Enhancing Precision in Process Conformance: Stability, Confidence and Severity

Jorge Munoz-Gama
Universitat Politecnica de Catalunya
Barcelona, Spain
jmunoz@lsi.upc.edu

Josep Carmona
Universitat Politecnica de Catalunya
Barcelona, Spain
jcarmona@lsi.upc.edu

***Abstract*—Process Conformance is becoming a crucial area due to the changing nature of processes within an Information System. By confronting specifications against system executions (the main problem tackled in process conformance), both system bugs and obsolete/incorrect specifications can be revealed. This paper presents novel techniques to enrich the process conformance analysis for the precision dimension. The new features of the metric proposed in this paper provides a complete view of the precision between a log and a model. The techniques have been implemented as a plug-in in an open-source Process Mining platform and experimental results witnessing both the theory and the goals of this work are presented.**

I. Introduction

The constant growth of *Information Systems* (IS) has been essential for the rising of new disciplines aiming at providing a systematic way of dealing with the vast available data that is generated. Among the various types of data related to an IS, *event logs* and *formal models* are two important sources of information. An event log is a set of traces produced by monitoring a particular set of activities in a IS, thus reflecting the reality. In contrast, formal models are used to precisely specify the operative behavior of the IS, e.g., like a contract of what must be done and in which order.

By focusing on only in the analysis of event logs, traditional areas like *Data Mining* can answer many questions regarding various aspects of the IS. However, it was only the advent of the *Process Mining* area that enabled to address the problem of confronting the reality (in the form of logs) with the specification (in the form of formal models). Process mining comprises three classes of techniques: *discovery*, *conformance* and *extension* [1]. It can be oriented to the control flow, the data or the social aspects of a process. This work proposes process conformance techniques to aid the analysis of the control flow aspect.

Process Conformance aims at evaluating the adequacy of a model in describing a log. Analyzing conformance is a complex task which involves the interplay of different and orthogonal dimensions [2].

- *Fitness*: indicates how much of the observed behavior is captured by (i.e. “fits”) the process model.
- *Precision*: refers to overly general models, preferring models with minimal behavior to represent as closely as possible the log.
- *Generalization*: addresses the degree of abstraction beyond observed behavior.
- *Structure*: refers to models minimal in structure which clearly reflect the described behavior.

The work presented in this paper proposes novel techniques to be applied within the precision dimension.

Different algorithms for conformance checking have been presented in the literature, getting more robust-aware all the time (e.g. [3]). A complete survey can be found in [4], [5]. In particular, some examples of approaches focused on precision are: [6] (measuring the percentage of potential traces in the model that are in the log), [7] (comparing two models and a log to see how much of the first model's behavior is covered by the second) used in [8], [9] (comparing the behavioral similarity of two models without a log), and [10] (using *minimal description length* to evaluate the quality of the model).

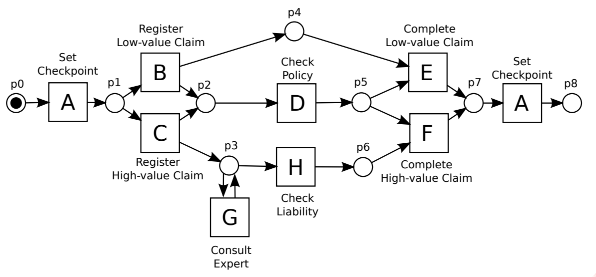
The work presented in this paper extends and complements the technique introduced in [11], where the precision between a log and a Petri net was considered. Using a log-guided traversal of the model, the refereed technique avoids the full state-space exploration of the Petri net, and provides a precision metric which evaluates the effort needed to derive a better (more precise) model. In this work, an extension of the technique is presented, which additionally considers potential variability in the log. Moreover, a novel technique to estimate the *confidence interval* of the metric is introduced, deriving an enhanced metric for precision which not only provides a value but also estimates its robustness. Finally, techniques to measure the *severity* of the imprecisions detected are developed, thus providing a very important step into two directions: improving the feedback provided by precision checking tools and bridging the gap towards automated techniques for precision oriented model refinement.

The techniques have been implemented as a ProM [12] plug-in. The experimental results reported provide insights on the real usefulness of the new precision analysis proposed in this paper.

This work has been partially supported by Intel Corporation and by the project FORMALISM (TIN2007-66523).

# Instances	Log Traces
1435	A B D E A
946	A C D G H F A
764	A C G D H F A
54	A C G H D F A
1	A C D G G H F A

(a) Log L_1



(b) Model M_1

Fig. 1. Running example of a plausible liability insurance claim process, in a banking scenario. The example is composed by two elements: (a) a log L_1 reflecting executions of the process and, (b) a possible model M_1 for the process. The modeling language used is Petri nets. For the reader not familiar with Petri nets: a transition (box) is enabled if every input place (circle) holds a token (black dot). If enabled, the transition can fire, removing tokens from its input places and adding tokens to its output places.

II. Preliminaries

In this section we introduce the two elements needed in any process conformance analysis: *logs* and *models*.

A. Logs

Event logs, or simply *logs*, contain executions of a system [1]. These executions represent the ordering between different tasks, but may also contain additional information, such as the task originator or its timestamp. For the purposes of this paper, all this information is abstracted. An example of a log is shown in Fig. 1a. Formally:

Definition 1 (Trace, Log): Let T be the set of tasks, and let $\mathcal{P}(S)$ denote the powerset over S, i.e., the set of possible subsets of elements of S. A *trace* σ is defined as $\sigma \in T^*$. And a *log* is a set of traces, i.e., $\mathsf{EL} \in \mathcal{P}(T^*)$.

Definition 2 (Prefixes, Occurrence of a Prefix): Given a log EL, define $pre(\mathsf{EL})$ as the set of prefixes of EL, i.e., $pre(\mathsf{EL}) = \{p \mid \sigma = px \in \mathsf{EL} \wedge p, x \in T^*\}$. Note that the empty trace ϵ is a prefix of any log. Given a prefix $p \in pre(\mathsf{EL})$, $p_{\#}^{\mathsf{EL}}$ denotes the occurrences of that prefix in the log, i.e., $p_{\#}^{\mathsf{EL}} = |\{\sigma \text{ s.t. } \sigma = px \in \mathsf{EL}\}|$. The subscript EL can be omitted whenever it is clear by the context.

B. Formal Models

The traces of a log EL can be represented with a formal model M. Given a model M and a trace σ, $\mathrm{avail}(\sigma, M)$ is the set of tasks available according to M after executing σ. Models for workflow representation like *Petri nets* [13] or its extensions (e.g., *YAWL* [14]) are considered in this paper. Due to the lack of space, the reader can follow the provided references to get insight into the theory underlying these models. An example of a Petri net is shown in Fig. 1b. Finally we present transition systems, the formal models used in the development of the algorithms proposed in this paper.

Definition 3 (Transition system [15]): A *transition system* (or TS) is a tuple (S, T, A, s_{in}), where S is a set of *states*, T is an alphabet of *actions*, $A \subseteq S \times T \times S$ is a set of *(labeled) transitions*, and $s_{in} \in S$ is the *initial state*.

III. Problem Statement and Approach

This work extends the precision approach presented in [11]. The refereed technique, consists on three steps for the precision analysis, that are explained and enhanced in Sect. III-A, III-B and III-C.

A. Defining Log Behavior

The first step to determine the precision between a model and a log is to define the behavior reflected in the log. This representation must contain exactly the same language as the log, but including also state information in order to be able to compare the log behavior with the model behavior. For this purpose, the *prefix automaton* of a log is defined.[1]

Definition 4 (Prefix Automaton): Given a log EL, the *prefix automaton* of EL is the transition system $\mathsf{TS} = (S, T, A, s_{in})$ such that $S = pre(\mathsf{EL})$, T corresponds with the tasks of the process, $A = \{(s, e, s') | s' = se\}$ where $e \in T$, and $s_{in} = \epsilon$.

Figure 2 shows the prefix automaton construction for the log in Fig. 1a. The number below each state s represents the occurrences of the prefix s in the log, i.e., $s_{\#}$. Note that all the states have an occurrence number greater than zero given that all them are log prefixes.

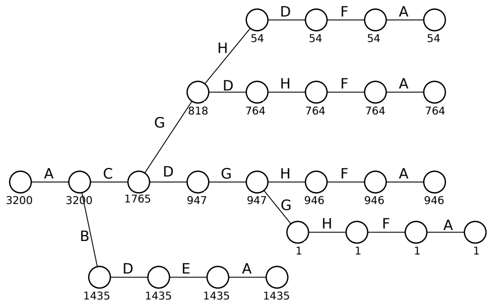


Fig. 2. Prefix Automaton

[1]The prefix automaton matches up with the transition system obtained using the approach presented in [16] with the parameters past, sequence and no horizon.

B. Log-based exploration of Model Behavior

The second step in the precision analysis is to determine the behavior of the model in order to compare it with the log behavior. However, the exploration of the model underlying behavior could be problematic in case of intractable (or even infinite) underlying behavior. Therefore, the technique presented in this work avoids the complete exploration of the model behavior, and instead this exploration is restricted to the observed log behavior. More concretely, the prefix automaton of the log is used as guideline for detecting the situations where the model allows more behavior than the log. The result will be an *extended* prefix automaton, containing both log and model behavior information.

Definition 5 (Extended Prefix Automaton): Given the prefix automaton $\mathsf{TS} = (S, T, A, s_{in})$ of EL and the model M, define the set of *extended states* and the set of *extended transitions* as:

$$\begin{aligned} S^+ &= \{s | s = s't \wedge s' \in S \wedge t \in \text{avail}(s', M) \wedge s \notin S\} \\ A^+ &= \{(s', t, s) | s = s't \wedge s \in S^+\} \end{aligned}$$

The *extended prefix automaton* is $\widehat{\mathsf{TS}} = (\widehat{S}, T, \widehat{A}, s_{in})$ with $\widehat{S} = S \cup S^+$, and $\widehat{A} = A \cup A^+$.

Notice that, for each state $s \in S^+$ the occurrence value $s_{\#}^{\mathsf{EL}} = 0$ because it does not appear as a prefix in the log, i.e., $s \in S^+$ denotes a state considered in the model but not reflected in the log.

In order to extend the automaton we must be able to, given any prefix σ in the log, compute the available tasks in the model after executing σ. These tasks must be a superset of the tasks observed in the log after σ. However, there may be traces in the log with prefixes that do not satisfy this condition, i.e, behavior in the log not included in the models behavior. These traces, called *non fitting traces*, are the scope of the *fitness dimension* of the conformance. Thus, the precision measure presented in this work will only take into account the part of the non fitting traces included in the model behavior, i.e. the one that precision dimension concerns about. Furthermore, in order to obtain a complete vision of the conformance, the precision measure should be included together with measures of the other dimensions like fitness (e.g., [17]).

A consideration must be made concerning about the *determinism* of replaying a trace in the model. The approach presented is able to deal with *duplicate tasks* (i.e., several tasks in the model associated with the same log event) and *invisible tasks* (i.e., tasks in the model associated with no event in the log), through the *invisible coverability graph* [11]. However, sometimes, the inclusion of these kind of tasks may produce indeterminism, i.e., given a trace, the possible set of tasks available after replaying the trace may not be unique [17], [11]. In such cases, the use of heuristics and best effort techniques becomes necessary. For sake of clarity, in this work a deterministic scenario is presented.

Following with the running example shown in Fig. 1, let M_1 in Fig. 1b be a possible model for the process. In this case, the modeling language used is Petri nets. Notice that M_1 is deterministic and includes the language of the log. If M_1 is used to extend the prefix automaton of the log (Fig. 2), it results in the extended prefix automaton shown in Fig. 3. Gray states in the figure correspond to states in S^+ (cf. Def 5).

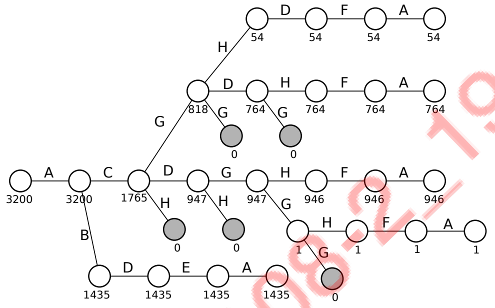


Fig. 3. Extended Prefix Automaton

C. Comparing Model and Log Behavior

Finally, once the prefix automaton contains information about both behaviors (log and model), it is possible to compare them, detecting the imprecisions between both behaviors. These imprecisions denote the situations where the model deviates from the log. A first straightforward way to define these deviations, is defining the *escaping states* concept:

Definition 6 (Simple Escaping States ([11])): Given an extended prefix automaton $\widehat{\mathsf{TS}} = (\widehat{S}, T, \widehat{A}, s_{in})$ and a state $s \in \widehat{S}$, the set of *simple escaping states* at s is:

$$E_S(s) = \{s' | (s, e, s') \in \widehat{A} \wedge s'_{\#} = 0\}$$

Note that the simple escaping states are always leaves of the prefix automaton, and they coincide with the extended states (S^+) introduced in Def. 5. The approach presented in [11] uses this definition of escaping edges, and is meant to measure the precision of a system. However, there are situations where this definition could be too restrictive, making the measurement of precision quite sensitive with respect to the size of the log. In order to reach a more stable measurement of precision, a refined version of the escaping states is introduced in this paper. This new definition takes into account the number of occurrences in each prefix in order to determine if there exists an escaping state or not.

Definition 7 (Escaping States and Outer States): Let $\widehat{\mathsf{TS}} = (\widehat{S}, T, \widehat{A}, s_{in})$ be the given extended prefix automaton for the log EL and the model M, and consider a threshold parameter $\gamma \in [0, 1]$. Given a state $s \in \widehat{S}$, the set of *escaping states* at s is defined as:

$$E_S^{\gamma}(s) = \{s' | (s, e, s') \in \widehat{A} \wedge (\gamma \cdot s_{\#}) \geq s'_{\#}\}$$

In other words, the occurrence value of the state s' (i.e., $s'_{\#}$) must exceed the threshold defined for this point (i.e., $\gamma \cdot s_{\#}$) in order to not be considered as an escaping state. The set of escaping states of a system is defined as:

$$E_S^{\gamma} = \{s \in E_S^{\gamma}(s_a) | \nexists s_a, s_b, s_c : s = s_b x \wedge s_b \in E_S^{\gamma}(s_c)\}$$

The set E_S^γ defines the *border* between the log and the model behavior. The states that fall out of this border are called *outer states* and are defined as:

$$O_S^\gamma = \{s \notin E_S^\gamma | \exists s' : s = s'x \wedge s' \in E_S^\gamma \wedge x \in T^*\}$$

Note that, some *escaping states of a given state* may not belong to the *escaping states of the whole system*, i.e., they belong to the *outer states set*. Following with the running example, Fig. 4 shows the escaping and the outer states for the extended prefix automaton of the log and model in Fig. 1, considering $\gamma = 0.03$. The states filled in dark gray are the escaping states, while the ones filled in light gray correspond to the outer states. For instance, the state ACDGG is considered escaping edge because its occurrence number (i.e., 1) is less than the threshold defined for this point (i.e., $0.03 \cdot 947 = 28.41$).

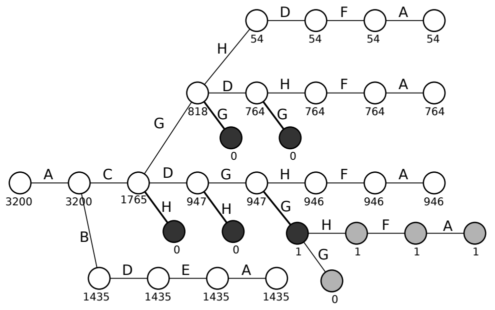


Fig. 4. Escaping States and Outer States of the extended prefix automaton of the example in Fig. 1 with $\gamma = 0.03$. The dark gray states (escaping states), define the border between log and model behavior. The light gray states (outer states), represent the states that fall out of the border.

IV. Evaluating the Precision

As it was mentioned in previous sections and it is proposed in [11], in order to estimate the precision of a system one can make use of the imprecisions detected when comparing the behavior of the model reflected in the log. The metric proposed in this section counts these imprecisions, weights them according to how often they occur, and compares them with the behavior allowed. This way of estimating precision is strongly linked with the idea of estimating the effort needed to achieve a model 100% precise with respect to the log.

Metric 1 (ETC Precision): Given a log EL, a model M, the extended prefix automaton $\widehat{\mathsf{TS}} = (\widehat{S}, T, \widehat{A}, s_{in})$, and the threshold parameter $\gamma \in [0, 1]$, define I as the set of states that are *in* the log behavior which are not escaping or outer states, i.e., $I = \{s \in [\widehat{S} \setminus (E_S^\gamma \cup O_S^\gamma)]\}$. The metric is defined as follows:

$$\text{etc}_P(\gamma) = 1 - \frac{\sum_{s \in I} (|E_S^\gamma(s)| \cdot s_\#)}{\sum_{s \in I} (|\text{avail}(s, M)| \cdot s_\#)} \qquad (1)$$

By dividing the set of escaping states by the set of allowed states, the metric evaluates the amount of *overapproximation* in each state in I. Note that $|E_S^\gamma(s)| \leq |\text{avail}(s, M)|$, and therefore $0 \leq \text{etc}_P \leq 1$. For instance, the metric value for the running example in Fig. 1 is 0.83.

V. Confidence over the Precision Metric

Together with a metric, it is convenient to provide also a *confidence interval*, i.e., some maximum and minimum values estimating the up and down variabilities over the metric computed, respectively. More specifically, the interval presented in this section will reflect the possible variations of the metric when considering k new traces. A narrow interval indicates that considering a set of k incoming traces, the metric should not vary significantly. On the other hand, a wide interval reflects the opposite: a low confidence in the metric provided, whose value could change drastically in the future. The number k used to compute the interval may depend on the future to consider: a low k is used to compute the confidence in a near future, whereas with a large k, a large set of traces is considered, and thus a longer term future is contemplated. Notice that, taking a percentage of the log size as a k, will not allow the comparison between logs of different size.

The confidence interval presented in this section is approximated: it is not measuring real bounds of the metric, but instead aims at estimating them with simple heuristics which can be computed in a systematic manner.

A. Upper Confidence Value

In order to compute the upper value of the confidence interval, the best possible scenario is considered: each one of the k traces reaches some escaping state, *covering* this state, i.e., the state will not be an escaping state any more (graphically, it is a dark gray state that will be colored white in Fig. 4). The number of traces needed to change the state from escaping to non escaping will depend on the γ considered (see Def. 7). Hence, the technique presented below estimates the gain (i.e., the precision increase) of covering each escaping state, and maximizes the total gain considering k as the maximum number of traces used to cover escaping states.

Definition 8 (Cost and Gain of Covering an imprecision): Let $s' \in \widehat{S}$ be an escaping state such that $(s, e, s') \in \widehat{A}$, and let γ be the parameter used to define the escaping states. The *cost* of covering s', denoted as $C(s') = l$ with $l \in \mathbb{N}$, is the minimum l that satisfies $(s_\# + l) \cdot \gamma < (s'_\# + l)$. The *gain* of s' is defined as $G(s') = s_\#$, i.e., the gain of reducing in one the number of escaping states of the parent state s.

By inspecting the fraction part of formula (1), one can see why the gain of covering the escaping state s' is $s_\#$: if in state s one escaping state is removed, then the new escaping estates in s are $|E_S^\gamma(s)| - 1$. Since this number is multiplied by $s_\#$ in the numerator part of the fraction, the numerator will be reduced exactly in $|E_S^\gamma(s)| \cdot s_\# - (|E_S^\gamma(s)| - 1) \cdot s_\# = s_\#$.

Once the gain and the cost of covering an escaping state have been defined, the maximum gain obtained with k traces must be computed. This problem is analogous to the well known *Knapsack* problem [18], which can be solved using *binary integer programming* (*BIP*) [19]. The following BIP model encodes the maximum gain over k traces:

1) *Variables: The variable*

$$X_i \in \{0, 1\} \qquad (2)$$

denotes if the escaping state i is covered or not.

2) *Constraints:* the total cost cannot exceed the number of new traces seen.

$$\sum_{i \in E_S^\gamma} C(i) \cdot X_i \leq k \tag{3}$$

3) *Cost function:* maximize the gain.

$$\max \sum_{i \in E_S^\gamma} G(i) \cdot X_i \tag{4}$$

The number of variables of the BIP model is $|E_S^\gamma|$. This number can be significantly reduced if the following fact is considered: if $\mathcal{C}$ is the set of possible costs for a given model, for each possible $l \in \mathcal{C}$, at most $\lfloor k/l \rfloor$ can be used in order to satisfy (3). Hence the real upper bound to the number of variables needed is $\sum_{l \in \mathcal{C}} \lfloor k/l \rfloor$.

Definition 9 (Upper Confidence Value): Let N and D be the numerator and denominator of the metric etc_P as defined in (1), i.e., $\text{etc}_P(\gamma) = 1 - (N \backslash D)$. Let $G_{\max}$ be the result obtained using the optimization problem modeled above. The *upper confidence value* is defined as follows:

$$U(k) = 1 - \frac{N - G_{\max}}{D} \tag{5}$$

This problem has been modeled and implemented within the ProM framework (cf. Sect. VII). Following with the running example of Fig 1, and considering $\gamma = 0.03$ and $k = 24$, the only escaping state with cost lower enough to be covered with this k is the state ACGDG. The gain of covering this state is 764. This value is subtracted from N, providing an upper interval value of 0.85 for this scenario.

B. Lower Confidence Value

The idea for computing the lower confidence value is similar to the upper value. However, in this case the k traces do not cover escaping states, but instead represent new possible behavior not observed yet, that may produce the rising of new escaping states. In order to estimate the lower confidence value, the average length of the traces in the log (let m) is assumed to be the length of the new traces. These traces denote a behavior not observed, and therefore not represented in the extended prefix automaton. In the worst case scenario, the inclusion of these traces in the automaton would result in $m \cdot k$ new states. Each of these new states may contain also new escaping states. Given that the worst case scenario is considered, the number of escaping states in each new state is $|T - 1|$, i.e., all the tasks are available but only one is followed by the trace. Given these considerations, the lower confidence value is defined as follows:

Definition 10 (Lower Confidence Value): Let N and D be the numerator and denominator of the metric etc_P as defined in (1), i.e., $\text{etc}_P(\gamma) = 1 - (N \backslash D)$. The *lower confidence value* is:

$$L(k) = 1 - \frac{N + (m \cdot k \cdot |T - 1|)}{D + (m \cdot k \cdot |T|)} \tag{6}$$

For instance, following with running example of Fig 1, being $m = 6$ the average length of the traces, being $T = 8$ the number of tasks of the process, and considering $\gamma = 0.03$ and $k = 24$, the lower bound for this scenario would be:

$$L(k) = 1 - \frac{N + (6 \cdot 24 \cdot 7)}{D + (6 \cdot 24 \cdot 8)} = 0.80$$

VI. Severity of an Imprecision

The computation of *escaping states* is an accurate mechanism to determine where exactly are the imprecisions of a system. Defining the *border* between log and model behaviors one can indicate where the efforts must be done in order to achieve a precise representation of the reality. Looking at the escaping states (Fig. 3) of the running example, it can be seen that the imprecisions are produced by the loop on G: a behavior present in the model, but not reflected in the log.

However, not all the imprecisions have the same importance: some refer to exceptional and infrequent parts of the process or are produced by the incompleteness of the log considered; other imprecisions are clear and affect important and frequent parts of the process. By assigning to each imprecision a *severity* degree, is is possible to compare and sort the imprecisions, prioritizing those that must be fixed first.

The severity of an imprecision is a complex multifactored concept with a strong subjective aspect that changes according to the importance that a person gives to each factor. Therefore, the severity of an imprecision can be defined as follows:

Definition 11 (Severity of an Imprecision): Let E_S^γ be the set of escaping states denoting the imprecisions of a system. Given $s \in E_S^\gamma$, the *severity* of the imprecision at s is

$$sev(s) = f(F_{ft}, A_{ft}, S_{ft}) \tag{7}$$

where F_{ft}, A_{ft}, S_{ft} correspond to the *frequency*, *alternation* and *stability* factors of the imprecision at s, and f is a user-defined function that weights these three factors. In the rest of the section techniques for the estimation of each factor are presented.

In the following definitions, this context is assumed: a model M, a log EL, the extended prefix automaton of EL on M $\widehat{\mathsf{TS}} = (\widehat{S}, T, \widehat{A}, s_{in})$, and $(s', e, s) \in \widehat{A}$ where $s \in E_S^\gamma$. In this case, the imprecision ACH (shown in Fig. 5) of the running example is used in order to illustrate the different factors of the severity concept.

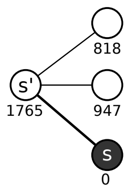


Fig. 5. Imprecision ACH of the running example of Fig. 1.

Factor 1 (Frequency): Let $\max_\# = \max(x_\# \mid x \in \widehat{S})$ be the maximum occurrence value of all the states in $\widehat{S}$. The *frequency factor* of the imprecision at s is defined as follows:

$$F_{ft}(s) = \frac{s'_\#}{\max_\#} \tag{8}$$

According to the formula above, imprecisions concerning more frequent parts of the process have more importance (i.e., higher severity) than others which are less frequent. For instance, imprecision ACH of Fig. 5 has a mid-value frequency factor of $\frac{1765}{3200} = 0.55$, where 3200 is the maximum occurrence value in the whole prefix automaton of Fig. 4.

The second factor addresses the possibility of choosing an incorrect option at a given state. In this case, choosing an incorrect option refers to choosing an imprecision, i.e., a path allowed by the model but never seen in the log. Given a situation where it is really likely to make a mistake, it must have more priority (i.e., higher severity), than other situations where choosing badly is not so probable.

Factor 2 (Alternation): Let $P_E(x)$ be the probability of choosing an escaping state being in the state x. The *alternation factor* of the imprecision at s is defined as $A_{ft}(s) = P_E(s')$, being s' the predecessor of s. The distribution of $P_E(x)$ may be different depending on the assumptions taken. If no assumption is considered, a uniform distribution, where each possible path has the same probability, must be applied. Therefore, the alternation factor can be estimated as:

$$A_{ft}(s) = \frac{|E_S^{\gamma}(s')|}{|\text{avail}(s', M)|} \tag{9}$$

i.e., it measures the amount of alternation in each imprecision. For instance, for the imprecision of Fig. 5, the alternation value is $\frac{1}{3} = 0.33$, denoting a mid-low probability of choosing a bad path (the one represented by the imprecision at s) at this point.

The last factor, the *stability factor*, addresses the stability or equilibrium of an imprecision, i.e., the probability of an imprecision to stop being an escaping state anymore, after a little perturbation.[2] In our setting, the perturbation consists on considering a small number of extra traces that pass through an imprecision, and the effect of this perturbation in the probability of the corresponding escaping state to remain being an escaping state. The number of new traces considered (z) is defined as a percentage (τ) of the total number of traces observed at this point ($s'_{\#}$). An escaping state with a high probability of remaining being an escaping state should be tackled first (i.e., higher severity), than another with a low probability, which denotes that it might disappear in the near future (i.e., considering a larger log).

Factor 3 (Stability): Let $\tau \in [0, 1]$ be the parameter indicating the percentage of new traces to be considered in order to determine the stability of an imprecision, i.e., $z = \lceil s'_{\#} \cdot \tau \rceil$ is the number of new traces to be considered. Let $l \in \mathbb{N}$ be the smallest number such that the equation $(s'_{\#} + z) \cdot \gamma < (s_{\#} + l)$ is satisfied, i.e., l defines the minimum number of traces the state s must receive in order to change from escaping to non-escaping state after considering z new traces in that point. For instance, in the example of Fig. 5, and considering $\gamma = 0.03$ and $\tau = 0.06$, z would be $\lceil 1765 \cdot 0.06 \rceil = 106$ and l would be $\lceil ((1765 + 106) \cdot 0.03) - 0 \rceil = 57$. The *stability factor* of the imprecision at s is the probability of s still being an escaping state after considering z new traces, i.e.,

$$S_{ft}(s, \tau) = P_s^z(< l) = \sum_{i=0}^{l-1} P_s^z(= i) \tag{10}$$

where $P_s^z(< x)$ and $P_s^z(= x)$ represent the probability that the state s receives less than x (or exactly x) of the new z traces considered in this point. Let p_s define the probability that a new trace in s' follows the state s, and let $1 - p_s$ be the probability that the trace follows one of the other successor states of s'. According to the *Binomial distribution* [21], the stability factor can be expressed as:

$$S_{ft}(s, \tau) = \sum_{i=0}^{l-1} \binom{z}{i} (p_s)^i (1 - p_s)^{z-i} \tag{11}$$

Formula (11) can be understood as follows: in order to s to still be an escaping state, i successes $(p_s)^i$ and $z-i$ failures $(1 - p_s)^{z-i}$ are needed. However, the i successes can occur anywhere among the z traces, and there are $\binom{z}{i}$ different ways of distributing i successes in a sequence of z traces. The probability p_s may depend on the assumptions taken. Again, if no knowledge regarding the distribution of the log is assumed, an uniform distribution is taken. Therefore, if c is the number of successor states of s', the probability of each successor state is $1/c$, and formula (11) can be rewritten as:

$$S_{ft}(s, \tau) = \sum_{i=0}^{l-1} \binom{z}{i} \left(\frac{1}{c}\right)^i \left(1 - \frac{1}{c}\right)^{z-i} \tag{12}$$

In the example of Fig. 4, given the imprecision s = ACGDG, and considering $\gamma = 0.03$ and $\tau = 0.06$, the stability factor value for s is 0.670, being $z = 46$ and $l = 25$. This factor reflects that this imprecision has a mid-probability of disappearing in the close future. This contrasts with the stability factor value of 1 obtained from the imprecision $s = ACH$ (Fig. 5), with same γ and τ parameters, reflecting a really stable imprecision.

Figure 6 shows two examples of severity diagrams (reflecting all three factors) for two imprecisions: The first diagram (a) corresponds to a really frequent imprecision. However, the situation of that imprecision is really unstable and the possibilities of choosing badly in that situation are really few. The second imprecision, shown in the diagram (b), is much more sever in general terms than the first one. It corresponds to a more stable and dangerous situation, but it is less frequent than the first one.

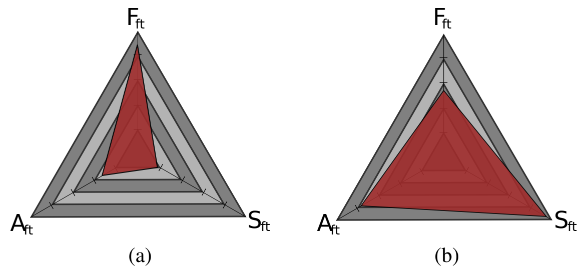


Fig. 6. Two examples of severity diagrams.

[2]The idea of introducing perturbations in order to estimate some property has been used successfully in other fields, such as the measurement of community robustness [20].

Bench	|Log|	γ	k	etc$_P$	Confidence	(s)
w01 10	20	.03	1	.818	.723 - 1.0	1.4
		.03	10	.818	.419 - 1.0	1.4
		.06	1	.818	.723 - .818	1.5
		.06	20	.818	.333 - 1.0	1.4
w01 100	200	.03	1	.818	.807 - .818	1.6
		.03	10	.818	.723 - 1.0	1.5
		.06	1	.818	.807 - .818	1.4
		.06	20	.818	.653 - 1.0	1.5
w02 1	2	.03	1	.2	.152 - .218	1.5
		.03	20	.2	.105 - .456	1.8
		.06	1	.2	.152 - 218	1.5
		.06	10	.2	.110 - .367	1.6
w02 100	200	.03	1	.2	.199 - .2	1.8
		.03	20	.2	.185 - .246	1.6
		.06	1	.2	.199 - .2	1.5
		.06	10	.2	.192 - .209	1.6

(a)

Bench		|Log|	γ	k	etc$_P$	Confidence		time(s)
a32	p20	180	.05	20	.543	.246 - .553	(307)	1 / 3 / 5
	p40	360			.564	.345 - .570	(225)	1 / 5 / 6
	p60	540			.576	.403 - .582	(179)	1 / 7 / 11
	p80	720			.583	.441 - .587	(146)	1 / 12 / 17
	p100	900			.592	.470 - .595	(125)	1 / 15 / 24
	p150	1350			.591	.504 - .595	(91)	2 / 16 / 23
	p200	1800			.591	.523 - .595	(72)	2 / 17 / 23
	p250	2250			.590	.534 - .594	(60)	2 / 16 / 24
	p300	2700			.591	.544 - .594	(50)	2 / 16 / 24
t32	p20	360	.05	20	.385	.250 - .387	(137)	2 / 67 / 121
	p40	720			.391	.305 - .392	(87)	4 / 180 / 229
	p60	1080			.392	.330 - .393	(63)	5 / 295 / 339
	p80	1440			.393	.345 - .394	(49)	6 / 336 / 496
	p100	1800			.393	.353 - .394	(41)	6 / 390 / 550
	p150	2700			.393	.365 - .393	(28)	6 / 411 / 562
	p200	3600			.393	.371 - .393	(22)	7 / 429 / 572
	p250	4500			.393	.376 - .393	(17)	9 / 440 / 579
	p300	5400			.393	.379 - .393	(14)	9 / 443 / 581

(b)

TABLE I

Benchmark	|Log|	|T|	τ	γ	etc$_P$	time (s)
large_01	25000	37	.06	.04	.705	34 / 38
large_02	25000	176	.06	.04	.627	98 / 102
large_03	10000	117	.06	.04	.465	31 / 33
large_04	10000	159	.06	.04	.524	37 / 39

TABLE II

VII. Implementation and Results

The approach presented in this paper has been implemented as an extension of the *ETConformance* plug-in within ProM 6 framework [12]. The tool uses the open-source linear programming solver *LPSolve* for computing the confidence estimation technique presented in Sect. V. The purpose of the experiments presented in this section is to illustrate the effect of the different parameters in the metric results, and to show the capacity of the technique in handling large specifications.

Table I (a) contains two simple sets of benchmarks, *w01* and *w02*, used to exemplify the main concepts presented in this paper.[3] For producing the benchmarks, a Petri net model has been created and simulated in order to derive each log. The model used in the *w01* benchmarks represents only 3 possible traces, while *w02* allows for the interleaving of 10 activities, i.e., 10! possible traces. Considering this two opposed scenarios allows to test the technique in the corner cases: the logs in *w01* contain only 2 different cases of the 3 allowed by the model, i.e., the precision tends to be high. On the other hand, the logs in *w02* contain only 2 different cases out of the 10! possible traces allowed by their model, i.e., the precision is low. The suffix *_X* in the name of each benchmark represents the number of instances of each case contained in the log, e.g., *w01_10* contains 10 traces of the first case, and 10 traces of the second case (and provided that only 2 different cases are included, the size of the log for this benchmark is 20). Finally, the evaluation of the metric and the time (in seconds) for different values of k and γ is reported.

The first conclusion one can draw from the numbers provided in Table I (a) is that etc$_P$ value is the same for both sets of benchmarks *w0X*. This is something desirable, given that in the *w01_100* and *w02_100* benchmarks, the instances of each one of the two only different traces has grown uniformly with respect to the small log, and therefore, the precision is the same. However, the confidence interval is not the same for each log, and as expected, it closely depends on the log size: the more traces considered, the more confidence in the metric value returned (i.e., the confidence interval becomes narrower). Table I (a) also shows the influence of the number of new traces to account for (k in Sect. V) in the confidence interval: the greater k is considered, the farther is the future considered, and therefore, the confidence in the metric value decreases (wider interval). Finally, for each possible *w01_X*, we show the use of different γ and its effect on the confidence interval. Considering greater values of γ, the number of traces needed to cover the escaping states increases, and consequently, less escaping states can be covered with only k traces.

Additionally, experiments on publicly available benchmarks have been performed. In Table I (b) a couple of these benchmarks are studied in depth (but additional benchmarks can be downloaded from [22]). These two benchmarks are based on the logs *a32f0n00_5* and *t32f0n00_5*, both publicly available in [23]. The Petri net models used are the ones obtained using the Petri net discovery ILPMiner [24] ProM plug-in. The experiments conducted in Table I (b) focus on illustrating how the growth of a log influences the metric and its confidence given a particular selection of the stability and confidence parameters presented in this paper. For these examples, greater values of γ that cut off more than 10% of the behavior are not considered. The column with *pX* reports the percentage of the log considered in each case, i.e. *p100* represents the original *a32f0n00_5* log whilst logs *pX* with $X < 100$ correspond to slices of the original log, e.g., *p20* contains the first 20% of the original log traces. Logs *pX* with $X > 100$ are obtained by choosing with uniform distribution among the existing traces in the log the extra traces needed to achieve the desired size. The wide spectrum of the set of benchmarks presented makes it possible to illustrate the evolution of the approach presented

[3]These benchmarks and other examples used in this paper available in [22].

in this paper and can be considered as real situations in an IS where trace sets are evaluated on a regular basis, e.g., monthly. Finally, the table includes the running time: the time after computing the metric, after computing the confidence, and the final running time that includes the creation and visualization of the imprecisions with its severity values in ProM.

A first conclusion on Table I (b) is the stability of the approach with respect to the size of the log. Note that etc_P value tends to increase as new behavior is considered, e.g., between p20 and p100 there is a difference of 0.05. However, this difference is extremely small considering that between p20 and p100 there is a 500% increment in the observed behavior. Moreover, as more traces are included in the previously observed behavior, the closer the metric value is to stabilizing. The second conclusion one can extract from this table is the dependency between the traces considered and the confidence in the metric. As it was observed in Table I (a), increasing the size of the trace set results in a narrower confidence interval (despite both lower and upper bounds increase due to the precision rising). Note also that both bounds are not symmetric, i.e., the loss caused by the *worst* trace is usually greater than the benefit of the *better* trace.

Finally, Table II is shown with the largest benchmarks, i.e., complex processes involving a large number of tasks and a high degree of parallelism, and some including also noise, duplicate and invisible tasks. These benchmarks have been generated with *PLG* tool [25]. In this table, the time reported corresponds to the time of computing only the metric, and the time also including the computation of the severity. The table illustrates that, even for large and complex benchmarks, the time needed to calculate the metric and severity is reasonable. However, given the prototyping nature of the plug-in, the *NP-HARD* complexity of *BIP*, and the academic nature of the Linear Programming solver used, the computation of the confidence intervals can only be performed for small/medium sized benchmarks. In future versions we will contemplate relaxations of the *BIP* problem and the use of more powerful solvers.

VIII. Conclusions and Future Work

In this work, important aspects have been incorporated in a precision checking framework: stability, confidence and severity. The techniques developed have been implemented as a ProM plug-in. The experiments performed reveal the usefulness and significance of the elaborated precision analysis presented in this paper.

As future work, several directions can be followed. First, techniques visualizing the severity of the imprecisions computed might be considered. On this regard, apart from coloring the sever nodes in the extended prefix automaton, other techniques that use alternatives representations can be explored. Second, the study of automated techniques for precision oriented model refinement is an interesting research problem to address. Third, proposing a partitional approach of the presented technique (i.e., not just one but disjunct prefix automatons) in order to deal with very large logs and to put less emphasis on the initial part of the model. Finally, adapting the current implementation to deal with other models apart from Petri nets would extend significantly the applicability of the approach presented in this paper.

Acknowledgment

The authors would like to thank Wil van der Aalst, Daniel Alonso and Adrià Gascón for their comments and observations.

References

[1] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: A survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.
[2] A. Rozinnat, "Process Mining: Conformance and Extension," Ph.D. dissertation, Technische Universiteit Eindhoven, 2010.
[3] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Towards Robust Conformance Checking," in *Business Process Intelligence*, 2010.
[4] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Towards an evaluation framework for process mining algorithms," *BPM Center Report BPM-07-06, BPMcenter.org*, 2007.
[5] J. D. Weerdt, M. D. Backer, J. Vanthienen, and B. Baesens, "A critical evaluation study of model-log metrics in Process Discovery," in *Business Process Intelligence*, 2010.
[6] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, "Discovering expressive process models by clustering log traces," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1010–1027, 2006.
[7] A. K. A. de Medeiros, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Quantifying process equivalence based on observed behavior," *Data Knowl. Eng.*, vol. 64, no. 1, pp. 55–74, 2008.
[8] W. M. P. van der Aalst, A. K. A. de Medeiros, and A. J. M. M. Weijters, "Genetic process mining," in *ICATPN*, vol. 3536, 2005, pp. 48–69.
[9] B. F. van Dongen, J. Mendling, and W. M. P. van der Aalst, "Structural patterns for soundness of business process models," in *EDOC*. IEEE Computer Society, 2006, pp. 116–128.
[10] T. Calders, C. W. Günther, M. Pechenizkiy, and A. Rozinat, "Using minimum description length for process mining," in *SAC*, S. Y. Shin and S. Ossowski, Eds. ACM, 2009, pp. 1451–1455.
[11] J. Munoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *Business Process Management*, Sep. 2010.
[12] "ProM 6 Framework," http://prom.win.tue.nl/tools/prom6/.
[13] T. Murata, "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.
[14] W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: yet another workflow language," *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.
[15] A. Arnold, *Finite Transition Systems*. Prentice Hall, 1994.
[16] W. M. P. van der Aalst, V. Rubin, H. M. W. E. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: a two-step approach to balance between underfitting and overfitting," *Software and Systems Modeling*, 2009.
[17] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior." *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.
[18] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.
[19] A. Schrijver, *Theory of Linear and Integer Programming*, 1998.
[20] B. Karrer, E. Levina, and M. E. J. Newman, "Robustness of community structure in networks," *Phys. Rev. E*, vol. 77, no. 4, p. 046119, Apr 2008.
[21] G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. New York: Wiley, 1978.
[22] "Benchmarks," http://www.lsi.upc.edu/~jmunoz/software.html.
[23] "Process Mining," http://www.processmining.org.
[24] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *Petri Nets*, ser. LNCS, vol. 5062, 2008, pp. 368–387.
[25] A. Burattin and A. Sperduti, "PLG: a Framework for the Generation of Business Process Models and their Execution Logs," in *Business Process Intelligence*, 2010.