PROCESS MINING AND SIMULATION: A MATCH MADE IN HEAVEN!

Wil M.P. van der Aalst

Process and Data Science (Informatik 9)
RWTH Aachen University, D-52056 Aachen, Germany
wvdaalst@pads.rwth-aachen.de

ABSTRACT

Event data are collected everywhere: in logistics, manufacturing, finance, healthcare, e-learning, e-government, and many other domains. The *events* found in these domains typically refer to *activities* executed by *resources* at particular *times* and for particular *cases*. Process mining provides the means to discover the real processes, to detect deviations from normative processes, and to analyze bottlenecks and waste from such events. However, process mining tends to be *backward-looking*. Fortunately, simulation can be used to explore different design alternatives and to anticipate future performance problems. This keynote paper discusses the link between both types of analysis and elaborates on the challenges process discovery techniques are facing. Quality notions such as *recall*, *precision*, and *generalization* are discussed. Rather than introducing a specific process discovery or conformance checking algorithm, the paper provides a *comprehensive set of conformance propositions*. These conformance propositions serve two purposes: (1) introducing the essence of process mining by discussing the relation between event logs and process models, and (2) discussing possible requirements for the quantification of quality notions related to recall, precision, and generalization.

Keywords: process mining, simulation, process discovery, conformance checking.

1 INTRODUCTION

Discrete-Event Simulation (DES)—simply referred to as *simulation* in this paper—is a widely used approach to *play-out* process models (Law and Kelton 1982, Kleijnen and Groenendaal 1992, Aalst and Stahl 2011, Aalst 2015). Based on the rules defined by the simulation model, events are generated. Each event occurs at a particular instant in time and marks a change of state in the system. States enable new events. A simulation run describes one of the possibly many ways in which the model can be played-out. Random-number generators are used to resolve choices when the model allows for different paths (e.g. XOR-splits) and to sample durations from predefined probability distributions to determine waiting and service times. Through simulation experiments various "what if" questions can be answered and redesign alternatives can be compared with respect to key performance indicators. However, making a good simulation model may be very time consuming, and it may be outdated by the time it is ready.

Process mining is fundamentally different as it starts from observed behavior rather than modeled behavior (Aalst 2016). *Event data*, recorded in a so-called *event log*, describe what really happened. Each event refers to an *activity* that occurred at a particular point in *time* for a particular *case*. The chronological ordering of events for a particular case yields a *trace*, i.e., a sequence of activities executed for that case. One such trace is similar to a simulation run; it is only one example of possibly many different behaviors. Process discovery aims to *play-in* process models, i.e., the example traces found in the event log are used to construct a process model. Other process mining techniques *replay* event data on a process model to check conformance and to analyze bottlenecks. Next, to providing insights into the real behavior of a system or process, these techniques help to understand and improve performance and compliance problems. The discovered process models may even be used to predict the trajectories of running cases (e.g., the remaining flow time) assuming that the process does not change. Whereas simulation focuses on *play-out*, process mining focuses on *play-in* and *replay*. However, both use or generate event data (e.g., simulation logs) and process models. Therefore, they nicely complement each other.

In the remainder, we assume that the reader has a good understanding of Discrete-Event Simulation (DES) approaches. The focus in the first part of the paper will be on the relation between simulation and process mining. We will show that together they form "A Match Made in Heaven!". Process mining can be used to make better simulation models and to compare simulation runs with actual behaviors. Simulation can be used to make process mining more forward-looking and explore different process changes.

The second part of the paper proposes a comprehensive set of *21 conformance propositions*. Each proposition describes a property of some quality measure that compares an event log and a process model. Consider for example recall, i.e., the ability to replay the behavior seen in the event log (sometimes called fitness). When a model and log have a high recall, most of the observed behavior in the log can be explained by the model. A possible proposition is that recall should be monotonic in terms of the model behavior, i.e., when the model allows for more behavior, recall can only improve. Similarly, precision cannot improve by adding behavior to the model that was never observed, and generalization cannot improve by removing behavior from the model. By providing a comprehensive set of conformance propositions, we aim to stimulate the reader to think about the relationship between observed behavior (event data) and modeled behavior (process models), thus revealing the essence of process mining. Moreover, the conformance propositions also reveal limitations of existing approaches that aim to quantify this relation.

Finally, we add probabilities to event logs and discuss conformance in the setting where the real process (i.e., the "ground truth") is known. This reveals the foundational challenges that need to be addressed by process discovery techniques. The idealized setting with a known "ground truth" can be used as a reality check for process mining research. Moreover, it further illustrates process mining concepts for readers with a simulation background.

The remainder of this paper is organized as follows. Section 2 discusses synergies between process mining and simulation. Section 3 introduces basic process mining concepts assuming a rather simplistic setting. This is used to present the 21 conformance propositions. These are interesting for the development of new recall, precision, and generalization measures. Section 4 describes a more informative setting where the real stochastic process and desired process model are known. Section 5 concludes the paper.

2 RELATING PROCESS MINING AND SIMULATION

Simulation has been used to analyze operational processes since the uptake of Simula and related languages in the 1960-ties (Dahl and Nygaard 1966). Through simulation experiments various "what if" questions can be answered and redesign alternatives can be compared with respect to key performance indicators (Law and Kelton 1982, Kleijnen and Groenendaal 1992, Aalst and Stahl 2011, Aalst 2015). Despite these *forward-looking* capabilities, rich history, and powerful simulation tools, the real-life application of simulation is limited. There are two main reasons for this. First of all, it is typically very time-consuming to build a good simulation model. Second, it is not easy to let a simulation model mimic reality. Therefore, simulation results can always be questioned: "It is just a modeled reality". Therefore, organizations are resorting increasingly to evidence-based approaches.

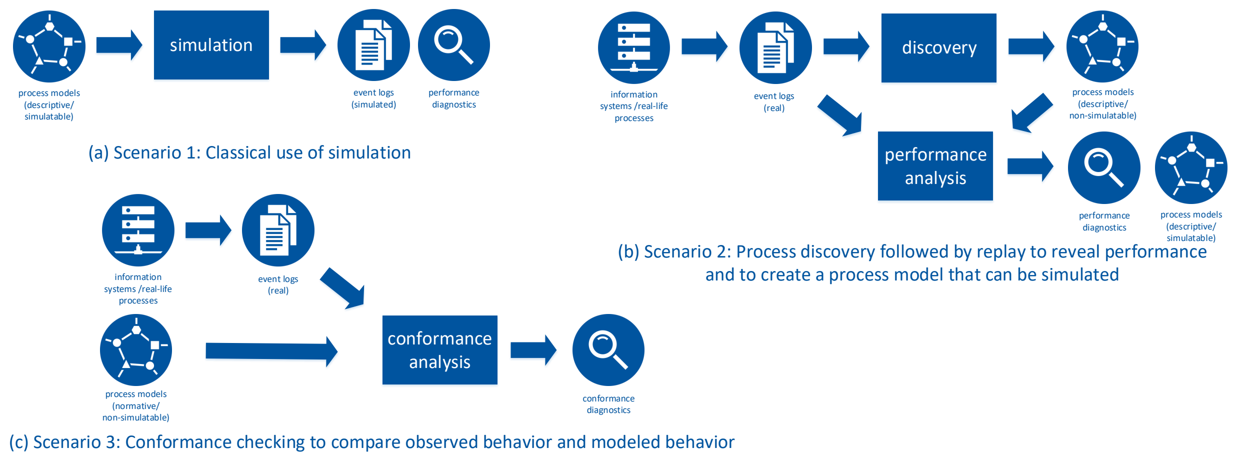

process models (descriptive/ simulatable)
simulation
event logs (simulated)
performance diagnostics
(a) Scenario 1: Classical use of simulation
information systems /real-life processes
event logs (real)
discovery
process models (descriptive/ non-simulatable)
performance analysis
performance diagnostics
process models (descriptive/ simulatable)
(b) Scenario 2: Process discovery followed by replay to reveal performance and to create a process model that can be simulated
information systems /real-life processes
event logs (real)
process models (normative/ non-simulatable)
conformance analysis
conformance diagnostics
(c) Scenario 3: Conformance checking to compare observed behavior and modeled behavior

Figure 1: Three analysis scenarios: (a) classical model-driven simulation, (b) discovery and performance analysis, and (c) conformance checking based on a normative model.

Process mining is a more recent development where operational processes are analyzed based on the event data they generate (Aalst 2016). Currently, there are over 25 commercial tools supporting process mining (e.g., Disco, Celonis, ProcessGold, QPR, minit, and myInvenio). ProM is the leading open-source process mining tool, serving as the de facto standard for the academic world. Using these tools processes can be constructed automatically based on event data. By replaying event data on process models (discovered or hand-made), one can answer compliance and performance questions. Since it is always possible to drill-down and show the actual event data, results become undeniable. However, the "Achilles heel of process mining" is the fact that it is *backward-looking*. Process mining can be used to diagnose problems (e.g., bottlenecks or non-compliance) and predict the paths taken by running process instances (i.e., cases), but it cannot be used to answer "what if" questions and explore radical redesigns.

Given the above, it is very natural to combine process mining and simulation. Figure 1 shows three analysis scenarios. Simulation starts from a process model and produces behavior and performance diagnostics (Figure 1(a)). Simulations can be used to generate event logs recording the simulated behavior. Figure 1(b) shows a process mining scenario where event data generated by some process or information system are used to discover a descriptive process model. By replaying the event data on the discovered process model, it is possible to analyze bottlenecks and add the temporal and stochastic behavior to the model. The result is a process model that can be used to simulate the process. Figure 1(c) shows a process mining scenario where there is a normative process model. By replaying the real behavior on the normative model, it is possible to diagnose deviations.

A deeper look at Figure 1 reveals possible interfaces between process mining and simulation: (1) the *event logs* generated through simulation can be analyzed using process mining techniques and (2) the *process models* generated by process mining can be used as input for simulation. The XES standard (www.xes-standard.org) can be used to exchange event logs between process mining and simulation tools. It is more difficult to exchange simulation models (cf. the attempted WfMC standard BPSim). Note that ProM provides exports to a limited number of simulation tools (e.g., CPN Tools).

As discussed in (Aalst 2015, Rozinat, Wynn, Aalst, Hofstede, and Fidge 2009) more advanced scenarios are possible. One of these scenarios is sketched in Figure 2. First, a simulation model is learned from event data. In the discovery step, a descriptive model is learned which is enriched into a simulation model by replaying the event log (adding probability distributions). At any point in time, the current state of the process can be loaded from the information system. Then a so-called *short-term simulation* can be performed. The key idea is to start all simulation runs from the current state and focus on transient behavior. This way a "fast-

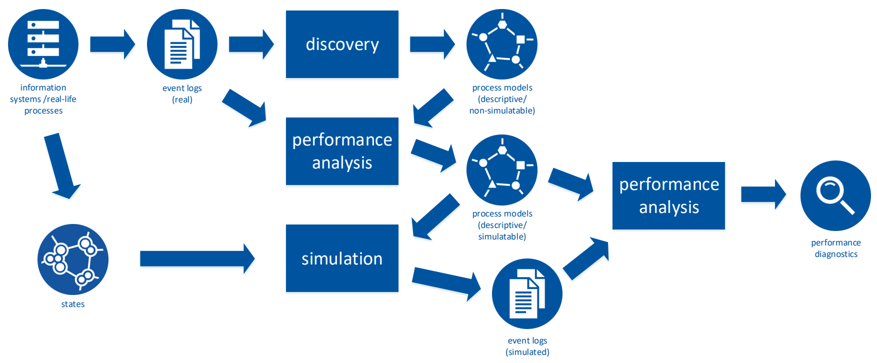


Figure 2: A more advanced scenario using a combination of simulation and process mining. Using a simulation model learned through process mining and an initialization using actual state information, it is possible to explore different "futures" (fast-forward capability).

forward button" into the future is provided (Aalst 2015, Rozinat, Wynn, Aalst, Hofstede, and Fidge 2009). For transient analysis the focus is on the initial part of future behavior, i.e., starting from the initial state the "near future" is explored. While for steady-state analysis the initial state is irrelevant and the simulation can be started without any cases in progress, this type of simulation relies on state information and a tight coupling between the information system and the simulation model. This is facilitated by process mining.

Figure 2 illustrates that new types of analysis are enabled by combining process mining and simulation. It is particularly interesting that *the difference between interpreting simulated behavior and real behavior is fading*. This facilitates data-driven exploration of "Ist" and "Soll" processes.

3 BASIC PROCESS MINING NOTIONS EXPLAINED THROUGH CONFORMANCE PROPOSITIONS

Process mining techniques focus on the relationship between observed behavior and modeled behavior. Therefore, we first formalize event logs (i.e., observed behavior) and process models (i.e., modeled behavior). To do this, we consider a very simple setting where we only focus on the control-flow, i.e., sequences of activities. Then, we discuss various quality notions and provide a set of conformance propositions.

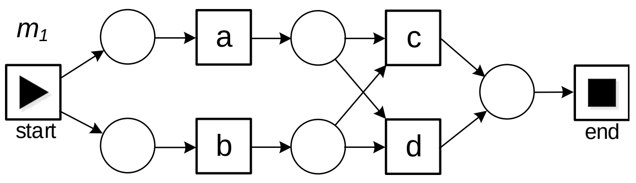
3.1 Event Logs and Process Models

The starting point for process mining is an event log. Each *event* in such a log refers to an *activity* possibly executed by a *resource* at a particular *time* and for a particular *case*. An event may have many more attributes, e.g., transactional information, costs, customer, location, and unit. Here, we focus on control-flow. Therefore, we only consider activity labels and the ordering of events within cases.

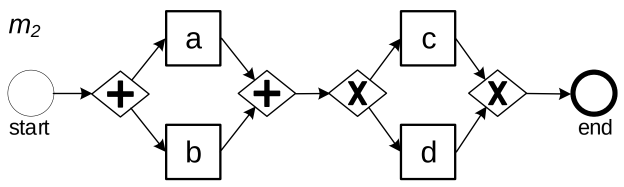
Definition 1 (Traces)**.** *$\mathcal{A}$ is the universe of* activities. *A* trace *$t \in \mathcal{A}^*$ is a sequence of activities. $\mathcal{T} = \mathcal{A}^*$ is the universe of traces.*

A trace $t = \langle a, b, a, b, a, b, c \rangle \in \mathcal{T}$ refers to 7 events belonging to the same case (i.e., $|t| = 7$). The events occurred in the order indicated, starting with a and ending with c. An event log refers to a collection of cases each represented by a trace.

Definition 2 (Event Log)**.** *$\mathcal{L} = \mathbb{B}(\mathcal{T})$ is the universe of event logs. An* event log *$l \in \mathcal{L}$ is a multiset of observed traces. $\widetilde{l} = \{t \in l\}$ is the set of traces appearing in $l \in \mathcal{L}$.*



(a) A Petri net model (with start and end transitions)



(b) A BPMN model allowing for the same behavior

Figure 3: Two process models m_1 and m_2 having the same behavior: $\widetilde{m_1} = \widetilde{m_2} = \{\langle a,b,c\rangle, \langle a,b,d\rangle, \langle b,a,c\rangle, \langle b,a,d\rangle\}$. m_1 is a Petri net with a special start activity (occurs once at the beginning) and a special end activity to signal the end of the trace. m_2 is a BPMN (Business Process Model and Notation) model.

An event log is a multiset of traces. Event log $l = [\langle a,b,c\rangle^5, \langle b,a,d\rangle^3, \langle a,b,d\rangle^2]$ refers to 10 cases (i.e., $|l| = 10$). Five cases are represented by the trace $\langle a,b,c\rangle$, three cases are represented by the trace $\langle b,a,d\rangle$, and two cases are represented by the trace $\langle a,b,d\rangle$. $l(t)$ is the number of times t appears in l, e.g., $l(\langle b,a,d\rangle) = 3$. We assume that the usual operators are defined for multisets. $l_1 \uplus l_2$ is the union of two multisets, $|l|$ is the number of elements, and $l_1 \setminus l_2$ is the difference. $l_1 \leq l_2$ means that l_1 is contained in l_2. For example, $[a^2,b] \leq [a^2,b^2,c]$, but $[a^2,b^3] \not\leq [a^2,b^2,c^2]$ and $[a^2,b^2,c] \not\leq [a^3,b^3]$. Next, we define process models where we distinguish between *representation* and *behavior* using the same simplistic setting (only control-flow).

Definition 3 (Process Model)**.** *$\mathcal{M}$ is the set of all process models. A* process model *$m \in \mathcal{M}$ allows for the set of traces denoted by $\widetilde{m} \subseteq \mathcal{T}$.*

$m \in \mathcal{M}$ could be represented in different modeling languages, e.g., Petri nets, BPMN models, UML activity diagrams, automata, and process trees. Here we abstract from the actual representation and focus on the behavior allowed by the model. $\widetilde{m} \subseteq \mathcal{T}$ denotes the set of traces possible according to the model. Figure 3 shows two process models that have the same behavior: $\widetilde{m_1} = \widetilde{m_2} = \{\langle a,b,c\rangle, \langle a,b,d\rangle, \langle b,a,c\rangle, \langle b,a,d\rangle\}$.

Definition 4 (Complement)**.** *$\overline{\widetilde{m}} = (\mathcal{T} \setminus \widetilde{m})$ is the complement of the set of traces allowed by model $m \in \mathcal{M}$. $\overline{\widetilde{l}} = (\mathcal{T} \setminus \widetilde{l})$ is the complement of the traces appearing in log $l \in \mathcal{L}$.*

$\overline{\widetilde{m}}$ is the set traces not allowed by the model. Note that $\widetilde{m}$ and $\overline{\widetilde{m}}$ partition the universe of traces $\mathcal{T}$ and may both contain infinitely many traces.

A discovery algorithm takes an event log as input and returns a process model. For example, m_1 and m_2 in Figure 3 could have been discovered based on event log $l = [\langle a,b,c\rangle^5, \langle b,a,d\rangle^3, \langle a,b,d\rangle^2]$. Ideally, the process model captures the (dominant) behavior observed, but also generalizes without becoming too imprecise. For example, m_1 and m_2 allow for trace $t = \langle b,a,c\rangle$ although this was never observed.

Definition 5 (Discovery Algorithm)**.** *A discovery algorithm can be described as a function $disc \in \mathcal{L} \to \mathcal{M}$ mapping event logs onto process models.*

We abstract from concrete discovery algorithms. Over 100 discovery algorithms have been proposed in literature (Aalst 2016). Merely as a reference to explain basic notions, we define three simple, but extreme, algorithms: $disc_{ofit}$, $disc_{ufit}$, and $disc_{nfit}$. Let $l \in \mathcal{L}$ be a log. $disc_{ofit}(l) = m_o$ such that $\widetilde{m}_o = \widetilde{l}$ produces an overfitting model that allows only for the behavior seen in the log. $disc_{ufit}(l) = m_u$ such that $\widetilde{m}_u = \mathcal{T}$ produces an underfitting model that allows for any behavior. $disc_{nfit}(l) = m_n$ such that $\widetilde{m}_n = \overline{\widetilde{l}}$ produces a non-fitting model that allows for all behavior *not* seen in the log.

3.2 Quality Dimensions

Since process mining focuses on the relationship between observed behavior and modeled behavior, it is important to quantify the relation between a log $l \in \mathcal{L}$ and model $m \in \mathcal{M}$. When learning a process model from event data, there is a trade-off between the following four quality dimensions (Aalst 2016): (1) *recall*: the discovered model should allow for the behavior seen in the event log (avoiding "non-fitting" behavior), (2) *precision*: the discovered model should not allow for behavior completely unrelated to what was seen in the event log (avoiding "underfitting"), (3) *generalization*: the discovered model should generalize the example behavior seen in the event log (avoiding "overfitting"), and (4) *simplicity*: the discovered model should be as simple as possible. The simplicity dimension refers to Occam's Razor: "one should not increase, beyond what is necessary, the number of entities required to explain anything". In the context of process mining, this is often operationalized by quantifying the complexity of the model (number of nodes, number of arcs, understandability, etc.). We do not consider the simplicity dimension in this paper, since we focus on behavior and abstract from the actual model representation. Recall is often referred to as *fitness* in process mining literature. Sometimes fitness refers to a combination of the four quality dimensions. To avoid later confusion, we use the term recall commonly used in pattern recognition, information retrieval, and (binary) classification.

In the remainder, we assume the existence of three functions: $rec()$, $prec()$, $gen()$. All three take a log and model as input and return a value between 0 and 1. The higher the value, the better. In process mining literature one can find many proposals for such functions. Here, we do not describe specific functions, but discuss their (desired) properties.

Definition 6 (Recall)**.** *A* recall measure $rec \in \mathcal{L} \times \mathcal{M} \rightarrow [0, 1]$ *aims to quantify the fraction of observed behavior that is allowed by the model.*

Definition 7 (Precision)**.** *A* precision measure $prec \in \mathcal{L} \times \mathcal{M} \rightarrow [0, 1]$ *aims to quantify the fraction of behavior allowed by the model that was actually observed.*

Definition 8 (Generalization)**.** *A* generalization measure $gen \in \mathcal{L} \times \mathcal{M} \rightarrow [0, 1]$ *aims to quantify the likelihood that new unseen cases will fit the model.*

Assume event log $l = [\langle a, c\rangle^5, \langle a, b, b, c\rangle^3, \langle a, b, b, b, b, b, c\rangle]$ containing 29 events relating to 9 cases. $m_o = disc_{ofit}(l) = \{\langle a, c\rangle, \langle a, b, b, c\rangle, \langle a, b, b, b, b, b, c\rangle\}$ allows only for the behavior seen in the log. $m_u = disc_{ufit}(l) = \mathcal{T}$ allows for any behavior. $m_n = disc_{nfit}(l) = \mathcal{T} \setminus \{\langle a, c\rangle, \langle a, b, b, c\rangle, \langle a, b, b, b, b, b, c\rangle\}$ allows for any behavior *not* seen in the log. The recall of models m_o and m_u is good because these models allow for all traces in event log l. The recall of m_n is poor because none of the observed traces is allowed. The precision of m_o is good because all behavior allowed has been observed. The precision of m_u and m_n is poor because most of the behavior allowed by these models was never observed. The generalization of m_u is good because it will also allow for unseen traces (e.g., $\langle a, b, c\rangle$ and $\langle a, b, b, b, c\rangle$). The generalization of m_o and m_n is poor because these models are unlikely to allow for a new trace generated by the process that also generated l. Table 1 summarizes the anticipated values for the different quality measures.

Table 1: Expected values for event log $l = [\langle a,c\rangle^5, \langle a,b,b,c\rangle^3, \langle a,b,b,b,b,b,c\rangle]$.

model	recall	precision	generalization
$m_o = \{\langle a,c\rangle, \langle a,b,b,c\rangle, \langle a,b,b,b,b,b,c\rangle\}$	good	good	poor
$m_u = \mathcal{T}$	good	poor	good
$m_n = \mathcal{T} \setminus \{\langle a,c\rangle, \langle a,b,b,c\rangle, \langle a,b,b,b,b,b,c\rangle\}$	poor	poor	poor

To explain the difference between recall and generalization, consider the event logs $l^1 = [\langle a,c\rangle^5, \langle a,b,b,c\rangle^3, \langle a,b,b,b,b,b,c\rangle]$, $l^2 = [\langle a,c\rangle^{10}, \langle a,b,b,c\rangle^6, \langle a,b,b,b,b,b,c\rangle^2]$, and $l^{100} = [\langle a,c\rangle^{500}, \langle a,b,b,c\rangle^{300}, \langle a,b,b,b,b,b,c\rangle^{100}]$. Note that l^2 is a duplication of the original event log. In l^{100} all traces appear 100 times as frequent as in l^1. Note that m_o (overfitting model), m_u (underfitting model), and m_n (non-fitting model) are the same for these three event logs (i.e., the models listed in Table 1). It seems reasonable to assume that given a model m, the recall and precision values will not be radically different for l^1, l^2, and l^{100}. However, generalization will be influenced by the absolute frequencies of traces in the event log even when the distribution does not change. Obviously, $gen(l^{100}, m_o) > gen(l^1, m_o)$. After observing 900 cases exhibiting only three unique traces, it is more likely that case 901 will again follow one of these three traces. After observing only 9 cases, this is less certain and it seems reasonable to think that for example trace $\langle a,b,c\rangle$ is still possible.

In the remainder, we use a "closed world assumption" with respect to the set of activities. We assume that $\mathcal{A}$ is limited to the activities that have been observed. Hence, models like $m_u = disc_{ufit}(l)$ and $m_n = disc_{nfit}(l)$ only refer to observed activities. It makes no sense to reason about activities that were never observed (although the propositions do not depend on this).

3.3 Conformance Propositions

Many recall measures have been proposed in literature (Aalst 2016, Rozinat and Aalst 2008, Medeiros, Weijters, and Aalst 2007, Aalst, Adriansyah, and Dongen 2012). In recent years, also several precision measures have been proposed (Adriansyah, Munoz-Gama, Carmona, Dongen, and Aalst 2015, Tax, Lu, Sidorova, Fahland, and Aalst 2018). Only few generalization measures have been proposed (Aalst, Adriansyah, and Dongen 2012). The goal of this paper is not to present new measures, but to define properties for such quality measures. Through this, we hope to facilitate a better understanding of process discovery and conformance checking. Moreover, these properties may help to choose an existing measure or to define new ones.

In the remainder, we provide *21 conformance propositions*. The Merriam-Webster dictionary defines the noun *proposition* as "an expression in language or signs of something that can be believed, doubted, or denied or is either true or false". Most of the conformance propositions have broad support from the community, i.e., there is consensus that these propositions should hold. These are marked with a "+". For example, the first two propositions are commonly accepted; the computation of a quality measure should be deterministic (**DetPro**$^+$) and only depend on behavioral aspects (**BehPro**$^+$). The latter is a design choice. We deliberately exclude simplicity notions. More controversial propositions are marked with a "0" (rather than a "+").

Proposition 1 (DetPro$^+$). *$rec()$, $prec()$, $gen()$ are deterministic functions, i.e., $rec(l,m)$, $prec(l,m)$, $gen(l,m)$ are fully determined by $l \in \mathcal{L}$ and $m \in \mathcal{M}$.*

Proposition 2 (BehPro$^+$). *For any $l \in \mathcal{L}$ and $m_1, m_2 \in \mathcal{M}$ such that $\widetilde{m}_1 = \widetilde{m}_2$: $rec(l,m_1) = rec(l,m_2)$, $prec(l,m_1) = prec(l,m_2)$, and $gen(l,m_1) = gen(l,m_2)$, i.e., the measures are fully determined by the behavior observed and the behavior allowed by the model (representation does not matter).*

3.4 Recall Propositions

First, we consider a few *recall propositions*. $rec \in \mathcal{L} \times \mathcal{M} \to [0,1]$ aims to quantify the fraction of observed behavior that is allowed by the model.

Proposition 3 (RecPro1$^+$). *For any $l \in \mathcal{L}$ and $m_1, m_2 \in \mathcal{M}$ such that $\widetilde{m}_1 \subseteq \widetilde{m}_2$: $rec(l, m_1) \leq rec(l, m_2)$.*

Proposition **RecPro1$^+$** states that extending the model to allow for more behavior can never result in a lower recall. Similarly, it cannot be the case that adding fitting behavior to the event logs, lowers recall (**RecPro2$^+$**). Adding non-fitting behavior to the log, cannot improve recall (**RecPro3$^+$**).

Proposition 4 (RecPro2$^+$). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \widetilde{m}$: $rec(l_1, m) \leq rec(l_2, m)$.*

Proposition 5 (RecPro3$^+$). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \overline{\widetilde{m}}$: $rec(l_1, m) \geq rec(l_2, m)$.*

For any natural number k: $l^k(t) = k \cdot l(t)$, e.g., if $l = [\langle a, b\rangle^3, \langle c\rangle^2]$, then $l^4 = [\langle a, b\rangle^{12}, \langle c\rangle^8]$. We use this notation to enlarge event logs without changing the distribution. One could argue that this should not influence recall (**RecPro4^0**), e.g., $rec([\langle a, b\rangle^3, \langle c\rangle^2], m) = rec([\langle a, b\rangle^{12}, \langle c\rangle^8], m)$. However, unlike the previous propositions, this requirement is debatable as is indicated by the "0" tag.

Proposition 6 (RecPro4^0). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$: $rec(l^k, m) = rec(l, m)$.*

Finally, we provide a proposition stating that recall should be 1 if all traces in the log fit the model (**RecPro5$^+$**). As a result, the empty log has recall 1 for any model.

Proposition 7 (RecPro5$^+$). *For any $l \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $\widetilde{l} \subseteq \widetilde{m}$: $rec(l, m) = 1$.*

Based on this proposition, $rec(l, disc_{ofit}(l)) = rec(l, disc_{ufit}(l)) = 1$ for any log l.

3.5 Precision Propositions

Precision ($prec \in \mathcal{L} \times \mathcal{M} \to [0,1]$) aims to quantify the fraction of behavior allowed by the model that was actually observed. In (Tax, Lu, Sidorova, Fahland, and Aalst 2018) several precision axioms were introduced. These partly overlap with the propositions below (but more are added and some are strengthened). **PrecPro1$^+$** states that removing behavior from a model that does not happen in the event log cannot lead to a lower precision. Adding fitting traces to the event log can also not lower precision (**PrecPro2$^+$**). However, adding non-fitting traces to the event log should not change precision (**PrecPro3^0**).

Proposition 8 (PrecPro1$^+$). *For any $l \in \mathcal{L}$ and $m_1, m_2 \in \mathcal{M}$ such that $\widetilde{m}_1 \subseteq \widetilde{m}_2$ and $\widetilde{l} \cap (\widetilde{m}_2 \setminus \widetilde{m}_1) = \emptyset$: $prec(l, m_1) \geq prec(l, m_2)$.*

Proposition 9 (PrecPro2$^+$). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \widetilde{m}$: $prec(l_1, m) \leq prec(l_2, m)$.*

Proposition 10 (PrecPro3^0). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \overline{\widetilde{m}}$: $prec(l_1, m) = prec(l_2, m)$.*

One could also argue that duplicating the event log should not influence precision because the distribution remains the same (**PrecPro4^0**), e.g., $prec([\langle a, b\rangle^{20}, \langle c\rangle^{20}], m) = prec([\langle a, b\rangle^{40}, \langle c\rangle^{40}], m)$.

Proposition 11 (PrecPro4^0). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$: $prec(l^k, m) = prec(l, m)$.*

If the model allows for the behavior observed and nothing more, precision should be maximal (**PrecPro5**$^+$). One could also argue that if all modeled behavior was observed, precision should also be 1 (**PrecPro6**^{0}). The latter proposition is debatable, because it implies that the non-fitting behavior cannot influence perfect precision. Consider for example extreme cases where the model covers just a small fraction of all observed behavior (or even more extreme situations like $\widetilde{m} = \emptyset$).

Proposition 12 (PrecPro5$^+$). *For any $l \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $\widetilde{m} = \widetilde{l}$: $prec(l, m) = 1$.*

Proposition 13 (PrecPro6^0). *For any $l \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $\widetilde{m} \subseteq \widetilde{l}$: $prec(l, m) = 1$.*

Based on proposition **PrecPro5**$^+$ or **PrecPro6**^{0}, $rec(l, disc_{ofit}(l)) = 1$ for any log l.

3.6 Generalization Propositions

Generalization ($gen \in \mathcal{L} \times \mathcal{M} \to [0, 1]$) aims to quantify the likelihood that new unseen cases will fit the model. This conformance dimension is a bit different than the other two dimensions, because it reasons about future *unseen* cases (i.e., not yet in the event log). If the recall is good and the log is complete with lots of repeating behavior, then future cases will most likely fit the model. Analogous to recall, model extensions cannot lower generalization (**GenPro1**$^+$), extending the log with fitting behavior cannot lower generalization (**GenPro2**$^+$), and extending the log with non-fitting behavior cannot improve generalization (**GenPro3**$^+$).

Proposition 14 (GenPro1$^+$). *For any $l \in \mathcal{L}$ and $m_1, m_2 \in \mathcal{M}$ such that $\widetilde{m}_1 \subseteq \widetilde{m}_2$: $gen(l, m_1) \leq gen(l, m_2)$.*

Proposition 15 (GenPro2$^+$). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \widetilde{m}$: $gen(l_1, m) \leq gen(l_2, m)$.*

Proposition 16 (GenPro3$^+$). *For any $l_1, l_2, l_3 \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $l_2 = l_1 \uplus l_3$ and $\widetilde{l_3} \subseteq \overline{\widetilde{m}}$: $gen(l_1, m) \geq gen(l_2, m)$.*

Duplicating the event log does not necessarily influence recall and precision. According to propositions **RecPro4**^{0} and **PrecPro4**^{0} this should have no effect on recall and precision. However, making the event log more redundant, should have an effect on generalization. For fitting logs, adding redundancy without changing the distribution can only improve generalization (**GenPro4**$^+$). For non-fitting logs, adding redundancy without changing the distribution can only lower generalization (**GenPro5**$^+$). Note that **GenPro4**$^+$ and **GenPro5**$^+$ are special cases of **GenPro6**^{0} and **GenPro7**^{0}. **GenPro6**^{0} and **GenPro7**^{0} consider logs where some traces are fitting and others are not. For a log where more than half of the traces is fitting, duplication can only improve generalization (**GenPro6**^{0}). For a log where more than half of the traces is non-fitting, duplication can only lower generalization (**GenPro6**^{0}).

Proposition 17 (GenPro4$^+$). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$ such that $\widetilde{l} \subseteq \widetilde{m}$: $gen(l^k, m) \geq gen(l, m)$.*

Proposition 18 (GenPro5$^+$). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$ such that $\widetilde{l} \subseteq \overline{\widetilde{m}}$: $gen(l^k, m) \leq gen(l, m)$.*

Proposition 19 (GenPro6^0). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$ such that $|[t \in l \mid t \in \widetilde{m}]| \geq |[t \in l \mid t \notin \widetilde{m}]|$: $gen(l^k, m) \geq gen(l, m)$.*

Proposition 20 (GenPro7^0). *For any $l \in \mathcal{L}$, $m \in \mathcal{M}$, and $k \geq 1$ such that $|[t \in l \mid t \in \widetilde{m}]| \leq |[t \in l \mid t \notin \widetilde{m}]|$: $gen(l^k, m) \leq gen(l, m)$.*

When the model allows for any behavior, clearly the next case will also be fitting (**GenPro8**^{0}). Nevertheless, it is marked as controversial because the proposition would also need to hold for an empty event log.

Proposition 21 (**GenPro8**^{0})**.** *For any $l \in \mathcal{L}$ and $m \in \mathcal{M}$ such that $\widetilde{m} = \mathcal{T}$: $gen(l,m) = 1$.*

This concludes this first inventory of conformance propositions. As mentioned, their purpose is twofold. First of all, they help to understand the goals and challenges of process discovery. Second, they provide a checklist for existing and future conformance measures. As demonstrated in (Tax, Lu, Sidorova, Fahland, and Aalst 2018) for precision, most of the existing approaches violate seemingly obvious requirements. This justifies the formulation of the above 21 conformance propositions.

4 TOWARDS A MORE REALISTIC SETTING

The propositions presented in Section 3 are based on a very simple setting where we know nothing about the underlying process and only see the event log. To provide another view on process discovery, we now provide a different setting where each trace has a likelihood. A stochastic process model is composed of a set of traces and a trace likelihood function. The focus is still on control-flow, but we add probabilities to show that "behavior is not binary".

Definition 9 (Trace Likelihood Function)**.** $\Pi = \mathcal{T} \to [0,1]$ *is the set of all* trace likelihood functions. $\pi \in \Pi$ *assigns a likelihood $\pi(t)$ to any trace $t \in \mathcal{T}$ such that $\sum_{t \in \mathcal{T}} \pi(t) = 1$.*

Note that the number of possible traces may be infinite, but it is easy to see that the universe of traces $\mathcal{T}$ is countable if $\mathcal{A}$ is finite (order by the length of the trace and then in lexicographical order). For pragmatic reasons, we can also restrict $\mathcal{T}$ to traces of a maximal length (e.g., the maximal length seen in any log). For any set of traces $X \subseteq \mathcal{T}$, $\pi(X) = \sum_{t \in X} \pi(t)$ is the probability that a run of the process π yields a trace in X. We assume that traces are sampled independently. When considering time this is not realistic (due to queueing), but for the control-flow (routing of a case through the model) this is a common assumption.

Definition 10 (Stochastic Process Model)**.** $\mathcal{S} = \mathcal{M} \times \Pi$ *is the set of all* stochastic process models. *A stochastic process model $s = (m, \pi) \in \mathcal{S}$ combines a model m and trace likelihood function π.*

The relation between m and π in $s = (m, \pi) \in \mathcal{S}$ is not very strict. Sometimes we will require $\widetilde{m} \subseteq \{t \in \mathcal{T} \mid \pi(t) > 0\}$ (all modeled traces are possible), $\{t \in \mathcal{T} \mid \pi(t) > 0\} \subseteq \widetilde{m}$ (model allows for all traces possible), or even $\widetilde{m} = \{t \in \mathcal{T} \mid \pi(t) > 0\}$. Model m may be an abstraction of the real process and leave out unlikely behavior. Due to simplification, the model may also allow for traces that cannot happen (e.g., particular interleavings or loops).

Next, we assume a highly idealized setting where we have a real stochastic process model $s_r = (m_r, \pi_r) \in \mathcal{S}$ and a discovered or manually designed stochastic process model $s_d = (m_d, \pi_d) \in \mathcal{S}$ (also see (Aalst 2013)). π_r is the real underlying stochastic process and m_r is the process we would like to obtain. As mentioned, m_r may abstract from unlikely behavior or include behavior that is impossible. $s_d = (m_d, \pi_d)$ is the model that was discovered (or obtained in some other way) and we would like to compare it with the real stochastic process model $s_r = (m_r, \pi_r)$. Although s_d could have been hand-made, we just refer to it as the discovered model.

This idealized setting leads to very natural notions of precision and recall. For example, $\pi_r(\widetilde{m}_r)$ is the coverage of m_r according to π_r, $\pi_r(\widetilde{m}_d)$ is the coverage of m_d according to π_r, $\pi_d(\widetilde{m}_r)$ is the coverage of m_r according to π_d, and $\pi_d(\widetilde{m}_d)$ is the coverage of m_d according to π_d.

Definition 11 (Precision and Recall)**.** *Let $s_r = (m_r, \pi_r), s_d = (m_d, \pi_d) \in \mathcal{S}$ be two stochastic process models (real and discovered).*

$$rec(s_r, s_d) = \frac{\pi_r(\widetilde{m}_d \cap \widetilde{m}_r)}{\pi_r(\widetilde{m}_r)} \qquad\qquad prec(s_r, s_d) = \frac{\pi_d(\widetilde{m}_d \cap \widetilde{m}_r)}{\pi_d(\widetilde{m}_d)} \tag{1}$$

To compute *recall*, we compare the traces allowed by both models ($\widetilde{m}_d \cap \widetilde{m}_r$) with the traces allowed by the real model ($\widetilde{m}_r$). π_r is used to quantify the fractions of traces. To compute *precision*, we compare the traces allowed by both models ($\widetilde{m}_d \cap \widetilde{m}_r$) with the traces allowed by the discovered model ($\widetilde{m}_d$). π_d is used to quantify the fractions of traces. Note that it does not make any sense to use π_r when computing precision. The behavior in $\widetilde{m}_d \setminus \widetilde{m}_r$ will be unlikely according to π_r. Hence, precision based on π_r would always yield a high value even when the discovered model allows for lots of additional behavior.

Definition 11 computes recall ($rec(s_r, s_d)$) and precision ($prec(s_r, s_d)$) by making the assumption that the real process is known. In real-life settings, this is not realistic. However, the definitions can be used when the ground truth is known (e.g., in process mining challenges and to evaluate discovery algorithms).

If $s_r = (m_r, \pi_r)$ is unknown, we can try to approximate it by a log-based estimate of the real process. Of course, this is very a crude approximation when only a fraction of the possible traces has been observed.

Definition 12 (Log-Based Precision and Recall)**.** *Let $l \in \mathcal{L}$ and $s_l = (m_l, \pi_l), s_d = (m_d, \pi_d) \in \mathcal{S}$ such that $\pi_l(t) = \frac{l(t)}{|l|}$ for $t \in \mathcal{T}$ and $\widetilde{m}_l = \widetilde{l}$.*

$$rec(s_l, s_d) = \frac{\pi_l(\widetilde{m}_d \cap \widetilde{m}_l)}{\pi_l(\widetilde{m}_l)} \qquad\qquad prec(s_l, s_d) = \frac{\pi_d(\widetilde{m}_d \cap \widetilde{m}_l)}{\pi_d(\widetilde{m}_d)} \tag{2}$$

It is interesting to investigate under which conditions the above precision and recall metrics satisfy the different propositions. Moreover, assuming that π_r and m_r are known may help to understand the interplay between generalization on the one hand and recall and precision on the other hand. See (Aalst 2018) for an analysis of a few simple conformance measures based on the propositions while including probabilities.

The above formulas also reveal the importance of the trace likelihood function. Any process model that has loops will allow for infinitely many traces. Therefore, we cannot reason about fractions of traces. Moreover, the likelihood of traces is very different. Consider the process that just does a's and after each a stops with a 0.5 probability. Hence, trace $\langle a \rangle$ has a likelihood of 0.5. Trace $\langle a, a, a, a, a \rangle$ has a likelihood of 0.03125 showing that it makes no sense to count traces without using some trace likelihood function.

5 OUTLOOK

In this paper, we discussed the relation between simulation and process mining. As shown, they complement each other well. Process mining can be made more *forward-looking* by using simulation. Moreover, process mining can *breathe new life into simulation research*. On the one hand, process mining can be used to automate parts of the modeling process and create much better fact-based simulation models. On the other hand, it provides ways to view the real processes and the simulated process in a unified manner. To introduce process mining, we followed a rather unusual approach. Instead of describing existing process mining algorithms, we reasoned about quality criteria using so-called *conformance propositions*. By providing 21 conformance propositions, we revealed the challenges that process discovery techniques are facing. Moreover, we related modeled behavior and real behavior (event logs or ground-truth models) in a novel way. This could serve as a basis for revisiting recall, precision, and generalization measures.

REFERENCES

Aalst, W. van der 2013. "Mediating Between Modeled and Observed Behavior: The Quest for the "Right" Process". In *IEEE International Conference on Research Challenges in Information Science (RCIS 2013)*, pp. 31–43, IEEE Computing Society.

Aalst, W. van der 2015. "Business Process Simulation Survival Guide". In *Handbook on Business Process Management 1*, edited by J. vom Brocke and M. Rosemann, International Handbooks on Information Systems, pp. 337–370, Springer-Verlag, Berlin.

Aalst, W. van der 2016. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin.

Aalst, W. van der 2018. "Relating Process Models and Event Logs: 21 Conformance Propositions". In *Algorithms & Theories for the Analysis of Event Data (ATAED 2018)*. CEUR Workshop Proceedings. Bratislava, Slovakia, June 2018.

Aalst, W. van der, A. Adriansyah, and B. van Dongen. 2012. "Replaying History on Process Models for Conformance Checking and Performance Analysis". *WIREs Data Mining and Knowledge Discovery* vol. 2 (2), pp. 182–192.

Aalst, W. van der, and C. Stahl. 2011. *Modeling Business Processes: A Petri Net Oriented Approach*. MIT press, Cambridge, MA.

Adriansyah, A., J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst. 2015. "Measuring Precision of Modeled Behavior". *Information Systems and e-Business Management* vol. 13 (1), pp. 37–67.

Dahl, O., and K. Nygaard. 1966, Sept. "SIMULA: An ALGOL Based Simulation Language". *Communications of the ACM* vol. 1, pp. 671–678.

Kleijnen, J., and W. Groenendaal. 1992. *Simulation: A Statistical Perspective*. John Wiley and Sons, New York.

Law, A., and D. Kelton. 1982. *Simulation Modeling and Analysis*. McGraw-Hill, New York.

Medeiros, A., A. Weijters, and W. van der Aalst. 2007. "Genetic Process Mining: An Experimental Evaluation". *Data Mining and Knowledge Discovery* vol. 14 (2), pp. 245–304.

Rozinat, A., and W. van der Aalst. 2008. "Conformance Checking of Processes Based on Monitoring Real Behavior". *Information Systems* vol. 33 (1), pp. 64–95.

Rozinat, A., M. Wynn, W. van der Aalst, A. ter Hofstede, and C. Fidge. 2009. "Workflow Simulation for Operational Decision Support". *Data and Knowledge Engineering* vol. 68 (9), pp. 834–850.

Tax, N., X. Lu, N. Sidorova, D. Fahland, and W. van der Aalst. 2018. "The Imprecisions of Precision Measures in Process Mining". *Information Processing Letters* vol. 135, pp. 1–8.

AUTHOR BIOGRAPHY

WIL M.P. VAN DER AALST is a professor at RWTH Aachen University where he chairs the Process and Data Science (PADS) group. His research interests lie in Data Science, Process Science, Process Mining, Business Process Management, Data Mining, Process Discovery, Conformance Checking, and Simulation. His website is www.vdaalst.com.