Information Systems and e-Business Management manuscript No.
(will be inserted by the editor)

Measuring Precision of Modeled Behavior

A. Adriansyah · J. Munoz-Gama · J. Carmona · B.F. van Dongen · W.M.P. van der Aalst

Received: date / Accepted: date

Abstract Conformance checking techniques compare observed behavior (i.e., event logs) with modeled behavior for a variety of reasons. For example, discrepancies between a normative process model and recorded behavior may point to fraud or inefficiencies. The resulting diagnostics can be used for auditing and compliance management. Conformance checking can also be used to judge a process model automatically discovered from an event log. Models discovered using different process discovery techniques need to be compared objectively. These examples illustrate just a few of the many use cases for aligning observed and modeled behavior. Thus far, most conformance checking techniques focused on replay fitness, i.e., the ability to reproduce the event log. However, it is easy to construct models that allow for lots of behavior (including the observed behavior) without being *precise*.

In this paper, we propose a method to measure *precision* of process models, given their event logs by first aligning the logs to the models. This way, the measurement is not sensitive to non-fitting executions and more accurate values can be obtained for non-fitting logs. Furthermore, we introduce several variants of the technique to deal better with incomplete logs and reduce possible bias due to behavioral property of process models. The approach has been implemented in the ProM 6 framework and tested against both artificial and real-life cases. Experiments show that the approach is robust to noise and applicable to handle logs and models of real-life complexity.

A. Adriansyah · B.F. van Dongen · W.M.P. van der Aalst
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: {a.adriansyah,b.f.v.dongen,w.m.p.v.d.aalst}@tue.nl

J. Munoz-Gama · J. Carmona
Universitat Politecnica de Catalunya
Barcelona, Spain
E-mail: {jmunoz,jcarmona}@lsi.upc.edu

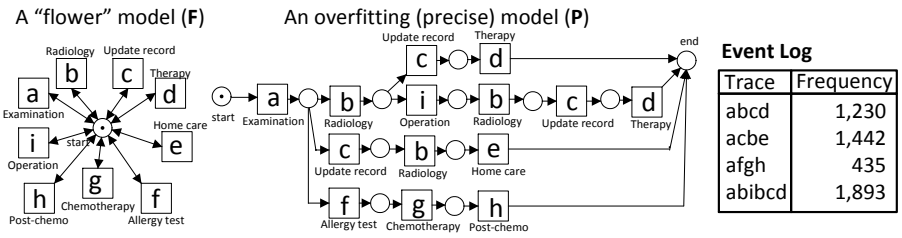


Fig. 1: Example of an extremely imprecise (underfitting) and precise model (overfitting) for a given log.

Keywords Precision measurement · Log-model alignment · Conformance checking · Process mining

1 Introduction

The starting point for most Business Process Management (BPM) activities are process models, as they provide insights into possible scenarios [21]. Process models are used for analysis (e.g., simulation [5]), enactment [21], redesign [18], and process improvement [28,29]. Therefore, they should reflect the dominant behavior accurately. The increasing availability of event data enables the application of *conformance checking* [2,4,30]. Conformance checking techniques compare recorded process executions in the form of *event logs* with process models to quantify how "good" are the models with respect to their executions.

Conformance can be viewed along multiple orthogonal dimensions: (1) fitness, (2) precision, (3) generalization, and (4) simplicity [2,14]. In this paper, we focus on the *precision* dimension. Given an event log and a process model, precision penalizes the model for allowing behavior that is unlikely given the observed behavior in the log. Take for example the two models and the event log in Figure 1. Both models show a cancer patient handling process in a hospital and are shown using Petri net formalism [27].[1] All traces in the log can be reproduced by both models, i.e., the traces perfectly *fit* the models. However, notice that the "flower" model (**F**) may provide misleading insights, as it also allows for much more behavior not appearing in the log. In contrast, the other model (**P**) only allows traces that occur in the log. Hence, the *precision* of model **P** is better than model **F** with respect to the log.

Many existing precision metrics (e.g., [25, 30, 34]) do not explicitly take into account possible deviations between the behavior observed in the event log with the behavior modeled in the models, while many case studies show

[1] For the reader not familiar with Petri nets, a Petri net is a bipartite graph that contains two types of nodes: places (circles) and transitions (boxes). A place may contain tokens (black dots), and a transition can fire if its predecessor places contain a token. When fired, a transition removes a token from each input place and adds a token to each successor place.

that such deviations often occur in practice (e.g., [11, 16, 19, 20, 23, 31, 35, 36]). Thus, these metrics might be biased due to unfitting logs and models. In this paper, *we explicitly take deviations between the observed behavior in event logs and the modeled behavior in process models into account* and propose a robust approach to measure the *precision* between a (possibly non-fitting) event log and a model. First, we *align* the log and the model to find, for each trace, those complete activity sequences in the model that are most similar to the trace. Then we use these alignments to measure precision between the original log and the model. In this paper, we generalize the approach presented in [9] by introducing various possible ways of computing precisions based on alignments, their log completeness requirements, and their issues in order to obtain accurate precision values.

The remainder of this paper is organized as follows: Section 2 shows the notations and preliminary concepts that are used throughout this paper. Alignments between event logs and models are explained in Section 3. The alignment-based precision approach is presented in Section 4. In Section 5 we propose a series of extensions for the basic precision approach. Experimental results are given in Section 6. Section 7 concludes the paper.

2 Preliminaries

Conformance checking requires as input both a process model and an event log. Therefore, we first formalize process models and logs after introducing a set of notations that is used in the remainder of this paper.

2.1 Sequence and Multiset

Let W be a set. For (finite) *sequences* of elements over a set W, we use ϵ to denote an empty sequence. A concatenation of sequences σ_1 and σ_2 is denoted with $\sigma_1 \cdot \sigma_2$. W^* denotes the set of all finite sequences over W. We refer to the i-th element of a sequence σ as $\sigma[i]$ and we use $|\sigma|$ to represent the length of sequence σ. We say that any $x \in (W \times W)$ is a *pair*. We use $\pi_1(x)$ and $\pi_2(x)$ to refer to the first and the second element of pair x respectively. We generalize this notation to sequences: $\pi_i(\sigma) = \langle \pi_i(\sigma[1]), \ldots, \pi_i(\sigma[|\sigma|]) \rangle$. For example, $\pi_1(\langle (a,b),(b,c),(b,d) \rangle) = \langle \pi_1((a,b)), \pi_1((b,c)), \pi_1((b,d)) \rangle = \langle a, b, b \rangle$. For all $Q \subseteq W$, $\sigma_{\downarrow Q}$ denotes the projection of $\sigma \in W^*$ on Q, e.g., $\langle a, a, b, c \rangle_{\downarrow \{a,c\}} = \langle a, a, c \rangle$. $\mathcal{P}(Q)$ denotes the powerset of Q, e.g., $\mathcal{P}(\{a, b\}) = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$.

A *multiset* m over W is a mapping $m : W \to \mathbb{N}$. We overload the set notation, using $\emptyset$ for the empty multiset and $\in$ for the element inclusion. We write e.g., $m = [p^2, q]$ or $m = [p, p, q]$ for a multiset m with $m(p) = 2$, $m(q) = 1$, and $m(x) = 0$ for all $x \notin \{p, q\}$. We use $|m|$ to indicate the total number of elements in multiset m (e.g., $|[p^2, q]| = 3$).

2.2 Event Log and Process Model

The starting point for conformance checking is an *event log*. An event log records the execution of all cases (i.e., process instances). Each case is described by a *trace*, i.e., an activity sequence. Different cases may have exactly the same trace. In reality, not all activities performed in a process are logged. We define the set of all logged activities from the universe of activities A as $A_L \subseteq A$. An event log over A_L is a multiset $L : {A_L}^* \to I\!N$. For example, the log in Figure 1 is formalized as $[\langle a,b,c,d\rangle^{1230}, \langle a,c,b,e\rangle^{1442}, \langle a,f,g,h\rangle^{435}, \langle a,b,i,b,c,d\rangle^{1893}]$. Note that for simplicity, we omit brackets for sequences of activities in Figure 1.

Similarly, a *process model* defines a set of sequences of activities that leads to proper termination of the process. Furthermore, some activities in a process may not appear in its model. Thus, we define a set of modeled activities over the set of all activities A as $A_M \subseteq A$. A process model is a (possibly infinite) set of complete activity sequences $M \subseteq {A_M}^*$, i.e., executions from the initial state to some final state. Consider for example the precise model (**P**) in Figure 1. Assuming that the end state is reached when the "end" place contains exactly one token, the model is formalized by the finite set $\{\langle a,b,c,d\rangle, \langle a,c,b,e\rangle, \langle a,f,g,h\rangle, \langle a,b,i,b,c,d\rangle\}$. Note that the set of modeled activities and the set of logged activities may be disjoint, i.e., $A_M \cap A_L$ can be the empty set. We consider activities that appear in event logs but not modeled in process models as activities that are allowed to occur anytime. Furthermore, modeled activities in process models that never occur in event logs are considered as unlogged activities. Thus, their absence in the logs is not counted as violations to the models.

3 Cost-Optimal Alignment

An alignment between an event log and a process model relates the occurrences of activities in the log to the execution steps of the model. As the execution of a case is often performed independently of the execution of another case, aligning is performed on the basis of traces.

For each trace in an event log that fits a process model, each "move" in the trace (i.e., an event observed in the log) can be mimicked by a "move" in the model (i.e., an action executed in the model). However, this is not the case if the trace does not fit the model perfectly. We use the symbol $\gg$ to denote "no move" in either the log or the model. Hence, we introduce the set $A_L^{\gg} = A_L \cup \{\gg\}$ where any $x \in A_L^{\gg}$ refers to a "move in log" and the set $A_M^{\gg} = A_M \cup \{\gg\}$ where any $y \in A_M^{\gg}$ refers to a "move in model". Formally, a *move* is represented by a pair $(x,y) \in A_L^{\gg} \times A_M^{\gg}$ such that:

- (x,y) is a *move in log* if $x \in A_L$ and $y = \gg$,
- (x,y) is a *move in model* if $x = \gg$ and $y \in A_M$,
- (x,y) is a *synchronous move/move in both* if $x \in A_L$, $y \in A_M$, and $x = y$,
- (x,y) is a *illegal move* in all other cases.

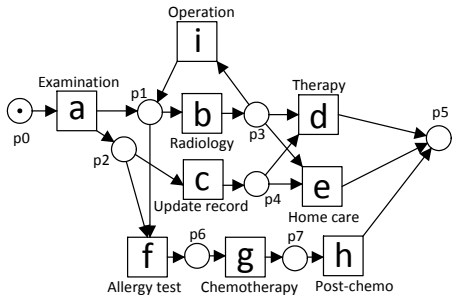


Fig. 2: Process model that is neither overfitting nor imprecise for the log in Figure 1.

$$\gamma_1 = \begin{vmatrix} a & \gg & b & d & e \\ a & c & b & \gg & e \end{vmatrix} \quad \gamma_2 = \begin{vmatrix} a & b & \gg & d & e \\ a & b & c & \gg & e \end{vmatrix} \quad \gamma_3 = \begin{vmatrix} a & \gg & b & d & e \\ a & c & b & d & \gg \end{vmatrix}$$

$$\gamma_4 = \begin{vmatrix} a & b & \gg & d & e \\ a & b & c & d & \gg \end{vmatrix} \quad \gamma_5 = \begin{vmatrix} a & b & d & \gg & e \\ a & b & \gg & c & e \end{vmatrix} \quad \gamma_6 = \begin{vmatrix} a & \gg & \gg & \gg & b & d & e \\ a & f & g & h & \gg & \gg & \gg \end{vmatrix}$$

Fig. 3: Some alignments between trace $\sigma_L = \langle a, b, d, e \rangle$ and the model in Figure 2.

We use A_{LM} to denote the set of all pairs of *legal moves*, i.e., all possible pairs of move in log, move in model, and move in both.

Along this section, let L be a log over A_L, let $\sigma_L \in L$ be a trace, and let M be a model. An *alignment* between σ_L and M is a sequence $\gamma \in {A_{LM}}^*$ where the projection of the first element (ignoring $\gg$) yields σ_L (i.e., $\pi_1(\gamma)_{\downarrow A_L} = \sigma_L$) and projection of the second element (ignoring $\gg$) yields a complete sequence of M (i.e., $\pi_2(\gamma)_{\downarrow A_M} \in M$).

Take for example an unfitting trace $\sigma_L = \langle a, b, d, e \rangle$ and the model in Figure 2. Assuming that the end state of the model is reached when place p_5 in the model contains exactly one token, the model has an infinite set of complete activity sequences (i.e., $\{\langle a, b, c, d \rangle, \langle a, c, b, d \rangle, \langle a, b, c, e \rangle, \langle a, c, b, e \rangle, \langle a, f, g, h \rangle, \langle a, b, i, c, b, e \rangle, \ldots\}$). Some possible alignments between σ_L and the model are shown in Figure 3.

The moves are represented vertically in Figure 3, e.g., the second move of γ_1 is $(\gg, c)$, indicating that the model moves c while the log does not make any move. Note that the projection of an alignment between a trace and a model to all of its movements on model yields a complete activity sequence allowed by the model. This property is not always ensured by other conformance checking approaches. For example, given a trace and a process model, when using the approach in [30], the so-called "missing tokens" are added to allow the activities that occur in the trace but not supposed to occur according to the model. The addition of such missing tokens introduces extra behavior that is not allowed in the original process model.

To measure the cost of an alignment, we define a *distance function* $\delta : A_{LM} \rightarrow I\!N$ where for all $(a_L, a_M) \in A_{LM}, \delta((a_L, a_M)) = 0$ if $a_L = a_M$

and $\delta(a_L, a_M) = 1$ otherwise.[2] The distance function can be generalized to alignments $\gamma \in {A_{LM}}^*$ by taking the sum of the costs of all individual moves: $\delta(\gamma) = \sum_{(a_L,a_M)\in\gamma} \delta((a_L, a_M))$. Using this function, the cost of alignment γ_1 is $\delta(\gamma_1) = \delta((a, a))+\delta((\gg, c))+\delta((b, b))+\delta((d, \gg))+\delta((e, e)) = 0+1+0+1+0 = 2$. Note that the function returns the number of mismatches in the alignment.

Given a trace from an event log and a process model, we are interested in an activity sequence from the model that is similar to the trace. Therefore, we define the set of *alignments* $\Gamma_{\sigma_L,M} = \{\gamma \in {A_{LM}}^* \mid \gamma \text{ is an alignment between } \sigma_L \text{ and } M\}$ to be all possible alignments between σ_L and M. Accordingly, we define the set of *optimal alignments* as the set of all alignments with minimum cost, i.e., $\Gamma^o_{\sigma_L,M} = \{\gamma \in \Gamma_{\sigma_L,M} \mid \forall_{\gamma'\in\Gamma_{\sigma_L,M}} \delta(\gamma) \leq \delta(\gamma')\}$. It is easy to see that there can be more than one optimal alignment between a trace and a model. For example, $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\}$ is the set of optimal alignments between the trace $\sigma_L = \langle a, b, d, e\rangle$ and the model in Figure 2.

For all alignments $\gamma \in {A_{LM}}^*, \bar{\lambda}_M(\gamma) = \pi_2(\gamma)_{\downarrow A_M}$ denotes the projection of γ to modeled activities. By definition, the bottom part of all alignments yields a complete activity sequence of the model. Thus, given an optimal alignment γ between σ_L and M, the projection $\bar{\lambda}_M(\gamma)$ provides an activity sequence that both perfectly fits M and closest to σ_L. In the example shown in Figure 2, $\bar{\lambda}_M(\gamma_1) = \langle a, c, b, e\rangle$ is one of the complete activity sequences of M that is most similar to trace $\langle a, b, d, e\rangle$.

Given a log and a model, constructing all optimal alignments between all traces in the log and the model is computationally expensive [7,8]. Thus, computing all optimal alignments between traces and process models with real-life complexity may not always be feasible in practice. Thus, instead of computing all optimal alignments between traces in the log and the model to obtain insights into deviations, one may also compute just some representative optimal alignments for each trace. In this paper, we investigate both approaches. We define three functions that provide optimal alignments between traces in the log and the model:

- $\Lambda^*_M : A^*_L \to \mathcal{P}({A_{LM}}^*)$ returns *all optimal alignments* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^*_M(\sigma_L) = \Gamma^o_{\sigma_L,M}$,
- $\Lambda^1_M : A^*_L \to {A_{LM}}^*$ returns *one optimal alignment* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^1_M(\sigma_L) \in \Gamma^o_{\sigma_L,M}$, and
- $\Lambda^R_M : A^*_L \to \mathcal{P}({A_{LM}}^*)$ returns *representatives of optimal alignments* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^R_M(\sigma_L) \subseteq \Gamma^o_{\sigma_L,M}$.

In [7, 8, 10] various approaches to obtain an optimal alignment between a trace and a model with respect to different cost functions are investigated. Given a trace σ_L of L and a model M, if there are multiple optimal alignments, Λ^1_M chooses one of them according to other external criteria. With our previous example, suppose that Λ^1_M selects an alignment that has the longest consecutive occurrence of synchronous moves in the beginning, $\Lambda^1_M(\sigma_L) = \gamma_4$.

[2] The distance function can be user-defined, but for simplicity we use a default distance function that assigns unit costs to moves in log/model only.

In [7,8], an $A^\star$-based algorithm is proposed to compute one optimal alignment between a trace and a model. The same algorithm can be extended to provide more than one optimal alignment between them. Given a trace σ_L of L and a model M, the algorithm constructs one optimal alignment by computing a shortest path from the initial to the final state of the state space of the synchronous product between σ_L and M. It is shown in [8] that all shortest paths from the initial to the final state of the state space yields an optimal alignment. For each state in the state space, the algorithm records a shortest path from the initial state to reach this state and thus, becomes the *representative* of all other shortest paths from the initial state to the state. An optimal alignment is constructed from a shortest path from the initial state to the final state that is also representing all other shortest paths that connect the same pair of states. By recording all represented shortest paths during state space exploration for each state, we can obtain all shortest paths from the initial to the final state of the state space (i.e., obtain *all optimal alignments*).

Furthermore, we can form groups of all shortest paths from the initial to the final state according to some criteria and take one representative path for each group. This way, we can get a number of *representatives* of all shortest paths between one up to the total number of all shortest paths from the initial to the final state. There are many possible ways of grouping shortest paths (i.e., grouping optimal alignments). One possibility is to group them based on their sub-path similarity (i.e., the followed sub-path in the state space). For example, one may group them based on the last step taken in the paths before they reach the final state. Such a grouping can be easily performed without much extra computation using the constructed state space. Moreover, this way of grouping allows computation of the exact number of represented optimal alignments for each representative by iterating through the state space. The interested reader is referred to [7,8] for details on the constructed state space with the $A^\star$-based algorithm approach. Note that to minimize the number of states that need to be explored, some optimizations can be performed to avoid visiting "similar" states more than once (e.g., pruning, prioritization of states [22,32]). In such cases, the constructed state space may be pruned. Thus, the number of represented shortest paths computed using the approach proposed before may only provide a lower bound to the actual number of represented shortest paths.

Given a set of representatives of all optimal alignments, each representative may represent a different number of optimal alignments. For all representatives $\gamma \in \Lambda^R_M(\sigma_L), rep_M(\gamma)$ denotes the number of optimal alignments represented by γ. Furthermore, due to possible pruning of state space, for all $\gamma_1, \gamma_2 \in \Lambda^R_M(\sigma_L) : \sum_{\gamma' \in \Lambda^R_M(\sigma_L)} rep_M(\gamma') \leq |\Gamma^o_{\sigma_L,M}|$, i.e., the total number of represented optimal alignments by the representatives is a lower bound of the total number of all optimal alignments.

Take for example a trace $\sigma_L = \langle a \rangle$. All optimal alignments between the trace and the model in Figure 2 are shown in Figure 4. Suppose that we define function Λ^R according to the extension to the $A^\star$ algorithm we described before, $\Lambda^R(\sigma_L) = \{\gamma_7, \gamma_9, \gamma_{10}\}$ where $rep_M(\gamma_7) = 1$ (γ_7 represents

$$\gamma_7 = \begin{array}{|c|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & f & g & h \\ \hline \end{array} \quad \gamma_8 = \begin{array}{|c|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & b & c & d \\ \hline \end{array} \quad \gamma_9 = \begin{array}{|c|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & c & b & d \\ \hline \end{array}$$

$$\gamma_{10} = \begin{array}{|c|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & c & b & e \\ \hline \end{array} \quad \gamma_{11} = \begin{array}{|c|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & b & c & e \\ \hline \end{array}$$

Fig. 4: All optimal alignments between trace $\sigma_L = \langle a \rangle$ and the model in Figure 2.

$\{\gamma_7\}$), $rep(\gamma_9) = 2$ (γ_9 represents $\{\gamma_8, \gamma_9\}$), and $rep(\gamma_{10}) = 2$ (γ_{10} represents $\{\gamma_{10}, \gamma_{11}\}$). In this example, $\bar{\lambda}(\gamma_7), \bar{\lambda}(\gamma_9), \bar{\lambda}(\gamma_{10})$ are $\langle a, f, g, h \rangle$, $\langle a, c, b, d \rangle$, and $\langle a, c, b, e \rangle$ respectively.

For simplicity, in the remainder we omit the model notation M in functions $\Lambda^*_M, \Lambda^1_M, \Lambda^R_M, \bar{\lambda}_M$, and rep_M if the context is clear. Note that in cases where a process model has duplicate tasks (more than one task to represent an activity) or invisible tasks (tasks whose execution are not logged), approaches to construct alignments (e.g., [7, 10]) keep the mapping from all model moves to the tasks they correspond to. Hence, given an alignment of a trace and such models, we know exactly which task is executed for each model move. We refer to [7, 10] for further details on how such mapping is constructed.

4 Computing Precision

Given an event log and a model, the technique described in the previous section provides a set of optimal alignments for each trace in the log. This section presents a technique to compute *precision* based on the use of these optimal alignments per trace. The technique considers 'one' or 'all' optimal alignments, and is based on the methods described in [24–26]. However, there is a fundamental difference: whereas in [24–26] precision is measured based on log-based model replay, the approach in this section is based on alignments [9]. The advantages are manifold. First of all, traces in the log do not need to be completely fitting. In [24–26] the non-fitting parts are simply ignored. For most real-life situations, this implies that only a fraction of the event log can be used for computing precision. Second, the existence of indeterminism in the model poses no problems when using the alignments. In [24–26], ad-hoc heuristics were used to deal with indeterminism. Finally, the use of alignments instead of log-based model replay improves the robustness of conformance checking. The remainder of this section is devoted to explain how precision can be calculated from the alignments.

Precision is estimated by confronting model and log behavior: *imprecisions* between the model and the log (i.e., situations where the model allows more behavior than the one reflected in the log) are detected by juxtaposing behavior allowed by the log and the one allowed by the model. This juxtaposition is done in terms of an automaton: first, an automaton is built from the alignments. Then, the automaton is enhanced with behavioral information of the model. Finally, the enhanced automaton is used to compute the precision. In the

L		$\bar{\lambda}(\Lambda^1)_L$	$\bar{\lambda}(\Lambda^*)_L$
trace	*freq*		
$\sigma_1 = \langle a \rangle$	1	$\langle a, f, g, h \rangle$	$\langle a, f, g, h \rangle$ $\langle a, c, b, e \rangle$ $\langle a, c, b, d \rangle$ $\langle a, b, c, d \rangle$ $\langle a, b, c, e \rangle$
$\sigma_2 = \langle a, b, c, d \rangle$	1	$\langle a, b, c, d \rangle$	$\langle a, b, c, d \rangle$
$\sigma_3 = \langle a, c, b, e \rangle$	1	$\langle a, c, b, e \rangle$	$\langle a, c, b, e \rangle$
$\sigma_4 = \langle a, f, g, h \rangle$	1	$\langle a, f, g, h \rangle$	$\langle a, f, g, h \rangle$
$\sigma_5 = \langle a, b, i, b, c, d \rangle$	1	$\langle a, b, i, b, c, d \rangle$	$\langle a, b, i, b, c, d \rangle$

Table 1: Model perspective of the alignments ('one' and 'all') for the model in Figure 2 and the log $[\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$.

remainder of the section we will use the following running example: the model shown in Figure 2 and the log $L = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$, containing the 5 traces that appear in in Table 1.

In order to build the automaton, log behavior must be determined in terms of model perspective, i.e., we consider the optimal alignments (Λ^1 or Λ^*) of each trace in the log for this purpose. For example, given the running example log $L = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$ and the model in Figure 2, the trace σ_1 has 5 optimal alignments, $\Lambda^*(\sigma_1) = \{\gamma_7, \gamma_8, \gamma_9, \gamma_{10}, \gamma_{11}\}$, shown in Figure 4. For this example, we assume that the alignment assigned to σ_1 by Λ^1 based on an external criterion corresponds to γ_7, i.e., $\Lambda^1(\sigma_1) = \gamma_7$. On the other hand, traces $\sigma_2 \dots \sigma_5$ are perfectly fitting, and therefore, each trace has only one optimal alignment containing only synchronous moves. In particular, given an alignment γ, in order to build the automaton, we only consider the projection of model moves, i.e., $\bar{\lambda}(\gamma)$. Table 1 shows all the projection of model moves for the alignments of log $L = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$. We use $\bar{\lambda}(\Lambda^1)_L$ and $\bar{\lambda}(\Lambda^*)_L$ to denote the application of function $\bar{\lambda}$ on all the alignments provided by the functions Λ^1 and Λ^* respectively for the traces in log L. We can omit the subindex L whenever the context is clear. Note that, by definition, any alignment projection $\bar{\lambda}(\gamma)$ is a valid complete activity sequence of the model.

Using Λ^1 (or Λ^*), the automaton is built considering all the prefixes for the sequences in $\bar{\lambda}(\Lambda^1)$ (or $\bar{\lambda}(\Lambda^*)$) as the states. For instance, given a sequence $\langle a, b, c, d \rangle$ resulting of $\bar{\lambda}(\Lambda^1)(\sigma_2)$, the states considered are ϵ, $\langle a \rangle$, $\langle a, b \rangle$, $\langle a, b, c \rangle$ and $\langle a, b, c, d \rangle$. Formally, the alignment automaton $A_A = (Q, \Sigma, \delta, \epsilon, \omega)$ is defined such that:

- The set of *states* Q corresponds to all prefixes.
- The set of *labels* Σ corresponds to the activities.
- The *arcs* $\delta : Q \times \Sigma \to Q$ define the concatenation between prefixes and activities, e.g., states $\langle a, b, c \rangle$ and $\langle a, b, c, d \rangle$ are connected by arc labeled d.
- The state corresponding with the empty sequence ϵ is the initial state.
- The function $\omega : Q \to \mathbb{R}$ determines the weight of each state according to its importance for the precision computation.

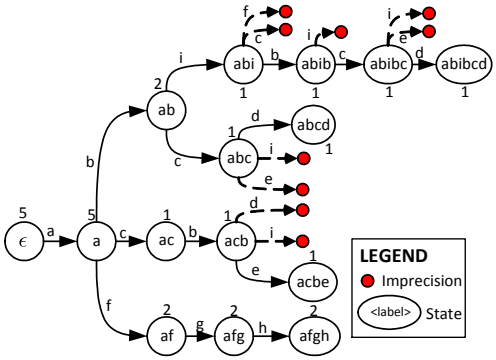


Fig. 5: Automaton using Λ^1 and the model of Figure 2.

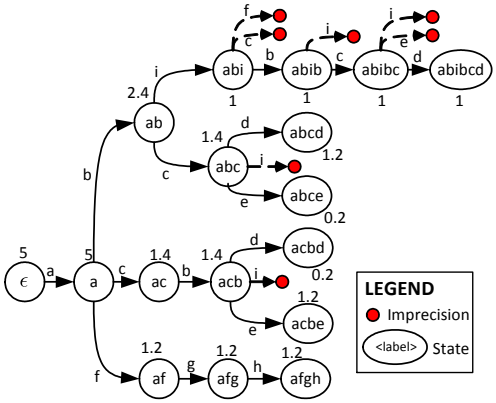


Fig. 6: Automaton using Λ^* and the model of Figure 2.

Figures 5 and 6 show the resulting automata for the model in Figure 2 and log L using the functions Λ^1 and Λ^* respectively.[3] Function ω is represented as the number in the states of the figures: informally, $\omega(s)$ represents the importance (frequency, but also alignment variation, as explained below) of state s. For instance, given the automaton of Λ^1 (Figure 5), the state $\langle a \rangle$ must have more weight than the state $\langle a, c \rangle$ because there are more traces with prefix $\langle a \rangle$ (all 5 sequences of $\bar{\lambda}(\Lambda^1)$) than the ones with prefix $\langle a, c \rangle$ (only $\langle a, c, b, e \rangle$). A naive approach for defining the ω function would be to split equally the importance of the process among all the alignment projections. This naive approach is valid for the automaton for Λ^1, since each trace in the log has just one alignment associated. However, in the case of the automaton for Λ^*, this naive approach will be biased, giving more importance to those log traces with more optimal alignments.

[3] For the sake of readability, in the figures, we use the label *abc* as an abuse of notation for referring to the sequence $\langle a, b, c \rangle$.

Hence we propose the ω function to distinguish the two situations that arise regarding the alignments used. Let us first define the ω function for alignment depending of the alignment automaton used:

- *Case* Λ^1: let M be a model, let L be an event log, let $\sigma_L \in L$ be a trace in L, let $L(\sigma_L)$ be the frequency of σ_L, and let $\gamma = \Lambda^1(\sigma_L)$ be the optimal alignment between σ_L and M obtained by Λ^1. The weight of γ is defined as: $\omega(\gamma) = L(\sigma_L)$, i.e., the weight is directly related with the frequency of the trace. For instance, given that all traces in the log of Table 1 have a frequency of 1, the weight of alignment $\Lambda^1(\sigma_5)$ is 1 (the same for the rest of alignments of Λ^1).
- *Case* Λ^*: let M be a model, let L be an event log, let $\sigma_L \in L$ be a trace in L, let $L(\sigma_L)$ be the frequency of σ_L, and let $\gamma \in \Lambda^*(\sigma_L)$ be one of the optimal alignments between σ_L and M. The weight of γ is defined as $\omega(\gamma) = L(\sigma_L) \cdot 1/|\Lambda^*(\sigma_L)|$, i.e., the weight is split equally among all the alignments of the log trace, taking into account the frequency of the trace within the log. For instance, the weight of alignment γ_9 of trace σ_1 is $1 \cdot 1/5 = 0.2$, while the weight of unique alignment of σ_5 is $1 \cdot 1/1 = 1$.

Once function ω is defined for alignments in Λ^1 and Λ^*, we define it for a prefix (intermediate state) as follows: let s be a state of the automaton, and let Λ be the sequences used to construct the automaton (Λ^1 or Λ^*). The weight of state s is defined as

$$\omega(s) = \sum_{\forall \gamma \in \Lambda} \omega(\gamma) \text{ if } s \text{ is a prefix of } \bar{\lambda}(\gamma) \text{ (or 0 otherwise)}$$

The weights of the states of figures 5 and 6 are shown next to the states. For example, in Figure 5, the state $\langle a \rangle$ appears in all 5 sequences of $\bar{\lambda}(\Lambda^1)$, and therefore, its weight is 5. In Figure 6, the state $\langle a, f \rangle$ appear in model moves projection of two alignments of Λ^* (one with weight 0.2 since the trace σ_1 has 5 optimal alignments, and the other with weight 1). Therefore, the weight of state $\langle a, f \rangle$ is 1.2.

Once the log behavior has been determined in terms of an automaton, the confrontation with the actual model behavior is required in order to determine the precision of the system. For each state of the automaton, we compute its set of *available actions*, i.e., possible direct successor activities according to the model (a_v), and then compare it with the set of *executed actions*, i.e., activities really executed in the log (e_x). Take for example state $\langle a, b, c \rangle$ of automaton created using the function Λ^1 shown in Figure 5, and the model in Figure 2. The set of executed actions of the state is $e_x(\langle a, b, c \rangle) = \{d\}$, i.e., for all traces with prefix $\langle a, b, c \rangle$, their direct successor is only d. The set of available actions for the state is $a_v(\langle a, b, c \rangle) = \{d, e, i\}$ because after performing the sequence of activities $\langle a, b, c \rangle$, the model allows to do d, e, or i. Note that, by construction $e_x(s) \subseteq a_v(s)$, i.e., the set of executed actions of a given state is always a subset of all available actions according to the model.

The activities that are allowed according to the model, but do not occur in the event log are used to collect the *imprecisions* of the system, i.e., an activity

that escapes from the log behavior. These imprecisions are represented as small filled states in the automaton in the figures. For example, the imprecisions of the state $\langle a, b, c\rangle$ are $\{d, e, i\} \setminus \{d\} = \{e, i\}$. The computation and analysis of these imprecisions are the cornerstone of the precision checking technique presented in this paper. All identified imprecisions can be analyzed and further used to correct the model and make it more precise. Furthermore, in order to globally estimate precision, these imprecisions in turn are weighted. Consequently, we define the *align-precision* (a_p) metric of a system represented by the automaton $A_A = (Q, \Sigma, \delta, \epsilon, \omega)$ as follows:

$$a_p(A_A) = \frac{\sum_{s \in Q} \omega(s) \cdot |e_x(s)|}{\sum_{s \in Q} \omega(s) \cdot |a_v(s)|}$$

For example, the precision for the automaton derived from Λ^1 shown in Figure 5 is 0.79. The precision for the automaton of Λ^* shown in Figure 6 is 0.83.

The presented precision approach in this section relies on the concept of alignments. Alignments are only defined for process models whose terminating states are reachable from their initial states. A process model whose termination state is not reachable from its initial state does not have any sequence of activities that leads to proper termination. Thus, the approach in this paper is limited to process models whose proper terminations are reachable from their initial states.

In practice, process models may have tasks whose execution are not logged or simply not labeled with any activity, i.e., invisible tasks. Furthermore, some tasks may have the same activity label, i.e., duplicate tasks. Both invisible and duplicate tasks may influence the behavior allowed by process models, therefore they need to be taken into account explicitly when measuring precision. We use tasks in place of activities when measuring precision of models with duplicate/invisible tasks, i.e., given a log and a model with duplicate/invisible tasks, a_p is computed by taking into account tasks that are executed in the log (obtained from alignments between traces in the log and the model) and possible direct successor tasks according to the model. Furthermore, although all models in this paper are shown as Petri nets, the approach is extendable to all process models for which translations to Petri nets are available, such as BPMN [17], YAWL [21], and Causal nets [3].

5 Extensions of the Precision Metric

The approach presented in Section 4 uses the prefix of complete activity sequences to represent states of the automaton. This implies that given a complete activity sequence σ, other sequences with slightly different permutation of activities are placed in different branches of constructed automaton than than σ. Given a process model that allows many possible interleaving of activities, and an event log, the approach can only provide a perfect precision

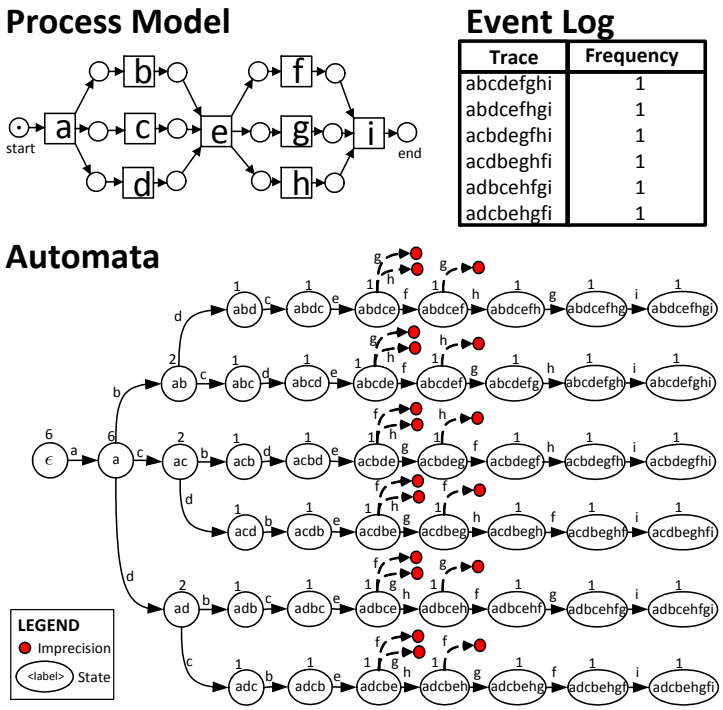


Fig. 7: Example of and event log and a process model that allows the interleaving of several activities. Precision of the model using the approach in [9] is less than perfect although all interleaving of activity b, c, d and f, g, h appear in the log.

value if all permutations of the interleaving activities are observed in the log. This requirement may be too restrictive in some cases.

Take for example the process model and the log shown in Figure 7. The model allows for the interleaved execution of b, c and d. This behavior is also reflected in the log, containing all possible permutations of b, c and d. The model also allows the interleaving of f, g and h, and all possible permutations of f, g and h are also contained in the log. One may expect a perfect precision of 1 for such model and log. However, given the presented approach, the precision is 0.8. The automaton of Figure 7 shows the imprecisions detected. Notice that prefix $\langle a, b, c\rangle$ of trace $\langle a, b, c, d, e, f, g, h, i\rangle$ and prefix $\langle a, c, b\rangle$ of trace $\langle a, c, b, d, e, g, f, h, i\rangle$ manifests as two different states even when the executed activities and their frequency in both prefixes are the same. For the given example, the minimum number of traces necessary to reach a precision of 1 is 36. This number increases exponentially with the increasing degree of concurrency of the considered model. In such cases, some level of abstraction in the way states are represented is desirable.

In Section 4 we show that different ways of aligning traces of event logs to process models (i.e., using 1 optimal alignment or all optimal alignments per trace) may also influence precision. While computing 1 optimal alignment is computationally less expensive than computing all optimal alignments, it only

provides approximations of the precision value without much more diagnostics information that can be exploited to obtain insights into precision. Representative alignments may offer a better degree of trade-off between measurement accuracy, additional insights into precision, and computation time.

Finally, other than the way states are represented, the *direction* for which the automaton is constructed may also influence precision measurement. Most of the existing approaches, e.g., [9, 24], construct an automaton from the beginning of the traces forward. Thus, choices that are made in the beginning of traces may have bigger influence on precision value than choices that are made towards the end of traces.

In this section we present three extensions of the technique presented in Section 4:

- Different *state representations* that do not take into account ordering of activities to deal with the possible incompleteness of the log (Section 5.1),
- Using *representative alignments* to get better trade-off between measurement accuracy and computation time (Section 5.2), and
- Different *directions to construct automaton* to deal with the possible bias produced by the direction used to compute precision (Section 5.3)

5.1 Abstraction on the State Representation

In [6], the states of an event log can be obtained by taking the set, multi-set, and sequence of activities. To measure precision, we propose two possible state representations that can be chosen depending on the desired level of abstractions:

- *Ordered*: A state is a *sequence* of activities. This is the same representation as the one used in [9, 24–26]. For example, the states for prefix $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ in Figure 7 are different.
- *Unordered*: A state is a *multi-set* of activities, i.e., the order among tasks does not matter, but the number of executions of each task does. For example, the states for $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ are the same, i.e., $[a, b, c]$. However, the states for $\langle a, b, i\rangle$ and $\langle a, b, i, b\rangle$ are not the same, i.e., $[a, b, i]$ and $[a, b^2, i]$ respectively, because frequency matters.

Figures 8 and 9 show the automata for the running example of Section 4, considering the unordered state representation. These automata contain differences with respect to their ordered homologous (Figures 5 and 6). For example, instead of having two states $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ for prefixes $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$, both prefixes are now represented as a single state $[a, b, c]$. This representation reduces the number of imprecisions and hence increases precision values. Using unordered state representation and precision calculation as explained in Section 4, the model in Figure 7 has a precision value of 1 (perfect). It is worth mention that in [6], the authors also propose the use of *set* as state representation. However, this is not applicable to our case: unlike

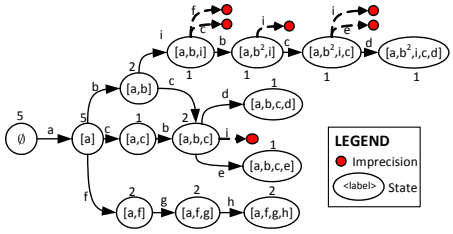


Fig. 8: Automaton using Λ^1 (unordered state representation), and the model of Figure 2.

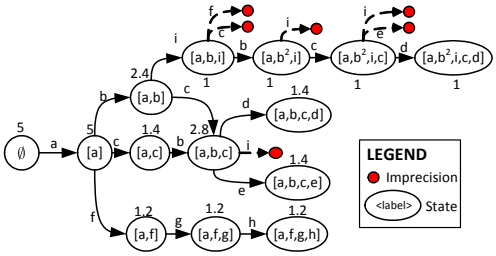


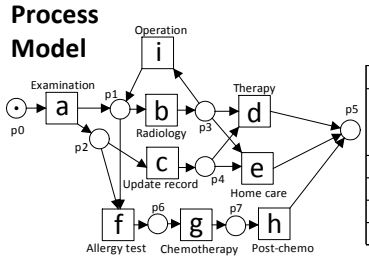
Fig. 9: Automaton using Λ^* (unordered state representation) and the model of Figure 2.

sequence or multiset, a set does not preserve the number of activities executed, and therefore, it may represent a (possible infinite) number of different model states. For example, given the model in Figure 2, the set $\{a, b, i\}$ represents $\langle a, b, i\rangle$,$\langle a, b, i, b\rangle$,$\langle a, b, i, b, i\rangle$, ... The idea of using the state representation of an aligned trace to measure precision is also introduced in [4], where precision is measured after representing the states of traces to the states of the models constructed with an ordered representation (i.e., sequence). However, similar to [9], the metric values are highly influenced by choices that are made in the beginning of the trace rather than the one that are made towards the end of the trace.

5.2 Representative Alignment

Given a trace and a process model, Λ^* provides all optimal alignments. However, as shown in [9], it is an expensive option in terms of computation time. The use of only one alignment per trace (i.e., Λ^1) solves this issue in cases where time is a priority, but may sacrifice accuracy. As a trade-off between time and accuracy, in this paper we propose precision measurement based on *representatives* of all optimal alignments (see Section 3). In this section, we revisit the precision measurement to include this notion.

Figure 10 shows the model moves projection of the representatives of all optimal alignments ($\bar{\lambda}(\Lambda^R)$) for the running example shown previously in Section 4. The construction of the automaton is defined identically as in the presented approach, with only one difference: the weight function of the alignments. Let σ_L be a trace of log L, let $L(\sigma_L)$ be the frequency of σ_L, let



Representatives

L

trace	*freq*
$\sigma_1 = \langle a \rangle$	1
$\sigma_2 = \langle a, b, c, d \rangle$	1
$\sigma_3 = \langle a, c, b, e \rangle$	1
$\sigma_4 = \langle a, f, g, h \rangle$	1
$\sigma_5 = \langle a, b, i, b, c, d \rangle$	1

$\bar{\lambda}(\Lambda^R)$	$rep_M(\gamma)$
$\langle a, f, g, h \rangle$	1
$\langle a, c, b, e \rangle$	2
$\langle a, c, b, d \rangle$	2
$\langle a, b, c, d \rangle$	1
$\langle a, c, b, e \rangle$	1
$\langle a, f, g, h \rangle$	1
$\langle a, b, i, b, c, d \rangle$	1

Fig. 10: Process model, traces, and representatives of all optimal alignments of all traces.

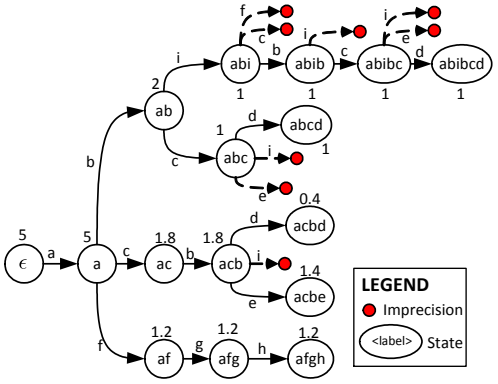


Fig. 11: Automaton using Λ^R (ordered state representation) and the model of Figure 2.

M be a model, and let $\gamma \in \Lambda^R(\sigma_L)$ be a representative alignment for trace σ_L. In such case, the weight of the alignment $\omega(\gamma)$ needs to be proportional to the number of alignments represented by γ, i.e., $rep(\gamma)$. Thus, we define $\omega(\gamma) = L(\sigma_L) \cdot rep(\gamma) / \sum_{\gamma' \in \Lambda^R(\sigma_L)} rep(\gamma')$. For instance, given the trace σ_1 in Figure 10, let γ_1 be the representative alignment such that $\bar{\lambda}(\gamma_1) = \langle a, c, b, d \rangle$. The number of alignments represented by γ_1 is $rep(\gamma_1) = 2$. The total number of optimal alignments represented by the representative alignments associated with σ_1 is $\sum_{\gamma' \in \Lambda^R(\sigma_1)} rep(\gamma') = 5$. Hence, the weight $\omega(\gamma_1) = 1 \cdot 2/5 = 0.4$. As another example in Figure 10, let γ_2 be the only representative alignment associated with σ_5, such that $\bar{\lambda}(\gamma_2) = \langle a, b, i, b, c, d \rangle$. The representative alignment γ_2 represents 1 optimal alignment. Since the number of all optimal alignments represented is $\sum_{\gamma' \in \Lambda^R(\sigma_5)} rep(\gamma') = 1$, the weight of γ_2 is $\omega(\gamma_2) = 1 \cdot 1/1 = 1$.

Figures 11 and 12 reflect the automata for the running example of the previous section, when representative alignments and different state representations are used.

Note that there can be more than one ways to compute representative alignments from a given model and a trace. Given an event log and a model, the selection of representative alignments between each trace in the log and the model obviously influences the automata that can be constructed between the log and the model.

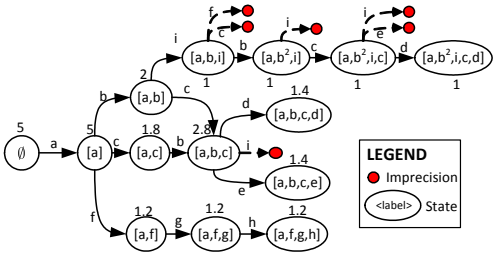


Fig. 12: Automaton using Λ^R (unordered state representation) and the model of Figure 2.

5.3 Forward and Backward Precision

In the approach presented in Section 4, the prefixes of the complete activity sequences are used to build the automaton. For example, given a complete activity sequence $\langle a, b, c, d\rangle$, the states constructed from the sequence are the empty sequence ϵ (corresponding with $\langle \bullet a, b, c, d\rangle$, where $\bullet$ indicates a point of interest in the sequence), $\langle a\rangle$ (for $\langle a \bullet b, c, d\rangle$), $\langle a, b\rangle$ (for $\langle a, b \bullet c, d\rangle$), $\langle a, b, c\rangle$ (for $\langle a, b, c \bullet d\rangle$) and finally $\langle a, b, c, d\rangle$ (for $\langle a, b, c, d\bullet\rangle$). In other words, only the activities in the past are used and we move *forward* on the complete activity sequences. This approach is used by all existing precision checking techniques [24–26].

In [6], the authors show that any point in the sequence (represented as $\bullet$) may represent two complementary visions: the *past* activities seen until that point (as it has been shown above), but also the *future* activities to come until the ending of the case. For instance, given $\langle a, b \bullet c, d\rangle$, $\langle a, b\rangle$ are the activities occurred, while $\langle c, d\rangle$ are the activities to happen. Both $\langle a, b\rangle$ and $\langle c, d\rangle$ are used in [6] as two different states that can be derived from the same point in the sequence. In this section, we use the same idea to present a *backward* precision measurement, that complements the *forward* approach presented before. The combination of both metrics will lead to a measurement unbiased by the direction of the precision checking. For the sake of clarity we will use ordered state representation to illustrate the remainder of the section, although the analogous procedure is applicable for unordered representation.

Let Λ be the option chosen to compute precision, i.e., Λ^1, Λ^* or Λ^R. In order to build the automaton for the backward precision measurement, we consider the prefixes of the *reversed* complete activity sequences in $\bar{\lambda}(\Lambda)$. In other words, given $\bar{\lambda}(\gamma) = \langle a, b, c, d\rangle$ of the alignment $\gamma \in \Lambda$, we use $\bar{\lambda}'(\gamma) = \langle d, c, b, a\rangle$ to determine the states, resulting in the following 5 states: ϵ (corresponding with $\langle \bullet d, c, b, a\rangle$), $\langle d\rangle$ (for $\langle d \bullet c, b, a\rangle$), $\langle d, c\rangle$ (for $\langle d, c \bullet b, a\rangle$), $\langle d, c, b\rangle$ (for $\langle d, c, b \bullet a\rangle$) and finally $\langle d, c, b, a\rangle$ (for $\langle d, c, b, a\bullet\rangle$). Analogously, the set of complete activity sequences of M is also reversed.[4] The rest of the precision checking is performed as it is described in Section 4.

[4] Notice that, for the case of Petri nets with one unique initial and final markings, the set of all reversed complete activity sequences can be generated by simulating the behavior of a net obtained from the original net by reversing its arcs and swapping their initial with final marking.

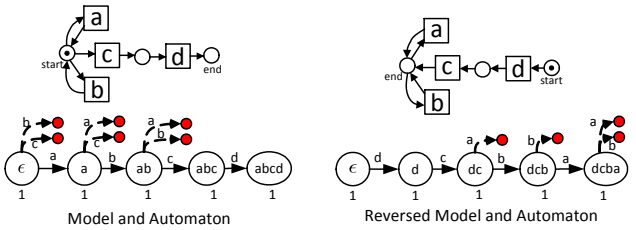


Fig. 13: Example of model and resulting automaton for both forward and backwards approaches.

Figure 13 shows an example of two automata constructed by moving in forward direction (left) and by moving backward (right). Notice the difference of identified imprecisions shown by the two automata. Finally, precision values obtained using forward and backward-constructed automaton can be combined (e.g., the average), resulting in a balanced precision metric unbiased by the direction of the automaton constructed. Note that more sophisticated and flexible combinations of both metrics are also possible. In Section 6, we investigate the differences in precision values produced by the various approaches using a variety of even logs and models.

6 Experiments

We have implemented the proposed precision calculation as a ProM 6 plugin named "Check Precision based on Align-ETConformance" in the "ETConformance" package, publicly available from `www.processmining.org`. We used it to perform a range of experiments to test the robustness of our proposed approach using both synthetic and real-life models (Petri nets) and logs.

6.1 Evaluating Unidimensionality of Metrics

The first set of experiments was performed to evaluate the precision measurements provided by the proposed metrics. In particular, we measured whether the proposed precision metrics are unidimensional [33], i.e., not sensitive to non-fittingness of event logs. We measured precision between various logs and models whose expected values are known. Furthermore, we compared the values obtained against existing state-of-the-art metrics for precision: etc_P [24], behavioral precision [34], and weighted behavioral precision [12].

By combining the models and log in Figure 1 in various ways, we created new models whose expected precision values are between the two extremes. Two models were combined by merging the end place of one with the initially marked place of another. The merged models were named according to the name of their original models, e.g., **PF** model is the result of merging the end place of **P** with the initially marked place of **F**. The activity names in the original models and logs were renamed before the models and logs were

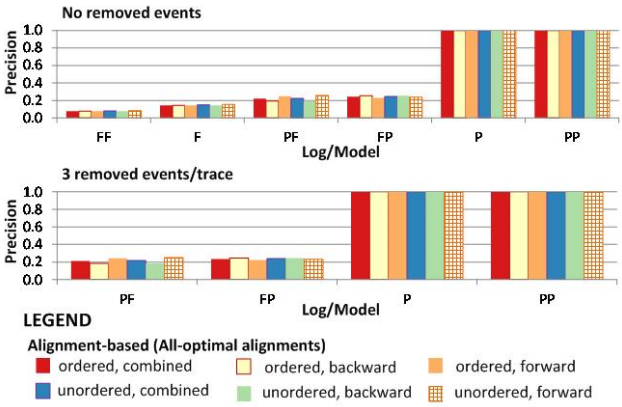


Fig. 14: Precision values of the logs/models in Figure 1 and their combinations provided by alignment-based approach (i.e., computed using all optimal alignments, ordered, and forward-constructed automata). If all behavior are observed in the original logs, all measurements are insensitive to non-fitting traces.

merged such that the original models and logs can be easily distinguished from the merged results. Precision values were measured 30 times using 30 event logs, each consists of 5,000 traces, generated by simulating the precise model (i.e., **PP**). For sake of completeness, we also measured the precision of the overfitting model (**P**) and the flower model (**F**) using 30 logs of 5,000 traces generated by simulating the **P** model. This way, *each log contains all the possible behavior of the model that generates it* (i.e., all directly follow relations between two activities that are allowed according to the model are recorded in the log).

The top part of Figure 14 shows the alignment-based precision values, measured using all optimal alignments per trace of the logs. The experiment with one and representative alignments per trace yields identical results. This result shows that by observing sufficiently enough behavior in the event logs, all alignment-based metrics provide similar intuition about precision of models, i.e., overfitting models have high precision values and "flower" models have low precision values. Note that there are slight differences between various configurations of metrics, i.e., states (ordered/unordered) and forward/backward constructed automata.

To evaluate the robustness of the metrics against non-fitting logs, we took the models and logs from the previous experiments and created unfitting logs by removing n random events per trace from the fitting logs. To ensure that the logs are unfitting, only activities that belong to the precise part (i.e., mapped to **P** part) are removed. Furthermore, the measurements are compared against existing metrics. We use the CoBeFra tool [13] to measure behavioral precision [34] and weighted behavioral precision [12]) and use ProM 6 to measure etc_P. The bottom part of Figure 14, Figures 15–16, and Table 2 show some of the results.

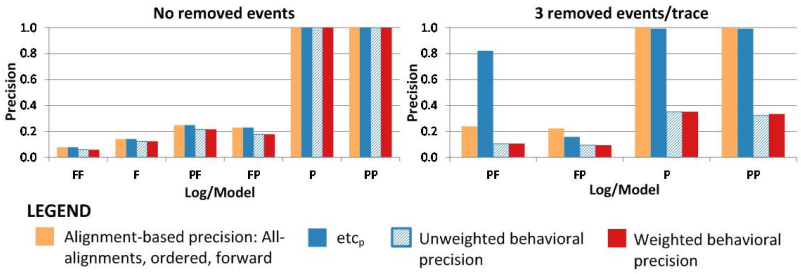


Fig. 15: Comparison between precision values obtained using alignment-based approach (i.e., computed using all optimal alignments, ordered, and forward-constructed automata) and other metrics (etc_P [24], behavioral precision [34], and weighted behavioral precision [12]). Only the alignment-based approach is not sensitive to non-fitting logs/models.

The bottom part of Figure 14 shows that the metrics proposed in this paper are robust to fitness problems. Even in cases where almost half of the events in all traces are removed, all alignment-based metrics provide similar value as the ones provided for perfectly fitting traces. Figure 15 shows a comparison between the precision values provided by alignment-based metrics and other existing metrics. For readability, we only show one alignment-based metric: the one computed using all-optimal alignments and forward-constructed automata whose states are constructed by taking into account activity ordering. Note that in cases where logs are perfectly fitting the models, all metrics provide similar precision intuition. In fact, the alignment-based precision values shown in Figure 15 are the same as the etc_P values. However, in cases where logs are non-fitting, other metrics may show misleading precision insights. The etc_P metric provides low precision for model **PF** with respect to perfectly fitting logs (i.e., 0.25). However, the value rises to 0.82 when 3 events are removed from the logs, because for all non-fitting traces it ignores the rest of the traces after the first non-fitting event occur. Similarly, both weighted and unweighted behavioral precision metrics provide lower precision values for non-fitting logs than the ones provided for perfectly fitting logs. Even for overly fitting models **P** and **PP**, both metrics provide precision values below half (i.e., indicating the models are imprecise). This occurs because both metrics mixed both perfectly-fitting and non-fitting traces in construction of artificial negative events, which leads to misleading construction of artificial negative events.

Figure 16 shows the influence of noise by removing some events in the logs. As shown in the figure, other than the alignment-based precision metric, precision values of all metrics may change significantly even with only one event removed from all traces. Due to the randomness of the location of removed events, the etc_P metric may both increases or decreases with the presence of non-fitting traces. Both weighted and unweighted behavioral precision metrics decreases when more events are removed because incorrect artificial negative events are introduced. Note that the number of negative events tends to decrease when traces in the log gets more vary because of the removal of events.

The set of experiments also shows some interesting insights into differences between alignment-based metrics. Table 2 reports the results for model **PF**. In

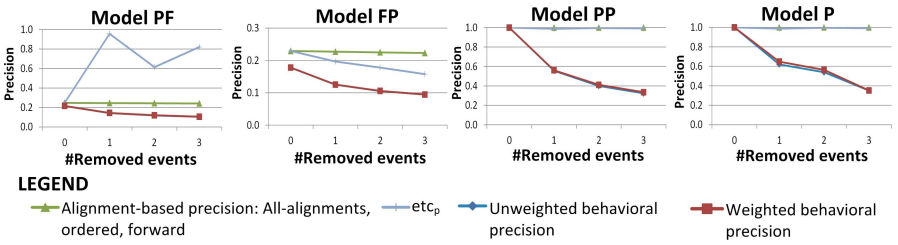


Fig. 16: Precision values of different metrics for perfectly fitting logs and non-fitting logs created by removing some events in the logs. Only the alignment-based approach metric (i.e., computed using all optimal alignments, ordered, and forward-constructed automata) is insensitive to non-fitting logs.

Table 2: Precision values of the **PF** model, measured using different state representations (ordered/unordered) and direction (forward/backward). If all behavior are observed, both 1-alignment and representative alignment provide good approximation of all-alignments.

		Automata Construction Direction											
		Forward				**Backward**				**Combined**			
#Removed		**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Ord.	one	0.25	0.24	0.24	0.24	0.19	0.19	0.19	0.18	0.22	0.22	0.21	0.21
	rep	0.25	0.25	0.24	0.24	0.19	0.19	0.19	0.19	0.22	0.22	0.22	0.21
	all	0.25	0.25	0.24	0.24	0.19	0.19	0.19	0.19	0.22	0.22	0.22	0.21
Unord.	one	0.26	0.25	0.25	0.25	0.19	0.19	0.19	0.18	0.22	0.22	0.22	0.22
	rep	0.26	0.26	0.25	0.25	0.19	0.19	0.19	0.19	0.22	0.22	0.22	0.22
	all	0.26	0.25	0.25	0.25	0.19	0.19	0.19	0.19	0.22	0.22	0.22	0.22

LEGEND

Ord./Unord : ordered/unordered state representations

One/rep/all : one/representative/all alignments

cases where the whole behavior is recorded in event logs, precision values only depend on the state representation of the automaton (ordered/non-ordered) and the direction for the automata construction. When all possible behavior are observed, the automata constructed using 1-alignment and all-alignments per trace are identical. Similar results are obtained from the experiments using the other models (**P,F,FP,PP,FF**). Table 2 shows slight differences between precision values that are measured using different state representations or different directions in the automata construction.

Figure 17 shows a comparison between precision values provided by the two metrics for models **PF** and **FP**. As shown in the figure, precision values of alignment-based metrics provided by forward-constructed automata for model **PF** is higher than the values provided by backward-constructed automata for the same model, regardless of the noise level and the state representation (ordered/unordered). In contrast, the values provided by the latter is higher than the former for the **FP** model. This shows that the position of the precise part of the models influences precision values. Precision values are higher when the direction of constructed automata starts with precise part of process models. In this case, we clearly see the influence of forward/backward direction of constructed automata to precision values. To balance the influence, one of the simplest way is to take the average between the values provided by both

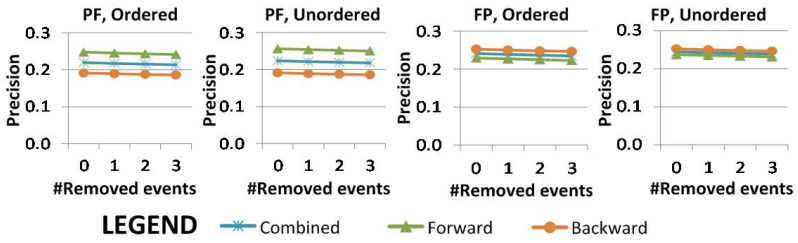


Fig. 17: Precision values of the **PF** and **FP** using all-alignments per trace, with different state representations (ordered/non-ordered) and direction (forward/backward). Higher precision is obtained when the direction of automata construction starts with precise part of the models.

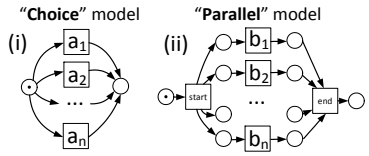


Fig. 18: (i) A model that only allows one activity per trace, and (ii) A model that allows interleaving between all activities.

directions. Figure 17 shows that the precision values obtained by combining both values are almost similar between model **PF** and **FP**.

In this section, non-fitting logs are created by removing activities randomly. Given a process model and a fitting trace, there are other ways to make the trace non-fitting, such as swapping some activities and add extra activities to the trace randomly. Regardless of the approach to introduce noise, an optimal alignment between a non-fitting trace and the model provides a good "guess" of a complete activity sequences allowed by the model that should have occurred instead of the trace. This way, precision is measured independently from other conformance metrics, i.e., the fitness metric. Other approaches investigated in this section do not explicitly handle such non-fittingness. Hence, they are not unidimensional and may yield misleading results as shown by the experiment results.

6.2 Observed Behavior Requirements

The second set of experiments were conducted to investigate how much behavior must be observed in the event logs in order to measure perfect precision accurately. We use two models that, despite having the same number of activities, have totally different number of complete activity sequences. The first model only allows choice among activities (i.e., is named **"Choice"** model) and the second model allows the interleaving of all activities (i.e., is named **"Parallel"** model) (see Figure 18). For our experiments, we used models that consist of 9 activities (with invisible task "start" and "end" for the **"Parallel"** model).

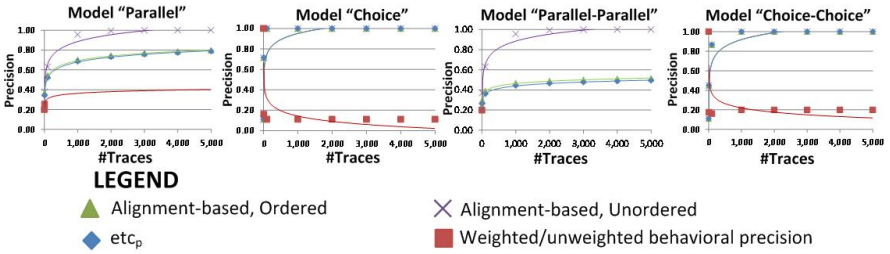


Fig. 19: Alignment-based precision values for **"Parallel"**, **"Choice"**, **"Parallel-Parallel"**, and **"Choice-Choice"** models. The values provided using unordered representation of states automata provide perfect precision without having to observe all interleaving behavior. Missing values on weighted/unweighted behavioral precision indicate that no result was obtained after 1 hour computation.

Similar to the set of experiments in Section 6.1, we randomly generated perfectly fitting logs for both models with various number of traces per log and then measured their precision values. Experiments are repeated 30 times for each combination of models and number of traces per log. We conducted the same experiments with models constructed by merging the two models in various order (**"Choice-Choice"**, **"Choice-Parallel"**, **"Parallel-Choice"**, **"Parallel-Parallel"**). The results of the experiments are shown in Figure 19 and Figure 20.

Both Figure 19 and Figure 20 reveal that even if logs are generated from models, all alignment-metrics require some degree of log completeness before they provide perfect precision value of 1.00. As expected, a perfect precision value for a **"Choice"** and a **"Choice-Choice"** models can be obtained after observing much fewer traces than the ones required to obtain the same precision value for both **"Parallel"** and **"Parallel-Parallel"** models. In theory, the minimum number of traces in an event log required to see all possible behavior of a **"Choice"** model with 9 activities is 9, while the minimum number of traces to see all possible interleaving of activities in a **"Parallel"** model is $9! = 362,880$ traces. In all experiments, alignment-based precision metrics with unordered automata state representation provide perfect precision values with less number of observed traces than the one with ordered automata. This shows that in conditions where not all behavior are observed in event logs, precision values computed using unordered automata state representation provides an upper-bound for the ones computed using ordered automata. The figure also shows that in all experiments, the etc_P values are the same as the alignment-based precision values computed using ordered automata state representation because all traces perfectly fit their models. Interestingly, in the experiments with model **"Choice"** and **"Choice-Choice"**, both the weighted and unweighted behavioral precision metrics provide a perfect value (1.00) for logs with only one trace but provide very low values (below 0.2) for other logs that contain more than one trace (i.e., logs with 10, 100, 1,000, to 5,000 traces). The reason the (un)weighted behavioral precision values is so high is that the artificial negative events construction only take into account logged activities. When an activity in a trace of the logs is replayed to construct ar-

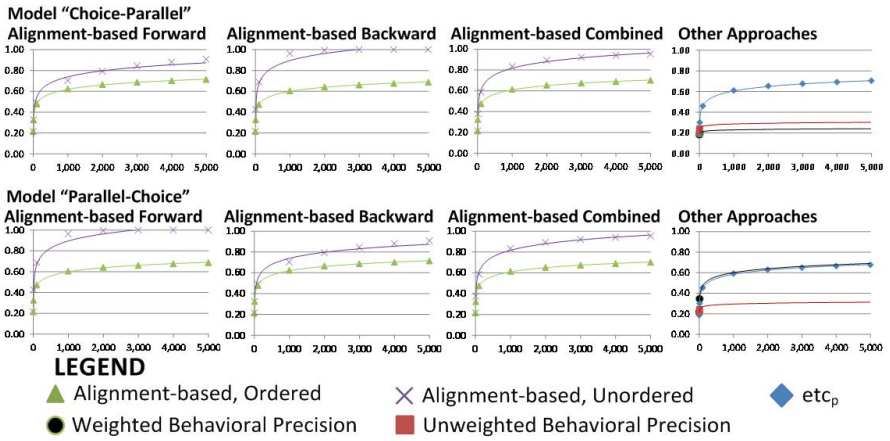


Fig. 20: Precision values of models with combination of choice and parallel control-flow patterns. Higher precision values are obtained when automata are constructed from the direction where the parallel part of the models exists (the first three figures). Missing values indicate that no result was obtained after 1 hour computation.

tificial negative events, other than the logged activity both models allow only unlogged activities (invisible tasks). Thus, no negative artificial events were constructed and therefore the precision of the models with respect to the logs are 1.00. Furthermore, the results also show that the time spent to compute alignment-based metrics is not necessarily higher than the time required to compute other existing metrics such as the (un)weighted behavioral precision. In some of the experiments with models **"Parallel"** and **"Parallel-Parallel"**, no result was obtained after 1 hour computation for (un)weighted behavioral precision while the alignment-based precision metrics were computed in less than 1 minute for each pair of model and log.

Figure 20 shows the precision values obtained from experiments with models **"Choice-Parallel"** and **"Parallel-Choice"**. Interestingly, the results of the experiment with **"Choice-Parallel"** model performed using forward automata construction (i.e., top-left-most of Figure 20) is identical to the one given by the experiment with **"Parallel-Choice"** model using backward automata construction (bottom-second from left of Figure 20). Similarly, the results of experiment with **"Parallel-Choice"** model performed using forward automata construction (i.e., bottom-left of Figure 20) is identical to the one given by the experiment with **"Choice-Parallel"** model using backward automata construction (top-second from left of Figure 20). These results show that precision values are influenced by the location of parallel-choice constructs: precision values are higher when automata are constructed from the direction where parallel construction lies. The combined precision value computed by averaging the precision values obtained from both forward and backward-constructed automata is less influenced by such construction as shown in the third figures from the left side of Figure 20. As shown in the figures, the measured precision values for both **"Choice-Parallel"** and

Table 3: Real-life logs and models used for experiments

Log	#Cases	#Events	Process Model		#Deviation/trace
			#Place	#Trans	
Bouw-1	139	3,364	33	34	9.75
Bouw-4	109	2,331	31	31	7.27
MLog1	3,181	20,491	15	12	5.33
MLog2	1,861	15,708	16	19	1.45
MLog3	10,271	85,548	24	21	14.50
MLog4	4,852	29,737	16	27	2.09
MLog5	25,846	141,755	14	24	1.21
IsalaLog	77	459	26	39	0.68

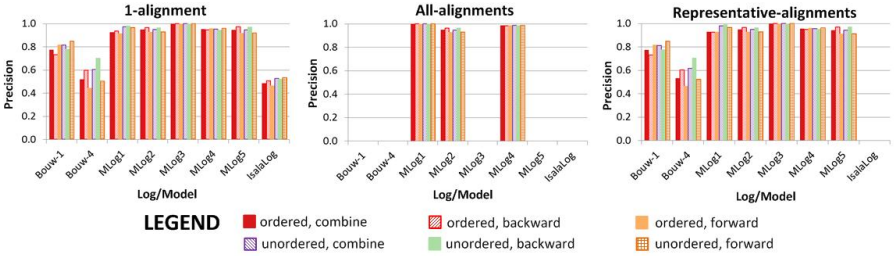


Fig. 21: Precision values of real-life logs and models. Only the 1-alignment approach manages to provide precision results for all logs/models.

"Parallel-Choice" models using the combined precision values are identical. None of non-alignment-based approaches in this set of experiments managed to provide perfect precision values. Note that no result was obtained after 1 hour of computation for both weighted and unweighted behavioral precision metric calculations and logs of size of 1,000 traces and larger.

6.3 Real-life Logs and Models

To evaluate the applicability of the approach to handle real life logs, we used 8 pairs of process models and logs from two different domains (see Table 3), where 7 logs and models were obtained from municipalities in the Netherlands. In particular, we took the collections of logs and models from the CoSeLoG project [1, 15]. The remaining pair of log and model is obtained from a hospital in the Netherlands[5]. The logs and models from municipalities are related to different types of building permission applications, while the hospital log is related to patient handling procedure. All processes have unlogged tasks, and some of the models allow loops. Table 3 shows an overview of the logs and models used in the experiments. #Deviations/trace column indicates the number of asynchronous moves after aligning all traces in the logs with their corresponding models. As shown in Table 3, all logs are not perfectly fitting to the corresponding models. We measure the precision values for all logs and the computation time required. The results are shown in Figure 21 and Figure 22.

[5] see `http://www.healthcare-analytics-process-mining.org/`

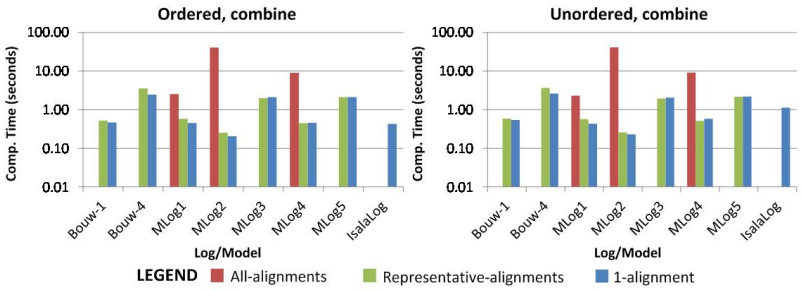


Fig. 22: Computation time comparison of alignment-based precision measurement using combined values (from backward and forward automata construction). Y-axis values are shown in a *logarithmic* scale.

Figure 21 reports the precision values obtained for real-life logs and models. Only the approach based on 1-alignment provides precision values for all real-life logs and models in the experiments. The approach based on all-optimal alignments per trace had out-of-memory problems when dealing with relatively complex process models and logs such as "Bouw-1" (33 places, 34 transitions), "Bouw-4" (31 places, 31 transitions), and "MLog-3" (24 places, 21 transitions). Precision measurements based on representative of optimal alignments also had the same problems dealing with the hospital log (i.e., "IsalaLog"). Although the model of the log is relatively small, it contains many unlogged tasks (tasks whose execution are not logged), allows loops, and allow many interleaving activities such that the size of state space required to compute even representative of all optimal alignments is large and does not fit memory.

Nevertheless, notice the similarity of the computed precision values using all three alignments (1-align, all-align, and representatives). From all pairs of logs and models, only 2 of them have precision value below 0.7. This shows that in reality, process models are made to be relatively precise such that meaningful insights into the process can be obtained. Interestingly, different precision values are provided by different metrics in the experiment with log and model "Bouw-4" when both one and representative alignments are used. The precision value provided by ordered-forward metric for the model is around 0.44 (showing imprecision) while the unordered-backward precision metric provides a value of 0.7 (i.e., precise). As discussed in Section 6.1 and Section 6.2, this indicates that more observations are required to measure the particular log and model accurately.

Figure 22 reports the computation time required to measure precision of real-life logs and models using the alignment-based approach with combined precision values between forward and backward-constructed automata. The y-axis of the charts are shown in logarithmic scale. As shown in the figure, the computation time of precision measurement with all-alignments takes much longer than the ones required by one or representative alignments. All measurements using 1-alignment/representative alignments were computed in less than 10 seconds. Notice the similarity between the left and right graph on the figure (except the IsalaLog that has out-of-memory problem in the approach with representative alignments). In fact, we obtained identical results

Table 4: Representative optimal alignments in real-life logs

Log	#Opt. Alignments (lower bound)			#Opt. Alignments represented		#Representatives per trace		
	Min	Max	Avg	Min	Max	Min	Max	Avg
Bouw-1	1	4,096	69.41	1	4,096	1	1	1.00
Bouw-4	1	5,300,287	146,617.87	1	5,300,287	1	2	1.01
MLog1	1	1,844	74.24	1	976	1	2	1.00
MLog2	1	6	1.05	1	6	1	1	1.00
MLog3	1	136	13.95	1	84	1	2	1.12
MLog4	1	48	3.65	1	48	1	1	1.00
MLog5	1	54	3.46	1	54	1	2	1.00

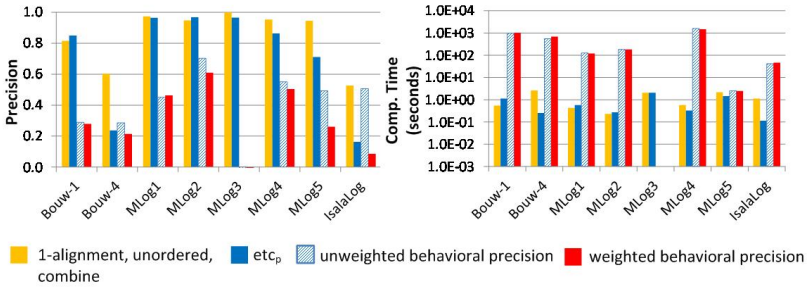


Fig. 23: Precision values (left) and computation time (right) comparison between alignment-based precision measurements and existing precision measurements using real-life logs and models. Y-axis values in the right chart are shown in a *logarithmic* scale. Missing values indicate that no result was obtained after 1 hour of computation

for all other combination of state representations (ordered/unordered) and directions where automata is constructed (forward/backward). This shows that the different directions of the automata construction and state representations are not significantly influencing computation time. Instead, most computation time of precision measurement is spent in the alignment of logs and process models. Another interesting observation is that the time spent to compute representative alignments are similar to the time spent to compute 1-alignment. Thus, we recorded the number of generated representatives for the experiments and other statistics to investigate this. The results are shown in Table 4.

Table 4 shows that the average number of representatives per trace that one can obtain using the extension of the $A^\star$ algorithm from real-life logs and models is close to one in all experiments (see #Representatives per trace column). This explains why the computation time between precision based on 1-alignment is not much different than the one based on representatives. Interestingly, some traces in real-life logs have more than ± 5.3 million optimal alignments (see log "Bouw-4"). Further investigations found that a trace with so many optimal alignments is incomplete (only consists of one event) while its model allows for many interleavings between activities. Hence, there are many possible activity interleavings that one can perform to complete the process and construct an optimal alignment from the trace. The optimal alignment of the same trace is also the one that represents more than ± 5.3 million optimal alignments. This shows that it is also important to take into account fitness values before measuring precision. Precision measurements that are based on severely non-fitting logs may be misleading.

Figure 23 shows a comparison of precision values and computation time between alignment-based precisions (represented by the ones computed using 1-alignment per trace, unordered state representation, and averaging values between forward-backward constructed automata) and other approaches. In most cases, the alignment-based approach yields higher values than other approaches. The right-side of the figure shows that the computation time of both weighted and unweighted behavioral precision is much higher than the computation time of both the alignment-based precision and etc_P.

7 Conclusions

The quality of process models is often measured merely based on the proportion of observed behavior in event logs that can be reproduced by the model (i.e., fitness). Models that allow for much more behavior than the behavior observed in event logs may provide misleading insights. Many approaches to quantify precision assume perfect fitness, while this assumption is rarely being satisfied in practice. This results in unreliable precision measurements as shown in this paper. Therefore, we have developed an approach that first aligns an event log and a process model. This step is crucial to measure precision more accurately, especially in those cases where the log is non-fitting. The prealignment of log and model makes it possible to explicitly identify deviations and measure precision more accurately.

Given a process model and an event log, we use an automata-based approach to measure precision. Automata are mainly used as a means to juxtapose the behavior of the model with the behavior observed in the log. We showed that the choice of state representation in the construction of the automata influences the precision value obtained. As illustrated by our experiments, a state representation that takes into account the ordering of activity and not just the frequency requires much more observed behavior to provide high precise value. In cases where the log is not complete (i.e., more behavior in reality may occur than the behavior recorded in the log), the state representation that ignores ordering can be used to provide an upper bound for the precision value of the model. Furthermore, we have identified several behavioral properties of process models that may cause a biased precision measurement depending on the choice of direction to construct automata. To minimize such bias, we proposed average precision value between the automata obtained using forward and backward directions.

Computing all optimal alignments between a process model and an event log is computationally expensive, being not feasible in practice. We showed that precision values based on both 1-alignment and representative optimal alignments are good approximation of the values obtained using the all-optimal alignments approach. We also showed that the precision measurement based on representative optimal alignments provides a trade-off between computation time and metric quality, providing more diagnostics information (i.e., lower bound of the number of optimal alignments). Nevertheless, identifying the "optimal" trade-off between computation time and rich diagnostic information remains a challenge for practical cases.

We stress that precision alone is not sufficient to determine the quality of process model with respect to its observed behavior. Other dimensions, such as fitness, generalization, and simplicity, must be considered altogether to provide a comprehensive evaluation on how "good" is a model, given its executions.

References

1. van der Aalst, W.: Business Process Configuration in The Cloud: How to Support and Analyze Multi-Tenant Processes? In: G. Zavattaro, U. Schreier, C. Pautasso (eds.) Proceedings of the 9th IEEE European Conference on Web Services (ECOWS 2011), pp. 3–10. IEEE Computer Society Press (2011)
2. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Verlag (2011)
3. van der aalst, W., Adriansyah, A., van Dongen, B.: Causal Nets: a Modeling Language Tailored Towards Process Discovery. In: Proceedings of the 22nd international conference on Concurrency theory, CONCUR'11, pp. 28–42. Springer-Verlag, Berlin, Heidelberg (2011)
4. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying History on Process Models for Conformance Checking and Performance Analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**(2), 182–192 (2012)
5. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge, MA, USA (2004)
6. van der Aalst, W., Rubin, V., Verbeek, H., van Dongen, B., Kindler, E., Gunther, C.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and System Modeling **9**(1), 87–111 (2010)
7. Adriansyah, A., van Dongen, B., van der Aalst, W.: Conformance Checking Using Cost-Based Fitness Analysis. In: Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference, Helsinki, Finland, EDOC '11, pp. 55–64. IEEE Computer Society (2011)
8. Adriansyah, A., van Dongen, B., van der Aalst, W.: Memory-Efficient Alignment of Observed and Modeled Behavior. Tech. Rep. BPM-03-03, BPMcenter.org (2013)
9. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B., van der Aalst, W.: Alignment Based Precision Checking. In: M.L. Rosa, P. Soffer (eds.) Business Process Management Workshops, *Lecture Notes in Business Information Processing*, vol. 132, pp. 137–149. Springer Berlin Heidelberg (2013)
10. Adriansyah, A., Sidorova, N., van Dongen, B.: Cost-Based Fitness in Conformance Checking. In: 11th International Conference on Application of Concurrency to System Design (ACSD), 2011, pp. 57–66 (2011)
11. Banescu, S., Zannone, N.: Measuring privacy compliance with process specifications. In: Proceedings of the 2011 Third International Workshop on Security Measurements and Metrics, METRISEC '11, pp. 41–50. IEEE Computer Society (2011)
12. vanden Broucke, S., de Weerdt, J., Baesens, B., Vanthienen, J.: Improved Artificial Negative Event Generation to Enhance Process Event Logs. In: J. Ralyt, X. Franch, S. Brinkkemper, S. Wrycza (eds.) Advanced Information Systems Engineering, *Lecture Notes in Computer Science*, vol. 7328, pp. 254–269. Springer Berlin Heidelberg (2012)
13. vanden Broucke, S., de Weerdt, J., Vanthienen, J., Baesens, B.: A Comprehensive Benchmarking Framework (CoBeFra) for Conformance Analysis between Procedural Process Models and Event Logs in ProM. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, part of the IEEE Symposium Series on Computational Intelligence 2013, SSCI 2013 (2013)
14. Buijs, J., van Dongen, B., van der Aalst, W.: On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In: R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, I. Cruz (eds.) On the Move to Meaningful Internet Systems: OTM 2012, *Lecture Notes in Computer Science*, vol. 7565, pp. 305–322. Springer Berlin Heidelberg (2012)
15. Buijs, J., van Dongen, B., van der Aalst, W.: Towards cross-organizational process mining in collections of process models and their executions. In: F. Daniel, K. Barkaoui, S. Dustdar (eds.) Business Process Management Workshops, *Lecture Notes in Business Information Processing*, vol. 100, pp. 2–13. Springer Berlin Heidelberg (2012)

16. Cook, J., Wolf, A.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology (TOSEM) **8**, 147–176 (1999)
17. Dijkman, R., Dumas, M., Ouyang, C.: Semantics and Analysis of Business Process Models in BPMN. Information and Software Technology **50**(12), 1281–1294 (2008)
18. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer (2013)
19. Gerke, K., Cardoso, J., Claus, A.: Measuring the Compliance of Processes with Reference Models. In: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I, OTM '09, pp. 76–93. Springer-Verlag, Berlin, Heidelberg (2009)
20. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering Expressive Process Models by Clustering Log Traces. IEEE Trans. on Knowl. and Data Eng. **18**, 1010–1027 (2006)
21. Hofstede, A., van der Aalst, W., Adams, M., Russell, N.: Modern Business Process Automation. Springer-Verlag (2010)
22. Kristensen, L., Schmidt, K., Valmari, A.: Question-guided Stubborn Set Methods for State Properties. Formal Methods in System Design **29**(3), 215–251 (2006)
23. M.Petkovic, Prandi, D., Zannone, N.: Purpose Control: Did You Process the Data for the Intended Purpose? In: W. Jonker, M. Petkovic (eds.) Secure Data Management, *Lecture Notes in Computer Science*, vol. 6933, pp. 145–168. Springer (2011)
24. Munoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: Proceedings of the 8th international conference on Business Process Management, BPM'10, pp. 211–226. Springer-Verlag, Berlin, Heidelberg (2010)
25. Munoz-Gama, J., Carmona, J.: Enhancing precision in process conformance: Stability, confidence and severity. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, April 11-15, 2011, Paris, France, pp. 184–191. IEEE (2011)
26. Munoz-Gama, J., Carmona, J.: A General Framework for Precision Checking. International Journal of Innovative Computing, Information and Control (IJICIC) **8**(7(B)), 5317–5339 (2012)
27. Murata, T.: Petri nets: Properties, Analysis and Applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
28. Pande, P., Neuman, R., Cavanagh, R.: The Six Sigma Way: How GE, Motorola, and Other Top Companies are Honing Their Performance. McGraw-Hill (2000)
29. Porter, L., Parker, A.: Total Quality Managementthe Critical Success Factors. Total Quality Management **4**(1), 13–22 (1993)
30. Rozinat, A., van der Aalst, W.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems **33**, 64–95 (2008)
31. Rozinat, A., de Jong, I., Günther, C., van der Aalst, W.: Process Mining Applied to the Test Process of Wafer Steppers in ASML. IEEE Transactions on Systems, Man and Cybernetics - Part C **39**, 474–479 (2009)
32. Schmidt, K.: Stubborn Sets for Standard Properties. In: Proceedings of the 20th International Conference on Application and Theory of Petri Nets, pp. 46–65. Springer-Verlag, London, UK, UK (1999)
33. de Weerdt, J., de Backer, M., Vanthienen, J., Baesens, B.: A Critical Evaluation Study of Model-log Metrics in Process Discovery. In: M. Muehlen, J. Su (eds.) Business Process Management Workshops, *Lecture Notes in Business Information Processing*, vol. 66, pp. 158–169. Springer Berlin Heidelberg (2011)
34. de Weerdt, J., de Backer, M., Vanthienen, J., Baesens, B.: A Robust F-measure for Evaluating Discovered Process Models. In: IEEE Symposium series on computational intelligence, pp. 148–155. IEEE (2011)
35. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process Compliance Measurement based on Behavioural Profiles. In: Proceedings of the 22nd international conference on Advanced information systems engineering, CAiSE'10, pp. 499–514. Springer-Verlag, Berlin, Heidelberg (2010)
36. Weijters, A., van der Aalst, W., de Medeiros, A.A.: Process Mining with the Heuristics Miner-algorithm. Tech. rep., Eindhoven University of Technology, Eindhoven (2006). BETA Working Paper Series, WP 166