes encoded in the event log traces. The precision and recall are usually defined as numeric quantities that take values between (and including) zero and one. The precision and recall values of zero represent a situation of no similarities between the traces of the event log and the model traces. The greater the precision and recall values, the greater the similarity between the event log and the process model traces. Finally, the precision and recall values of one represent the situation when the event log and process model capture the same behavior.

Precision and recall are central performance measures for information retrieval systems [9]. Given a set of relevant documents and a set of documents retrieved/discovered by an information retrieval system, *precision* is the fraction of relevant retrieved documents over the retrieved documents, whereas *recall* is the fraction of relevant retrieved documents over the relevant documents. Similarly, given traces recorded in an event log (a.k.a. relevant traces) and a set of traces described by a process model discovered from the event log, e.g., using techniques proposed in [2]–[4], (a.k.a. discovered traces), *precision* is the fraction of relevant discovered traces over the discovered traces, whereas *recall* is the fraction of relevant discovered traces over the relevant traces.

Despite being conceptually clean and intuitive, the proposed above conformance measures of precision and recall face at least one major challenge. Unlike in information retrieval, where both collections of relevant and retrieved documents are finite, the set of traces encoded in a (discovered) process model is often infinite. This makes it practically difficult to define conformance measures with the desired properties, e.g., determinism and monotonicity, as it is not immediate to define how to measure infinite collections of traces. In [8], we overcame this challenge by proposing process conformance measures of precision and recall based on the notion of *topological entropy* over regular languages.

Although the entropy-based precision and recall are deterministic and monotonic measures, they perform unsatisfactorily in the presence of non-identical traces, even when traces differ only in a single process step. For instance, if each trace in an event log is *not* described in a process

model but differs with some trace of the model in only one process step, precision and recall values between the event log and model are equal to zero, just like in the situation when these traces are completely different. In this paper, we overcome this limitation. We achieve this by "diluting" the traces captured in the compared event log and process model and then comparing these diluted traces. In our approach, the dilution of a trace results in the inclusion of all its sub-traces into the collection of traces captured in the corresponding event log or process model. We demonstrate that the extended measures possess intuitively desired properties for precision and recall in the presence of partially matching traces, viz. the more (in the number of traces) and the longer (in the number of matched process steps) are the partial matches between the traces captured in an event log and process model, the greater are the precision and recall values between them.

To summarize, this paper makes these contributions:

- Extends the entropy-based precision and recall measures proposed in [8] to support the partial matching between traces via comparisons of their sub-traces;
- Introduces, for the first time, properties for conformance measures that address the partial matching of traces; note that all the so far introduced properties for conformance measures [8], [10], [11] only address the comparisons of identical traces; and
- Reports on qualitative and quantitative evaluations of the proposed precision and recall measures that demonstrate the feasibility of using them in industrial settings.

The next section gives a motivating example. Section III introduces the basic notions. Section IV presents the entropy-based measures, whereas Section V extends them with partial matching support. Section VI presents the results of our evaluation, before stating concluding remarks.

II. Motivating Example

Consider a simple process of booking a flight ticket captured as a finite automaton in Figure 1a. The process starts with the user opening a booking window. Then, she fills her name and passport data (in any order). Finally, the booking is confirmed and the window is closed. Suppose that the user behavior in the real-world can deviate from the one specified in the reference model. For example, a user may insert the name twice and close the window without the confirmation; this sequence of steps is captured in the automaton shown in Figure 1b. Despite the fact that the automata presented in Figure 1 describe similar traces, precision and recall measures that rely on exact comparisons of traces cannot appraise this similarity (i.e., the corresponding values of precision and recall equal to zero).

To address this limitation, we observe that the traces captured in the example automata coincide if certain steps get replaced with silent (τ) steps. If one "mirrors" step *confirm* in the automaton in Figure 1a and one step *fill name* (any of the two) in the only trace of the automaton in Figure 1b with silent steps, cf. the dashed transitions in the figure, then both resulting automata describe the observable trace

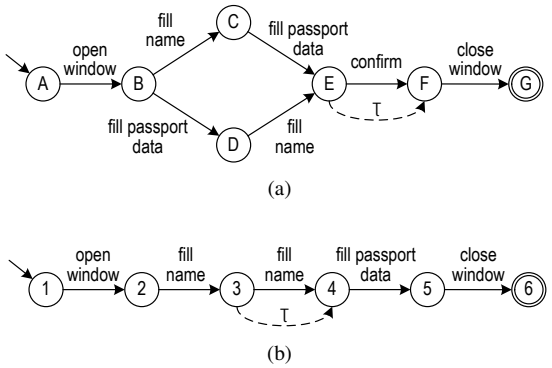

Figure 1: Two finite automata.

$\langle open\ window, fill\ name, fill\ passport\ data, close\ window \rangle$. This idea for identifying partially matching traces through their "dilution" with silent steps lies at the heart of the solution proposed in the work at hand.

According to the conformance measures proposed in this paper, precision and recall values between the automata in Figure 1 are 0.859 and 0.694, respectively. If one inserts step *confirm* between *fill passport data* and *close window* in the only trace of the automaton in Figure 1b, i.e., makes it more similar to the traces of the automaton in Figure 1a, the values of precision and recall increase to 0.978 and 0.973, respectively, capturing the monotonic nature of our measures.

III. Preliminaries

This section introduces basic notations and definitions.

A. Sequences, Languages, and Event Logs

Let X be a set of elements. A *power set* over set 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 $x_1, x_2, \ldots, x_k \in X$, $k \in \mathbb{N}_0$, we denote a *sequence* of elements from X of length k. The *empty sequence* of zero length is represented by $\langle\rangle$. 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 equals to $\langle x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_l \rangle$. X^* designates the set of all finite sequences over X (including the *empty sequence*).

An *alphabet* is any nonempty finite set. The elements of an alphabet are its *labels*. A (formal) *language* L over an alphabet Σ is a (not necessarily finite) set of *sequences*, or *words*, of elements 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. τ designates a special *silent* label, such that $\tau \notin \Xi$ and $\langle \tau \rangle = \langle\rangle$. Let L_1 and L_2 be two languages. Then, their concatenation is the language $\{l_1 \cdot l_2 \,|\, l_1 \in L_1, l_2 \in L_2\}$ denoted by $L_1 \circ L_2$.

Let E be a finite nonempty set of events. A finite language $L \subseteq E^*$ is an *event log* and its words are called *traces*.

B. Finite Automata

A *deterministic finite automaton*, or a DFA, is a 5-tuple $(Q, \Lambda, \delta, q_0, A)$, where Q is a finite set of *states*, $\Lambda \subset \Xi$ is the *input alphabet*, such that Q and Λ are disjoint, $\delta : Q \times (\Lambda \cup$

$\{\tau\}) \to Q$ is the *transition function*, $q_0 \in Q$ is a *start state*, and $A \subseteq Q$ is a set of *accepting states*.

A *computation* of a DFA $B = (Q, \Lambda, \delta, q_0, A)$ is either the empty sequence or a sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$, $n \in \mathbb{N}$, where $a_i \in \Lambda \cup \{\tau\}$, $i \in [1..n]$, and there is a sequence of states $\langle q_0, q_1, \ldots, q_n \rangle$, where $q_j \in Q$, $j \in [0..n]$, such that for every $k \in [1..n]$ it holds that $\delta(q_{k-1}, a_k) = q_k$. We say that σ *leads to* q_n. By convention, the empty sequence always leads to the start state. Note that when referring to computations silent labels can be omitted, as $\langle a_1, a_2, ..., a_i, \tau, a_{i+1}, ..., a_k \rangle = \langle a_1, a_2, ..., a_i \rangle \cdot \langle \tau \rangle \cdot \langle a_{i+1}, ..., a_k \rangle = \langle a_1, a_2, ..., a_i \rangle \cdot \langle \rangle \cdot \langle a_{i+1}, ..., a_k \rangle = \langle a_1, a_2, ..., a_k \rangle$. In what follows, we only consider computations without silent labels. We say that B *accepts* $\gamma \in \Lambda^*$ iff γ is a computation of B that leads to an accepting state $q \in A$. The *language* of B is denoted by $lang(B)$ and is the set of all sequences B accepts; we also say that B *recognizes* $lang(B)$.

A DFA $(Q, \Lambda, \delta, q_0, A)$ is *ergodic* if its underlying graph is strongly irreducible, i.e., for all $(x, y) \in Q \times Q$ there exists a sequence of states $\langle q_1, \ldots, q_n \rangle \in Q^*$, $n \in \mathbb{N}$, such that $q_1 = x$, $q_n = y$, and for every $k \in [1 \,..\, n-1]$ there exists $\lambda \in \Lambda \cup \{\tau\}$ such that $\delta(q_k, \lambda) = q_{k+1}$.

A language $L \subseteq \Sigma^*$ is *regular* iff it is recognized by a DFA. $L \subseteq \Sigma^*$ is *irreducible* if, given two words w_1 and w_2 in L, there exists a word $w \in \Sigma^*$, such that $w_1 \cdot w \cdot w_2 \in L$. A regular language L is irreducible iff it is a language of an ergodic DFA [12].

IV. Entropy-based Conformance Checking

In this paper, we consider models such that their behaviors can be described in terms of DFAs. These can be Petri nets or BPMN models that induce finite reachability graphs, or (not necessary deterministic) finite automata, which can always be converted into equivalent DFAs [13]. As an event log induces a finite language, it can, as well, be encoded as a DFA.

To compute precision and recall between a model and event log, we compare the languages the corresponding DFAs recognize. Specifically, we measure the ration of the traces the model and log share to the traces captured in the model or log. To obtain such a measure over collections of traces, we estimate the cardinality of the corresponding languages, i.e., the number of words that belong to the languages.

Next, in Section IV-A, we present the notion of topological entropy. We use topological entropy to define the notion of the short-circuit entropy of a regular language (see Section IV-B), which, in turn, in Section IV-C, is used to define the entropy-based precision and recall between a model and log.

A. Topological Entropy

The cardinality of a finite language $L \subseteq \Sigma^*$ can be naturally defined as $|L|$, i.e., the number of words in L. However, this definition is not particularly useful for infinite languages. One can use *topological entropy* to estimate the cardinality of an irreducible language L [12], which is defined as follows:

$$ent(L) = \lim_{n \to \infty} \sup \frac{\log |C_n(L)|}{n}.$$

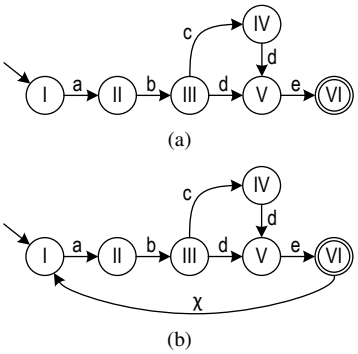

Figure 2: Two finite automata; (b) is ergodic.

Thus, the topological entropy of an irreducible language estimates the number of distinct words in the language with respect to the length of these words. One can compute the topological entropy of an irreducible language L as the logarithm of the Perron-Frobenius eigenvalue of the adjacency matrix of a DFA that recognizes L [12].

B. Short-circuit Entropy

Because topological entropy is defined over irreducible languages, one cannot use it to estimate cardinality of infinite non-irreducible regular languages or finite languages, e.g., event logs. To allow estimating cardinality of an arbitrary regular language, in [8], we proposed the notion of a *short-circuit measure* over languages. Given a measure over languages $m : \mathcal{L} \to \mathbb{R}_0^+$, where $\mathcal{L} \subseteq \mathcal{P}(\Sigma^*)$ and $\Sigma \subset \Xi$, its *short-circuit* version $m\bullet$ is defined as $m\bullet(L) = m((L \circ \{\langle \chi \rangle\})^* \circ L)$, where $L \subseteq \Sigma^*$ and $\chi \in \Xi \setminus \Sigma$. Because given a regular language L, $(L \circ \{\langle \chi \rangle\})^* \circ L$ is irreducible [8], $ent\bullet(L)$ can always be computed.

Hence, to compute the *short-circuit (topological) entropy* of a regular language L, i.e., $ent\bullet(L)$, one can transform a DFA that recognizes L by inserting additional transitions marked by a special label χ not observed in L that connect all the accepting states of the DFA with its start state, and then compute the topological entropy of the language recognized by the resulting automaton. For example, $ent\bullet(L)$, where L is the language recognized by the DFA in Figure 2a, can be computed as topological entropy of the language recognized by the DFA in Figure 2b.

It holds that $ent\bullet(\emptyset) = 0$. Moreover, for two regular languages L_1 and L_2, such that $L_1 \subset L_2$, it holds that $ent\bullet(L_1) < ent\bullet(L_2)$, refer to [8] for details. Next, we show two additional properties of the short-circuit entropy measure over regular languages. These properties of language measures are used to establish several new properties of the entropy-based precision and recall measures, which also apply for the new conformance measures that support partial matching of processes, refer to Section V.

If for every length, one regular language has no more words of that length than the other regular language, then the short-circuit entropy of the former language is less than or equal to the short-circuit entropy of the latter language.

Theorem 1 (Monotonicity on the number of words)**.**
Let L_1 and L_2 be two regular languages such that $\forall\, k \in \mathbb{N}: |C_k(L_1)| \leq |C_k(L_2)|$. Then, $ent\bullet(L_1) \leq ent\bullet(L_2)$.

Proof. Let $x_n = \frac{\log |C_n(L_1)|}{n}$ and $y_n = \frac{\log |C_n(L_2)|}{n}$, $n \geq 1$. Suppose that $\{x_{n_l}\}_{l=1}^{\infty}$, where $n_1 < n_2 < \ldots$, is a subsequence of $\{x_n\}_{n=1}^{\infty}$, such that $\lim_{l\to\infty} x_{n_l} = ent\bullet(L_1)$. Consider the corresponding sequence $\{y_{n_l}\}_{l=1}^{\infty}$. As follows from the theorem conditions, $\forall l \in \mathbb{N}, l \geq 1: x_{n_l} \leq y_{n_l}$, then $\lim_{l\to\infty} \sup x_{n_l} \leq \lim_{l\to\infty} \sup y_{n_l}$. Hence, $ent\bullet(L_1) = \lim_{l\to\infty} x_{n_l} = \lim_{l\to\infty} \sup x_{n_l} \leq \lim_{l\to\infty} \sup y_{n_l} \leq ent\bullet(L_2)$. □

Next, we refine the above property.

Theorem 2 (Strict monotonicity on the number of words)**.**
Let L_1 and L_2 be two regular languages such that $\forall\, k \in \mathbb{N}: |C_k(L_1)| \leq |C_k(L_2)|$ and $\exists\, k_0 \in \mathbb{N}: |C_{k_0}(L_1)| < |C_{k_0}(L_2)|$. Then, $ent\bullet(L_1) < ent\bullet(L_2)$.

Proof. Language L_1 does not contain the maximum number of sequences of length k_0, because $|C_{k_0}(L_1)| < |C_{k_0}(L_2)|$. Let l be a sequence such that $l \in \Sigma^*$, $|l| = k_0$ and $l \notin L_1$. Let us consider language $L_1' = L_1 \cup \{l\}$. Then, $ent\bullet(L_1) < ent\bullet(L_1')$, because of the monotonicity of the short-circuit entropy measure, see [8]. According to Theorem 1, $ent\bullet(L_1') \leq ent\bullet(L_2)$. Thus, $ent\bullet(L_1) < ent\bullet(L_2)$. □

Theorems 1 and 2 allow comparing short-circuit entropy measures of languages that are not in the containment relationship. According to Theorem 2, for example, for two languages $L_1 = \{\langle a, a\rangle, \langle a, a, a\rangle, \langle a, a, a, a\rangle, ...\}$ and $L_2 = \{\langle b\rangle, \langle b, b\rangle, \langle b, b, b\rangle, \langle b, b, b, b\rangle, ...\}$ it holds that $ent\bullet(L_1) < ent\bullet(L_2)$.

C. Precision and Recall

Let M and L be two regular languages that capture the traces of the model and log, respectively. The intersection of M and L is a regular language, refer to [13]. One can use $ent\bullet(M \cap L)$ to estimate the cardinality of the collection of all the traces shared by the model and log. Consequently, we define the entropy-based precision ($prec$) and recall ($recall$) between M and L as follows:

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

In [8], we showed that as the number of traces shared by the model and log increases, the entropy-based precision and recall also increase. Next, we demonstrate three additional properties of the entropy-based precision and recall, which also hold for their extensions proposed in Section V.

Theorem 3 (Monotonicity)**.**
Let L_1 and L_2 be two event logs (let M_1 and M_2 be two regular languages) such that $L_1 \subset L_2$ ($M_1 \subset M_2$). Let M be a regular language (let L be an event log). Then, $prec(M, L_1) \leq prec(M, L_2)$ ($recall(M_1, L) \leq recall(M_2, L)$).

Proof. Since it holds that $L_1 \subset L_2$ ($M_1 \subset M_2$), it also holds that $M \cap L_1 \subseteq M \cap L_2$ ($M_1 \cap L \subseteq M_2 \cap L$). Because of the monotonicity of the short-circuit entropy, it holds that $ent\bullet(M \cap L_1) \leq ent\bullet(M \cap L_2)$ ($ent\bullet(M_1 \cap L) \leq ent\bullet(M_2 \cap L)$). Hence, it also holds that $prec(M, L_1) \leq prec(M, L_2)$ ($recall(M_1, L) \leq recall(M_2, L)$). □

Theorem 3 shows that the "monotonicity of languages" implies the monotonicity of the corresponding precision and recall values. Next, we demonstrate two additional properties of precision and recall that follow from Theorems 1 and 2. These properties are discussed and exemplified in Section V.

Theorem 4 (Generalized monotonicity)**.**
Let L_1 and L_2 be two event logs (let M_1 and M_2 be two regular languages) and let M be a regular language (let L be an event log) such that $\forall k \in \mathbb{N}: |C_k(M \cap L_1)| \leq |C_k(M \cap L_2)$ ($\forall k \in \mathbb{N}: |C_k(M_1 \cap L)| \leq |C_k(M_2 \cap L)|$). Then, $prec(M, L_1) \leq prec(M, L_2)$ ($recall(M_1, L) \leq recall(M_2, L)$).

Theorem 4 follows from Theorem 1 applied to $M \cap L_1$ and $M \cap L_2$ ($M_1 \cap L$ and $M_2 \cap L$). This result can be conveniently refined into the next one.

Theorem 5 (Generalized strict monotonicity)**.**
Let L_1 and L_2 be two event logs (let M_1 and M_2 be two regular languages) and let M be a regular language (let L be an event log) such that $\forall\, k \in \mathbb{N}: |C_k(M \cap L_1)| \leq |C_k(M \cap L_2)|$ ($\forall\, k \in \mathbb{N}: |C_k(M_1 \cap L)| \leq |C_k(M_2 \cap L)|$) and $\exists\, k_0 \in \mathbb{N}: |C_{k_0}(M \cap L_1)| < |C_{k_0}(M \cap L_2)|$ ($\exists\, k_0 \in \mathbb{N}: |C_{k_0}(M_1 \cap L)| < |C_{k_0}(M_2 \cap L)|$). Then, $prec(M, L_1) < prec(M, L_2)$ ($recall(M_1, L) < recall(M_2, L)$).

Theorem 5 follows immediately from the application of Theorem 2 to languages $M \cap L_1$ and $M \cap L_2$ ($M_1 \cap L$ and $M_2 \cap L$).

Consider the two DFAs shown in Figure 3a and Figure 3b that recognize languages M_1 and M_2, respectively, and three simple event logs $L_1 = \{\langle a, b\rangle, \langle b, a\rangle\}$, $L_2 = \{\langle\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle, \langle b, a, b\rangle\}$, and $L_3 = \{\langle b, a, d\rangle, \langle b, a, b\rangle\}$.

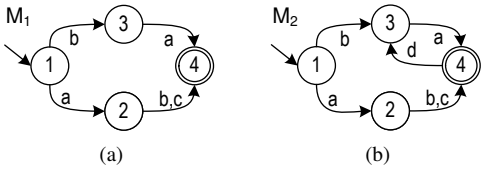

Figure 3: Two finite automata.

The entropy-based precision and recall values for all the combinations of these two regular languages and three event logs are listed in Table I below.

Table I: The entropy-based precision and recall values.

Model	Log	*prec*	*recall*
M_1	L_1	0.874	1.000
M_1	L_2	1.000	0.745
M_1	L_3	0.000	0.000
M_2	L_1	0.754	1.000
M_2	L_2	0.863	0.745
M_2	L_3	0.000	0.000

The values in Table I justify, e.g., that M_1 contains all the traces from L_1 ($recall(M_1, L_1) = 1$) and does not contain some traces from L_2 ($recall(M_1, L_2) < 1$). One

can also conclude that L_2 "covers" all the behavior of M_1 ($prec(M_1, L_2) = 1$). Furthermore, M_2 does not contain some traces from L_2 ($recall(M_2, L_2) < 1$). Interestingly, according to the results in [8], it holds that $prec(M_2, L_1) < prec(M_1, L_1)$ and $prec(M_2, L_2) < prec(M_1, L_2)$, because $L_1 \subset M_1 \subset M_2$, $M_1 \cap L_2 = M_2 \cap L_2$, and $(M_1 \cap L_2) \subset M_1$.

Note that L_3 does not intersect with M_1 (or M_2). Consequently, despite some shared subsequences in their traces, e.g., $\langle b, a \rangle$, the corresponding precision and recall values equal to zero. This limitation is addressed in the next section.

V. Partially Matching Approach

In this section, we extend the conformance checking approach summarized in the previous section with support for partial matching of the compared model and log. We first define additional notions and discuss their properties, refer to Section V-A. We then use these notions to propose new precision and recall measures, refer to Section V-B.

A. Entropy and τ-closure of Regular Languages

Let $B = (Q, \Lambda, \delta, q_0, A)$ be a DFA that recognizes language L, i.e., $lang(B) = L$. We construct the τ-closure of B, denoted by B', as follows: $B' = (Q, \Lambda, \delta', q_0, A)$, $\delta'(q_1, a) = q_2$ iff $(\delta(q_1, a) = q_2) \vee ((a = \tau) \wedge (\exists\, a' : \delta(q_1, a') = q_2))$. In other words, for each two states connected via a transition in B, we add an additional silent transition that connects these states. We call B' ($lang(B')$) the *τ-closure* of B (L). Figure 4 shows the τ-closure of the automaton from Figure 2a.

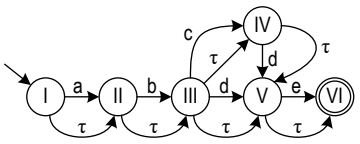

Figure 4: τ-closure of the DFA from Figure 2a.

By L', we denote the language recognized by B'. Note that $L \subseteq L'$. Next, we state two important properties of the short-circuit entropy over the τ-closures of regular languages.

Theorem 6 (Monotonicity of τ-closure)**.**
Let L_1 and L_2 be two regular languages such that $L_1 \subset L_2$. Then, it holds that $ent\bullet(L_1') \leq ent\bullet(L_2')$.

Proof. For each $\alpha \in L_1$ it holds that $\alpha \in L_2$. Consequently, all the sequences obtained from α by the τ-closure operation belong to both L_1' and L_2'. This implies $L_1' \subseteq L_2'$. Consequently, $ent\bullet(L_1') \leq ent\bullet(L_2')$ because of the monotonicity of the $ent\bullet$ measure. □

Hence, τ-closure relaxes the strict monotonicity. Note that in case the τ-closure operation does not preserve the strict monotonicity, both measures, the exact trace matching and the partial matching, can be applied to obtain the complete information about deviations between the languages. We now define a condition under which one obtains strict monotonicity.

Theorem 7 (Strict monotonicity of τ-closure)**.**
Let L_1 and L_2 be two regular languages, such that $\exists\, \alpha \in L_2 : \alpha \notin L_1'$ and $\forall\, \beta \in L_1 : \beta \in L_2'$, then $L_1' \subset L_2'$ and $ent\bullet(L_1') < ent\bullet(L_2')$.

Proof. Since for each $\beta \in L_1$ it holds that $\beta \in L_2'$, it also holds that $\{\beta\}' \subseteq L_2'$, because if a τ-closure of a language contains β, it contains all the sequences obtained from β using the τ-closure operation. Consequently, $L_1' \subseteq L_2'$. Additionally, since $\exists\, \alpha \in L_2 : \alpha \notin L_1'$, it is a strict inclusion $L_1' \subset L_2'$, and $ent\bullet(L_1') < ent\bullet(L_2')$. □

Suppose that L_1 and L_2 are two regular languages, such that $L_1' = (L_2')\backslash\{\alpha\}$ and $\alpha \in L_2$, i.e., their τ-closures differ in one sequence α. Both L_1' and L_2' contain all "sub-words" of α, i.e., $\{\alpha\}'\backslash\alpha \subseteq L_1'$ and $\{\alpha\}'\backslash\alpha \subset L_2'$. According to Theorem 7, it holds that $ent\bullet(L_1') < ent\bullet(L_2')$. Consequently, one may note that a "long" sequence α belonging to L_2 which cannot be generated by constructing a τ-closure of L_1 plays a role when comparing $ent\bullet(L_1')$ and $ent\bullet(L_2')$, even if L_1' and L_2' share the same set of its "sub-words".

Consider two models that recognize languages $M_1 = \{\langle a, b \rangle, \langle a, c \rangle \langle b, c \rangle \langle d \rangle\}$ and $M_2 = \{\langle a, b, c \rangle, \langle d \rangle\}$. Obviously, $M_2' = M_1' \cup \{\langle a, b, c \rangle\}$. Let $L = \{\langle a, b, c \rangle\}$ be a language. According to Theorem 7, $recall(M_2', L') > recall(M_1', L')$, as the "long" sequence $\langle a, b, c \rangle$ belongs to M_2 and cannot be obtained by applying the τ-closure operation to M_1.

B. Precision and Recall

Let M and L be languages recognized by DFAs that encode a model and log, respectively. The relations between M and L and their τ-closures are represented in Figure 5.

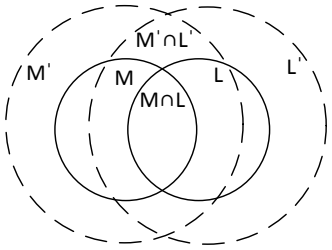

Figure 5: Intersection of two languages and their τ-closures.

We propose to measure precision and recall between the model and log based on the τ-closures of M and L as follows:

$$prec_\tau(M, L) = \frac{ent\bullet(M' \cap L')}{ent\bullet(M')},$$

$$recall_\tau(M, L) = \frac{ent\bullet(M' \cap L')}{ent\bullet(L')},$$

where M' and L' are the τ-closures of M and L, respectively. Note that first M', L', and $M' \cap L'$ are constructed and only after that the short-circuit entropy is computed. Table II extends Table I by also showing the new precision and recall values for the corresponding models and logs.

These example values show that although log L_3 has no common traces with the models, they can be partially matched. According to Theorem 7, it holds that $L_1' \subset L_2'$, and according to Theorem 3 (applied to L_1', L_2', M_1', and M_2'; note that

Table II: The entropy-based precision and recall values, both original and based on the τ-closures of languages.

Model	Log	$prec$	$recall$	$prec_\tau$	$recall_\tau$
M_1	L_1	0.874	1.000	0.873	1.000
M_1	L_2	1.000	0.745	1.000	0.960
M_1	L_3	0.000	0.000	0.873	0.811
M_2	L_1	0.754	1.000	0.615	1.000
M_2	L_2	0.863	0.745	0.704	0.960
M_2	L_3	0.000	0.000	0.733	0.966

τ-closures of regular languages are also regular languages), new precision values for L_1 (0.873 and 0.615) are indeed less than or equal to the corresponding new precision values for L_2 (1.000 and 0.704). Interestingly, $prec_\tau(M_1, L_2) = 1$ because $M_1' \subseteq L_2'$. Also, note that the absolute values of the reported measures are of minor importance, as those are their relations that provide useful insights. According to Theorem 7, $L_1' \subset L_3'$, similarly, the new precision values for L_3 (0.873 and 0.733) are greater than or equal to the corresponding new precision values for L_1 (0.873 and 0.615).

It is easy to verify that $M_1' \subset M_2'$, refer to Figure 3. Then, according to Theorem 3, recall values for M_2 are always greater or equal to the corresponding recall values for M_1.

Let us take a closer look at logs L_2 and L_3. None of these logs includes the other; same holds for logs L_2' and L_3'. It holds that $M_2' \cap L_2' = \{\langle\rangle, \langle a\rangle, \langle b\rangle, \langle c\rangle, \langle a,b\rangle, \langle b,a\rangle, \langle a,c\rangle\}$ and $M_2' \cap L_3' = \{\langle\rangle, \langle a\rangle, \langle b\rangle, \langle d\rangle, \langle a,b\rangle, \langle b,a\rangle, \langle a,d\rangle, \langle b,d\rangle, \langle b,a,d\rangle\}$. The former language contains less number of sequences of the length two and three. Hence, by Theorem 5, it must hold that $prec_\tau(M_2, L_2) < prec_\tau(M_2, L_3)$. Indeed, according to the values in Table II: $prec_\tau(M_2, L_2) = 0.704$ and $prec_\tau(M_2, L_3) = 0.733$. Note that this property allows comparing languages over different alphabets.

VI. Experimental Results

The proposed partial matching technique has been implemented and is publicly available.[1] Next, we evaluate this implementation over synthetic and real-life datasets.

A. Synthetic Dataset

In this section, we experiment with a synthetic event log and a set of corresponding process models described in [14], [15]. The event log is defined as this set of traces: $L = \{\langle A,B,D,E,I\rangle, \langle A,C,D,G,H,F,I\rangle, \langle A,C,G,D,H,F,I\rangle, \langle A,C,H,D,F,I\rangle, \langle A,C,D,H,F,I\rangle\}$. The set of process models consists of nine Petri nets shown in Figure 6. The reachability graphs of all these nine Petri nets can be encoded as DFAs. Table III shows precision and recall values between languages recognized by these DFAs and L. All these values were computed in close to real-time (in milliseconds) using Intel Core i3-3110M CPU @2.40 GHz with 4 GB RAM.

The values in the left-most $recall$ column in Table III measure the share of traces from the log that are also the traces of the model. Six models can "replay" all the log traces, hence the recall values of one. The *Single trace* model accepts only one trace from the log, which leads to the recall value of less than one. The *All parallel* model, which imposes a restriction that all the labels must appear in a trace, and the *Round robin* model, which executes all the activities in a particular order without skipping them, do not accept a single trace from the log. Therefore, the corresponding recall values equal to zero. Note that the recall values that are based on the τ-closures of languages show that the *All parallel* and *Round robin* model describe traces that preserve the order (assuming skips in the model traces) of events as they appear in the traces of the event log, see the corresponding recall values of 1.000.

Table III: Precision and recall values for synthetic log.

Model	$prec$	$recall$	$prec_\tau$	$recall_\tau$
$Original\ model$	0.979	1.000	0.998	1.000
$Single\ trace$	1.000	0.798	1.000	0.732
$Separated\ traces$	1.000	1.000	1.000	1.000
$Flower\ model$	0.125	1.000	0.479	1.000
$H\ and\ G\ in\ parallel$	0.889	1.000	0.986	1.000
$H\ and\ G\ in\ loops$	0.568	1.000	0.933	1.000
$D\ in\ a\ loop$	0.758	1.000	0.970	1.000
$All\ parallel$	0.000	0.000	0.656	1.000
$Round\ robin$	0.000	0.000	0.479	1.000

Let us now examine the precision values. We ranked the models in accordance with their precision values in Table IV.

The table shows our rankings (last two columns) and compares them with the rankings of other conformance techniques calculated for the same models and log in [14], [15]. Concretely, these techniques are used (refer to columns 2–8 in the table): *Set difference* (*SD*) [16], *Negative events* (*NE*) [17], *Escaping edges* (*ETC*) [18], *Alignment-based ETC precision* (*ETCa*) [19], *Projected conformance checking* (*PCC*) [20], *Anti-alignment precision* (*AA*) [14], [21], and *k-order Markovian abstractions* (*MAP*^{k}) [15]. Greater values of k for the latter technique correspond to less abstraction in the encodings of models and logs and, thus, more "precise" precision measurements. Column *EB* stands for the *entropy-based* approach from Section IV, while *EB*$^\tau$ for its extension from Section V.

The *entropy-based* approach (*EB*) ranks the models quantifying the behavior which is "covered" by the traces of the event log. Because models *All parallel* and *Round robin* are not covered at all, their precision w.r.t. the log is the least, while *Single trace* and *Separated traces* models are totally covered and, thus, they have the best precision w.r.t. the log.

The ranking produced by the partial matching approach (*EB*$^\tau$) is closest to that one by *anti-alignment precision* (*AA*). The difference is that both our approaches show that model *H and G in loops* has a lower precision than the model *D in a loop*. Indeed, these models share the same traces with the log, but the behavior described in *H and G in loops* model has more variability, i.e., the number of sequences between C and F is higher for *H and G in loops* model. The monotonicity of the approach reported in [14], [21] is based on finite concepts, such as maximal length of the log trace, while we propose a more general approach capable of assessing infinite behaviors described in models. Finally, note that in contrast to the anti-alignment precision, using both our precision measures, i.e., *EB* and *EB*$^\tau$, allows for a more complete analysis by differentiating between *Flower* model and *Round robin* model.

[1] https://github.com/akalenkova/eigen-measure.

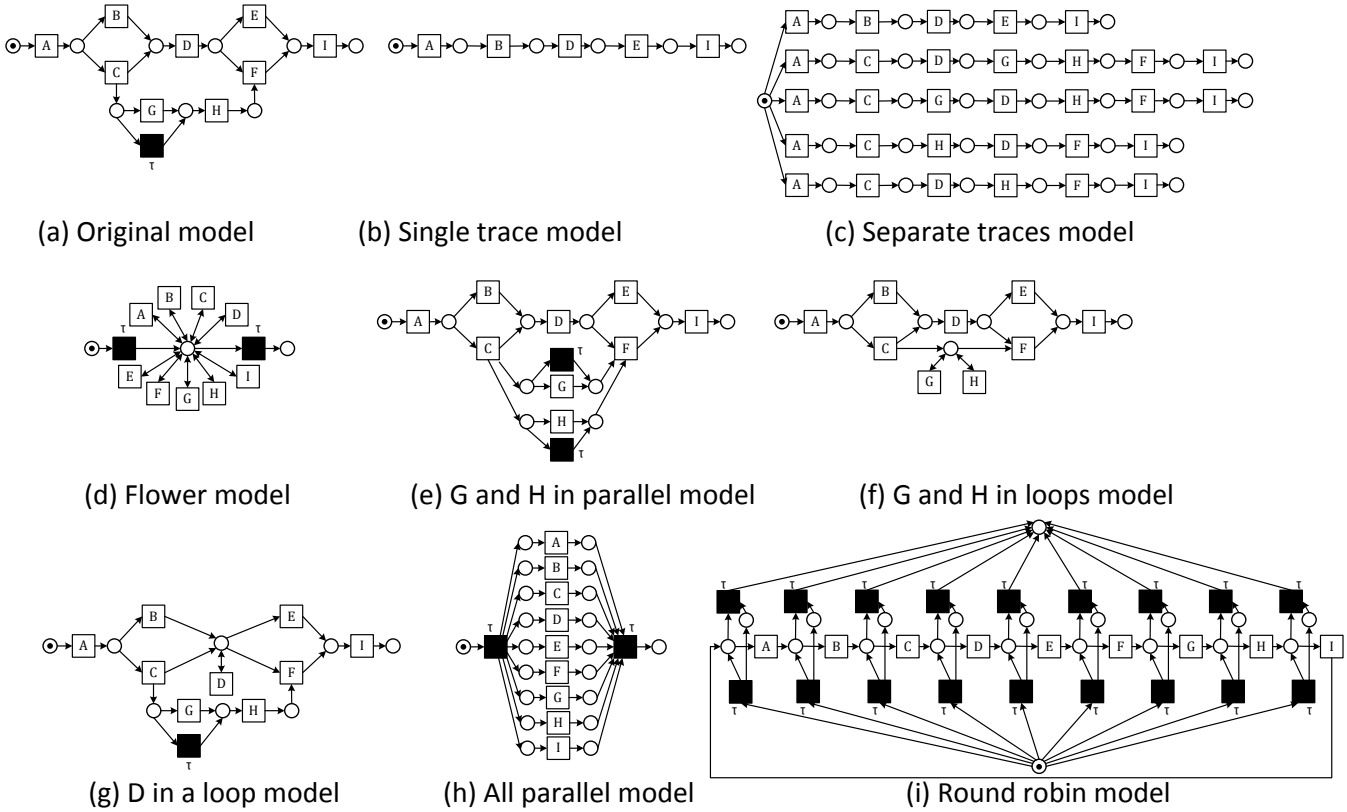

Figure 6: Artificial process models from [14], [15].

Table IV: Rankings of precision values for synthetic log.

Model	SD	ETC_a	NE	PCC	AA	MAP^1	$MAP^{2\text{-}7}$	EB	EB^τ
$Original$	7	7	9	8	7	7	7	7	7
$Single\ trace$	8	8	6	8	8	7	8	8	8
$Separated\ traces$	8	8	8	7	8	7	8	8	8
$Flower$	1	1	1	1	1	1	1	3	1
$H\ and\ G\ in\ parallel$	6	3	7	6	6	5	6	6	6
$H\ and\ G\ in\ loops$	1	5	5	4	5	3	3	4	4
$D\ in\ a\ loop$	1	6	4	5	4	5	4	5	5
$All\ parallel$	1	2	2	2	3	2	2	1	3
$Round\ robin$	1	4	3	3	1	4	5	1	1

B. Real-life Event Data

Next, we investigate the scalability of our approach to verify whether it can be applied to real-life event logs. We have analyzed BPI Challenge (BPIC) event logs[2], which are publicly available logs of real-life IT systems, and an event log of a booking flight system (BFS). Prior to the analysis, we filtered out infrequent events that appear less than in 80% of traces using *Filter Log using Simple Heuristics* Process Mining Framework (ProM) [22] plugin.[3] For the filtered event logs, the number of traces in a single log varies from 596 to 11,636 and the overall length of all the traces in a single event log varies from 1,403 to 164,144. From each event log, a Petri net was discovered using the Inductive miner [3]. This discovery technique constructs bounded Petri nets, such that their reachability graphs are DFAs. We used these DFAs as representations of model behaviors to apply the partial matching technique described in this paper. Since the entropy-based approach is applicable to τ-free DFAs only, the DFAs with silent transitions were converted to equivalent τ-free DFAs. It was crucial to estimate the applicability of our approach in order to show that despite the potential state space explosion, it is still computationally feasible to construct an equivalent τ-free DFA from an initial DFA with silent transitions [13] obtained via the τ-closure operation. In our experiments, the number of automaton states increases by no more than 10 times. The results of experiments, including the sizes of the automatons, time (in seconds) taken for the determinization of automata, entropy and precision and recall values calculation, are presented in Table V. In these experiments, we used Intel Core(TM) i7-3970X CPU @3.50 GHz with 64 GB RAM.

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

[3]All the filtered logs, including the BFS log, are distributed with the implementation at https://github.com/akalenkova/eigen-measure.

VII. Conclusion

In this paper, we proposed an extension of our entropy-based precision and recall conformance measures, proposed

Table V: Time of determinization and entropy calculation for real-life event logs (in seconds).

Event log	Automaton	# States / # Transitions	Determiniz. time	Entropy calc. time	Precision / Recall
BPIC'12	L'	90,557 / 446,847	141.455	4,641.421	
	$M' \cap L'$	90,557 / 446,847	-	4,733.990	0.709 /
	M'	3 / 33	0.001	0.013	1.000
BPIC'13 closed	L'	216 / 629	0.171	0.235	
	$M' \cap L'$	216 / 629	-	0.661	0.960 /
	M'	1 / 3	0.001	0.010	1.000
BPIC'13 incidents	L'	24,336 / 72,994	1,909.794	2.187	
	$M' \cap L'$	24,336 / 72,994	-	1.552	0.995 /
	M'	1 / 3	0.001	0.011	1.000
BPIC'13 open	L'	17 / 31	0.012	0.123	
	$M' \cap L'$	17 / 31	-	0.003	0.980 /
	M'	1 / 2	0.000	0.011	1.000
BFS'13	L'	22,359 / 200,254	37.947	2.427	
	$M' \cap L'$	7,542 / 45,163	-	1.516	0.939 /
	M'	514 / 3,340	1.625	0.153	0.825

in [8], which quantify the similarity between traces described in a designed process model and its corresponding executed traces recorded in an event log. While precision quantifies how well the traces of the model are represented in the log, recall measures how well the traces in the log are represented in the model. The extension addresses the phenomenon of partially matching traces, i.e., non-identical traces that nevertheless describe similar sequences of process steps. The extended measures exhibit the intuitively desired property which establishes that more partially matching traces with matches of greater length in the compared model and log lead to greater precision and recall values. In [8], we showed that the entropy-based precision and recall fulfill a range of important properties of conformance measures that are not fulfilled by the state of the art measures. The extended measures inherit all those properties of the original measures, while all the properties of the extended measures demonstrated in this paper also apply to the original measures.

The reported qualitative and quantitative evaluation of the proposed measures suggests that they can be useful in industrial settings. We acknowledge that the extended measures, still, have several limitations, which naturally give rise to future work. Firstly, the proposed measures do not support systems which cannot be described as DFAs, for example infinite-state systems. To address this limitation, coverability graphs may yield useful. The proposed extension based on the dilution of the model and log traces has proven effective, but also demonstrated a significant negative impact on the efficiency of the measures. To cope with this deficiency, secondly, we plan to exploit the properties of topological entropy and structural features of log automata to simplify certain "expensive", in terms of time, computations. Thirdly, the proposed approach considers distinct traces only and can be extended to take into account frequencies of traces in event logs. Finally, even though the state of the art conformance measures do not satisfy the properties for completely matching traces, refer to [6], [10] for details, they may satisfy the properties for partially matching traces. Therefore, one can verify if other measures satisfy the properties put forward in this work. Also, we envision that the discussion on the desired properties for conformance measures will continue within the process mining community.

Acknowledgment

Artem Polyvyanyy was partly supported by the Australian Research Council Discovery Project DP180102839. Anna Kalenkova was supported by the Basic Research Program at the Higher School of Economics.

References

[1] W. van der Aalst, *Process Mining: Data Science in Action*, 2016.
[2] W. van der Aalst, A. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.
[3] S. Leemans, D. Fahland, and W. van der Aalst, "Discovering Block-Structured Process Models from Incomplete Event Logs," in *ATPN'2014*, ser. LNCS. Springer, 2014, vol. 8489, pp. 91–110.
[4] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *KAIS*, pp. 1–34, 2018.
[5] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.
[6] A. Polyvyanyy, W. van der Aalst, A. ter Hofstede, and M. Wynn, "Impact-driven process model repair," *ACM Transactions on Software Engineering and Methodology*, vol. 25, no. 4, pp. 1–60, 2017.
[7] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking—Relating Processes and Models*. Springer, 2018.
[8] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *CoRR*, vol. abs/1812.07334, 2018.
[9] W. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*. NJ, USA: Prentice-Hall, Inc., 1992.
[10] N. Tax, X. Lu, N. Sidorova, D. Fahland, and W. van der Aalst, "The imprecisions of precision measures in process mining," *Information Processing Letters*, vol. 135, pp. 1–8, 2018.
[11] W. van der Aalst, "Relating process models and event logs—21 conformance propositions," in *Proceedings of ATAED satellite event for ATPN 2018*, ser. CEUR Workshop Proceedings, vol. 2115, 2018, pp. 56–74.
[12] T. Ceccherini-Silberstein, A. Machì, and F. Scarabotti, "On the entropy of regular languages," *Theor. Comp. Sci.*, vol. 307, pp. 93–102, 2003.
[13] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*, Boston, USA, 2006.
[14] B. van Dongen, J. Carmona, and T. Chatain, "A unified approach for measuring precision and generalization based on anti-alignments," in *Business Process Management*. Cham: Springer, 2016, pp. 39–56.
[15] A. Augusto, A. Armas-Cervantes, R. Conforti, M. Dumas, M. La Rosa, and D. Reissner, "Abstract-and-compare: A family of scalable precision measures for automated process discovery," in *Business Process Management*. Cham: Springer International Publishing, 2018, pp. 158–175.
[16] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, "Discovering expressive process models by clustering log traces," *IEEE Trans. on Knowl. and Data Eng.*, vol. 18, no. 8, pp. 1010–1027, 2006.
[17] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A robust f-measure for evaluating discovered process models," in *CIDM*. IEEE, 2011, pp. 148–155.
[18] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 211–226.
[19] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst, "Measuring precision of modeled behavior," *Inf. Syst. and e-Business Management*, vol. 13, no. 1, pp. 37–67, 2015.
[20] S. Leemans, D. Fahland, and W. van der Aalst, "Scalable process discovery and conformance checking," *Software & Systems Modeling*, vol. 17, no. 2, pp. 599–631, 2018.
[21] T. Chatain and J. Carmona, "Anti-alignments in conformance checking – the dark side of process models," in *ATPN'2016*, pp. 240–258.
[22] B. van Dongen, A. de Medeiros, H. Verbeek, A. Weijters, and W. van der Aalst, "The ProM Framework: A new era in process mining tool support," in *ATPN'2005*, pp. 444–454.