Mediating Between Modeled and Observed Behavior: The Quest for the "Right" Process

Wil M.P. van der Aalst
Department of Mathematics and Computer Science
Eindhoven University of Technology
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
WWW: vdaalst.com

***Abstract*—Operational processes leave trails in the information systems supporting them. Such event data are the starting point for process mining – an emerging scientific discipline relating modeled and observed behavior. Whereas an event log describes example behavior of the underlying process, a process model aims to describe an abstraction of the same process. Models may be descriptive or normative. Descriptive models aim to describe the underlying process and are used for discussion, performance analysis, obtaining insights, and prediction. Normative models describe the desired behavior and are used for workflow management, system configuration, auditing, compliance management, and conformance checking. Differences between modeled and observed behavior may point to undesirable deviations or inadequate models. In this paper, we discuss challenges related to finding the "right" process, i.e., the process model that describes the real underlying process or a process that behaves as desired.**

I. Introduction

Recently, *process mining* emerged as a new scientific discipline on the interface between process models and event data [1]. Conventional *Business Process Management* (BPM) [2] and *Workflow Management* (WfM) [3] approaches and tools are mostly model-driven with little consideration for event data. *Data Mining* (DM) [4], *Business Intelligence* (BI), and *Machine Learning* (ML) [5] focus on data without considering end-to-end process models. Process mining aims to bridge the gap between BPM and WfM on the one hand and DM, BI, and ML on the other hand (cf. Figure 1).

The practical relevance of process mining is increasing as more and more event data become available (cf. the recent attention for "Big Data"). Process mining techniques aim to *discover, monitor and improve real processes by extracting knowledge from event logs*. The two most prominent process mining tasks are: (i) *process discovery*: learning a process model from example behavior recorded in an event log, and (ii) *conformance checking*: diagnosing and quantifying discrepancies between observed behavior and modeled behavior.

Starting point for any process mining task is an *event log*. Each *event* in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a *process instance*). The events belonging to a case are ordered, and can be seen as one "run" of the process. Such a run is often referred to as a *trace*. It is important to note that an event log contains only example behavior, i.e., we cannot assume that all possible runs have been observed.

Given a process model (discovered or made by hand) and an event log one can try to *align modeled and observed behavior*. An alignment relates a trace in a event log to its corresponding path in the model. If there is not a direct match, the trace is aligned with the closest or most likely path. Such alignments can be used to answer *performance-oriented* and *compliance-oriented* questions (cf. Figure 1). Alignments can be used to show how often paths are taken and activities are being executed. Moreover, events often bear a timestamp which can be used to compute flow times, waiting times, service times, etc. For example, alignments can be used to highlight bottlenecks in the process model. Similarly, alignments can be used to show where model and event log disagree. This is commonly referred to as *conformance checking*.

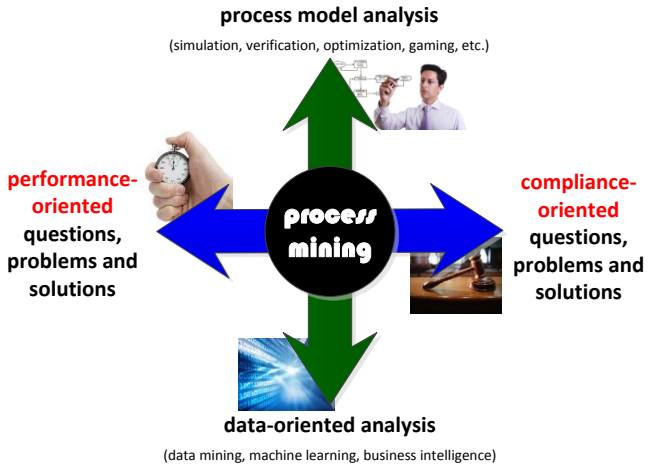

Fig. 1. Process mining is on the interface between process model analysis and data-oriented analysis and can be used to answer a variety of performance and compliance-related questions.

This paper focuses on the *relation between modeled and observed behavior* and considers two use cases: (i) evaluating the results of process discovery ("How good is the model I discovered?") and (ii) comparing discrepancies between some normative or descriptive model and an event log ("Where do model and log disagree?"). The latter question is relevant for compliance-related questions (e.g., auditing). Also note that discrepancies between model and event log may point to *problems in the model* (e.g., the model is a poor representation of reality and as such has no predictive power) or *problems in the process* described by the model (e.g., people deviate from the desired process or cases are handled too late). The second

part of the title of this paper – The Quest for the "Right" Process – refers to both types of problems.

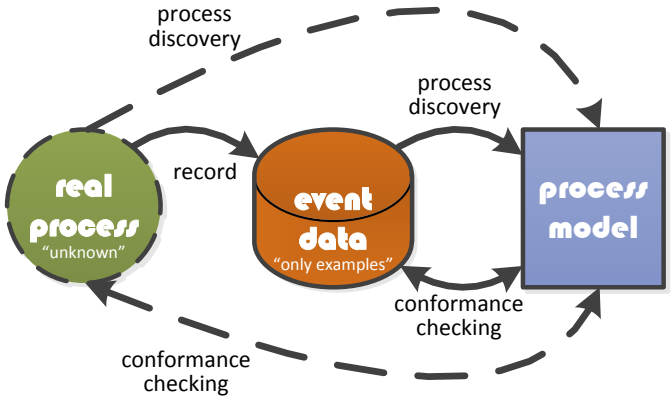

Fig. 2. Process discovery and conformance checking relate a descriptive or normative process model to an *unknown* process using event data.

Unlike existing approaches, we initially assume that the "real process" is known to better understand the quality of process discovery and conformance checking. Figure 2 provides an overview of the problem investigated in this paper. The *real process* is generally unknown. We can only see the *event data* generated by this process (arrow *record*). These event data can be used to discover a *process model* (arrow *process discovery*). Moreover, the process model (discovered or man-made) and the event log can be compared as illustrated by the solid double-headed arrow *conformance checking*. However, both process discovery and conformance checking aim to tell something about the *unknown* real process rather than the example traces in the event log (see the two dashed arrows in Figure 2).

In the remainder we investigate *the relationship between the real process, the event log, and a given process model*. Section II reviews existing approaches and literature related to conformance checking. Section III provides formal definitions for processes, process models, and event logs. Using these definitions and assuming that the real process is known, we discuss various *precision* and *recall* notions in Section IV. In Section V the assumption that the real process is known is dropped and event data are used to "guess" the real process. Section VI reflects on the findings in the previous sections and discusses additional challenges. Related work is considered in Section VII. Section VIII concludes the paper.

II. Existing Conformance Checking Approaches

To introduce conformance checking and to describe some of the existing approaches, we use the process model shown in Figure 3. A *labeled Petri net* is used to represent the process. A Petri net is a bipartite graph consisting of places and transitions. Transitions are the active components and places are the passive components. Places may contain tokens. The network structure does not change, but transitions may consume and produce tokens. A transition may have a label describing the corresponding activity. For example, transition $t1$ models activity "register request" having short name a and $t2$ models the skipping of activity "examine file" modeled by transition $t3$. Transitions without a label, e.g., $t2$, are invisible and do not correspond to actual activities.

The process starts with a token in place *start* and ends with a token in place *end*. Figure 3 allows for traces such as $\langle a, c, b, d, f, g\rangle$, $\langle a, c, d, h\rangle$, and $\langle a, b, c, d, e, c, d, g, f\rangle$ (because of the loop there are infinitely many possible traces). The notation itself is not relevant: any other notation could have been used (e.g., BPMN, UML activity diagrams, EPCs, BPEL, etc.).

An *event log* is a multiset of traces. Each trace is a sequence of activities. Multiple cases may follow the same trace. An example log is $L_1 = [\langle a, c, d, f, g\rangle^{10}, \langle a, c, d, h\rangle^5, \langle a, b, c, d, e, c, d, g, f\rangle^5]$. L_1 contains information about 20 cases, e.g., 10 cases followed trace $\langle a, c, d, f, g\rangle$. Figure 3 shows only three of these 20 cases. There are $10 \times 5 + 5 \times 4 + 5 \times 9 = 115$ events in total. $L_2 = [\langle a, c, d, f\rangle^{10}, \langle a, c, d, c, h\rangle^5, \langle a, b, d, e, c, d, g, f, h\rangle^5]$ is another event log also containing 20 cases.

Conformance checking techniques investigate how well an event log L and a process model M fit together. Let M be the labeled Petri net in Figure 3. Clearly, L_1 is perfectly fitting M whereas L_2 is not.

A. Four Quality Dimensions for Comparing Model and Log

There are four quality dimensions for comparing model and log: (1) *fitness*, (2) *simplicity*, (3) *precision*, and (4) *generalization* [1]. A model with good *fitness* allows for most of the behavior seen in the event log. A model has a perfect fitness if all traces in the log can be replayed by the model from beginning to end. The *simplest* model that can explain the behavior seen in the log is the best model. This principle is known as *Occam's Razor*. Fitness and simplicity alone are not sufficient to judge the quality of a discovered process model. For example, it is very easy to construct an extremely simple Petri net ("flower model") that is able to replay all traces in an event log (but also any other event log referring to the same set of activities). Similarly, it is undesirable to have a model that only allows for the exact behavior seen in the event log. Remember that the log contains only example behavior and that many traces that are possible may not have been seen yet. A model is *precise* if it does not allow for "too much" behavior. Clearly, the "flower model" lacks precision. A model that is not precise is "underfitting". Underfitting is the problem that the model over-generalizes the example behavior in the log (i.e., the model allows for behaviors very different from what was seen in the log). At the same time, the model should generalize and not restrict behavior to just the examples seen in the log. A model that does not *generalize* is "overfitting". Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior (i.e., the model explains the particular sample log, but there is a high probability that the model is unable to explain the next batch of cases).

Figure 4 shows that the four quality dimensions may be competing. Just like the four forces enabling a airplane to fly (lift, gravity, drag, and thrust), one needs to balance fitness, simplicity, precision, and generalization when discovering process models from event logs.

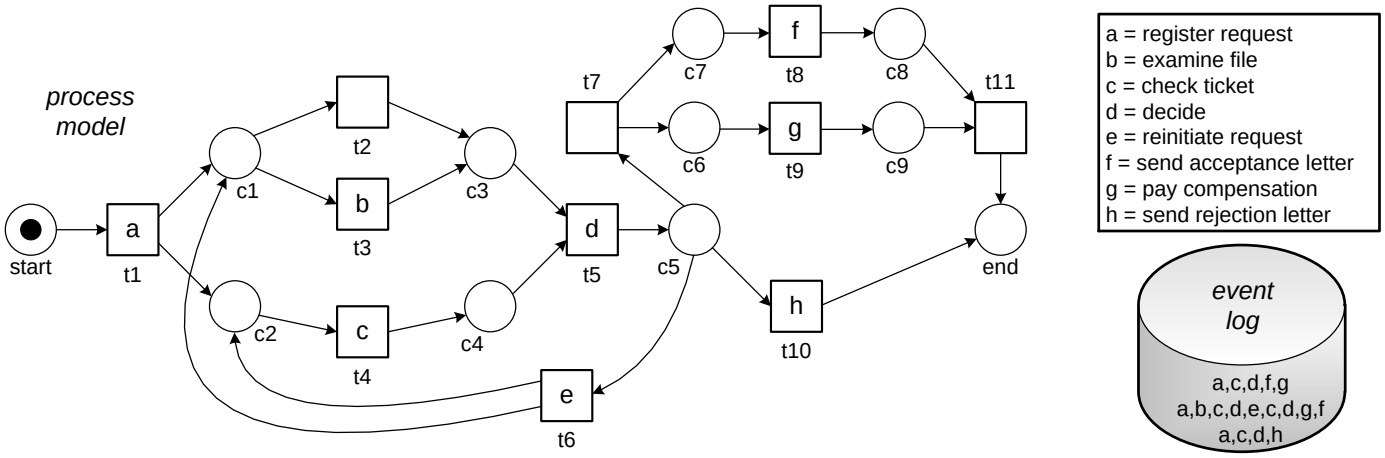

Fig. 3. A process model represented using a labeled Petri net; the event log shows three example traces that could have been generated by this process.

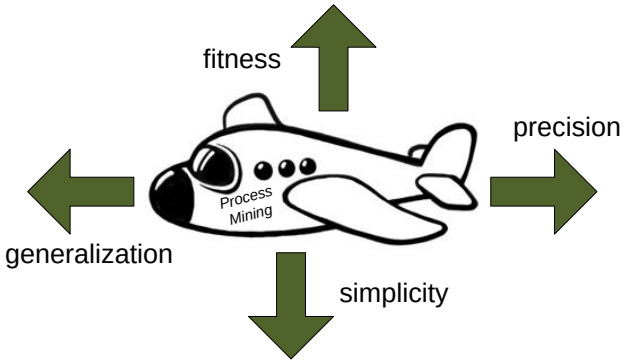

Fig. 4. The four forces of process mining: lift = *fitness* (ability to explain observed behavior), gravity = *simplicity* (Occam's Razor), drag = *precision* (avoiding underfitting), and thrust = *generalization* (avoiding overfitting).

Many fitness notions have been defined in literature [1], [6], [7], [8], [9], [10], [11], [12]. Independent of the notion used, all approaches will indicate that L_1 is perfectly fitting M whereas L_2 is clearly not. Precision can be quantified by counting "escaping edges" [13], [10], [11]. Also notions for simplicity and generalization have been defined [1], [6], [14]. The problem is that many different conformance metrics can be defined. These metrics all aim to capture an intuitive notion and struggle with the problem that the real process is unknown.

B. Token Based Replay

A naïve approach towards conformance checking would be to simply count the fraction of cases that can be "parsed completely". In terms of Figure 3, this would be the fraction of cases corresponding to firing sequences leading from the state with a token in *start* to the state with a token in *end*. Let M be the process model in Figure 3 and L_1 and L_2 as defined before. 100% of the cases in L_1 are fitting M whereas 0% of the cases in L_2 are fitting M. However, the latter example shows that such a naïve approach is unable to distinguish between an almost fitting trace and a trace that has little in common with any path in the model.

Therefore, more sophisticated approaches have been developed. A good example is the token-based replay approach that counts produced p, consumed c, missing m, and remaining r tokens [1], [12]. Consider the perfectly fitting trace $\sigma_1 = \langle a, c, d, f, g \rangle$ and the labeled Petri net in Figure 3. Initially, all four counters are set to zero: $p = c = m = r = 0$. Then the environment produces token for place *start* to initialize the process: $p = 1$. When firing transition $t1$ for the first activity in σ (a), one token is consumed and two tokens are produced: $p = 1 + 2 = 3$ and $c = 0 + 1 = 1$. When executing $t4$ for the second activity in σ (c), one token is consumed and one token is produced: $p = 3 + 1 = 4$ and $c = 1 + 1 = 2$. To execute the third activity in σ (d), we first need to execute $t2$: $p = 4{+}1 = 5$ and $c = 2{+}1 = 3$. Then we can execute $t5$ which consumes two tokens and produces one token: $p = 5 + 1 = 6$ and $c = 3 + 2 = 5$. Etc. After replaying the entire trace, the environment consumes the token from place *end* to close the process. After this the total number of produced tokens is $p = 11$ and the total number of consumed tokens is $c = 11$. Clearly, there are no problems when replaying the σ_1, i.e., there are no missing or remaining tokens ($m = r = 0$).

The fitness of a case with trace σ is defined as follows [12]:

$$\mathit{fitness}(\sigma, M) = \frac{1}{2}\left(1 - \frac{m}{c}\right) \; + \; \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

The first parts computes the fraction of missing tokens relative to the number of consumed tokens. $1 - \frac{m}{c} = 1$ if there are no missing tokens ($m = 0$) and $1 - \frac{m}{c} = 0$ if all tokens to be consumed were missing ($m = c$). Similarly, $1 - \frac{r}{p} = 1$ if there are no remaining tokens and $1 - \frac{r}{p} = 0$ if none of the produced tokens was actually consumed. $\mathit{fitness}(\sigma_1, M) = 1$ because there are no missing or remaining tokens.

Let us now consider a trace that cannot be replayed properly: $\sigma_2 = \langle a, b, d, h \rangle$. The first two activities can be executed without any problems; fire $t1$ and $t2$. When trying to execute $t5$ for the third activity in σ_2 (d), we encounter a problem because place $c4$ is empty. We record the missing token $m = 1$ and continue. At the end a token remains in place $c2$. Hence, $r = 1$. After replaying σ_2 completely: $p = 6$, $c = 6$, $m = 1$, and $r = 1$. Hence, $\mathit{fitness}(\sigma_2, M) = \frac{1}{2}(1 - \frac{1}{6}) + \frac{1}{2}(1 - \frac{1}{6}) \approx 0.83$.

The same approach can be used to compute the fitness of an event log consisting of many cases: $\mathit{fitness}(L, M)$. Simply take the sums of all produced, consumed, missing, and remaining tokens, and apply the same formula. If multiple cases follow the same trace, the produced, consumed, missing, and remaining tokens should be multiplied accordingly.

C. Aligning Event Log and Process Model

Token-based replay has problems when dealing with more complex process models having duplicate and silent activities and event logs with long traces. The results may be misleading and there is no explicit relation between model and log. *Alignments* were introduced to tackle these problems [6], [7], [13], [15], [16]. Consider the following three alignments for the traces in L_1 and model M in Figure 3:

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline a & c & \gg & d & \gg & f & g & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array}$$

$$\gamma_2 = \begin{array}{|c|c|c|c|c|} \hline a & c & \gg & d & h \\ \hline a & c & \tau & d & h \\ t1 & t4 & t2 & t5 & t10 \\ \hline \end{array}$$

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & c & d & e & c & \gg & d & \gg & g & f & \gg \\ \hline a & b & c & d & e & c & \tau & d & \tau & g & f & \tau \\ t1 & t3 & t4 & t5 & t6 & t4 & t2 & t5 & t7 & t9 & t8 & t11 \\ \hline \end{array}$$

The top row of each alignment corresponds to "moves in the log" and the bottom two rows correspond to "moves in the model". Moves in the model are represented by the transition and its label. This is needed because there could be multiple transitions having the same label. If a move in the model cannot be mimicked by a move in the log, then a "$\gg$" ("no move") appears in the top row. For example, in the third position of γ_1 the log cannot mimic the invisible transition $t2$. The τ above $t2$ indicates that $t2$ does not correspond to a visible activity. Note that all "no moves" (i.e., the seven $\gg$ symbols) in $\gamma_1 - \gamma_3$ are "caused" by invisible transitions.

Two example alignments for L_2 and process model M in Figure 3:

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline a & c & \gg & d & \gg & f & \gg & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array}$$

$$\gamma_5 = \begin{array}{|c|c|c|c|c|c|} \hline a & c & \gg & d & c & h \\ \hline a & c & \tau & d & \gg & h \\ t1 & t4 & t2 & t5 & & t10 \\ \hline \end{array}$$

Alignment γ_4 shows a "$\gg$" ("no move") in the top row that does not correspond to an invisible transition. The model makes a g move (occurrence of transition $t9$) that is not in the log. If a move in the log cannot be mimicked by a move in the model, then a "$\gg$" ("no move") appears in the bottom row. For example, in γ_5 the second c move in the log is not mimicked by a move in the model. Note that the "no moves" not corresponding to invisible transitions point to deviations between model and log.

A *move* is a pair $(x, (y, t))$ where the first element refers to the log and the second element refers to the model. For example, $(a, (a, t1))$ means that both log and model make an "a move" and the move in the model is caused by the occurrence of transition $t1$. $(\gg, (g, t9))$ means that the occurrence of transition $t9$ with label g is not mimicked by a corresponding move of the log. $(c, \gg)$ means that the log makes an "c move" not followed by the model.

An *alignment* is a sequence of legal moves such that after removing all $\gg$ symbols, the top row corresponds to a trace in the log and the bottom row corresponds to a firing sequence starting in the initial state (token in *start*) and ending in the final state (token in *end*). γ_1–γ_3 are examples of alignments for the traces in L_1 and their corresponding firing sequences in Figure 3. γ_4 and γ_5 are examples of alignments for the first two traces in L_2 and complete firing sequences of the same process model.

Given a log trace and a process model there may be many (if not infinitely many) alignments. Consider the following two alignments for $\langle a, c, d, f \rangle \in L_2$:

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline a & c & \gg & d & \gg & f & \gg & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array}$$

$$\gamma_4' = \begin{array}{|c|c|c|c|c|c|c|} \hline a & c & \gg & d & \gg & f & \gg \\ \hline a & c & b & d & \tau & \gg & h \\ t1 & t4 & t3 & t5 & t7 & & t10 \\ \hline \end{array}$$

γ_4 seems to be better alignment than γ_4' because it has only one deviation (move in model only; $(\gg, (g, t9))$) whereas γ_4' has three deviations: $(\gg, (b, t3))$, $(f, \gg)$, and $(\gg, (h, t10))$. To select the most appropriate one, we associate *costs* to undesirable moves and select an alignment with the lowest total costs. To quantify the costs of misalignments we introduce a cost function δ. Moves where log and model agree have no costs, i.e., $\delta(x, (x, t)) = 0$ for all $x \in A$. Moves in model only have no costs if the transition is invisible, i.e., $\delta(\gg, (\tau, t)) = 0$. $\delta(\gg, (x, t)) > 0$ is the cost when the model makes an "x move" without a corresponding move of the log. $\delta(x, \gg) > 0$ is the cost for an "x move" in just the log. These costs may depend on the nature of the activity, e.g., skipping a payment may be more severe than sending too many letters. The cost function can also be used to assign a penalty to events that are executed too late or by the wrong person [17].

An *optimal alignment* has the lowest possible costs, i.e., the observed trace is related to a path in the model that is "closest" in terms of the cost function δ. If the optimal alignment has non-zero costs, there was a deviation. The total cost of an optimal alignment quantifies non-conformance. It is possible to convert such misalignment costs into a fitness values between 0 (poor fitness, i.e., maximal costs) and 1 (perfect fitness, zero costs). This can be done at the level of a single trace or at the level of an entire event log. We refer to [6], [7] for details.

The notion of an optimal alignment can be used to create a so-called *oracle*. Given a full or partial trace, the oracle returns the most likely path in the model (or a set of paths). The resulting aligned path(s) can be used to evaluate the full or partial trace, e.g., to compute costs, time, risks, energy consumption, etc. Hence, there may be different cost functions to evaluate paths returned by the oracle. In fact, the cost functions and oracle can also be used to make predictions about the completion of cases.

Once the oracle established an optimal alignment for every trace in the event log, these alignments can also be used as a basis to quantify other conformance notations such as precision and generalization [6]. For example, precision can be computed by counting "escaping edges" as shown in [13], [10], [11]. Alignments can also be used to project additional information extracted from event logs onto the model. For example, timestamps in the event log can be used to identify bottlenecks in the process and to predict delays in running processes [1], [18], [19].

D. Problem

The approaches based on replay and alignments illustrate that there are various ways to quantify fitness. Literature identifies four quality dimensions for comparing model and log [1], [6], [14], [20]. Unfortunately, these do not seem orthogonal and for each of the four dimensions different metrics can be defined [21]. Even for an intuitively clear notion such as replay fitness various metrics are possible [1], [6], [7], [8], [9], [10], [11], [12], [20], [22]. Therefore, we take a step back and reconsider the relation between modeled and observed behavior. In order to do this we first assume that we know the real process and use the concepts *precision* (the fraction of retrieved items that are indeed relevant) and *recall* (the fraction of relevant items that are indeed retrieved) from information retrieval.

III. Processes, Event Logs and Models

In this section, we formalize the basic concepts used in the remainder. Process models, processes, and event logs share common notions such as *activity* and *trace* (sequence of activities).

Definition 1 (Universe of Activities, Universe of Traces): $\mathcal{A}$ is the universe of *activities*, i.e., the set of all possible and relevant activities. Other activities cannot be observed (or are abstracted from). Elements of $\mathcal{A}$ may have *attributes*, e.g., costs, resource information, duration information, etc. A *trace* $\sigma \in \mathcal{A}^*$ is a sequence of activities found in an event log or corresponding to a run of some process model. $\mathcal{U} = \mathcal{A}^*$ is the universe of traces.

We assume that an activity is identified by attributes relevant for learning, i.e., irrelevant attributes are removed and attribute values may be coarsened. $|\mathcal{A}|$ is the number of unique activities. Process models with hundreds of activities (or more) tend to be unreadable. In the remainder we will refer to activities using a single letter (e.g. a), however, an activity could also be $decide(gold, manager, reject)$ to represent a decision to reject a gold customer's request by a manager.

In a process a trace $\sigma \in \mathcal{U}$ has a likelihood $\pi(\sigma)$. We assume the process is in steady state and that cases do not influence each other.

Definition 2 (Process): A *process* is a discrete probability function $\pi \in \mathcal{U} \rightarrow [0, 1]$ which assigns a probability $\pi(\sigma)$ to any trace $\sigma \in \mathcal{U}$.

Note that $\sum_{\sigma \in \mathcal{U}} \pi(\sigma) = 1$. An event log is similar to a process, but each trace has a frequency $L(\sigma)$ rather than a likelihood $\pi(\sigma)$. Formally, an event log is a multiset of *sample* traces from a known or unknown process.

Definition 3 (Event Log): An *event log* $L \in \mathbb{B}(\mathcal{U})$ is a multiset of observed traces.

The same trace may appear multiple times. For example in event log $L_1 = [\langle a, c, d, f, g\rangle^{10}, \langle a, c, d, h\rangle^5, \langle a, b, c, d, e, c, d, g, f\rangle^5]$, trace $\langle a, c, d, f, g\rangle$ appears 10 times.

$\pi(X) = \sum_{\sigma \in X} \pi(\sigma)$ for a set or multiset of traces X (counting elements multiple times if needed). For example, $\pi(L_1) = 10 \times \pi(\langle a, c, d, f, g\rangle) + 5 \times \pi(\langle a, c, d, h\rangle) + 5 \times \pi(\langle a, b, c, d, e, c, d, g, f\rangle)$.

Like any process model, the labeled Petri net in Figure 3 defines a set of possible traces. In this paper, a process model is defined as a set of traces, i.e., we abstract from the concrete notation.

Definition 4 (Process Model): A *process model* is a set of traces $M \subseteq \mathcal{U}$.

A process model splits the universe of traces $\mathcal{U}$ into two classes: M (all traces possible according to the model) and $\overline{M}$ (all traces impossible according to the model).

Definition 5 (Complement): $\overline{M} = \mathcal{U} \setminus M$ is the complement of some model M.

In Figure 2 we used the terms real process, event data, and process model. The *real process* is represented by process $\pi_0 \in \mathcal{U} \rightarrow [0, 1]$. The *event data* are represented by event log $L \in \mathbb{B}(\mathcal{U})$. The *process model* obtained through process discovery or used for conformance checking is represented by $M_1 \subseteq \mathcal{U}$.

Let us assume that we also have $M_0 \subseteq \mathcal{U}$ as the "ideal" or "desired" model derived from the real process π_0. For example, M_0 could be based on π_0 and some predefined threshold. Consider the following two exemplary model derivations:

- $M_0 = \{\sigma \in \mathcal{U} \mid \pi_0(\sigma) \geq \epsilon\}$ for some trace probability threshold $\epsilon \in [0, 1]$.
- $M_0 \subseteq \mathcal{U}$ such that M_0 is the smallest set that satisfies: $\pi_0(M_0) \geq \tau$ and $\forall_{\sigma \in M_0} \forall_{\sigma' \in \overline{M_0}} \pi_0(\sigma) > \pi_0(\sigma')$ for some threshold $\tau \in [0, 1]$.

The latter way of computing M_0 can be explained intuitively, e.g., if $\tau = 0.8$, then M_0 is the so-called "80% model" covering at least 80% of the process behavior with a preference for more likely traces. The $80-20$ rule (also known as the Pareto principle) informally states that often 80% of the behavior can be explained by 20% of the most frequent traces.

From a more philosophical point of view one could argue that anything is possible (just wait long enough and it will happen): Murphy's law for process mining. Taking this viewpoint, it is not interesting to look for an M_0 with $\pi_0(M_0) = 1$, because $M_0 = \mathcal{U}$ as a consequence (i.e., the model does not contain any information).

In the remainder, $\pi_1 \in \mathcal{U} \rightarrow [0, 1]$ defines the probability of a trace according to the model. In a stochastic process model (e.g., a generalized stochastic Petri net or Markov chain [23]) such probabilities are defined explicitly. If the model does not contain such information, π_1 is estimated (e.g., assume equal probabilities for all choices or use domain knowledge and/or historic information).

The following table summarizes the conventions used in the remainder:

	process (probability)	model (selection)
real process ("ideal" or "desired")	$\pi_0 \in \mathcal{U} \rightarrow [0,1]$	$M_0 \subseteq \mathcal{U}$
descriptive or normative process	$\pi_1 \in \mathcal{U} \rightarrow [0,1]$	$M_1 \subseteq \mathcal{U}$

IV. Precision and Recall of Models Assuming an A Priori Distribution

In reality we do not know the real process π_0 and only see event log L as a reflection of the actual process. However, assuming that π_0, M_0, π_1, and M_1 are known, we can use standard information retrieval notions such as *precision* and *recall*.[1] To illustrate this consider Figure 5. The area TP (True Positives) corresponds to all traces that are possible according to both the ideal/desired model M_0 and the descriptive/normative model M_1. The area FN (False Negatives) corresponds to all traces that are not possible according to the descriptive/normative model M_1, but that are possible according to the ideal/desired model M_0. TN (True Negatives) and FP (False Positives) are defined in a similar fashion. TP, FN, TN, and FP can be used to define precision and recall. However, since there are infinitely many possible traces in $\mathcal{U}$ and also models may describe infinitely many traces, we cannot simply count the number of traces in the four classes. Therefore, we assign weights to traces based on their likelihood. However, there are two ways to determine the likelihood: based on π_0 (real likelihood) or π_1 (modeled/estimated likelihood). Therefore, there are multiple precision and recall notions.

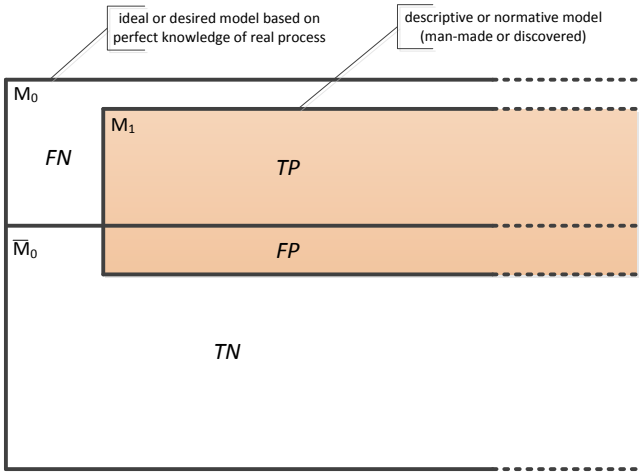

Fig. 5. Comparing M_0 and M_1: $TP = M_0 \cap M_1$, $TN = \overline{M}_0 \cap \overline{M}_1$, $FP = \overline{M}_0 \cap M_1$, and $FN = M_0 \cap \overline{M}_1$.

Definition 6 (Precision and Recall): Let $\pi_0 \in \mathcal{U} \rightarrow [0,1]$ be the real process having $M_0 \subseteq \mathcal{U}$ as its "ideal" or "desired" model. Let $M_1 \subseteq \mathcal{U}$ be the "descriptive" or "normative" model, and (optionally) $\pi_1 \in \mathcal{U} \rightarrow [0,1]$ a function determining the likelihood of traces according to the model. $TP = M_0 \cap M_1$, $TN = \overline{M}_0 \cap \overline{M}_1$, $FP = \overline{M}_0 \cap M_1$, and $FN = M_0 \cap \overline{M}_1$. Precision and recall are defined as follows:

$$precision_{\pi_0,M_0,M_1} = \frac{\pi_0(TP)}{\pi_0(M_1)} = \frac{\pi_0(TP)}{\pi_0(TP) + \pi_0(FP)}$$

$$recall_{\pi_0,M_0,M_1} = \frac{\pi_0(TP)}{\pi_0(M_0)} = \frac{\pi_0(TP)}{\pi_0(TP) + \pi_0(FN)}$$

$$precision_{\pi_1,M_0,M_1} = \frac{\pi_1(TP)}{\pi_1(M_1)} = \frac{\pi_1(TP)}{\pi_1(TP) + \pi_1(FP)}$$

$$recall_{\pi_1,M_0,M_1} = \frac{\pi_1(TP)}{\pi_1(M_0)} = \frac{\pi_1(TP)}{\pi_1(TP) + \pi_1(FN)}$$

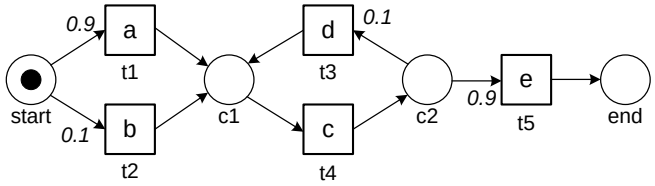

Fig. 6. $M_0 = \{\langle a,c,e\rangle, \langle b,c,e\rangle, \langle a,c,d,c,e\rangle, \ldots\}$ and the probabilities are as shown, e.g., $\pi_0(\langle a,c,e\rangle) = 0.9 \times 0.9 = 0.81$, $\pi_0(\langle b,c,e\rangle) = 0.1 \times 0.9 = 0.09$, $\pi_0(\langle a,c,d,c,e\rangle) = 0.9 \times 0.1 \times 0.9 = 0.081$, etc.

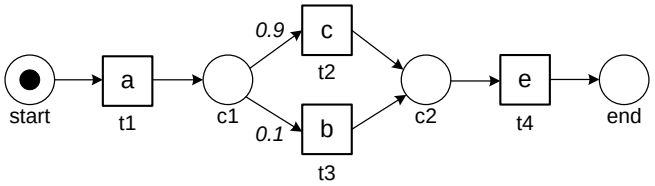

Fig. 7. $M_1 = \{\langle a,c,e\rangle, \langle a,b,e\rangle\}$, $\pi_1(\langle a,c,e\rangle) = 0.9$, and $\pi_1(\langle a,b,e\rangle) = 0.1$.

To illustrate these notions consider π_0 and M_0 shown in Figure 6 and π_1 and M_1 shown in Figure 7. $TP = \{\langle a,c,e\rangle\}$, $\pi_0(TP) = 0.81$, and $\pi_1(TP) = 0.9$.

$$precision_{\pi_0,M_0,M_1} = \frac{\pi_0(TP)}{\pi_0(M_1)} = \frac{0.81}{0.81} = 1$$

$$recall_{\pi_0,M_0,M_1} = \frac{\pi_0(TP)}{\pi_0(M_0)} = \frac{0.81}{1} = 0.81$$

$$precision_{\pi_1,M_0,M_1} = \frac{\pi_1(TP)}{\pi_1(M_1)} = \frac{0.9}{1} = 0.9$$

$$recall_{\pi_1,M_0,M_1} = \frac{\pi_1(TP)}{\pi_1(M_0)} = \frac{0.9}{0.9} = 1$$

Next to precision and recall, related notions such as *error* ($\frac{\pi(FP \cup FN)}{\pi(\mathcal{U})} = \pi(FP \cup FN)$), *accuracy* ($\frac{\pi(TP \cup TN)}{\pi(\mathcal{U})} = \pi(TP \cup TN)$), and *F1-measure* can be defined. The F1-measure is the harmonic mean of precision and recall: $F_{\pi,M_0,M_1} = 2 \times \frac{precision_{\pi,M_0,M_1} \times recall_{\pi,M_0,M_1}}{precision_{\pi,M_0,M_1} + recall_{\pi,M_0,M_1}}$. Note that π may refer to π_0 or π_1.

[1] Note that the notion of precision used here (i.e., the fraction of retrieved items that are indeed relevant) is different from the notion of precision used in Figure 4 and Section II-A.

At first it may seem odd to consider both π_0 (real but generally unknown probabilities) and π_1 (modeled and possibly inaccurate probabilities). However, both views are needed. This can be illustrated as follows.

If M_1 allows for many traces that rarely or never happen in reality, then $precision_{\pi_0,M_0,M_1}$ is hardly affected by this because the contribution of these traces to $\pi_0(FP)$ is marginal. Extending M_1 with behavior that is impossible according to π_0, does not increase $precision_{\pi_0,M_0,M_1}$. In fact, if $\pi_0(\overline{M}_0) = 0$, then $precision_{\pi_0,M_0,M_1} = 1$ by definition. Hence, $precision_{\pi_0,M_0,M_1}$ is unable to detect the loss of precision. However, adding behavior to M_1 that is impossible according to π_0 will impact $precision_{\pi_1,M_0,M_1}$ as π_1 is likely to reflect the extra behavior allowed by M_1. Hence, $precision_{\pi_1,M_0,M_1}$ will be able to signal the loss of precision.

Similarly, it seems reasonable to assume that $\pi_1(\overline{M}_1) = 0$ (i.e., traces that are impossible according to the model have a model-based probability of 0). Hence, $\pi_1(FN) = 0$ and $recall_{\pi_1,M_0,M_1} = 1$. This illustrates that recall based on π_1 tends to give optimistic values. It is better to use $recall_{\pi_0,M_0,M_1}$ instead.

We cannot avoid using a weight function π because there may be infinitely many traces (due to loops) and we need to quantify the surfaces in Figure 5 in some way. It is essential to distinguish between highways (frequent paths) and dirt roads (infrequent paths). From a conformance point of view, it is more relevant that both models agree on the highways than their consensus on insignificant paths. However, using a weight function complicates matters as π_0 is unknown and π_1 may not be given.

V. Precision and Recall Based On Event Data

In the previous section we did not consider the event log and assumed that π_0, M_0, π_1, and M_1 are known. Typically, π_0 and M_0 are unknown. The only things that are known are event log L, model M_1, and (optionally) π_1. Therefore, we use event log L to approximate π_0, M_0, and possibly π_1. Given approximate values π_L and M_L, we can apply the earlier definitions of precision and recall.

Figure 8 relates L to the unknown M_0 and the known M_1. The larger event log L is, the more it will cover M_0. Moreover, more likely traces in the real process tend to appear more frequently in event log L. This observation can be used to approximate the real process.

Definition 7 (Estimator Based on Log): Let $L \in \mathbf{B}(\mathcal{U})$ be an event log. $\pi_L \in \mathcal{U} \rightarrow [0,1]$ is an *estimator* of π_0 based on L: $\pi_L(\sigma) = \frac{L(\sigma)}{|L|}$ for $\sigma \in \mathcal{U}$.

Just like M_0 can be derived from π_0, M_L can be derived from π_L. For example, $M_L \subseteq \mathcal{U}$ such that M_L is the smallest set that satisfies: $\pi_L(M_L) \geq \tau$ and $\forall_{\sigma \in M_L} \forall_{\sigma' \in \overline{M}_L} \pi_L(\sigma) > \pi_L(\sigma')$ for some threshold $\tau \in [0,1]$. M_L is the *estimator model* for M_0 based on L.

If there is no information about π_1 in the "descriptive" or "normative" model, we can use replay algorithms to estimate probabilities for choices [1], [24]. For example, for a choice between a and b in the model we can measure the fraction of times a (and b) occurs in the event log and use this to extend the model with stochastic information.

In the remainder we assume that (1) π_L is an estimator for π_0, (2) M_L is an estimator for M_0, (3) M_1 is given, and (4) π_1 is known (either it is given or estimated based on replaying log L on M_1).

Now we can apply all of the earlier computations using these estimators: $TP_L = M_L \cap M_1$, $TN_L = \overline{M}_L \cap \overline{M}_1$, $FP_L = \overline{M}_L \cap M_1$, $FN_L = M_L \cap \overline{M}_1$,

$$precision_{\pi_L,M_L,M_1} = \frac{\pi_L(TP_L)}{\pi_L(M_1)}$$

$$recall_{\pi_L,M_L,M_1} = \frac{\pi_L(TP_L)}{\pi_L(M_L)}$$

$$precision_{\pi_1,M_L,M_1} = \frac{\pi_1(TP_L)}{\pi_1(M_1)}$$

$$recall_{\pi_1,M_L,M_1} = \frac{\pi_1(TP_L)}{\pi_1(M_L)}$$

Consider again the process shown in Figure 6 and π_1 and M_1 shown in Figure 7. However, now assume that π_0 and M_0 are unknown. Instead, we have an event log $L = [\langle a,c,e\rangle^{16}, \langle b,c,e\rangle^2, \langle a,c,d,c,e\rangle^1, \langle a,c,d,c,d,c,e\rangle^1]$. $\pi_L(\langle a,c,e\rangle) = \frac{16}{20} = 0.8$, $\pi_L(\langle b,c,e\rangle) = \frac{2}{20} = 0.1$, $\pi_L(\langle a,c,d,c,e\rangle) = \frac{1}{20} = 0.05$, and $\pi_L(\langle a,c,d,c,d,c,e\rangle) = \frac{1}{20} = 0.05$. Assume $M_L = \{\sigma \in L\}$, i.e., the process model able to reproduce all observed traces. $TP_L = \{\langle a,c,e\rangle\}$, $\pi_L(TP_L) = 0.8$, and $\pi_1(TP_L) = 0.9$.

$$precision_{\pi_L,M_L,M_1} = \frac{\pi_L(TP_L)}{\pi_L(M_1)} = \frac{0.8}{0.8} = 1$$

$$recall_{\pi_L,M_L,M_1} = \frac{\pi_L(TP_L)}{\pi_L(M_L)} = \frac{0.8}{1} = 0.8$$

$$precision_{\pi_1,M_L,M_1} = \frac{\pi_1(TP_L)}{\pi_1(M_1)} = \frac{0.9}{1} = 0.9$$

$$recall_{\pi_1,M_L,M_1} = \frac{\pi_1(TP_L)}{\pi_1(M_L)} = \frac{0.9}{0.9} = 1$$

Let $\overline{L} = \{\sigma \in \mathcal{U} \mid \sigma \notin L\}$. By definition $\pi_L(\overline{L}) = 0$, so things outside the event log are assumed to be impossible. If $M_L = \{\sigma \in \mathcal{U} \mid \pi_L(\sigma) > 0\}$, then $\pi_L(M_L) = 1$ and $\pi_L(\overline{M}_L) = 0$. As a result $\pi_L(FP_L) = \pi_L(TN_L) = 0$ and $precision_{\pi_L,M_L,M_1} = 1$. This is a concern as precision is not affected by adding non-observed behavior to M_1.

If $M_1 = \{\sigma \in \mathcal{U} \mid \pi_1(\sigma) > 0\}$, then $\pi_1(M_1) = 1$ and $\pi_1(\overline{M}_1) = 0$. As a result $\pi_1(TN_L) = \pi_1(FN_L) = 0$ and $recall_{\pi_1,M_L,M_1} = 1$.

Hence, in practice the only two meaningful metrics are $recall_{\pi_L,M_L,M_1}$ and $precision_{\pi_1,M_L,M_1}$.

If we assume $\pi_L(M_L) = 1$ (all observed observed behavior is possible in M_L) and $\pi_1(M_1) = 1$ (behavior that is not modeled is considered to be impossible), these two metrics can be simplified as follows:

$$precision_{\pi_1,M_L,M_1} = \frac{\pi_1(TP_L)}{\pi_1(M_1)} = \pi_1(TP_L) = \pi_1(M_L)$$

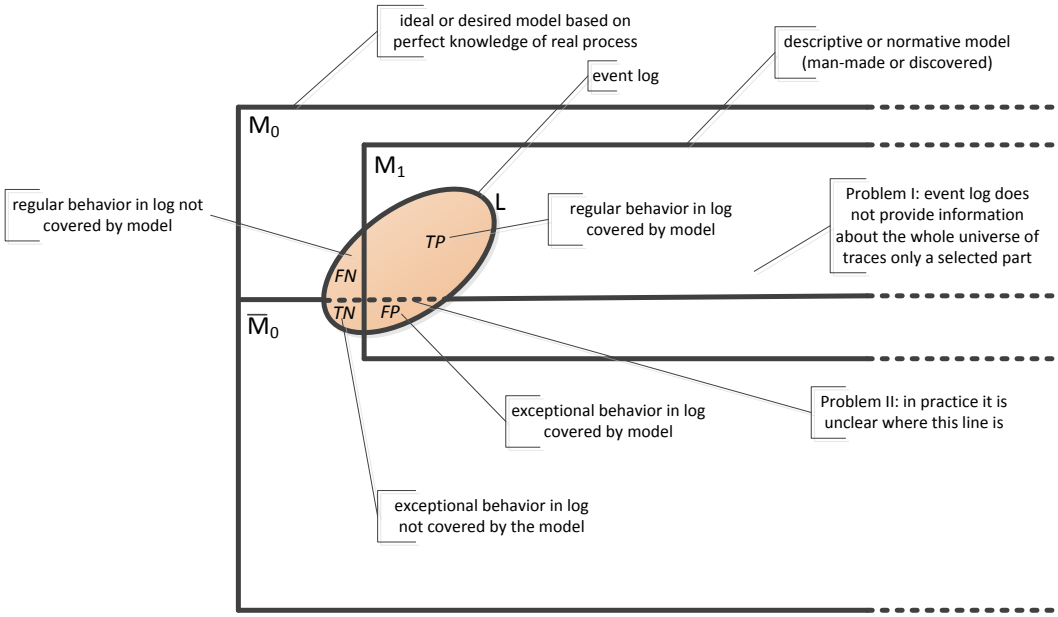

Fig. 8. Event log L is finite and contains only samples of the real process. A trace $\sigma \in L$ typically fits into the unknown ideal or desired model M_0. However, σ may be infrequent in reality and not part of M_0. Trace σ may fit the descriptive or normative model M_1 (or not).

$$recall_{\pi_L, M_L, M_1} = \frac{\pi_L(TP_L)}{\pi_L(M_L)} = \pi_L(TP_L) = \pi_L(M_1)$$

These correspond to the traditional precision and fitness notions [1], [6] discussed in Section II-A. The notion of *fitness* (ability to replay event log) in Figure 4 corresponds to $recall_{\pi_1, M_L, M_1}$ and the notion of *precision* (not allowing for too much behavior unrelated to event log) in Figure 4 corresponds to $precision_{\pi_1, M_L, M_1}$.

Interestingly, in Section IV we already concluded that precision based on π_1 and recall based on π_0 seem to be the only two notions that make sense. Hence, whether we approximate π_0 by π_L or not, we come to the same conclusion.

VI. Beyond Precision and Recall

In the previous section we concluded that $recall_{\pi_L, M_L, M_1}$ and $precision_{\pi_1, M_L, M_1}$ are the only two meaningful metrics. As indicated, $recall_{\pi_1, M_L, M_1}$ refers to the classical notion of fitness and $precision_{\pi_1, M_L, M_1}$ refers to the classical notion of precision described in Section II-A. However, Figure 4 shows two additional quality dimensions: *simplicity* and *generalization*.

Simplicity (often referred to as Occam's Razor) is a concern not addressed by precision and recall. Simplicity seems only indirectly related to the notions used before (i.e., π_0, M_0, π_L, M_L, π_1, and M_1). For example, the complexity of process model $M_1 \subseteq \mathcal{U}$ should not be defined as the number of possible traces. A simple process model having a loop has infinitely many possible traces. Concurrency may introduce a factorial number of possible interleavings, but does not need to be more complex than a model with choices instead of concurrency. Hence, $|M_1|$ is a bad indicator for a model's perceived simplicity. Consider for example the "flower model" that is able to generate any trace over some alphabet. The model is easy to understand and can be represented compactly, but is the "largest model" in terms of corresponding traces.

There are various techniques to quantify model complexity. The complexity of the model could be defined by the number of nodes and arcs in the underlying graph. Also more sophisticated metrics can be used, e.g., metrics that take the "structuredness" or "entropy" of the model into account. A detailed discussion of these complexity metrics is outside the scope of this paper. We refer to [25] for pointers to over twenty metrics described in literature.

The notion of *generalization* described in Section II-A seems to be related to the quality of π_L as an estimator for π_0. A model that does not generalize is "overfitting". Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior. *Generalization can be defined as the probability that the next, not yet observed, case can be replayed by the process model.* In other words, the probability that for the next observed trace σ_N: $\sigma_N \in M_1$. It is difficult to reason about generalization because the notion refers to unseen examples. In [6], [14] possible metrics are discussed.

If we have an event log L with just a few traces and most traces are unique, then π_L is a poor estimator for π_0. In this case, the next trace σ_N is likely to be different from all traces seen before. Hence, a model allowing for just the traces in L will not allow for σ_N. If we have an event log with many traces and most traces appear many times, then π_L is a much better estimator for π_0 due to the *strong law of large numbers* that states that the sample average converges to the expected value (assuming independence between the different cases). Hence, a model allowing for the traces in L will, most likely, also allow for σ_N. These extreme examples illustrate the connection between generalization in Figure 4 and the quality of π_L as an estimator for the unknown π_0.

In the context of discovery algorithms, often a notion of *completeness* is defined. An event log is complete if it contains enough example behavior to discover the underlying process. For example, in the context of the α algorithm [26] a log is complete if all direct successions have been observed, i.e., if a can be followed by b in the process it should be observed at least once in the event log. Clearly, completeness is related to generalization and the quality of π_L as an estimator for π_0.

The above reasoning reconfirms the necessity of the four conventional quality dimensions described in Section II-A. This is quite surprising since a completely different starting point was used. Unlike existing process mining literature, our reasoning is based on the assumption that we know the real process π_0 and we consider probabilities explicitly (rather than considering only the possibility of a trace). Yet, we come to the same conclusion. Nonetheless, the new insights may help to provide more objective metrics for precision and generalization.

Moreover, we advocate the use of a so-called *Pareto front* for process discovery. Rather than aggregating the four quality dimensions into a single value and selecting the best model according to an aggregate value, we suggest building a set of process models that are *Pareto-efficient*. A process model is Pareto-efficient if there is no other model that scores better with respect to one dimension (e.g., fitness) and not worse with respect to the other three dimensions (e.g., simplicity, generalization, and precision). The Pareto front is very natural in the context of genetic process mining [9], [14] where many different possible models are explored.

The Achilles' heel of the precision and recall metrics defined in Section V is the definition of π: π_0 is unknown, π_L is just an approximation, and π_1 may be very different from reality. Moreover, depending on the choice of π the interpretation of $recall_{\pi,M,M_1}$ and $precision_{\pi,M,M_1}$ changes. It would be interesting to investigate hybrid approaches using an estimator $\pi' \in \mathcal{U} \rightarrow [0,1]$ that uses both model and log, e.g., the likelihood of a trace is based on the frequency in the log and the likelihood of the corresponding path in the model. The weight of the log-based part could be based on the log's size or completeness.

Function π can be used to select the desired perspective (model or log). Moreover, it is interesting to think of a parameterized function π_c where c is a timestamp or some context attribute. For example, in December, when it is raining, the probability of trace σ is higher than in June when the sun shines. The probability of σ may depend on the month, the day of the week, the weather, the workload, etc. This relates to notions such as concept drift [27] and contextual process mining [28].

VII. Related Work

See [1] for an introduction to process mining and the Process Mining Manifesto [29] for the main challenges in process mining.

Cook et al. [30], [31] were among the first to quantify the relationship between event logs and process model. They compare event streams of the model with event steams generated from the event log.

Several authors proposing process discovery algorithms also provide a quality metric (often related to fitness). For example, in [9] the authors define a fitness function for searching for the optimal model using a genetic approach. In [32] a "process mining evaluation framework" for benchmarking process discovery algorithms is proposed.

The first comprehensive approach to conformance analysis was proposed in [12] by Rozinat and Van der Aalst. Two different types of metrics are proposed: (a) *fitness metrics*, i.e., the extent to which the log traces can be associated with valid execution paths specified by the process model, and (b) *appropriateness metrics*, i.e., the degree of accuracy in which the process model describes the observed behavior, combined with the degree of clarity in which it is represented. Fitness in [12] is measured by "replaying the event log" and counting the number of missing and remaining tokens. This typically results in rather high fitness values as also pointed out in [16], [21]. In [12] four appropriateness metrics are defined. *Simple behavioral appropriateness* looks at the average number of enabled transitions. If most transitions are continuously enabled, the model is likely to lack precision (i.e., underfitting). *Advanced behavioral appropriateness* compares the "footprint" of the log (follows and precedes relationships) to the "footprint" of the model. *Simple structural appropriateness* and *advanced structural appropriateness* quantify the complexity of the model.

One of the drawbacks of the approach in [12] and most other approaches that "play the token game", is that fitness is typically overestimated. When a model and log do not fit well together, replay will overload the process model with superfluous tokens. As a result, the model will allow for too much behavior. Approaches such as the one in [12] also have problems when the model has "invisible activities" (silent steps that are not recorded in the event log) or "duplicate activities" (multiple transitions bearing the same label). To deal with such phenomena state-space exploration and heuristics are needed to replay the event log. In fact, most conformance techniques give up after the first non-fitting event or simply "guess" the corresponding path in the model. Therefore, Adriansyah et al. formulated conformance checking problems as an optimization problem [16], [7].

Lion's share of attention in conformance checking has been devoted to checking fitness. However, in recent papers researchers started to explore the other quality dimensions [6], [14]. For example, Munoz-Gama et al. quantified additional precision notions [13], [10], [11].

As shown in this paper, it is difficult to use classical quality notions such as *precision* and *recall* for process mining. The main reason is that event logs only contain positive examples, i.e., one can see what "did happen" but not what "could not happen". Therefore, some authors suggest inserting artificially generated "negative events" [8], [33]. Goedertier et al. proposed such events for both process discovery and conformance checking [8]. De Weerdt et al. defined a so-called F-measure based on artificially generated negative events [33]. The authors of the latter paper also conducted a comparative analysis of several conformance metrics [21], [20].

In [34] a so-called completeness metric and soundness metric are defined. These metrics compare the traces of the

model with the traces in the log. This approach suffers from several drawbacks. First of all, only complete traces are compared. Second, it is assumed that the model's behavior can be enumerated. Finally, it is assumed that the log contains all possible traces.

In [35], the techniques presented in [16], [7] are generalized to artifact-centric processes (the so-called Proclets). The conformance notions in [35] also take interactions between process instances into account.

Several approaches create so-called behavioral footprints to compare event log and model [1]. The key idea is that a footprint can be based on observed behavior and modeled behavior as described in [1]. Another example of such a footprint is the so-called "behavioral profile" [36]. The problem of this approach is that it cannot handle loops properly (unlike [1], [12]).

All techniques discussed thus far, compare model and log. There are also many compliance approaches that compare a model and another model or a model and a set of rules [37], [38], [39], [40]. These approaches are very different from the techniques discussed in this paper as they do not take the actual observed behavior into account. In fact, these approaches do not even take the likelihood of traces into account. This problem is discussed in [41], [42] where process equivalence is quantified based on observed behavior.

Although not shown in this paper, alignments can also be used for performance analysis as most event logs contain timestamps [1]. Replaying event logs with timestamps allows for bottleneck analysis and prediction as demonstrated in [18], [19].

VIII. Conclusion

In this paper, we investigated the relation between observed and modeled behavior. This topic is highly relevant as torrents of event data have become available. As a result, models can be discovered using process mining techniques and it is possible to reflect on existing process models using real event data.

In literature typically four quality dimensions for comparing model and log are considered: *fitness*, *simplicity*, *precision*, and *generalization* [1]. At a first glance these dimensions seem arbitrary and not well defined. Therefore, this paper used well-known notions from information retrieval (precision and recall) to compare observed and modeled behavior. *Unlike existing papers we started from the assumption that the real process is known. Moreover, we also take the likelihood of traces into account.*

As shown, it is crucial to consider the likelihood of traces when comparing the ideal/desired model and the descriptive/normative model. There are two main reasons. There may be infinitely many traces (e.g., in case of loops). Therefore, it does not make sense to count traces without considering their likelihood. Moreover, some traces may be very frequent (highways) whereas other traces occur rarely (dirt roads). If observed and modeled behavior disagree on a "highway", this is more severe than disagreement on some infrequent trace.

In case the real process π_0 and the desired process model M_0 are known, $precision_{\pi_1,M_0,M_1}$ and $recall_{\pi_0,M_0,M_1}$ make most sense. If only an event log L is available and π_0 and M_0 are unknown, then $precision_{\pi_1,M_L,M_1}$ and $recall_{\pi_L,M_L,M_1}$ are the two obvious metrics to consider. Surprisingly these correspond to the traditional precision and fitness notions.

π_L is a log-based estimate of π_0. The quality of such an approximation is an important factor when comparing observed and modeled behavior using precision and recall. The approximation's quality is closely related to the notion of generalization. Simplicity is another obvious concern that is not covered by precision and recall. Hence, despite taking a completely different starting point, we end up with the conventional four quality dimensions for analyzing the relation between observed and modeled behavior [1].

The second part of the title of this paper – The Quest for the "Right" Process – can be viewed from two angles. When the process model and the real process (reflected in the event log) disagree, one can "blame" (1) the model or (2) the actual process. When evaluating process discovery techniques, the real process is always "right" and the discovered process model is "wrong" in case of deviations. For other applications, the process model and/or the real process may be "wrong". For example, if people often deviate from the normative process model, then conformance analysis may trigger measures to influence the behavior of these people. However, deviations may also point to inefficiencies in the normative process. For example, people may bypass activities for good reasons. In this case the model need to be repaired based on observed behavior [43]

Acknowledgment

The author would like to thank all process mining researchers and ProM developers that contributed to the insights reported in this paper. Unfortunately, just a few can be mentioned here. Anne Rozinat laid the foundations for conformance checking as we know it today [12], [32], [44]. She was the first to define clear conformance notions outside the scope of a specific discovery algorithm (including token-based replay). Before the work of Anne, fitness was investigated by Ana Karla Alves de Medeiros in the context of genetic mining [9]. Arya Adriansyah (with great support from Boudewijn van Dongen) developed the notion of alignments and various A^*-based algorithms [6], [15], [16], [7], [13]. Jorge Munoz Gama and Josep Carmona made the precision notion more concrete by quantifying "escaping edges" [13], [10], [11]. Joos Buijs and Boudewijn van Dongen contributed the idea to consider the "real process" in the context of evaluating discovered process trees [14].

References

[1] W. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.

[2] ——, "Business Process Management: A Comprehensive Survey," *ISRN Software Engineering*, pp. 1–37, 2013, doi:10.1155/2013/507984.

[3] W. van der Aalst and K. van Hee, *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2004.

[4] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. MIT press, Cambridge, MA, 2001.

[5] T. Mitchell, *Machine Learning*. McGraw-Hill, New York, 1997.

[6] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying History on Process Models for Conformance Checking and Performance Analysis," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[7] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Conformance Checking using Cost-Based Fitness Analysis," in *IEEE International Enterprise Computing Conference (EDOC 2011)*, C. Chi and P. Johnson, Eds. IEEE Computer Society, 2011, pp. 55–64.

[8] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust Process Discovery with Artificial Negative Events," *Journal of Machine Learning Research*, vol. 10, pp. 1305–1340, 2009.

[9] A.K. Alves de Medeiros, A. Weijters, and W. van der Aalst, "Genetic Process Mining: An Experimental Evaluation," *Data Mining and Knowledge Discovery*, vol. 14, no. 2, pp. 245–304, 2007.

[10] J. Munoz-Gama and J. Carmona, "A Fresh Look at Precision in Process Conformance," in *Business Process Management (BPM 2010)*, ser. Lecture Notes in Computer Science, R. Hull, J. Mendling, and S. Tai, Eds., vol. 6336. Springer-Verlag, Berlin, 2010, pp. 211–226.

[11] ——, "Enhancing Precision in Process Conformance: Stability, Confidence and Severity," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, N. Chawla, I. King, and A. Sperduti, Eds. Paris, France: IEEE, April 2011, pp. 184–191.

[12] A. Rozinat and W. van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[13] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst, "Alignment Based Precision Checking," in *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2012)*, ser. Lecture Notes in Business Information Processing, M. Rosa and P. Soffer, Eds., vol. 132. Springer-Verlag, Berlin, 2013, pp. 137–149.

[14] J. Buijs, B. van Dongen, and W. van der Aalst, "On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery," in *OTM Federated Conferences, 20th International Conference on Cooperative Information Systems (CoopIS 2012)*, ser. Lecture Notes in Computer Science, R. Meersman, S. Rinderle, P. Dadam, and X. Zhou, Eds., vol. 7565. Springer-Verlag, Berlin, 2012, pp. 305–322.

[15] A. Adriansyah, N. Sidorova, and B. van Dongen, "Cost-based Fitness in Conformance Checking," in *International Conference on Application of Concurrency to System Design (ACSD 2011)*. IEEE Computer Society, 2011, pp. 57–66.

[16] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Towards Robust Conformance Checking," in *BPM 2010 Workshops, Proceedings of the Sixth Workshop on Business Process Intelligence (BPI2010)*, ser. Lecture Notes in Business Information Processing, M. Muehlen and J. Su, Eds., vol. 66. Springer-Verlag, Berlin, 2011, pp. 122–133.

[17] M. de Leoni, W. van der Aalst, and B. van Dongen, "Data- and Resource-Aware Conformance Checking of Business Processes," in *Business Information Systems (BIS 2012)*, ser. Lecture Notes in Business Information Processing, W. Abramowicz, D. Kriksciuniene, and V. Sakalauskas, Eds., vol. 117. Springer-Verlag, Berlin, 2012, pp. 48–59.

[18] W. van der Aalst, M. Pesic, and M. Song, "Beyond Process Mining: From the Past to Present and Future," in *Advanced Information Systems Engineering, Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*, ser. Lecture Notes in Computer Science, B. Pernici, Ed., vol. 6051. Springer-Verlag, Berlin, 2010, pp. 38–52.

[19] W. van der Aalst, M. Schonenberg, and M. Song, "Time Prediction Based on Process Mining," *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[20] J. De Weerdt, M. Backer, J. Vanthienen, and B. Baesens, "A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs," *Information Systems*, vol. 37, no. 7, pp. 654–676, 2012.

[21] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A Critical Evaluation of Model-Log Metrics in Process Discovery," in *BPM 2010 Workshops, Proceedings of the Sixth Workshop on Business Process Intelligence (BPI2010)*, ser. Lecture Notes in Business Information Processing, M. Muehlen and J. Su, Eds., vol. 66. Springer-Verlag, Berlin, 2011, pp. 158–169.

[22] J. Buijs, B. van Dongen, and W. van der Aalst, "A Genetic Algorithm for Discovering Process Trees," in *IEEE Congress on Evolutionary Computation (CEC 2012)*. IEEE Computer Society, 2012, pp. 1–8.

[23] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Wiley series in parallel computing. Wiley, New York, 1995.

[24] A. Rozinat, R. Mans, M. Song, and W. van der Aalst, "Discovering Simulation Models," *Information Systems*, vol. 34, no. 3, pp. 305–327, 2009.

[25] J. Mendling, G. Neumann, and W. van der Aalst, "Understanding the Occurrence of Errors in Process Models Based on Metrics," in *Proceedings of the OTM Conference on Cooperative information Systems (CoopIS 2007)*, ser. Lecture Notes in Computer Science, F. Curbera, F. Leymann, and M. Weske, Eds., vol. 4803. Springer-Verlag, Berlin, 2007, pp. 113–130.

[26] 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.

[27] R.P. Jagadeesh Chandra Bose, W. van der Aalst, I. Zliobaite, and M. Pechenizkiy, "Handling Concept Drift in Process Mining," in *International Conference on Advanced Information Systems Engineering (Caise 2011)*, ser. Lecture Notes in Computer Science, H. Mouratidis and C. Rolland, Eds., vol. 6741. Springer-Verlag, Berlin, 2011, pp. 391–405.

[28] W. van der Aalst and S. Dustdar, "Process Mining Put into Context," *IEEE Internet Computing*, vol. 16, no. 1, pp. 82–86, 2012.

[29] IEEE Task Force on Process Mining, "Process Mining Manifesto," in *BPM Workshops*, ser. Lecture Notes in Business Information Processing, vol. 99. Springer-Verlag, Berlin, 2011.

[30] J. Cook and A. Wolf, "Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model," *ACM Transactions on Software Engineering and Methodology*, vol. 8, no. 2, pp. 147–176, 1999.

[31] J. Cook, C. He, and C. Ma, "Measuring Behavioral Correspondence to a Timed Concurrent Model," in *Proceedings of the 2001 International Conference on Software Mainenance*, 2001, pp. 332–341.

[32] A. Rozinat, A.K. Alves de Medeiros, C. Günther, A. Weijters, and W. van der Aalst, "The Need for a Process Mining Evaluation Framework in Research and Practice," in *BPM 2007 International Workshops (BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws)*, ser. Lecture Notes in Computer Science, A. Hofstede, B. Benatallah, and H. Paik, Eds., vol. 4928. Springer-Verlag, Berlin, 2008, pp. 84–89.

[33] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A Robust F-measure for Evaluating Discovered Process Models," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, N. Chawla, I. King, and A. Sperduti, Eds. Paris, France: IEEE, April 2011, pp. 148–155.

[34] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, "Discovering Expressive Process Models by Clustering Log Traces," *IEEE Transaction on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1010–1027, 2006.

[35] D. Fahland, M. de Leoni, B. van Dongen, and W. van der Aalst, "Conformance Checking of Interacting Processes with Overlapping Instances," in *Business Process Management (BPM 2011)*, ser. Lecture Notes in Computer Science, S. Rinderle, F. Toumani, and K. Wolf, Eds., vol. 6896. Springer-Verlag, Berlin, 2011, pp. 345–361.

[36] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske, "Process Compliance Measurement Based on Behavioral Profiles," *Information Systems*, vol. 36, no. 7, pp. 1009–1025, 2011.

[37] A. Elgammal, O. Turetken, W. Heuvel, and M. Papazoglou, "Root-Cause Analysis of Design-time Compliance Violations on the basis of Property Patterns," in *Proceedings of Service-Oriented Computing (ICSOC 2010)*, ser. Lecture Notes in Computer Science, P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds., vol. 6470. Springer-Verlag, Berlin, 2010, pp. 17–31.

[38] G. Governatori, J. Hoffmann, S. Sadiq, and I. Weber, "Detecting Regulatory Compliance for Business Process Models through Semantic Annotations," in *4th International Workshop on Business Process Design*, 2008.

[39] G. Governatori, Z. Milosevic, and S. Sadiq, "Compliance Checking Between Business Processes and Business Contracts," in *10th Inter-*

national Enterprise Distributed Object Computing Conference (EDOC 2006). IEEE Computing Society, 2006, pp. 221–232.

[40] N. Lohmann, "Compliance by Design for Artifact-Centric Business Processes," in *Business Process Management (BPM 2011)*, ser. Lecture Notes in Computer Science, S. Rinderle, F. Toumani, and K. Wolf, Eds., vol. 6896. Springer-Verlag, Berlin, 2011, pp. 99–115.

[41] W. van der Aalst, A.K. Alves de Medeiros, and A. Weijters, "Process Equivalence: Comparing Two Process Models Based on Observed Behavior," in *International Conference on Business Process Management (BPM 2006)*, ser. Lecture Notes in Computer Science, S. Dustdar, J. Fiadeiro, and A. Sheth, Eds., vol. 4102. Springer-Verlag, Berlin, 2006, pp. 129–144.

[42] A.K. Alves de Medeiros, W. van der Aalst, and A. Weijters, "Quantifying Process Equivalence Based on Observed Behavior," *Data and Knowledge Engineering*, vol. 64, no. 1, pp. 55–74, 2008.

[43] D. Fahland and W. van der Aalst, "Repairing Process Models to Reflect Reality," in *International Conference on Business Process Management (BPM 2012)*, ser. Lecture Notes in Computer Science, A. Barros, A. Gal, and E. Kindler, Eds., vol. 7481. Springer-Verlag, Berlin, 2012, pp. 229–245.

[44] A. Rozinat, I. de Jong, C. Günther, and W. van der Aalst, "Process Mining Applied to the Test Process of Wafer Scanners in ASML," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 39, no. 4, pp. 474–479, 2009.