Abstract and Compare: A Framework for Defining Precision Measures for Automated Process Discovery

Adriano Augusto[1,2], Abel Armas-Cervantes[2], Raffaele Conforti[2], Marlon Dumas[1], Marcello La Rosa[2], and Daniel Reissner[2]

[1] University of Tartu, Estonia
{adriano.augusto, marlon.dumas}@ut.ee
[2] The University of Melbourne, Australia
{raffaele.conforti, marcello.larosa, abel.armas}@unimelb.edu.au

Abstract. Automated process discovery techniques allow us to extract business process models from event logs. The quality of process models discovered by these techniques can be assessed with respect to various quality criteria related to simplicity and accuracy. One of these criteria, namely *precision*, captures the extent to which the behavior allowed by a discovered process model is observed in the log. While numerous measures of precision have been proposed in the literature, a recent study has shown that none of them fulfils a set of five axioms that capture intuitive properties behind the concept of precision. In addition, several existing precision measures suffer from scalability issues when applied to models discovered from real-life event logs. This paper presents a versatile framework for defining precision measures based on behavior abstractions. The key idea is that a precision measure can be defined by three ingredients: a function that abstracts a process model (e.g. as a transition system), a function that does the same for an event log, and a function that compares the behavior abstraction of the model with that of the log. We show empirically that different instances of this framework allow us to strike different tradeoffs between scalability and sensitivity. We also show that two instances of the framework based on lossless abstraction functions yield a precision measure that fulfils all the above-mentioned axioms.

1 Introduction

Modern enterprise information systems store detailed records of the execution of the business processes they support, such as records of the creation of process instances (a.k.a. *cases*), the start and completion of tasks, and other events associated with a case. These records can generally be extracted as event logs consisting of a set of traces, each trace itself consisting of a sequence of events associated with a case. Automated process discovery techniques [3] allow us to extract process models from such event logs. The quality of process models discovered in this way can be assessed with respect to several quality criteria related to simplicity and accuracy.

Two commonly used criteria for assessing accuracy are fitness and precision. *Fitness* captures the extent to which the behavior observed in an event log is allowed by the discovered process model (i.e. Can the process model generate every trace observed in the event log?). Reciprocally, *precision* captures the extent to which the behavior allowed by a discovered process model is observed in the event log. A low precision indicates that the model under-fits the log, i.e. it can generate traces that are unrelated

or only partially related to traces observed in the log, while a high precision indicates that it over-fits (i.e. it can only generate traces in the log and nothing more).[1]

While several precision measures have been proposed, a recent study has shown that none of them fulfils a set of five axioms that capture intuitive properties behind the concept of precision [20]. In addition, most of the existing precision measures suffer from scalability issues when applied to models discovered from real-life event logs.

This paper presents a framework for defining precision measures based on behavior abstraction and comparison. In this framework, a precision measure is defined by three ingredients: a function that constructs a behavior abstraction from a process model, a function that does the same for an event log, and a function that computes the difference between the behavior abstraction of the model and that of the log. The paper studies instances of this framework where the behavior abstractions are Transitions Systems (TS). In particular, the paper proposes a family of measures based on k^{th}-order Markovian abstractions and shows that this family of measures fulfils four of the abovementioned axioms, and all five axioms for a sufficiently large k (dependent on the process model). The paper also studies an automata-based measure that fulfils all five axioms.

In addition to analysing formal properties of the proposed measures, the paper empirically compares them using: (i) a set of process models previously used to assess the suitability of precision measures; and (ii) a set of process models discovered from 12 real-life event logs using three automated process discovery techniques. The evaluation shows that the k^{th}-order Markovian measures allow us to strike a tradeoff between scalability and sensitivity (i.e. ability to distinguish between behaviorally similar models).

The rest of the paper is structured as follows. Section 2 introduces existing precision measures as well as the five axioms proposed previously. Section 3 introduces the framework, while Sections 3 and 4 discuss instances thereof. Finally, Section 5 presents the empirical evaluation and Section 6 draws conclusions and directions for future work.

2 Background and Related Work

One of the earliest precision measures was proposed by Greco et al. [7], based on the *set difference* (SD) between the model behavior and the log behavior, each represented as a set of traces. Despite it closely operationalizes the definition of precision, this measure is not suitable for models with cycles since in these cases the precision is always zero.

Later, Rozinat and van der Aalst [17] proposed the *advanced behavioral appropriateness* (ABA) precision. The ABA precision is based on the comparison between the sets of activity pairs that sometimes but not always follow each other, and the set of activity pairs that sometimes but not always precede each other. The comparison is performed on the sets extracted both from the model and the log behaviors. The ABA precision is non-deterministic [20] and it does not scale to large models. Moreover, this precision is undefined for models with no routing behavior (i.e. models without concurrency or conflict relations).

De Weerdt et al. [6] proposed the *negative events* precision measure (NE). This method works by inserting inexistent (so-called negative) events to enhance the traces in the log. A negative event is inserted after a given prefix of a trace if this event is never observed preceded by that prefix anywhere in the log. The traces extended with negative events are then replayed on the model. If the model can parse some of the negative events, it means that the model has additional behavior. This approach is however heuristic: it does not guarantee that all additional behavior is identified.

[1] A third accuracy criterion in automated process discovery is *generalization*: the extent to which the process model captures behavior that, while not observed in the log, is implied by it.

Muñoz-Gama and Carmona [15] proposed the *escaping edges* (ETC) precision. Using the log behavior as reference, it builds a *prefix automaton* and, while replaying the process model behavior on top of it, counts the number of *escaping edges*, i.e. edges not in the prefix automaton which represent extra behavior of the process. Subsequently, to improve the robustness of the ETC precision for logs containing non-fitting traces, the ETC precision evolved into the *alignments-based ETC* precision (ETC_a) [1] where the replay is guided by alignments.

Despite its robustness, ETC_a does not scale well to real-life datasets. To address this issue, Leemans et al. [12] proposed the *projected conformance checking* (PCC) precision. This precision, starting from the log behavior and the model behavior builds a projected automaton (an automaton where a reduced number of activities are encoded) from each of them, i.e. A_l and A_m. These two automata are then used to generate a third automaton capturing their common behavior, i.e. $A_{l,m}$. The precision value is then computed as the ratio between the number of outgoing edges of each state in $A_{l,m}$ and the number of outgoing edges of the corresponding states occurring in A_m.

Finally, van Dongen et al. [22] proposed the *anti-alignment* precision (AA). This measure analyses the anti-alignments of the process model behavior to assess the model's precision. An anti-alignment of length n is a trace in the process model behavior of length at most equal to n, which maximizes the Levenshtein distance from all traces in the log.

In a recent study, Tax et al. [20] proposed 5 axioms to capture intuitive properties behind the concept of precision advising that any precision measure should fulfill these axioms. Before presenting them, we introduce preliminary concepts and notations.

Definition 1. ***[Trace]*** *Given a set of activity labels Σ, we define a trace on Σ as a sequence $\tau_\Sigma = \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, such that $\forall 1 \leq i \leq n, t_i \in \Sigma$. Furthermore, we denote with $\tau[i]$ the activity label in position i, and we use the symbol Γ_Σ to refer the universe of traces on Σ. With abuse of notation, from here on, we refer to any $t \in \Sigma$ as an activity instead of an activity label.*

Definition 2. ***[Subtrace]*** *Given a trace $\tau_\Sigma = \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, with the notation $\tau^{i \to j}$, we refer to the subtrace $\langle t_i, t_{i+1}, \ldots, t_{j-1}, t_j \rangle$, where $0 < i < j \leq n$. Additionally, we extend the subset operator to traces. Given two traces τ_Σ, and $\bar{\tau}_\Sigma$, $\bar{\tau}_\Sigma$ is contained in τ_Σ, i.e. $\bar{\tau}_\Sigma \subset \tau_\Sigma$, if and only if $\exists i, j \in \mathbb{N} \mid \tau^{i \to j} = \bar{\tau}_\Sigma$.*

Definition 3. ***[model behavior]*** *Given a process model P (regardless of its representation) and being Σ the set of its activities. We refer to the model behavior as $\mathscr{B}_P \subseteq \Gamma_\Sigma$, where $\forall \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle \in \mathscr{B}_P$ there exists an execution of P that allows to execute the sequence of activities $\langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, with t_1 being the first activity executed, and t_n the last.*

Definition 4. ***[Event Log Behavior]*** *Given a set of activities Σ, an event log L is a finite multiset of traces defined over Σ. The event log behavior of L is defined as $\mathscr{B}_L = support(L)$.*[2]

Definition 5. ***[Precision Axioms]***

- ***Axiom-1.*** *A precision measure is a deterministic function $prec : \mathscr{L} \times \mathscr{P} \to \mathbb{R}$. Where, $\mathscr{L}$ is the universe of event logs, and $\mathscr{P}$ is the universe of processes.*
- ***Axiom-2.*** *Given two process models fully containing the behaviour of a log, if the behavior or the former is equal to or contained in the behavior of the latter, the precision value of the latter must be equal to or lower then the precision value of the former. Formally, let P_1, P_2 be two processes and L be an event log. If $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2} \Longrightarrow prec(L, P_1) \geq prec(L, P_2)$.*
- ***Axiom-3.*** *Given two process models fully containing the behaviour of a log, if the behavior or the former is contained in the behavior of the latter, the precision value of the latter must be lower then the precision value of the former. Formally, let P_1, P_2 be two processes and L an event log. If $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2} = \Gamma_\Sigma \Longrightarrow prec(L, P_1) > prec(L, P_2)$.*

[2] The support of a multiset is the set containing the distinct elements of the multiset.

- ***Axiom-4.*** *Given two process models, if the behavior or the former is equal to the behavior of the latter, their precision values must be equal. Formally, let P_1, P_2 be two processes and L an event log. If $\mathscr{B}_{P_1} = \mathscr{B}_{P_2} \implies prec(L, P_1) = prec(L, P_2)$.*
- ***Axiom-5.*** *Given two logs, having their behavior fully contained in the behavior of a process model, if the behavior or the former is equal to or contained in the behavior of the latter, the precision value of the model measured over the first log must be equal to or lower then the precision value measured over the second log. Formally, let P be a process and let L_1, L_2 be two event logs. If $\mathscr{B}_{L_1} \subseteq \mathscr{B}_{L_2} \subseteq \mathscr{B}_P \implies prec(L_2, P) \geq prec(L_1, P)$.*

Tax et al. [20] showed that none of the existing measures fulfils all the axioms.

3 Framework

We propose a framework to decompose a precision measure into three functions: 1) a function to abstract the behavior of a process model, 2) a function to abstract the behavior recorded in an event log, and 3) a function to compare the two behavioral abstractions. Thus, we set *behavior abstraction* and *comparison type* as the two dimensions characterizing our framework.

The proposed framework is displayed in Fig. 1 and is pre-populated with different variants of two novel precision measures introduced in this paper (Sections 4 and 5). The framework is founded on two axes: *behavior abstraction* – exact or approximate – and *comparison type* – structural or behavioral. This framework can be used to study the trade-offs of existing precision measures, for instance, to see impact in the coarseness of the precision measure and computation time by using approximate behavioral abstractions, in comparison to exact behavioral abstractions (and, similarly, for structure-based versus behavioral-based comparison techniques). Below we describe each of the axes of our framework in more detail.

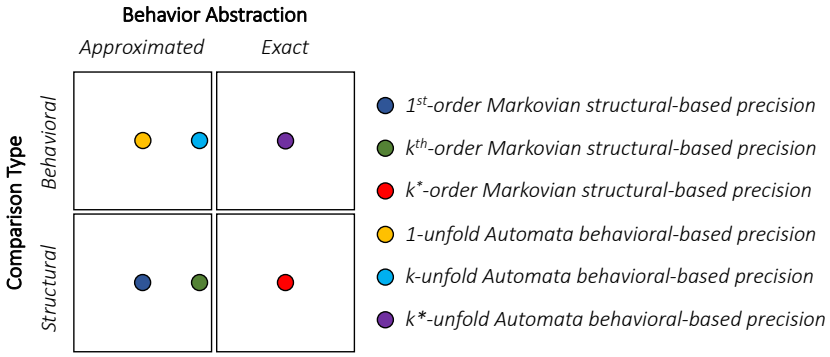


Fig. 1: Proposed framework and precision measures.

Behavior Abstraction The *behavior abstraction dimension* covers the first two components of a precision measure: the function to abstract the process model behavior and the function to abstract the behavior captured in an event log. Note that even though both, models and event logs, could describe essentially the same behavior, event logs will always contain a *finite amount* of behavior, which is not always true for process models. In the latter case, process models can capture an *infinite amount* of behavior, for instance, when containing loops. Nevertheless, in order to compare the behavior of a model and a log, it is necessary to use a unified behavioral abstraction to represent

both. Examples of some behavioral abstractions that can be used for encoding the models and event logs include: collections of traces (i.e. *trace abstraction*), Markov models, finite state machines, and automatons. Note, however, that the encoding of the behavior can be *exact*, when no alteration of the behavior occurs (neither addition nor deletion of behavior occurs), or *approximate*, when some behavior either added or removed.

Despite the fact that many behavioral abstractions are available in the literature, in our study we refer only to transition systems-based (TS) abstractions. We remark that the use of TS abstractions in process mining has been widely used. For example, [21] puts forward the possibility of using Markovian abstractions (namely *maximal horizon* abstractions) to predict the completion time of running process instances, though it neither considers their application in process discovery nor conformance checking.

Comparison Type The *comparison type dimension* corresponds to the third ingredient of a precision measure: the function to compare the behavioral abstractions of both the process model and the log. We distinguish two types of comparison that can be applied over a behavioral abstraction: *structural* and *behavioral*. In the first case, a *structural comparison* aims at identifying the differences between the two abstractions focusing exclusively on the topology of the abstractions (e.g. considering the abstractions as graphs). Examples of structural comparisons include *set difference* and *graph-based matchings*. On the other hand, a *behavioral comparison* aims at finding differences related to behavior, e.g., by focusing on wether an abstraction can be replayed over the other (simulation), and detecting and characterising the cases where such a replay is not possible; the *simulation-based graph similarity* of Sokolsky et al.[19] is a case in point.

The selection of a behavioral abstraction directly influences the selection of the type of comparison to be performed. For instance, by considering trace abstractions, we can choose a structural comparison based on set difference (as proposed by Greco et al. [7]), but not one based on graph matching. Instead, when considering TSs, we can choose comparisons either based on set difference operators (e.g. either on the sets of states or edges), graph matching or simulation graph similarity.

4 Markovian Structural-based Precision (MPS)

We present a novel precision measure based on Markovian abstractions. Following the structure of the proposed framework, we first discuss the behavior abstraction, then we illustrate the comparison type and the computation of the precision measure.

4.1 Behavioral Abstraction

A Markovian abstraction (M^k-abstraction) is composed by a set of states and edges, where every state represents a (sub)trace, and every edge connects two states, such that the (sub)trace represented by the source state occurs before the (sub)trace represented by the target state. Every state in this representation is unique, in the sense that there are no two events representing the same (sub)trace. This abstraction is defined w.r.t. a given order k, which defines the size of the (sub)traces encoded in the states. An M^k-abstraction contains a fresh state $-$ representing the sink and source of the M^k-abstraction. Intuitively, a state is either a trace of length less-or-equal-to k or a subtrace of length k. The edges connect states whose traces' overlapping is still either a trace or a subtrace of length $k+2$ in the input behavior. The M^k-abstraction captures how all the traces of the input behavior evolves in chunks of length k. The definition below shows the detailed construction of $M^k - abstraction$ from a given $\mathscr{B}_X$.

Definition 6. ***[k^{th}-order Markovian Abstraction]*** *Given a set of traces $\mathscr{B}_X$, the k-order Markovian Abstraction (M^k-abstraction) is the graph $M^k_X = (S,E)$ where S is a set of the states and $E \subseteq S \times S$ is a set edges between the states, such that (s.t.):*

- $S = \{-\} \cup \{\tau \in \mathscr{B}_X \ : \ |\tau| \leq k\} \cup \{\tau^{i \to j} \ : \ \exists \tau \in \mathscr{B}_X \ s.t. \ |\tau| > k \ \wedge \ \left|\tau^{i \to j}\right| = k\}$
- $E = \{(-,\tau),(\tau,-) : |\tau| \leq k\} \cup \{(-,\tau^{x \to y}) : \exists \tau' \in \mathscr{B}_X \ s.t. \ \tau^{x \to y} = \tau'^{1 \to k}\} \cup \{(\tau^{x \to y},-) : \exists \tau' \in \mathscr{B}_X \ s.t. \ \tau^{x \to y} = \tau'^{(|\tau'|-k+1) \to |\tau'|}\} \cup \{(\tau^{g \to h}, \tau^{i \to j}) : \tau^{g \to h} \oplus \tau^{i \to j}[j-i+1] = \tau^{g \to h}[1] \oplus \tau^{i \to j} \ \wedge \ \exists \tau' \in \mathscr{B}_X \ s.t. \ \tau^{g \to h} \oplus \tau^{i \to j}[j-i+1] = \tau'^{x \to y} \ for \ 1 \leq x < y \leq |\tau'|\} \ for \ all \ \tau \in \mathscr{B}_X.$

Note that, M^1-abstraction is equivalent to a *directly-follows graph* (a well-known behavior abstraction used as starting point by many process discovery approaches [10, 4, 23, 24]). Instead, if k approaches to infinite then M^∞-abstraction is equivalent to listing all the traces. The M^k-abstraction of a process model can be built from its reachability graph by replaying it. The time complexity of such operation strictly depends on k, and it ranges from polynomial time ($k = 1$) to exponential time for greater values of k. Instead, the M^k-abstraction of an event log can be built always in polynomial time, since the log behavior is a finite set of traces.

Varying the order k of the M^k-abstraction we can tune the level of behavior approximation. For example, let us consider the event log L^* as in Tab. 1, and the Process-X (P_x) in Fig. 2b. Their respective M^1-abstractions: $M^1_{L^*}$ and $M^1_{P_x}$ are shown in Fig. 3d and 3b. We can notice that $M^1_{L^*} = M^1_{P_x}$, though $\mathscr{B}_{P_x}$ is infinite whilst $\mathscr{B}_{L^*}$ is not. This happens because the M^1-abstraction over-approximate both L^* and P_x behaviors. However, if we increase k the approximation reduces and the behavioral differences between L^* and P_x can be detected, as shown in Fig. 4d and 4b. We remark that for k equal to the longest trace in the log, the behavior abstraction of this latter is exact. Unfortunately, a similar reasoning cannot be done for a model behavior, since the longest trace may be infinite.

4.2 Comparison

We introduce the concept of graph matching algorithm, which is the core of the comparison of the M^k-abstractions representing process and log behaviors.

Traces
$\langle a,a,b \rangle$
$\langle a,b,b \rangle$
$\langle a,b,a,b,a,b \rangle$

Table 1: Log L^*.

Definition 7. ***[Weighted Edge-based Graph Matching Algorithm (GMA)]*** *Let $G_1 = (N_1,E_1)$ and $G_2 = (N_2,E_2)$ be two graphs and $\mathscr{I}_C : E_1 \to (E_2 \cup \{\varepsilon\})$ be a mapping function given by a deterministic graph matching algorithm. The cost associated to a single match is given by $C : E_1 \times (E_2 \cup \{\varepsilon\}) \to [0,1]$. The element ε is mapped to every edge $e \in E_1$ when the graph matching algorithm did not find a matching for e in E_2. Furthermore, the following relation hold: $\forall e_1,e_2 \in E_1 . \mathscr{I}_C(e_1) = \mathscr{I}_C(e_2) \iff \mathscr{I}_C(e_1) = \varepsilon \wedge \mathscr{I}_C(e_2) = \varepsilon$;*

Given a GMA $\mathscr{I}_C$, an event log L and a process P as inputs, the k^{th}-order Markovion abstraction-based precision (hereby MSP^k) is estimated applying equation 1.

$$MSP^k(L,P) = 1 - \frac{\Sigma_{e \in E_1} C(e, \mathscr{I}_C(e))}{|E_P|} \tag{1}$$

The selected GMA for the implementation of our MSP^k is an adaptation of the Hungarian method [9], where: the cost of a match between two edges is defined as the average of the Levenshtein distance between the source states and the target states; and the final matching is the one minimising the total costs of the matches.

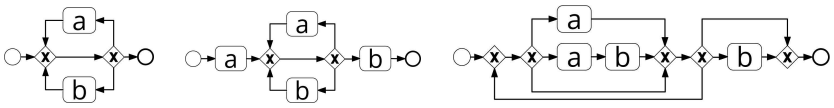


Fig. 2: Examples of processes in BPMN notation.

Figure 2 shows three models in BPMN notation. Their respective Markovian abstractions are captured in Fig. 3 and 4, for $k = 1$ and $k = 2$. We can observe that by increasing k, the behavior approximation decreases. Consequently, also the MSP^k decreases achieving a finer result. Although we acknowledge that the value returned by MSP^k strictly depends on k, in Section 6, we empirically show that: given two processes such that for $k = \bar{k}$ a precision ranking is identified, for any $k > \bar{k}$ the precision ranking of the two process does not change.

We now turn our attention to show that our Markov models-based precision measure fulfils the axioms presented in Section 2. For the remaining part of the section, let L_x be a log, P_x be a process model, and $M^k_{L_x} = (S_{L_x}, E_{L_x}, \phi_{L_x})$ and $M^k_{P_x} = (S_{P_x}, E_{P_x}, \phi_{P_x})$ be the M^k-abstractions of the log and the model, respectively.

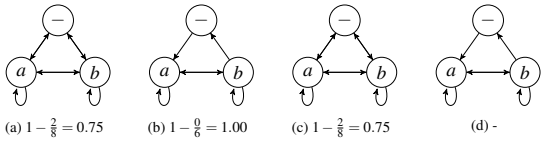


Fig. 3: From left to right, the M^1-abstraction of the Flower Process, Process-X, Process-Y and the event log L^*. The respective labels report the value of their MSP^1.

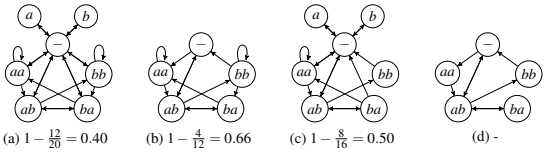


Fig. 4: From left to right, the M^2-abstraction of the Flower Process, Process-X, Process-Y and the event log L^*. The respective labels report the value of their MSP^2.

- **Axiom-1.** $MSP^k(L,P)$ is a deterministic function. Given a log L and a process P, The construction of M^k_L and M^k_P is fully deterministic (see Definition 6) for $\mathscr{B}_P$ and $\mathscr{B}_L$. Furthermore, the graph matching algorithm $\mathscr{I}_C$ is deterministic, and thus the function $MSP^k(L,P)$ of E_P, E_L and GMA, is also deterministic with codomain $\mathbb{R}$.
- **Axiom-2.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2}$, then $MSP^k(L,P_1) \geq MSP^k(L,P_2)$. By construction, the following relation holds, $E_L \subseteq E_{P_1} \subseteq E_{P_2}$. Then, we distinguish two possible cases:
  1. if $E_{P_1} = E_{P_2}$, then it follows straightforward $MSP^k(L,P_1) = MSP^k(L,P_2)$ from the proof of Axiom-1.

2. if $E_{P_1} \subset E_{P_2}$, then $E_L \subset E_{P_2} \wedge \ (|E_{P_2}| - |E_{P_1}|) > 0$. In this case, we show that $MSP^k(L)(P_2) - MSP^k(L)(P_1) < 0$ is always true.

$$1 - \frac{\Sigma_{e_2 \in E_{P_2}} C(e_2, \mathscr{I}_C(e_2))}{|E_{P_2}|} - \left(1 - \frac{\Sigma_{e_1 \in E_{P_1}} C(e_1, \mathscr{I}_C(e_1))}{|E_{P_1}|}\right) = \frac{\Sigma_{e_1 \in E_{P_1}} C(e_1, \mathscr{I}_C(e_1))}{|E_{P_1}|} - \frac{\Sigma_{e_2 \in E_{P_2}} C(e_2, \mathscr{I}_C(e_2))}{|E_{P_2}|} < 0$$

Since $\forall e_1 \in E_{P_1} \cap E_L \Longrightarrow C(e_1, \mathscr{I}_C(e_1)) = 0$ and $\forall e_1 \in E_{P_1} \setminus E_L \Longrightarrow C(e_1, \mathscr{I}_C(e_1)) = C(e_1, \varepsilon) = 1$, it follows $\Sigma_{e_1 \in E_{P_1}} C(e_1, \mathscr{I}_C(e_1)) = |E_{P_1}| - |E_L|$. Similarly, since $\forall e_2 \in E_{P_2} \cap E_L \Longrightarrow C(e_2, \mathscr{I}_C(e_2)) = 0$ and $\forall e_2 \in E_{P_2} \setminus E_L \Longrightarrow C(e_2, \mathscr{I}_C(e_2)) = C(e_2, \varepsilon) = 1$, it follows $\Sigma_{e_2 \in E_{P_2}} C(e_2, \mathscr{I}_C(e_2)) = |E_{P_2}| - |E_L|$.

The above inequality becomes: $\frac{|E_{P_1}| - |E_L|}{|E_{P_1}|} - \frac{|E_{P_2}| - |E_L|}{|E_{P_2}|} = \frac{|E_L|(|E_{P_1}| - |E_{P_2}|)}{|E_{P_1}||E_{P_2}|} < 0$

This latter is always true, since $(|E_{P_2}| - |E_{P_1}|) > 0$.

– **Axiom-3.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2} = \Gamma_\Sigma$ then $MSP^k(L,P_1) > MSP^k(L,P_2)$. For any $k \in \mathbb{N}$, the relation $MSP^k(L,P_1) \geq MSP^k(L,P_2)$ holds for Axiom-2. A case $MSP^k(L,P_1) = MSP^k(L,P_2)$ occurs when $M^k_{P_1}$ over-approximates the behavior of P_1.
For any $\mathscr{B}_{P_1}$ there exists always a k^* s.t. $E_{P_1} \subset E_{P_2}$. This is always true since $\exists \tau_\Sigma \in \mathscr{B}_{P_2} \setminus \mathscr{B}_{P_1}$, s.t. for $k^* = |\tau_\Sigma| + 1 \Longrightarrow \exists (-, \tau_\Sigma) \in E_{P_2} \setminus E_{P_1}$ (see Definition 6). Consequently, for any $k \geq k^*$, having $E_{P_1} \subset E_{P_2}$, $MSP^k(L,P_1) > MSP^k(L,P_2)$ holds (see proof Axiom-2, case 2). It can be shown that $k^* \leq min(|\bar{\tau}_\Sigma|, max(|\bar{\tau}_\Sigma|, |\Sigma|))$, where $\bar{\tau}_\Sigma$ is the shortest trace of the set $\{\tau^{i \to j} \mid \tau_\Sigma = \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle \in \mathscr{B}_{P_1} \wedge t_x = t_y \ \forall x, y \in [i, j]\}$. In real cases, $k^* = 1$ for any process model[3] having at least one activity not in self-loop. Furthermore, if a similar restriction on k is applied for the Axiom-2, the MSP^k guarantees $MSP^k(L,P_1) > MSP^k(L,P_2)$ instead of $MSP^k(L,P_1) \geq MSP^k(L,P_2)$ for any P_1, P_2 and L, such that $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2}$.
– **Axiom-4.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$ then $MSP^k(L,P_1) = MSP^k(L,P_2)$. If $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$, then $E_{P_1} = E_{P_2}$ (by construction). It follows straightforward that $MSP^k(L,P_1) = MSP^k(L,P_2)$ (see proof Axiom-1).
– **Axiom-5.** Given two event logs L_1, L_2 and a process P, s.t. $\mathscr{B}_{L_1} \subseteq \mathscr{B}_{L_2} \subseteq \mathscr{B}_P$, then $MSP^k(L_2,P) \geq MSP^k(L_1,P)$. Consider the two following cases:
  1. if $\mathscr{B}_{L_1} = \mathscr{B}_{L_2}$, then $E_{L_1} = E_{L_2}$ (by construction). Likewise the proof of Axiom-4, it follows $MSP^k(L_2,P) = MSP^k(L_1,P)$.
  2. if $\mathscr{B}_{L_1} \subset \mathscr{B}_{L_2}$, then $E_{L_1} \subset E_{L_2}$ (by construction). In this case, the graph matching algorithm would find matchings for a larger number of edges between M^k_P and $M^k_{L_2}$, than between M^k_P and $M^k_{L_1}$ (this follows from $\mathscr{B}_{L_1} \subset \mathscr{B}_{L_2}$). Thus, less edges will be mapped to ε in the case of $MSP^k(L_2,P)$ increasing the value for the precision, i.e., $MSP^k(L_2,P) \geq MSP^k(L_1,P)$.

5 Automata Behavioral-based Precision (ABP)

This section introduces a novel precision measure based on a behavioral comparison of automata-based behavioral abstractions of process models and event logs. The computation of the precision between a model and a log is divided into five steps. 1) A deterministic and acyclic finite state automaton is constructed from the event log. 2) A reachability graph is derived from the process model. 3) A representative set of traces is extracted from the reachability graph. 4) For each selected trace in the model, the most

[3] Without activities with duplicate labels.

similar trace in the log is determined (optimal alignment). 5) Finally, the precision metric is computed from all the computed (optimal) trace alignments. First, we introduce the behavioral abstraction used in this section and, second, we present the machinery for the comparison and, by the same token, we define the precision measure.

5.1 Behavioral Abstraction

The behavioral abstraction used for this precision measure is finite state automata, which can used to define lossless behavioral representations for process models and event logs. Given a set of activity labels Σ, a finite state automata is a directed graph, denoted as a tuple $\mathscr{F} = (N, A, n_0, F)$, where N is a finite non-empty set of states, $A \subseteq N \times \Sigma \times N$ is a set of arcs, $n_0 \in N$ is an initial state, and $F \subseteq N$ is a set of final states. The execution of an activity $l \in \Sigma$ is represented as an arc in a FSA, denoted as a triplet (n_s, l, n_t), where n_s is the *source* state, n_t is the *target* state and l is the *label* associated to the arc. The set of incoming and outgoing arcs of a state n is defined as $\bullet n = \{(n_s, l, n_t) \in A \mid n = n_t\}$ and $n\bullet = \{(n_s, l, n_t) \in A \mid n = n_s\}$, respectively. Finally, a sequence of (contiguous) arcs in a FSA is called a *path*. This formalism is used to concretely define the behavioral representations for the event log (DAFSA) and for the process model (reachability graph), which are presented next.

An event log can be represented as a directed acyclic FSA [16] by merging common prefixes and suffixes between traces. The resulting FSA is deterministic (i.e. every outgoing arc of a state needs to have a unique label and no arc is associated with an invisible label) and acyclic (i.e. no path contained in the FSA contains a duplicate state). A deterministic and acyclic FSA is formally defined as follows:

Definition 8. *[Deterministic Acyclic Finite State Automata (DAFSA)] A Deterministic Acyclic Finite State Automata is a FSA $\mathscr{D} = (N_{\mathscr{D}}, A_{\mathscr{D}}, n_{0,\mathscr{D}}, F_{\mathscr{D}})$, that additionally is deterministic, i.e. $\nexists n \in N_{\mathscr{D}}((n, l_x, n_x), (n, l_y, n_y) \in n\bullet \mid l_x = l_y \vee l_x = \tau \vee l_y = \tau)$ and acyclic, i.e. $\nexists path \in \mathscr{D}(n_x, n_y \in path \mid n_x = n_y)$.*

Given a path from the initial state to a state $n \in N_{\mathscr{D}}$, we refer to the labels associated to the arcs in the path as the *prefix* of n, and, analogously, given a path from n to a final state, we refer to the labels associated to such path as the *suffix* of n. Of special interest is the set of suffixes of a state n that is represented by the function $\mathit{suff}(n) = \bigcup_{(n_s,l,n_t)\in n\bullet}\{l \oplus x \mid x \in \mathit{suff}(n_t)\}$, where $\oplus$ denotes the concatenation operator. If n is a final state then $\{\langle\rangle\} \in \mathit{suff}(n)$. The complexity of building the DAFSA is $O(|\Sigma| \cdot \log n)$, where L is the set of distinct event labels, and n is the number of states in the DAFSA.

The computation of the reachability graph from a process model requires the ability to compute the execution state space of the process. Thus, we use Petri nets, which has a well defined execution semantics, as the formalism to represent the process models.

The execution semantics of Petri nets can be defined in terms of markings and firings. A marking describes an execution state that can be reached as result of firing a sequence of transitions. All possible reachable markings are represented via a reachability graph [14]. Intuitively, a reachability graph is a deterministic FSM where states denote markings, and arcs denote the transitions fired to go from one marking to another. In this paper we restrict ourselves to Petri nets having a finite number of execution states.

Definition 9. *[Reachability graph] The* reachability graph *of a Petri net $\mathscr{N}$ is a deterministic finite state machine $\mathscr{R} = (M, A_{\mathscr{R}}, m_0, M_f)$, where M is the set of reachable markings, $A_{\mathscr{R}}$ is the set of arcs $A_{\mathscr{R}} = \{(m_1, \lambda(t), m_2) \in M \times \Sigma \times M \mid m_2 = m_1 - \bullet t + t\bullet\}$ and $M_f = \{m \in M \mid \nexists t \in T$, such that $\bullet t \subseteq m\}$.*

The reachability graph can be reduced to remove τ labelled transitions while preserving the original behavior of the Process model [16]. This reduction would also help coping with possible complexity arising from the large number of silent transitions. In the following, we assume that τ labelled transitions have been removed with the technique in [16], and the resulting FSA is non-deterministic and can possibly contain cycles, thus making impracticable the a comparison to the log behavior (due to the infinite amount of traces represented by the cylces).

In the light of the above, we abstract from cyclic behavior in a reachability graph by selecting a finite set of model traces. The selected traces are those which represent either acyclic behavior or cyclic behavior unfolded up to k-number of times in each of its possible traces. Roughly speaking, we first determine all possible simple paths from the initial marking to all final markings [18], then we identify a list of elementary cycles and their corresponding entry states [8], and then generate all possible combinations of cycles with up to k repetitions. This last step is done by plugging-in all possible permutations of cycles, whenever is necessary in the simple paths. The number of traces identified from the model in this way is factorial, i.e. the number of duplications for a single cycle unfolding is $(2^{c*k})!$, where c is the number of cycles in a certain entry state and k is the number of iterations. From a theoretical perspective, the accuracy of the precision metric will increase with k and it should be chosen with regards to the log, i.e. the number of iterations of each cycle + 1. To prevent a factorial explosion of the set of traces, however, a compromise between accuracy and time performance has to be made with regards to k.

5.2 Metric

The precision measure defined for the FSA is grounded on the concept of alignments, note that this notion is orthogonal to trace fitness defined in [2]. Alignments computes the distance between a pair of traces (sequences of of activity labels) via three operations: (1) synchronized move (*match*), the model trace and the event log can execute the same task/event w.r.t. label; (2) model move (*rhide*), a task in the model can occur, but the corresponding event is missing in the log; and (3) log move (*lhide*), an event observed in the log cannot occur in the model. If the cost of an alignment is minimal then such alignment is called optimal.

The idea of alignments is to synchronously traverse a trace in the log and a trace in the model, and *match* those activities that are observed in both traces. If a label of an activity is observed in one trace, but not the other, then it has to be hidden with an *rhide* operation in case it is not observed in the model trace, and with an *lhide* operation otherwise. An alignment is composed by *synchronizations*, representing the operation performed and the activity label in the log and an activity label in the model affected by such operation. In the case of *rhide* and *lhide*, which affect only one label, we use $\perp$ to denote the absence of the other. The synchronizations and alignments between

Definition 10. *[Synchronization and Alignment] Let $\Sigma_{\mathscr{D}}$ and $\Sigma_{\mathscr{R}}$ be the alphabets of activity labels of the DAFSA and the reachability graph, respectively. A synchronization β is a triplet $\beta \subseteq op \times \Sigma_{\mathscr{D}} \times \Sigma_{\mathscr{R}}$, where $op \in \{match, lhide, rhide\}$. The set of all synchronizations is denoted as S. Given a set of synchronizations S, an alignment is defined as $\varepsilon = \langle \beta_1, ..., \beta_n \rangle$ with $\beta_i \in S, 1 \leq i \leq n$. All possible alignments are denoted as $\mathscr{C}$.*

Any trace in the Model behavior can be aligned with any possible path of the DAFSA, however we are interested only in the optimal ones, i.e., those containing the minimum number of *rhides* and *lhides*, and the maximum number of matches. In order

to compute those matches, we adapted the technique presented in a previous work [16] by focussing on aligning each model trace with the DAFSA. However, in order to ensure determinism (property used later in this Section), we extended the current implementation to define a total order (lexicographic order over the activity labels and over the operations: *match* > *lhide* > *rhide*) while computing the optimal alignments. Details about the computation of the optimal alignments can be found in [16].

To formalize the precision measure, let us define two functions auxiliary functions (1) to count the number of matches in a given alignment $\#M : \mathscr{C} \to \mathbb{N}$, where $\#M(\varepsilon) = |\{(op, l_{\mathscr{D}}, l_{\mathscr{R}}) \in \varepsilon \mid op = match\}|$, and (2) to count the number of rhides in a given alignment $\#R : \mathscr{C} \to \mathbb{N}$, where $\#R(\varepsilon) = |\{(op, l_{\mathscr{D}}, l_{\mathscr{R}}) \in \varepsilon \mid op = rhide\}|$. Note that we are not interested in lhides, because those are related to operations present in the log, which does not reflect the essence of the precision measure for quantifying the behavior allowed by the model and found in the log. Using these defined notions, we can now define our new notion of precision based on one optimal alignment per trace of the model behavior $\mathscr{B}_P$ as:

$$ABP(\mathscr{B}_P, \mathscr{D}_L) = \frac{\Sigma_{c \in \mathscr{B}_M} \#M(\Psi(c))}{\Sigma_{c \in \mathscr{B}_M} \#M(\Psi(c)) + \#R(\Psi(c))} \tag{2}$$

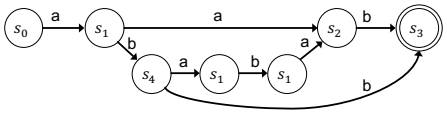
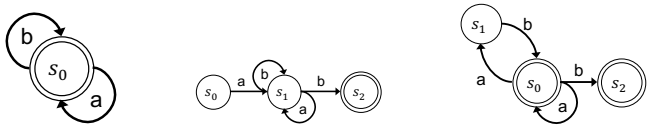


Fig. 5: Sample Log DAFSA of Table 1.



(a) Flower Proc.(0.47) (b) Process X (0.55) (c) Process Y (0.62)

Fig. 6: Examples of tauless reachability graphs and their precision for k=3.

Figure 5 shows the DAFSA created from our sample event log of Table 1. The reachability graphs of the BPMN models are then shown in Figure 6 together with their corresponding precision values for $k = 3$ cycle unfolding. We can see, that with an increasing amount of possible behavior a decreasing value for our precision metrics, i.e. c) >b) >a).

We now investigate to what extent the Automata-based precision measure fulfils the axioms of a precision metric from Section 2. For the remaining section, let L_x be a log, P_x. Furthermore, let $\mathscr{D}_x$ and $\mathscr{B}_{P_x}$ be the behavior abstractions of the log and the model respectively.

– **Axiom-1.** $ABP(\mathscr{B}_P, \mathscr{D})$ is a deterministic function. Given a log L_x and a process P_x, the construction of $\mathscr{D}_{L_x}$ and M_P^k is fully deterministic for $\mathscr{B}_P$ and $\mathscr{B}_L$. The selection of the traces

from the reachability graph is also deterministic, and by the determinism of the computation of the optimal matchings, the same alignments are always computed for the same set of traces extracted from the model and the log. Thus, the precision measure $ABP(\mathscr{B}_P, \mathscr{D})$ is always the same for the same pair of model-log.

- **Axiom-2.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2}$, then $ABP(\mathscr{B}_{P_1}, \mathscr{D}_L) \geq ABP(\mathscr{B}_{P_2}, \mathscr{D}_L)$. In this case, it would need to hold that:

$$\frac{\Sigma_{c \in \mathscr{B}_{P_1}} \#M(\Psi(c))}{\Sigma_{c \in \mathscr{B}_{P_1}} \#M(\Psi(c)) + \#R(\Psi(c))} \geq \frac{\Sigma_{c \in \mathscr{B}_{P_2}} \#M(\Psi(c))}{\Sigma_{c \in \mathscr{B}_{P_2}} \#M(\Psi(c)) + \#R(\Psi(c))} \tag{3}$$

  This equation holds if:

$$\frac{\Sigma_{c \in \mathscr{B}_{P_1}} \#M(\Psi(c))}{\Sigma_{c \in \mathscr{B}_{P_1}} \#R(\Psi(c))} \geq \frac{\Sigma_{c \in \mathscr{B}_{P_2}} \#M(\Psi(c))}{\Sigma_{c \in \mathscr{B}_{P_2}} \#R(\Psi(c))} \tag{4}$$

  The proposed precision metric $ABP(\mathscr{B}_P, \mathscr{D}_L)$ fulfils Axiom 2 only if the proportion of matches to model hides of P_1 is higher or equal to that of P_2. That is, for example, when the additional traces of $\mathscr{B}_{P_2}$, which are not in $\mathscr{B}_{P_1}$, are not in the alphabet of L. In the rest of t he cases, our precision measure cannot fulfil this axiom.
- **Axiom-3.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2} = \Gamma_\Sigma$ then $ABP(\mathscr{B}_{P_1}, \mathscr{D}_L) > ABP(\mathscr{B}_{P_2}, \mathscr{D}_L)$. If k is chosen high enough, i.e. k = the number of repetitions in the log + 1, then $\mathscr{B}_{P_2}$ will contain longer traces than any other model $\mathscr{B}_{P_1}$ as the flower model contains the highest number of cycles starting in one state. Additionally, the traces will be longer than all log traces by construction. Thus equation 4 will always hold, when comparing the flower model P_2 with any other model P_1, as with longer traces than the log, the flower model will automatically include more rhide operations than any process P_1 with shorter traces.
- **Axiom-4.** Given two processes P_1, P_2 and an event log L, s.t. $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$ then $ABP(\mathscr{B}_{P_1}, \mathscr{D}_L) = ABP(\mathscr{B}_{P_2}, \mathscr{D}_L)$. If $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$, then they contain the same set of traces to be aligned with $\mathscr{D}_L$. It follows straightforward that $ABP(\mathscr{B}_{P_1}, \mathscr{D}_L) = ABP(\mathscr{B}_{P_2}, \mathscr{D}_L)$, since the Alignments are deterministic (see proof Axiom-1).
- **Axiom-5.** Given two event logs L_1, L_2 and a process P, s.t. $\mathscr{B}_{L_1} \subseteq \mathscr{B}_{L_2} \subseteq \mathscr{B}_P$, then $ABP(\mathscr{B}_P, \mathscr{D}_{L_2}) \geq ABP(\mathscr{B}_P, \mathscr{D}_{L_1})$. Consider the two following cases:
  1. if $\mathscr{B}_{L_1} = \mathscr{B}_{L_2}$, then the A^*-search will find the same optimal alignment for each $c \in \mathscr{B}_P$, such that $ABP(\mathscr{B}_P, \mathscr{D}_{L_2}) = ABP(\mathscr{B}_P, \mathscr{D}_{L_1})$ holds (see proof Axiom 1).
  2. if $\mathscr{B}_{L_1} \subset \mathscr{B}_{L_2}$, then $\mathscr{D}_{L_2}$ contains all paths from $\mathscr{D}_{L_1}$ and will additionally contain a set of paths that is not in $\mathscr{D}_{L_1}$. The optimal alignment for each $c \in \mathscr{B}_P$ can always be chosen from the former set of paths in $\mathscr{D}_{L_1}$, such that at least $ABP(\mathscr{B}_P, \mathscr{D}_{L_2}) = ABP(\mathscr{B}_P, \mathscr{D}_{L_1})$ holds. If at least one optimal alignment is from a *path* $\notin \mathscr{D}_{L_1}$, then for this trace c either $\#M(\Psi(c))$ is higher or $\#R(\Psi(c))$ is lower than for L_1 and thus $\mathscr{D}_{L_1}$, such that at least $ABP(\mathscr{B}_P, \mathscr{D}_{L_2}) > ABP(\mathscr{B}_P, \mathscr{D}_{L_1})$ holds.

Our metric does not fulfil Axiom 2 if Equation (4) does not hold. In this case, the ratio of *match* and *rhide* operations in P_2 is higher than that of P_1. We argue that it makes sense that the precision of P_2 is higher in this case, as P_2 contains overall more traces with a higher concordance with the log traces, than P_1 does. Indeed, Axiom 2 was formulated to test strict inclusion of a trace in the language of the log, i.e. if a trace of a process model is not also in the log, it should decrease the value of precision. In contrast, our precision measure does not necessarily decrease with the increase of the number of traces that are not in the log, as it is based on partial language containment.

6 Evaluation

We implemented the two precision measures presented in Sections 4 and 5 as open-source standalone tools[4] and used these tools to carry out a two-pronged comparative evaluation with state-of-the-art precision measures.

[4] Available from `http://apromore.org/platform/tools`

6.1 Qualitative evaluation

First, we used synthetic data to assess the quality of the results produced by our measures. To this purpose, we repeated the experiment carried out by van Dongen et al. [22], which relies on an artificial event log (see Table 2) and a collection of eight variants of a given process model (shown in Fig. 7). These include a *single trace* model capturing the most frequent trace, a model incorporating all *separate traces*, the *flower model* of all activities in the log, a model with activities G and H in parallel (*Opt. G || Opt. H*), one with G and H in self-loop ($\circlearrowleft G$, $\circlearrowleft H$), a model with D in self-loop ($\circlearrowleft D$), a model with all activities in parallel (*All parallel*), and a final model where all activities are in round robin (*Round robin*).

Traces	#
$\langle A,B,D,E,I\rangle$	1207
$\langle A,C,D,G,H,F,I\rangle$	145
$\langle A,C,G,D,H,F,I\rangle$	56
$\langle A,C,H,D,F,I\rangle$	23
$\langle A,C,D,H,F,I\rangle$	28

Table 2: Test log [22].

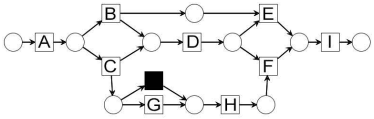


Fig. 7: Original model [22].

Using each model-log pair, we compared our precision measures (MSP^k with k up to 7, and *ABP*) with those reported in [22], namely: alignment-based ETC precision (ETC_a), negative events precision (NE) and anti-alignment precision (AA).[5] To these, we added the traces set difference precision (SD) and the projected conformance checking (PCC). We left out the advanced behavioral appropriateness (ABA) as it is not defined for some of the models in this dataset.

Table 3 reports the results of this first experiment. Notwithstanding the inability of the existing measures to satisfy the axioms in [20], in commenting these results, we take as a reference the values of *ABP*. This is the only measure based on an exact representation of the behavior of log and model, and a full characterization of the additional model behavior not contained in the log, based on behavioral comparison (as opposed to structural comparison). As a result, *ABP* has a higher sensitivity (i.e. ability to discriminate between behaviorally similar models) than the other measures. For example, we can observe that *ABP* discriminates well the behavior of the all-parallel model from that of the flower model and of the round robin model. These three models yield a very large number of traces due to the combinatorial explosion of the states produced by the unfolding of cycles and by the interleavings of parallel activities. As such, other measures, such as NE, PCC and AA, produce results very close to each other (e.g. AA has values ranging from 0.000 to 0.033). This means that a minimum behavioral variation in these models may not be picked up by these measures.

On the one hand, the Markovian abstraction-based measure becomes more sensitive at picking up differences between acyclic models as we increase k. For example, MSP^7 can distinguish well the original model from the single trace and the separate traces models, which is not possible with $k = 1$ (all values equal to 1.000). On the other hand, this measure loses sensitivity with models yielding large state spaces as we increase of k. This is the case of the flower model, which for $k = 7$ is no longer distinguishable

[5] Some values differ from those in [22] as we used each measure's latest implementation.

Process Variant	SD	ETC_a	NE	PCC	AA	MSP^1	MSP^3	MSP^7	*ABP*
Original model	0.833	0.900	0.995	1.000	0.871	1.000	0.880	0.852	0.974
Single trace	1.000	1.000	0.893	1.000	1.000	1.000	1.000	1.000	1.000
Separate traces	1.000	1.000	0.985	0.978	1.000	1.000	1.000	1.000	1.000
Flower model	0.000	0.153	0.117	0.509	0.000	0.189	0.003	0.000	0.000
Opt. G || Opt. H	0.417	0.682	0.950	0.974	0.800	0.895	0.564	0.535	0.948
↻G, ↻H	0.000	0.719	0.874	0.896	0.588	0.810	0.185	0.006	0.394
↻D	0.000	0.738	0.720	0.915	0.523	0.895	0.349	0.069	0.426
All parallel	0.000	0.289	0.158	0.591	0.033	0.210	0.006	0.000	0.406
Round robin	0.000	0.579	0.194	0.594	0.000	0.814	0.496	0.274	0.333

Table 3: Comparison of the precision metrics on artificial data.

from the all-parallel model and is hardly distinguishable from the model with G and H in self-loop. This is a direct result of satisfying Axioms 2 and 3 for this measure, which postulate a decreasing precision as the state space of the model increases, unless we allow partial language matching as *ABP* does. For this reason, *ABP* obtains the highest precision value for the all-parallel model across all measures, since on average it matches the highest number of labels for each trace, due to the various possible interleavings.

Finally, in terms of models ranking (i.e. the order in which the models are ranked based on their precision), the Markovian abstraction measures concord the most with the ranking made by *ABP* (6 models out of 9), closely followed by SD and AA (5 models), while all other measures only concord in ranking for less than half of the models (ETC_a and PCC 4, NE only 1).

6.2 Quantitative evaluation

In the second evaluation, we used twelve publicly-available real-life logs to measure the time performance of our Markovian measures (up to $k = 11$) compared to that of ETC_a. We chose ETC_a as a representative of the state of the art because this measure is widely accepted (and most-frequently used) by the community. We excluded our *ABP* measure since, while theoretically appealing, it does not scale to real-life datasets. For scalability reasons, we also excluded AA, despite being more recent than ETC_a.

The logs, extracted from the 4TU Data Center,[6] are the same as those used in a recent benchmark of automated discovery methods [3], and cover business processes in different domains, ranging from finance through to healthcare and government. For each log we measured the precision on the process models discovered using three state-of-the-art automated discovery methods: Inductive Miner Infrequent [11], Split Miner [4] and Structured Heuristics Miner [5].

For each model, we started with $k = 1$ and increased it to 11 unless a 30 minutes timeout was reached, or a stack overflow exception was raised by the tool. For half of the process models discovered by Inductive Miner, it was not possible to evaluate the Markovian-based precision for $k > 5$, due to the huge amount of states (over a million) typical of flower-like models as those discovered by Inductive Miner. Nevertheless, we noticed that for every log our measure could quickly identify a precision ranking of the process models discovered by the three discovery methods. Across all logs, if one

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

of the three process models scored a higher precision than the other two, increasing k, the ranking would not change. Specifically, we observed a perfect rank correlation between second-level and higher-level Markovian abstractions. This can be explained by the fact that the automated process discovery methods employed in this experiment use the Directly-Follows Graph of the log as an intermediate representation, which is a first-order Markovian abstraction. The reason why we need to go to the second-order Markovian abstraction to see a perfect rank correlation with higher-order abstractions is due to the fact that the first-order Markovian abstraction confuses self-loops and short-loops with parallelism.

For the majority of the logs (9 out of 12) it was possible to determine the most precise process model already with $k = 1$, whilst in the worst case (for the simplest model) a ranking was determined at $k = 7$. Altogether, these results show that our Markovian-based precision is a tunable measure that becomes more sensitive for acyclic models, yet less discriminative for cyclic models, by increasing the value of k.

Fianlly, Table 4 reports statistics on the time performance of MSP^k ($k \in [1,3]$) against ETC_a, for each discovery method, across all twelve logs. Our measure scales well to real-life datasets, being quite fast for models with small state spaces like those produced by Split Miner and Structured Heuristics Miner, while ETC_a remains slow. As we increase k, the performance of MSP^k reduces sharply for flower-like models as those produced by Inductive Miner. Yet, it scores similar values as ETC_a.

	Split Miner				Inductive Miner				Struct. Heuristics Miner			
Precision	avg	max	min	total	avg	max	min	total	avg	max	min	total
ETC_a	60.0	351.9	0.3	720.3	84.2	642.7	0.1	1009.8	34.0	101.4	0.2	305.9
MSP^1	2.0	8.9	0.1	24.3	1.9	7.5	0.1	22.4	5.9	18.5	0.9	70.5
MSP^2	1.9	7.3	0.1	23.2	5.4	15.2	0.1	65.3	6.2	24.4	0.4	74.3
MSP^3	2.0	7.7	0.1	22.5	109.6	426.7	0.1	1205.7	18.5	59.9	0.2	203.7

Table 4: Time performance statistics (in seconds) using the twelve real-life logs.

All experiments were performed on an HP Elitebook with an Intel Core i5-6200U CPU @ 2.30 GHz and 16GB RAM running Windows 10 Pro (64-bit) and JVM 8 with 12GB RAM (8GB Stack and 4GB Heap).

7 Conclusion

The contributions of this paper are: (i) a framework for defining precision measures based on behavior abstraction and comparison; (ii) a family of instances of this framework based on k^{th}-order Markovian abstractions that fulfil four of five axioms of precision measures and all five for a sufficiently large k; and (iii) a measure based on simulation of lossless automata abstractions fulfiling all axioms. The evaluation shows that the Markovian abstractions offer a suitable tradeoff between scalability and sensitivity. The execution times of the 3^{rd}-order Markovian abstraction are comparable or lower than ETC_a precision, while the resulting measurements have a perfect rank correlation with those provided by higher-order abstractions. The automata-based measure, while theoretically appealing, does not scale up in its current form to real-life logs.

A possible avenue for future work is to develop scalable variants of the automata-based abstraction that still fulfil all axioms. Another direction is to extend the proposed

abstract-and-compare approach to define measures of fitness and generalization with provable formal properties. Yet another direction is to investigate other abstractions that can explicitly handle concurrency, such as abstractions based on event structures.

Acknowledgements. This research is partly funded by the Australian Research Council (DP180102839) and the Estonian Research Council.

References

1. A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst. Measuring precision of modeled behavior. *ISeB*, 13(1), 2015.
2. A. Adriansyah, B. van Dongen, and W. van der Aalst. Conformance checking using cost-based fitness analysis. In *Proc. of EDOC*. IEEE, 2011.
3. A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F.M. Maggi, A. Marrella, M. Mecella, and A. Soo. Automated discovery of process models from event logs: Review and benchmark. Technical report, arXiv:1705.02288, 2017.
4. A. Augusto, R. Conforti, M. Dumas, and M. La Rosa. Split miner: Discovering accurate and simple business process models from event logs. In *Proc. of IEEE ICDM*. IEEE, 2017.
5. A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and G. Bruno. Automated discovery of structured process models: Discover structured vs. discover and structure. In *Proc. of ER*, LNCS 9974. Springer, 2016.
6. J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A robust f-measure for evaluating discovered process models. In *Proc. of IEEE Symposium on CIDM*. IEEE, 2011.
7. G. Greco, A. Guzzo, L. Pontieri, and D. Sacca. Discovering expressive process models by clustering log traces. *IEEE TKDE*, 18(8), 2006.
8. D. Johnson. Finding all the elementary circuits of a directed graph. *SICOMP*, 4(1), 1975.
9. H.W. Kuhn. The hungarian method for the assignment problem. *NRL*, 2(1-2), 1955.
10. S. Leemans, D. Fahland, and W. van der Aalst. Discovering block-structured process models from event logs - a constructive approach. In *Proc. of Petri Nets*. Springer, 2013.
11. S. Leemans, D. Fahland, and W. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Proc. of BPM Workshops*. Springer, 2014.
12. S. Leemans, D. Fahland, and W. van der Aalst. Scalable process discovery and conformance checking. *Software & Systems Modeling*, 2016.
13. N. Lohmann, E. Verbeek, and R. Dijkman. Petri net transformations for business processes–a survey. In *Transactions on Petri Nets and other models of concurrency II*. Springer, 2009.
14. E. Mayr. An algorithm for the general petri net reachability problem. *SICOMP*, 13(3), 1984.
15. J. Munoz-Gama and J. Carmona. A fresh look at precision in process conformance. In *Proc. of BPM*. Springer, 2010.
16. D. Reißner, R. Conforti, M. Dumas, M. La Rosa, and A. Armas-Cervantes. Scalable conformance checking of business processes. In *Proc. of CoopIS*. Springer, 2017.
17. A. Rozinat and W. van der Aalst. Conformance checking of processes based on monitoring real behavior. *ISJ*, 33(1), 2008.
18. F. Rubin. Enumerating all simple paths in a graph. *IEEE TCS*, 25(8), 1978.
19. O. Sokolsky, S. Kannan, and I. Lee. Simulation-based graph similarity. In *Proc. of TACAS*. Springer, 2006.
20. N. Tax, X. Lu, N. Sidorova, D. Fahland, and W. van der Aalst. The imprecisions of precision measures in process mining. *Information Processing Letters*, 135, 2018.
21. W. van der Aalst, M. Schonenberg, and M. Song. Time prediction based on process mining. *ISJ*, 36(2), 2011.
22. B. van Dongen, J. Carmona, and T. Chatain. A unified approach for measuring precision and generalization based on anti-alignments. In *Proc. of BPM*. Springer, 2016.
23. S. vanden Broucke and J. De Weerdt. Fodina: a robust and flexible heuristic process discovery technique. *DSS*, 2017.
24. A. Weijters and J. Ribeiro. Flexible heuristics miner (FHM). In *Proc. of CIDM*. IEEE, 2011.