Alignment Based Precision Checking

A. Adriansyah[1], J. Munoz-Gama[2], J. Carmona[2], B.F. van Dongen[1], and W.M.P. van der Aalst[1]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{a.adriansyah,b.f.v.dongen,w.m.p.v.d.aalst}@tue.nl
[2] Universitat Politecnica de Catalunya
Barcelona, Spain
{jmunoz,jcarmona}@lsi.upc.edu

Abstract. Most organizations have process models describing how cases need to be handled. In fact, legislation and standardization (cf. the Sarbanes-Oxley Act, the Basel II Accord, and the ISO 9000 family of standards) are forcing organizations to document their processes. These processes are often not enforced by information systems. However, torrents of event data are recorded by today's information systems. These recorded events reflect how processes are really executed. Often reality deviates from the modeled behavior. Therefore, measuring the extent process executions *conform* to a predefined process model is increasingly important. In this paper, we propose an approach to measure the *precision* of a process model with respect to an event log. Unlike earlier approaches, we first *align* model and log thus making our approach more robust, even in case of deviations. The approach has been implemented in the ProM 6 tool and evaluated using both artificial and real life cases.

Keywords: Precision measurement, Log-model alignment, Conformance checking, Process mining

1 Introduction

Process models are the starting point for most Business Process Management (BPM) activities, as they provide insights into possible scenarios [10]. Process models are used for analysis (e.g. simulation), enactment, redesign, and process improvement. Therefore, they should reflect the dominant behavior accurately. The increasing availability of event data enables the application of *conformance checking* [9, 12, 13]. Conformance checking techniques compare *event logs* with process models such that deviations can be diagnosed and quantified.

Conformance can be viewed along multiple orthogonal dimensions: (1) Fitness, (2) Simplicity, (3) Precision, and (4) Generalization [12]. In this paper, we focus on the *precision* dimension. Precision penalizes a process model for allowing behavior that is unlikely given the observed behavior in the event log. Take for example the two models and the event log in Figure 1. All traces in the log can be reproduced by both models, i.e. the traces perfectly *fit* the models.

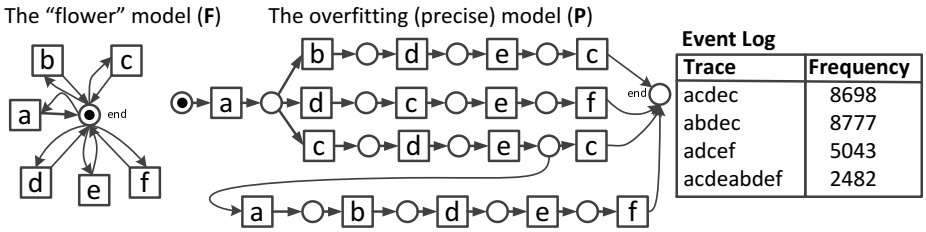

Fig. 1. Example of an extremely precise (overfitting) and imprecise model (underfitting) for a given log.

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. [4, 7, 9]) assume that the event log perfectly fits the model, while many case studies show that this assumption does not hold (e.g. [5, 8, 14]). In this paper, we do not use such assumptions and propose a robust approach to measure the *precision* between an event log and a model. This way we combine our earlier work on *precision* [6, 7] and *alignments* [1, 2].

The paper is organized as follows: Section 2 shows the notations and preliminary concepts that are used throughout this paper. Alignment between event logs and models is explained in Section 3. Alignment-based precision measurements are presented in Section 4. Experimental results are given in Section 5. Section 6 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.

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 σ_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 $sel_1(x)$ and $sel_2(x)$ to refer to the first and the second element of pair x respectively. We generalize this notation to sequences: $sel_i(\sigma) = \langle sel_i(\sigma_1), ..., sel_i(\sigma_{|\sigma|})\rangle$. For all $Q \subseteq W$, $\sigma_{\downarrow Q}$ denotes the projection of $\sigma \in W^*$ on Q, e.g., $\langle a, a, b, c\rangle_{\downarrow\{a,c\}} = \langle a, a, c\rangle$. For simplicity, we omit brackets for sequences whenever their elements are clearly distinguishable, e.g. we write aac instead of $\langle a, a, c\rangle$.

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 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 $L : {A_L}^* \to \mathbb{N}$. For example, the log in Figure 1 is formalized as $L = [acdec^{8698}, abdec^{8777}, adcef^{5043}, acdeabdef^{2482}]$.

Similarly, a *process model* defines a set of sequences of activities that leads to proper termination of the process. Some activities described in a model may not be logged. 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 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 are formalized by the set $\{acdec, abdec, adcef, acdeabdef\}$. Note that the set of modeled activities and the set of logged activities may be disjoint.

3 Cost-Optimal Alignment

An alignment between an event log and a process model relates occurrences of activities in the log to 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 $\perp$ to denote "no move" in either the log or the model. Hence, we introduce the set $A_L^\perp = A_L \cup \{\perp\}$ where any $x \in A_L^\perp$ refers to a "move in log" and the set $A_M^\perp = A_M \cup \{\perp\}$ where any $y \in A_M^\perp$ refers to a "move in model". Formally, a *move* is represented by a pair $(x, y) \in A_L^\perp \times A_M^\perp$ such that:

- (x, y) is a *move in log* if $x \in A_L$ and $y = \perp$,
- (x, y) is a *move in model* if $x = \perp$ 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.

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 $\sigma_M \in M$ be a complete execution of the model. An *alignment* between σ_L and σ_M is a sequence $\gamma \in {A_{LM}}^*$ where the projection of the first element (ignoring $\gg$) yields σ_L (i.e. $sel_1(\gamma)_{\downarrow A_L} = \sigma_L$) and projection of the second element yields σ_M (i.e. $sel_2(\gamma)_{\downarrow A_M} = \sigma_M$).

Take for example a trace $\sigma_L = aacef$ and an activity sequence $adcef$ allowed by model $\mathbf{P}$ in Figure 1. Some possible alignments between the two are:

$$\gamma_1 = \left|\begin{matrix}a\\a\end{matrix}\right|\left|\begin{matrix}a\\\gg\end{matrix}\right|\left|\begin{matrix}\gg\\d\end{matrix}\right|\left|\begin{matrix}c\\c\end{matrix}\right|\left|\begin{matrix}e\\e\end{matrix}\right|\left|\begin{matrix}f\\f\end{matrix}\right| \quad \gamma_2 = \left|\begin{matrix}a\\\gg\end{matrix}\right|\left|\begin{matrix}a\\a\end{matrix}\right|\left|\begin{matrix}\gg\\d\end{matrix}\right|\left|\begin{matrix}c\\c\end{matrix}\right|\left|\begin{matrix}e\\e\end{matrix}\right|\left|\begin{matrix}f\\f\end{matrix}\right| \quad \gamma_3 = \left|\begin{matrix}a\\a\end{matrix}\right|\left|\begin{matrix}\gg\\d\end{matrix}\right|\left|\begin{matrix}a\\\gg\end{matrix}\right|\left|\begin{matrix}c\\c\end{matrix}\right|\left|\begin{matrix}e\\e\end{matrix}\right|\left|\begin{matrix}f\\f\end{matrix}\right| \quad \gamma_4 = \left|\begin{matrix}a\\\gg\end{matrix}\right|\left|\begin{matrix}a\\a\end{matrix}\right|\left|\begin{matrix}c\\\gg\end{matrix}\right|\left|\begin{matrix}\gg\\d\end{matrix}\right|\left|\begin{matrix}\gg\\c\end{matrix}\right|\left|\begin{matrix}e\\e\end{matrix}\right|\left|\begin{matrix}f\\f\end{matrix}\right|$$

The moves are represented vertically, e.g., the first move of γ_2 is $(a, \gg)$, indicating that the log moves a while the model does not make any move. Note that the projections of all moves in model in all alignments are by definition complete activity sequences allowed by the model. This property is not always guaranteed in some other approaches that also relates occurrences of observed activities in the logs to execution steps in process models (e.g. [9]).

To measure the cost of an alignment, we define a *distance function* $\delta : A_{LM} \rightarrow \mathbb{N}$ where for all $(x, y) \in A_{LM}, \delta((x, y)) = 0$ if $x = y$ and $\delta((x, y)) = 1$ otherwise[3]. The distance function can be generalized to alignments $\gamma \in {A_{LM}}^*$ by taking the sum of the costs of all individual moves: $\delta(\gamma) = \sum_{(x,y)\in\gamma} \delta((x, y))$. Using this function, the cost of alignment γ_1 is $\delta(\gamma_1) = \delta((a, a)) + \delta((a, \gg)) + \delta((\gg, d)) + \delta((c, c)) + \delta((e, e)) + \delta((f, f)) = 0 + 1 + 1 + 0 + 0 + 0 = 2$.

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* $\Gamma_{\sigma_L,M} = \{\gamma \in {A_{LM}}^* \mid \exists_{\sigma_M \in M} : \gamma$ is an alignment between σ_L and $\sigma_M\}$ to be all possible alignments between σ_L and complete activity sequences of M. Accordingly, we define the set of *optimal alignments* as the set of all alignments with minimum cost, i.e. $\Gamma^o_{\sigma_L,M} = \{\gamma \in \Gamma_{\sigma_L,M} \mid \forall_{\gamma'\in\Gamma_{\sigma_L,M}} \delta(\gamma) \leq \delta(\gamma')\}$. The union of all optimal alignments is defined as $\Gamma^o_{L,M} = \bigcup_{\sigma_L\in L} \Gamma^o_{\sigma_L,M}$. It is easy to see that there can be more than one optimal alignment between a trace and a model. For example, $\{\gamma_1, \gamma_2, \gamma_3\}$ is the set of optimal alignments between the trace $\sigma_L = aacef$ and model $\mathbf{P}$ in Figure 1.

Given a log and a model, one can measure precision based on all optimal alignments between traces in the log and the model or take just one representative element for each trace. In this paper, we investigate both approaches. We define a function $\lambda_M \in {A_L}^* \rightarrow {A_{LM}}^*$ that maps each trace in the log to an optimal alignment, i.e. for any $\sigma_L \in L$, $\lambda_M(\sigma_L) = \gamma$, where $\gamma \in \Gamma^o_{\sigma_L,M}$. If there are multiple optimal alignments, λ_M chooses one of them according to other external criteria. For instance, with our previous example, suppose that λ_M selects an alignment that has the earliest occurrence of non-synchronous moves, $\lambda_M(\sigma_L) = \gamma_2$.

[3] 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.

We define a function $\overline{\lambda}_M \in {A_L}^* \rightarrow M$ based on λ_M such that for any trace σ_L in log L and a model M, $\overline{\lambda}_M(\sigma_L) = sel_2(\lambda_M(\sigma_L))_{\downarrow A_M}$. Function $\overline{\lambda}_M$ provides an "oracle" that produces one complete activity sequence allowed by models. In [1,2] various approaches to obtain an optimal alignment with respect to different cost function are investigated. As a result, for any given trace and model, *we can always obtain an activity sequence closest to the trace that perfectly fits the model.*

Note that in cases where process model has duplicate tasks (more than one tasks to represent an activity) or unlogged tasks (tasks whose execution are not logged), approaches to construct alignments (e.g. [1,2]) keep the mapping from all model moves to the tasks they corresponds to. Hence, given an alignment of a trace and such models, we know exactly which task is executed for each model move. Due to space constraints, we refer to [1,2] for further details on how such mapping is constructed.

4 Alignment-Aware Precision

Given an event log and a model, the technique described in the previous section provides one optimal alignment (through the λ_M function) or all optimal alignments (through the $\Gamma^o_{\sigma_L,M}$ set) for each trace in the log. This section presents a technique to compute *precision* based on the use of these optimal alignments. The technique is grounded on the methods described in [6,7]. However, there is a fundamental difference: whereas in [6,7] traces in the log are simply replayed in the model, our new approach is based on alignments.

The advantages of the approach presented in this paper are manifold. First of all, traces in the log do not need to be completely fitting. In [6,7] the non-fitting parts are simply ignored. For most real-life logs 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 [6,7], ad-hoc heuristics were used to deal with non-determinism. Finally, the use of alignments instead of log-based model replay improves the robustness of conformance checking (as will be demonstrated later when we present the experimental results). The remainder of this section is devoted to explain how precision can be calculated.

Precision is estimated by confronting model and log behaviors: *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 and analyzed. For instance, there are 5 clear cases of imprecision (b, c, d, e, f) in the initial state of the **F** model in Figure 1, where a, b, c, d, e, f are possible activities according to the model but only a occurs in the initial state according to the log.

First, log behavior must be determined in terms of model perspective, i.e., we consider the optimal alignments of each trace for this purpose. In particular, the projection of the second element of each optimal alignment, i.e., $sel_2(\gamma)_{\downarrow A_M}$. These sequences are used to build the *alignment automaton*, i.e., a prefix automaton that includes information of all log traces. Depending if all the possible

optimal alignments are used to build the automaton (i.e., $\Gamma^{o}_{\sigma_L,M}$) or just one (i.e., λ_M), we will refer to the instantiation of the automaton as $\mathcal{A}$ or $\mathcal{A}^1$ respectively. Clearly, $\mathcal{A}$ provides more information than $\mathcal{A}^1$, and hence the precision value will be closer to the reality. But for large logs it may be difficult to compute all optimal alignments. Apart from providing individual precision metrics for each one of these two automata, the experiments demonstrate that using $\mathcal{A}^1$ in the precision metric is a good approximation to the value provided by using $\mathcal{A}$.

Take for example the model and the log $L = [\sigma_1, \sigma_1, \sigma_2, \sigma_2]$ in Figure 2, where $\sigma_1 = abcde$ and $\sigma_2 = acbde$. The set of optimal alignments for the two possible traces consists of:

$$\gamma_5 = \left|\begin{array}{|c|c|c|c|c|c|}\hline a & b & c & d & e & \bot \\ \hline \bot & b & \bot & d & e & c \\ \hline\end{array}\right| \quad \gamma_6 = \left|\begin{array}{|c|c|c|c|c|}\hline a & b & c & d & e \\ \hline \bot & b & c & \bot & \bot \\ \hline\end{array}\right| \quad \gamma_7 = \left|\begin{array}{|c|c|c|c|c|c|}\hline a & b & c & d & e & \bot \\ \hline \bot & b & \bot & d & e & a \\ \hline\end{array}\right|$$

$$\gamma_8 = \left|\begin{array}{|c|c|c|c|c|c|}\hline a & c & b & d & e & \bot \\ \hline \bot & \bot & b & d & e & a \\ \hline\end{array}\right| \quad \gamma_9 = \left|\begin{array}{|c|c|c|c|c|c|}\hline a & c & b & d & e & \bot \\ \hline \bot & \bot & b & d & e & c \\ \hline\end{array}\right|$$

In the first part of this section we consider the case where only one optimal alignment per trace is used, i.e., we use function $\lambda_M(\sigma_L)$ rather than $\Gamma^{o}_{\sigma_L,M}$ to construct the automaton $\mathcal{A}^1$ (the case where all the optimal alignments are considered is detailed at the end of the section). Considering the distance function and an external criteria, the optimal alignments selected for the traces in the log could be, for instance, $\lambda_M(\sigma_1) = \gamma_5$ and $\lambda_M(\sigma_2) = \gamma_8$. The projection of the second element of each optimal alignment (e.g., $\overline{\lambda}_M(\sigma_1) = bdec$ and $\overline{\lambda}_M(\sigma_2) = bdea$) is used to build the automaton $\mathcal{A}^1$, where the states of that automaton are determined by complete set of all the prefixes of the alignment projections (e.g., $\{\epsilon, b, bd, bde, bdea, bdec\}$ on this example).

Formally, the alignment automaton is defined such that:

- The set of *states* corresponds to all prefixes.
- The set of *labels* corresponds to the activities.
- The *arcs* define the concatenation between prefixes and activities, e.g., states bd and bde are connected by arc labeled E.
- The state corresponding with the empty sequence ϵ is the initial state.
- The function ω determines the weight of each state according to its importance for the precision computation. Graphically it is shown as a number inside the state.

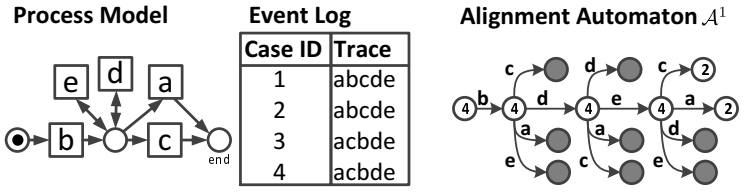

Fig. 2. Example of a model with an unfitting log and its alignment automaton, considering one optimal alignment per trace ($\mathcal{A}^1$).

Function ω is used to determine the importance of the states based on frequencies. In this example, where only one alignment per trace is considered, value of ω for a state is the number of occurrences of the state in the multiset of all visited states when replaying the log, e.g. using the example in Figure 2, $\omega(b) = 4$ because b is a prefix of both $\overline{\lambda}_M(\sigma_1)$ and $\overline{\lambda}_M(\sigma_2)$ and $L(\sigma_1)+L(\sigma_2) = 2+2 = 4$. $\omega(bdea) = 2$ because $bdea$ is only a prefix of $\overline{\lambda}_M(\sigma_2)$ and $L(\sigma_2) = 2$.

Note that the alignment automaton is similar to the *prefix automaton* presented in [6]. However, the alignment automaton is build from proper firing sequences, i.e., the projections of the alignments. Therefore, any sequence of activities corresponding with a prefix of the automaton can be replayed unambiguously on the model. This also ensures that occurrences of activities that are modeled but not logged (i.e. unlogged tasks) and duplicate tasks (i.e. which task an event is mapped to) are identified correctly. This is not the case on the construction of the prefix automaton in [6].

Once the log behavior has been determined in terms of the model's perspective, the confrontation with the actual model behavior is required in order to determine the precision of the system. We compute the set of available actions for each state of the process according to the model ($a_v(\sigma)$), and then compare this set with the set of actions really executed ($e_x(\sigma)$), and therefore reflected accordingly in the log. Given a prefix σ corresponding to a state of $\mathcal{A}^1$, we define the set of *executed actions* $e_x(\sigma)$ as the labels of the outgoing arcs of σ. For example, $e_x(bde) = \{a, c\}$. We also define the set of *available actions* $a_v(\sigma)$ as the set of available actions of the model, e.g., for Petri net models it corresponds with the set of transitions enabled in the marking reached after firing the sequence σ. For example, $a_v(bde) = \{a, c, d, e\}$, i.e., the transitions enabled after firing bde. Note that, by construction $e_x(\sigma) \subseteq a_v(\sigma)$, i.e., given that the replay automaton is built from valid firing sequences in the model (after removing the $\perp$'s), the set of outgoing arcs of a given state is always a subset of all actions enabled according to the model.

The actions available in a state but never observed in the log are used to collect the *imprecision* of the system, i.e., an activity that escapes from the log behavior. These imprecisions are represented in gray in the automaton of Figure 2. For example, the imprecisions of the state bde are $\{a, c, d, e\} \setminus \{a, c\} = \{d, e\}$. The computation and analysis of these imprecisions are the cornerstone of the precision checking 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 pondered by their weight within the process.

The *align-based precision metric* (a_p^1) of a system, where only one alignment per trace is considered (hence, using automaton $\mathcal{A}^1$), is determined by the formula:

$$a_p^1(\mathcal{A}^1) = \frac{\sum_{\sigma \in S} \omega(\sigma) \cdot |e_x(\sigma)|}{\sum_{\sigma \in S} \omega(\sigma) \cdot |a_v(\sigma)|}$$

where S is the set of states of the alignment automaton $\mathcal{A}^1$.

To compute this metric, all the executed activities for each state (weighted with the importance of the state within the process) are collected, and then compared with the set of activities allowed by the model for the same state (also weighted), i.e., imprecisions are quantified. For example, given automaton of Figure 2, precision is computed as:

$$\frac{1 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 + 2 \cdot 4 + 0 \cdot 2 + 0 \cdot 2}{1 \cdot 4 + 4 \cdot 4 + 4 \cdot 4 + 4 \cdot 4 + 0 \cdot 2 + 0 \cdot 2} = 0.38$$

where each $s \cdot w$ summand refers to a state on the automaton, i.e., s groups the activities (executed or available) of that state and ponders them by the weight w of the state.

In order to focus on the important parts of the process and to mitigate the effects produced by rarely occuring traces or incomplete traces, the precision defined above could be restricted to consider only such states with a weight greater than a given pruning threshold (called τ). In the remainder, we assume no pruning (i.e., $\tau = 0$), unless it is stated otherwise. The effects of the pruning can be seen in [7]. Additionally, it is also possible to consider the precision with a severity factor associated to the activity that escapes from the log behavior.

The case considered so far is the one where only one optimal alignment per trace is used to build the automaton ($\mathcal{A}^1$). However, the same idea can be used to propose a metric for the general case (denoted a_p) where all the best alignments of a trace are used to build the alignment automaton ($\mathcal{A}$). For instance, following the running example and considering a similar scenario, there are three optimal alignments for the trace σ_1 (γ_5,γ_6 and γ_7), and two for the trace σ_2 (γ_8 and γ_9). The process of building the alignment automaton $\mathcal{A}$ (see Figure 3) and computing the metric a_p is the same as computing a_p^1, except the definition of the function ω.

Unlike the case with one alignment, in this case the importance of each state does not depend exclusively on the frequency, but must also be equally balanced among all its alignments. Consider for instance the state corresponding with the prefix b. This prefix appears in all the optimal alignments of all the traces in the log (σ_1 and σ_2). So, the weight of this state is 4 (1 for each trace and both traces occur twice in the log), as shown in Figure 3. However, this is not the case for the state bc. This state only appears in the set of optimal alignments of only one trace (σ_1) that occurs twice in the log. The first naive attempt would be then assign to this state a weight of 2 (1 for each occurrence). However, note that, there are cases where the number of optimal alignments of one trace may not

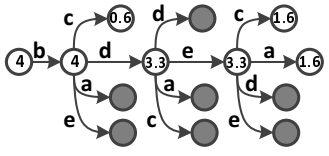

Fig. 3. Alignment automaton $\mathcal{A}$ of model and log in Figure 2, considering all optimal alignments per trace.

coincide with another trace, e.g. σ_1 has 3 alignments and σ_2 has 2. In order to eliminate the bias produced by traces with many optimal alignments, this value needs to be normalized, i.e., we consider also the number of optimal alignments of the trace and also in how many of them the prefix appears. In that sense, for the state *bc* the weight is 1/3 (it appears only in one of the 3 optimal alignments of the trace σ_1) times 2 (σ_1 occurs twice in the log), i.e., $1/3 \cdot 2 \approx 0.6$ as is shown in Figure 3. Formally, the function ω can be defined then as follows:

$$\omega(s) = \sum_{\sigma \in \Lambda(s)} \frac{\text{num\_occurrences}(\sigma, s)}{\text{num\_total\_align}(\sigma)}$$

where $\Lambda(s) = \{\sigma | \sigma \in L \wedge \exists \gamma \in \Gamma^o_{L,M} : \gamma = (\sigma, (s \cdot \beta))\}$, i.e., the set of traces in the log with both one optimal alignment associated and having s as a prefix. The functions num_occurrences(σ, s) and num_total_align(σ) return the number of best optimal alignments where s is a prefix, and the number total of alignments, respectively, associated with σ.

Note that there are theoretical differences concerning the imprecisions of $\mathcal{A}^1$ and $\mathcal{A}$. For instance, in the running example, *bc* is an imprecision in $\mathcal{A}^1$ but not in $\mathcal{A}$. This difference is reflected in the values of a_p and a_p^1 (0.47 and 0.38 respectively). Since all the optimal alignments are taken into account, a more complete characterization of log behavior is considered in $\mathcal{A}$. However, the experiments show that, the use of a_p^1 is a good approximation of a_p, in such cases where complexity is an issue (see Sec. 5).

The metrics presented in this section coincide with the intuition for precision presented in the introduction of this paper. This is illustrated by the the results of the a_p^1 and a_p metrics for the example log and model in Figure 1. As expected, the precision for models **P** is high (1.00 for both a_p and a_p^1) while the precision for model **F** is low (0.20 for both a_p and a_p^1).

5 Experiments

We have implemented the proposed precision calculation as a ProM 6 plugin, 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 and logs.

5.1 Artificial cases

The first set of experiments was performed to evaluate the values provided by the proposed metrics. We measured precision between various logs and models whose expected values are known and compare them against etc_P [7] precision as benchmark for existing precision metrics. We created new models whose expected precision values are between the two extremes by combining the models and log in Figure 1 (**P** and **F**) in different orders. Two models are combined by merging the end place of one with the initially marked place of another. Merged models

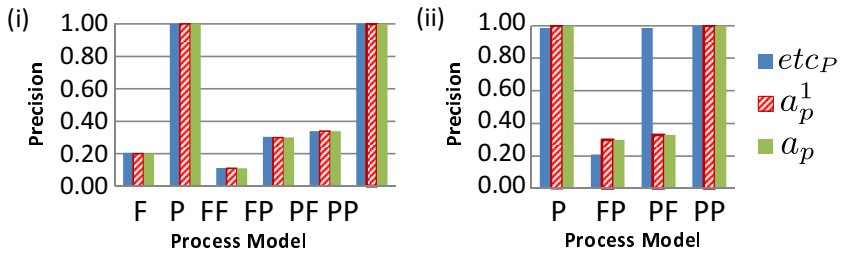

Fig. 4. Precision values for (i) perfectly fitting logs, and (ii) unfitting logs where 4 events are removed from each trace in the logs.

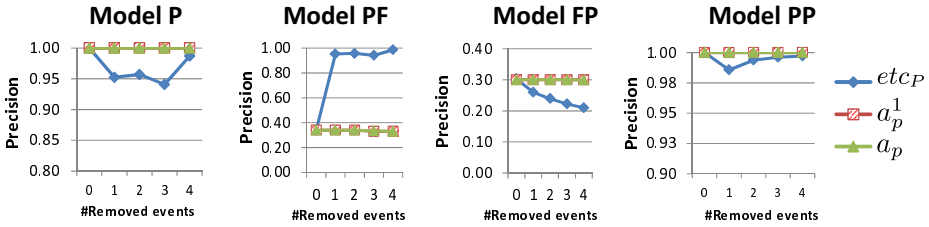

Fig. 5. Robustness of Precision to Unfitting Logs

are named according to the 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 merged models and logs are renamed such that splitting the logs and models into two parts yields back the original logs and models. Precision values were measured 10 times for event logs consisting of 5,000 traces, generated by simulating the precise model (i.e. **PP**). The results are shown in Fig. 4(i).

As shown in Figure 4(i), both a_p and a_p^1 give the same values as etc_P. In cases where logs are perfectly fit to models and activity execution can be mapped unambiguously to tasks in the models, values of both a_p, a_p^1, and etc_P are the same as there is only one optimal alignment per trace.

The second set of experiments were conducted to evaluate the robustness of the proposed metric against non-fitting logs. We took the models and log from the previous experiment and create unfitting logs by removing n number of events randomly per trace from the fitting log. To ensure that the created logs are unfitting, only events that belong to the precise part (i.e. mapped to **P** part) are removed. Figure 4(ii) and Figure 5 show the results.

As it is shown in Figure 4(ii) and Figure 5, our metrics are more robust to noise than etc_P. Even in cases where almost half of the events in all traces are removed, both metrics provide the same value as the ones given for perfectly fitting traces. In contrast, the etc_P value may change significantly because for all non-fitting traces, it ignores the rest of the traces after the first non-fitting event occur. In the experiment with model **PF**, etc_P value changes significantly even when only one event is removed per trace as the remaining events that belong to the imprecise model are ignored. In the experiment with model **FP**, etc_P values

gets closer to the precision value of the **F** model as the number of removed events increases, because non fitting events always occur in the precise part of the model (i.e. **P**). Figure 5 also shows that a_p^1 values are good approximation to a_p values because the aggregation of all selected optimal alignments for each trace in the logs cover all traces allowed by the **P** part of all four models.

5.2 Real-life Logs and Models

To evaluate the applicability of the approach to handle real life logs, we use 5 pairs of process models and logs from the CoSeLoG project [3, 11]. The models and logs were obtained from participating municipalities in the Netherlands. We consider processes related to five types of building permission applications. All processes have unlogged tasks, and two of the models allow loops. We have compared the proposed precision measurements with related metrics such as the etc_P metric [7] and the advanced behavioral appropriateness a'_b [9]. The results are shown in Table 1.

An important conclusion that can be drawn from Table 1 is that the computation time of a_p takes much longer than a_p^1. From all evaluated precision metrics, a_p^1 managed to provide precision values for all logs and models under 12 seconds, while a_p calculation takes much longer. However, this is not a problem because a_p^1 provides a close estimation to a_p. Moreover, Table 1 shows that in reality, the observed traces are not perfectly fitting the corresponding models (see #not synchronous moves/case) and hence justifies the need of having precision measurements that are robust to non-fitting logs.

6 Conclusion

Lion's share of conformance checking literature has been focusing on fitness, i.e., quantifying the proportion of the event log that is possible according to a given model. However, it is also important to analyze precision. A process model that allows for behavior unrelated to the example behavior seen in the log is too general. Existing approaches for quantifying precision are time consuming and have problems dealing with non-fitting traces. This results in unreliable precision measurements as shown in this paper. Therefore, we developed an approach that first aligns event log and model. The pre-alignment of log and model makes it

Table 1. Precision values from experiments on real-life logs and models

Log	#Cases	#Events	Process model		#Not sync.	a_p^1	time	a_p	time	etc_P	a'_b
			#place	#trans.	moves/case		(sec)		(sec)		
MLog1	3181	20491	15	12	5.33	0.92	11.3	1.00	321.1	0.97	0.82
MLog2	1861	15708	16	19	1.45	0.93	3.7	0.93	53.5	0.97	0.92
MLog4	4852	29737	16	27	2.09	0.96	4.1	0.99	15.7	0.86	0.75
Bouw-1	139	3364	33	34	9.46	0.82	0.7	n/a	n/a	0.85	0.95
Bouw-4	109	2331	31	31	6.18	0.44	2.4	n/a	n/a	0.34	n/a

* n/a : not found in 6 hours.

possible to measure precision more accurately. In this work we presented two metrics (a_p^1 and a_p) to measure the precision, considering just one or all possible optimal alignments respectively. The results show experimentally the usefulness and the robustness of the approach proposed.

References

1. A. Adriansyah, N. Sidorova, and B.F. van Dongen. Cost-Based Fitness in Conformance Checking. *International Conference on Application of Concurrency to System Design*, pages 57–66, 2011.
2. A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst. Conformance Checking Using Cost-Based Fitness Analysis. *IEEE International Enterprise Distributed Object Computing Conference*, pages 55–64, 2011.
3. J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. Towards cross-organizational process mining in collections of process models and their executions. In *Business Process Management Workshops*, volume 100 of *Lecture Notes in Business Information Processing*. Springer Berlin Heidelberg, 2012.
4. T. Calders, C.W. Günther, M. Pechenizkiy, and A. Rozinat. Using Minimum Description Length for Process Mining. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 1451–1455, New York, USA, 2009. ACM.
5. K. Gerke, J. Cardoso, and A. Claus. 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, pages 76–93, Berlin, Heidelberg, 2009. Springer-Verlag.
6. J. Munoz-Gama and J. Carmona. A Fresh Look at Precision in Process Conformances. In *Proceedings of the 8th International Conference on Business Process Management*, BPM'10, pages 211–226, Berlin, Heidelberg, 2010. Springer-Verlag.
7. J. Munoz-Gama and J. Carmona. Enhancing precision in process conformance: Stability, confidence and severity. In *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011*, pages 184–191. IEEE, April 2011.
8. A. Rozinat, I.S.M. de Jong, C.W. Günther, and W.M.P. van der Aalst. Process Mining Applied to the Test Process of Wafer Steppers in ASML. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 39:474–479, 2009.
9. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, March 2008.
10. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation*. Springer-Verlag, 2010.
11. W.M.P. van der Aalst. Business Process Configuration in The Cloud: How to Support and Analyze Multi-Tenant Processes? In G. Zavattaro, U. Schreier, and C. Pautasso, editors, *Proceedings of the 9th IEEE European Conference on Web Services (ECOWS 2011)*, pages 3–10. IEEE Computer Society Press, 2011.
12. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.
13. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
14. M. Weidlich, A. Polyvyanyy, N. Desai, and J. Mendling. Process Compliance Measurement based on Behavioural Profiles. In *Proceedings of the 22nd international conference on Advanced information systems engineering*, CAiSE'10, pages 499–514, Berlin, Heidelberg, 2010. Springer-Verlag.