Measuring the Precision of Multi-perspective Process Models

Felix Mannhardt[1,2], Massimiliano de Leoni[1⋆] , Hajo A. Reijers[3,1], Wil M.P. van der Aalst[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Lexmark Enterprise Software, Naarden, The Netherlands
[3] VU University Amsterdam, Amsterdam, The Netherlands
{f.mannhardt,m.d.leoni,h.a.reijers,w.m.p.v.d.aalst}@tue.nl

Summary. Process models need to reflect the real behavior of an organization's processes to be beneficial for several use cases, such as process analysis, process documentation and process improvement. One quality criterion for a process model is that they should precise and not express more behavior than what is observed in logging data. Existing precision measures for process models purely focus on the control-flow dimension of a process model, thereby ignoring other perspectives, such as the *data objects* manipulated by the process, the *resources* executing process activities, and *time-related* aspects (e.g., activity deadlines). Focusing on the control-flow only, the results may be misleading. This paper extends existing precision measures to incorporate the other perspectives and, through an evaluation with a real-life process and corresponding logging data, demonstrates how the new measure matches our intuitive understanding of precision.

Key words: Process Mining, Process Model Quality, Precision, Multi-perspective Process Mining

1 Introduction

Process mining is a quickly developing field that aims to discover, monitor, and improve real processes by extracting knowledge from event logs readily available in today's information systems. For most use cases of BPM, the discovered process model needs to *adequately* reflect the real behavior of the process. An obvious question is then: How does one know if a model is adequate? Clearly, a process model should be able to explain the behavior of the process using the process model: The model should *recall* the observed behavior. In other words, using process-mining terminology, the model should *fit* [1] the real behavior observed in the event log. However, the model should also be *precise* [1]: It should not allow for more behavior than observed in the event log and, thus, allow for

⋆ This work has received funding from the European Community's Seventh Framework Program FP7 under grant agreement num. 603993 (CORE).

Table 1. Event log $\mathcal{E}$ recorded by a fragment of a credit application process

Id	Case	Activity	Resource	Loan
1	1	**H**andle Request	Rory	750
2	1	**S**imple Check	Rory	$\perp$
3	1	**C**all Customer	Amy	$\perp$
4	1	**D**ecide	Amy	$\perp$
5	2	**H**andle Request	Rory	750
6	2	**C**all Customer	Amy	$\perp$
7	2	**S**imple Check	Rory	$\perp$
8	2	**D**ecide	Rory	$\perp$
9	3	**H**andle Request	Rory	1250
10	3	**S**imple Check	Rory	$\perp$
11	3	**C**all Customer	Amy	$\perp$
12	3	**D**ecide	Amy	$\perp$
..	..	..	..	..
13	4	**H**andle Request	Rory	1500
14	4	**S**imple Check	Rory	$\perp$
15	4	**C**all Customer	Amy	$\perp$
16	4	**D**ecide	Amy	$\perp$
17	5	**H**andle Request	Rory	1500
18	5	**E**xtensive Check	Rory	$\perp$
19	5	**C**all Customer	Amy	$\perp$
20	5	**D**ecide	Rory	$\perp$
21	6	**H**andle Request	Rory	5000
22	6	**E**xtensive Check	Rory	$\perp$
23	6	**C**all Customer	Amy	$\perp$
24	6	**D**ecide	Amy	$\perp$

behavior without empirical support. Therefore, the precision of a model is not an absolute value but it is relative to an event log. In other words, precision depends on what has been observed.

Multiple measures for *precision* have been proposed in the literature [2, 3, 4, 5, 6, 7]. However, these approaches can only be used to measure precision of models that do not encompass data-, resource and time-related aspects. This is a serious limitation, since these aspects play an important role in real business processes. The importance of data in business processes is, for example, paramount as it is often data that drives the decisions that participants make [8]. In industrial practice, the modeling of additional perspectives is picking up, too. Consider, for example, that support for the standard *Decision Model And Notation* (DMN) was recently added to the process modeling tool of a major vendor[1].

We wish to illustrate the problem of measuring precision while ignoring perspectives beyond control flow. Let us consider a fragment of a credit application process that generated the event log shown in Table 1. Figures 1 and 2 show BPMN models that describe the entire behavior of the process, i.e., the models are perfectly fitting with respect to the event log $\mathcal{E}$. When disregarding the data perspective, model M_1 (Fig. 1) can be seen as a precise representation of the observed behavior. The difference between models M_1 and M_2 (Fig. 2) is that the latter specifies additional rules: Depending on the requested loan amount either activity `Simple Check` or activity `Extensive Check` needs to be executed. For certain loan amounts between 1,000 and 2,000 the decision between `Simple Check` or `Extensive Check` is left to the process worker.[2] Moreover, in M_2, a separation-of-duty constraint is implemented between activities `Call Customer` and `Handle Request`: These must be performed by different resources.[3] Intuitively, these rules based on process data make the process model M_2 more precise than M_1: Their presence provides additional constraints that reduce the amount of allowed behavior. For example, model M_1 would allow to

[1] `http://www.signavio.com/news/managing-business-decisions-with-dmn-1-0/`

[2] We apply this non-standard BPMN semantics as simplification.

[3] We use an annotation as this rule cannot be expressed in BPMN.

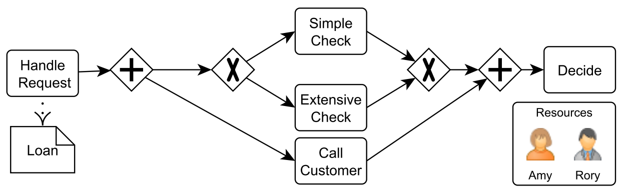


Fig. 1. BPMN model M_1 without rules and perfect fitness

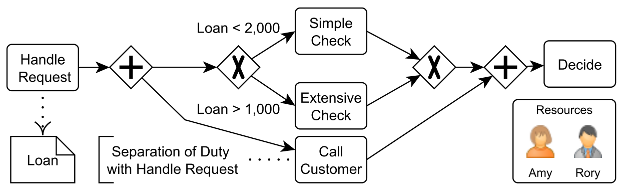


Fig. 2. BPMN model M_2 with data and resource rules and perfect fitness

execute `Simple Check` for any amount, but `Simple Check` is only considered for an amount smaller than 2,000 in the event log. The major insight here is that *existing* approaches [2, 3, 4, 5, 6, 7] would return the *same* precision score for both models.

The main contribution of this paper is a technique that generalizes the *precision* measure proposed in [6, 9] to incorporate additional rules relating to multiple perspectives. More precisely, it supports multi-perspective rules that can be encoded as constraints over data attributes and directly influence the execution of the process. The approach in [6] returns, for both M_1 and M_2, the same precision score of 0.913, because it ignores the constraints of duty separation as well as those at decision points. By contrast, our approach, when applied to the shown process model and event log, returns a lower precision score 0.757 for process model M_1 and a higher precision score 0.848 for the process model M_2. Thus, the precision added by specifying data-driven rules for choices in process models is reflected in our measure. Note that the scores returned by our approach should not be compared directly to the scores returned by the approach in [6], since we compute the precision in a new, more generic manner that acknowledges the precision added by those rules and penalizes their absence. We implemented the multi-perspective precision measure as a plug-in in the ProM framework, and evaluated it in the context of a real-life process.

The remainder of the paper is organized as follows. In Sect. 2, we define the precision measure for multi-perspective process models, illustrate the measure using examples and describe its actual implementation. Sect. 3 evaluates the introduced measure using several process models created for a real-life event log. Finally, Sect. 4 concludes with a summary and sketches future work.

2 Precision of Multi-perspective Process Models

In this section, we define a precision measure for multi-perspective process models that determines the precision of process models in relation to event logs.

2.1 Event Log and Process Model

In Sect. 1 we informally introduced event log $\mathcal{E}$ by listing the recorded events in Table 1. Each of the recorded events refers to the execution of an activity in a process instance, therefore, each event is unique. In the remainder of our paper, we define an **event log** $\mathcal{E}$ as a collection of unique events: $\mathcal{E} = \{e_1, \ldots, e_n\}$ [1, 9]. Each event is associated with a set of pairs (v, u) indicating that the event assigns value u to a process attribute v. In the remainder of the paper, we use V to define the set of attribute names that are relevant for the process in question and U to indicate the universe of possible values for attributes. Each event records some special attributes such as the case identifier `Case` and the activity name `Activity`. Each event and its location in the trace is uniquely identified through an `Id`. We denote with $A \subseteq U$ the set of recorded activity names. Given an event log $\mathcal{E}$, we also introduce the function $act \in \mathcal{E} \to A$ that extracts the name of the executed activity from an event.

Our approach to measure the precision of multi-perspective process models is independent of the formalism used to model the process, e.g., BPMN, EPCs or YAWL. To safeguard this independence, we use a transition-system notation to represent a process model. We use a transition system, which can be considered as a foundational formalism to capture processes:

Definition 1 (Process Model). *A process model defines a transition system $\mathcal{P} = (S, s_0, S_F, F)$ consisting of a set S of states, an initial state $s_0 \in S$, a set of final states $S_F \subseteq S$, and a transition function $F \in \big(S \times A \times (V \not\rightarrow U)\big) \not\rightarrow S$.*[4]

Here, we abstract from details on the *state* function to be configurable for different settings. For process models expressed as Petri nets, the reachability graph is an example of a possible transition-system representation. Several translations from BPMN models to transition systems are available. In the remainder of the paper, we assume the existence of a transition system having the following structure. Given a function f, we use $\mathsf{dom}(f)$ to denote the domain of that function. Given the set V of model attributes, the set U of potential values, and the set A of labels of BPMN activities, the transition system of BPMN models is a tuple $\mathcal{P} = (S, s_0, S_F, F)$ where

- $S \subseteq (A^* \times (V \not\rightarrow U))$;
- the initial state is $s_0 = (\langle\rangle, ass_0)$ with ass_0 being the function with an empty domain (initially no model attributes take on values);
- the set of final states $S_F \subseteq S$ contains all activity sequences (with the latest value assignments to model attributes) that reach the final BPMN event;

[4] Symbol $\not\rightarrow$ is used to indicate partial functions.

– from any state $(\sigma, ass) \in S$, a state transition $\delta = (a, w) \in A \times (V \not\rightharpoonup U)$ is defined if the BPMN activity $a \in A$ together with the value assignments $w \in V \not\rightharpoonup U$ can be executed in state (σ, ass);
– for each state $s = (\sigma, ass) \in S$, for each transition $\delta = (a, w) \in A \times (V \not\rightharpoonup U)$ defined in that state, the state-transition function is defined as follows: $F\big((\sigma, ass), \delta\big) = (\sigma', ass')$ with $\sigma' = \sigma \oplus \langle a \rangle$ and:

$$ass'(v) = \begin{cases} w(v) & \text{if } v \in \mathsf{dom}(w) \\ ass(v) & \text{otherwise.} \end{cases}$$

Events in the log can be related to a state of the transition system as follows:

Definition 2 (State Prior to the Occurrence of an Event). *Given an event log $\mathcal{E}$ and a process model $\mathcal{P} = (S, s_0, S_F, F)$, we define function $state_{\mathcal{P}} : \mathcal{E} \to S$ that, for each event, returns the state reached in the transition system just before the event happened.*

For the sake of simplicity, we assume that the event log fits the process model. Also, the names/labels of attributes and their values observed in the event log (including the activity labels as a special case) are matched with the ones in the process model. Moreover, if the process model contains unobservable routing activities (i.e., invisible transitions) or multiple activities sharing the same label (i.e., duplicate activities), then we assume that the event log contains information to uniquely identify all executed activities including unobservable ones. For any event log that does not meet these requirements, we can transform the log to the closest event log matching the requirements. This can be done, for example, by using alignment-based techniques for multi-perspective process models as [10], which "squeeze" any non-compliant log portion into a compliant one, adds events for required unobservable activities, and uniquely identifies every executed activities. In [6] it is reported that this alignment has little effect on the precision measurement even for event logs with major deviations.

2.2 Precision Measure

The *precision* of a process model in relation to an event log must take into account the extra behavior allowed by the model that is not seen in the event log. In Sect. 1, we mentioned that the *precision* of a process model is computed with respect to an event log that records executions of such a process. It is the ratio between the amount of observed behavior as recorded in the log and the amount of possible behavior as allowed by the model. All behavior that is allowed by the model yet never observed in the log makes a model less precise.

More precisely, we define *possible behavior* with respect to each event $e \in \mathcal{E}$. It consists of the the possible activities that can be executed in the state prior to the occurrence of e according to the process model.

Definition 3 (Possible Behavior). *Let $\mathcal{P} = (S, s_0, S_F, F)$ the transition system of a process model. Let $\mathcal{E}$ be an event log. The possible behavior when event e occurs as allowed by a model can be represented as a function $pos_{\mathcal{P}} : \mathcal{E} \to 2^A$:*

$$pos_{\mathcal{P}}(e) = \{a \in A \mid \exists w \in V \not\rightarrow U : \exists(state_{\mathcal{P}}(e), a, w) \in \mathsf{dom}(F)\}.$$

In a similar way, we define the observed behavior prior to the occurrence of any event $e \in \mathcal{E}$ as the activities that can observed in the whole event log when being in the same state as prior to the occurrence of e:

Definition 4 (Observed Behavior). *Let $\mathcal{P} = (S, s_0, S_F, F)$ be the transition system of a process model. Let $\mathcal{E}$ be an event log, and $e \in \mathcal{E}$ an event. The observed behavior as seen in the event log can be represented as a function $obs_{\mathcal{P}} : \mathcal{E} \rightarrow 2^A$:*

$$obs_{\mathcal{P}}(e) = \{a \in A \mid \exists\, e' \in \mathcal{E} : state_{\mathcal{P}}(e) = state_{\mathcal{P}}(e') \wedge act(e') = a\}.$$

Using the definitions of possible and observed behavior in the context of an event, we define the precision of a multi-perspective process model $\mathcal{P}$ according to an event log $\mathcal{E}$ as follow.

Definition 5 (Precision of a Process Model wrt. an Event Log). *Let $\mathcal{P}$ be the transition system of a process model. Let $\mathcal{E}$ be an event log. The precision of $\mathcal{P}$ with regard to $\mathcal{E}$ is a function $precision : \mathcal{P} \times \mathcal{E} \rightarrow [0, 1]$:*

$$precision(\mathcal{P}, \mathcal{E}) = \frac{\sum_{e \in \mathcal{E}} |obs_{\mathcal{P}}(e)|}{\sum_{e \in \mathcal{E}} |pos_{\mathcal{P}}(e)|}.$$

Since for each event $e \in \mathcal{E}$, $|obs_{\mathcal{P}}(e)| \leq |pos_{\mathcal{P}}(e)|$, precision scores are always between 0 and 1. Note that the transition system is finite: a state s is only considered if there is an event $e \in \mathcal{E}$ such that $state_{\mathcal{P}}(e) = s$. Since the number of events is finite, the number of states to consider is also finite.

For each event $e \in \mathcal{E}$, computing $state_{\mathcal{P}}(e)$ is $O(|\mathcal{E}|)$, because, in the worst case, one needs to iterate over all events in $\mathcal{E}$ to reconstruct the state. Once $state_{\mathcal{P}}(e)$ is computed, computing $obs_{\mathcal{P}}(e)$ and $pos_{\mathcal{P}}(e)$ is linear in the number of activities: $O(|A|)$. Since these functions need to be computed for each $e \in \mathcal{E}$, the worst-case time complexity of computing precision is $O(|\mathcal{E}|(|A| + |\mathcal{E}|))$, which is $O(|\mathcal{E}|^2)$ as $|\mathcal{E}| >> |A|$ (the number of events is way larger than the process activity).

2.3 Illustration of the Measure

We proceed to show that our definition is intuitive by discussing a series of illustrative examples. In this section activities and attribute names are abbreviated with their first letter; also, with abuse of notation, any model M_i also refers to its transition-system representation as defined in Sect. 2.1. We obtain the following sets of observed and possible behavior for the events listed in Table 1 and the initial model M_1:

$$\begin{array}{l} pos_{M_1}(e_1) = \{\mathtt{H}\},\ pos_{M_1}(e_2) = \{\mathtt{S}, \mathtt{E}, \mathtt{C}\},\ pos_{M_1}(e_3) = \{\mathtt{C}\},\ pos_{M_1}(e_4) = \{\mathtt{D}\}, \ldots \\ obs_{M_1}(e_1) = \{\mathtt{H}\},\ obs_{M_1}(e_2) = \{\mathtt{S}, \mathtt{C}\},\quad obs_{M_1}(e_3) = \{\mathtt{C}\},\ obs_{M_1}(e_4) = \{\mathtt{D}\}, \ldots \end{array}$$

For example, the set of observed behavior for e_2 is $\{\mathtt{S}, \mathtt{C}\}$ because the execution of both activities `Simple Check` and `Call Customer` can be observed in those

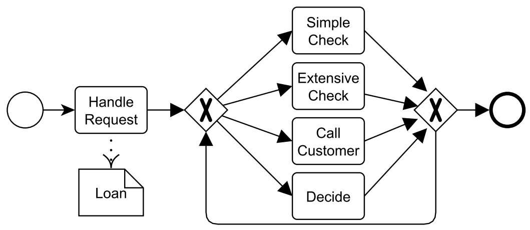


Fig. 3. BPMN model M_3 for log $\mathcal{E}$ with a precision of 0.359

events that are carried out when the transition system is in the state prior to the occurrence of e_2: $state_{M_1}(e_2) = (\langle \mathtt{H} \rangle, \{\mathtt{R} := \mathit{Rory}, \mathtt{L} := 750\})$. This state is reached when activity `Handle Request` has already been executed and the latest values assigned to the attributes `Resource` and `Loan` are *Rory* and 750 respectively. By consulting Table 1, it becomes clear that events e_2 and e_6 contribute to the set of observed behavior for e_2. Please note that the e_2 and e_6 are events from different traces, i.e., the whole event log is considered when computing the observed behavior. Continuing in the same manner with the remaining events yields a precision of $\mathit{precision}(M_1, \mathcal{E}) = \frac{28}{37} \approx 0.76$. Applying the same measure on the process model M_2, we get:

$$pos_{M_2}(e_1) = \{\mathtt{H}\},\ pos_{M_2}(e_2) = \{\mathtt{S}, \mathtt{C}\},\ pos_{M_2}(e_3) = \{\mathtt{C}\},\ pos_{M_2}(e_4) = \{\mathtt{D}\}, \ldots$$
$$obs_{M_2}(e_1) = \{\mathtt{H}\},\ obs_{M_2}(e_2) = \{\mathtt{S}, \mathtt{C}\},\ obs_{M_2}(e_3) = \{\mathtt{C}\},\ obs_{M_2}(e_4) = \{\mathtt{D}\}, \ldots$$

and subsequently a value of $\mathit{precision}(M_2, \mathcal{E}) = \frac{28}{33} = 0.848$. It is easy to see that the added constraints regarding the attribute `Loan` limits the set of possible activities for event e_2 to `Simple Check` and `Call Customer`. The activity `Extended Check` cannot be executed anymore in the state prior to the occurrence of e_2, as the value of the attribute `Loan` would need to be higher than 1,000. This improves the precision of model M_2. Moreover, the observed parallelism of activities `Simple Check` and `Call Customer` for a `Loan` value of 750 is not seen as imprecision in either case, as reflected in the set of observed behavior $\{\mathtt{S}, \mathtt{C}\}$ for event e_2. Note that the assignment of data attributes needs to be *exactly* the same in order to detect parallelism in the model as a precise representation of the observed behavior. Otherwise, when parallelism is observed with different attribute values, it is seen as an imprecision because a more precise process model using the different attribute values for a data rule can be created.

Next to the two process models introduced in Fig. 1 and Fig. 2, we consider two additional, illustrative process models representing extreme cases. Fig. 3 shows an example of a process model that is very imprecise in relation to event log $\mathcal{E}$. Always starting with `Handle Request`, model M_3 in Fig. 3 allows to execute the remaining activities any arbitrary number of times and, also, in any order. On the opposite side of the spectrum, model M_4 in Fig. 4 is very precise, as only exactly the observed behavior in $\mathcal{E}$ is possible. For instance, the order of the activities `Simple Check` and `Call Customer` depends on the value of the

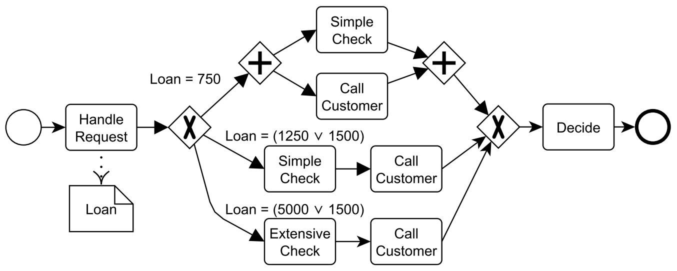


Fig. 4. BPMN model M_4 for log $\mathcal{E}$ with a precision of 1

data attribute `Loan`; if it has 750 as value both activities are carried out in any order, for the values 1,500 and 1,250 only the modeled order is observed.

For the process model M_3, we obtain $\mathit{precision}(M_3, \mathcal{E}) = \frac{28}{78} \approx 0.359$, which is less than half of the precision measure of M_1 in Fig. 2 (0.848). On the other end of the spectrum, model M_4 in Fig. 4 has a perfect precision: $\mathit{precision}(M_4, \mathcal{E}) = 1$, as only the behavior seen in the event log is allowed. These examples demonstrate that the computed values for *precision* behave intuitively, as model M_3 is the least precise, and model M_4 is the most precise of the presented examples.

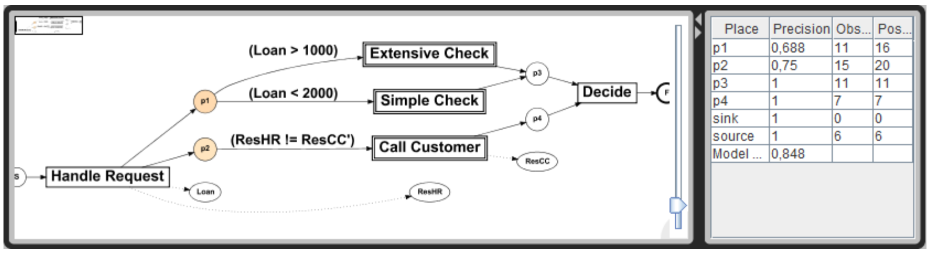
The example above also shows that models scoring a very high precision value are not always the most preferable. In particular, model M_4 in Fig. 4 with a perfect precision score allows for exactly the behavior observed in the event log and nothing more. Given that event logs only contain example behavior as observed in a limited time frame, one cannot assume that all possible behavior has been observed. Hence, a process model should to some extent *generalize* and allow for more behavior than what has simply been observed, yet is potentially admissible. In other words, using data-mining terminology, model M_4 is probably *over-fitting* the event log. Therefore, when using the proposed precision measure to, e.g., rank the quality of discovered process models it should be balanced with other quality criteria [9], rather than aiming for a perfect precision score.

2.4 Implementation

This section shows the concrete implementation of the precision measure as a plug-in for the process mining framework ProM[5]. We use Data Petri Nets (DPN-nets) [10] as modeling language with simple and clear semantics.

A DPN-net is a Petri net [11] extended with variables (i.e., data attributes). Transitions update the values of variables through so-called *write operations* and can be associated with guards that further constrain when transitions are enabled to fire. A transition in a DPN-net can fire only if all its input places contain at least one token and the guard, if any, is satisfied. Guards can be formulated as an expression over the process variables. Fig. 5 shows how the BPMN model M_2 (Fig. 2) can be expressed using the DPN-net notation. For instance, transition

[5] Available at `http://www.promtools.org` in the *DataAwareReplayer* package



Place	Precision	Obs...	Pos...
p1	0,688	11	16
p2	0,75	15	20
p3	1	11	11
p4	1	7	7
sink	1	0	0
source	1	6	6
Model ...	0,848		

Fig. 5. Screenshot of the implementation in ProM showing the precision visualization of M_2 and $\mathcal{E}$. The darker the color of a place in the DPN-net, the lower its precision

(i.e., activity) `Extensive Check` is enabled if place $p1$ contains at least a token and the value of variable *Loan*, which has been written by transition `Handle Request`, is larger than 1,000. However, at times one may wish to constrain the values that a transition is allowed to write. In those cases, variables are post-fixed with a *prime* symbol, indicating the value assigned by the transition. For instance, the guard of transition `Call Customer` states that the resource executing the activity must differ from the resource who executed the `Handle Request`: in this way, we can enforce a separation of concerns. Readers are referred to [10] for more details. The behavior of a DPN-net can be represented as a transition system similarly as discussed in Sect. 2.1 for BPMN models.

To provide diagnostics on the precision measurement, we can compute a local precision score for each place by including only those events that correspond to DPN-net transitions that consume tokens from the place. A visualization of this diagnostics for model M_2 is shown in Fig. 5. Each place is colored according to its local precision score (darker colors correspond to lower precision). Additionally, the table on the right side provides an overview about the precision scores.

3 Evaluation

The evaluation is based on a real-life case study, which is concerned with the process of handling road-traffic fine by an Italian local police force [10]. Specifically, we employed a real-life event log[6] that records the execution of 11 different activities and contains 550,000 events grouped into 150,000 traces. In total there are 9 data attributes. All experiments were conducted with a memory limit of 2 GB, which is lower than what current-day, regular computers contain.

For this evaluation, we used five different models:

Model A, which is discovered using the Inductive Miner (IM) set to guarantee perfect fitness [12];

Model B, which extends model A with guards as discovered by the decision-tree miner (DTM) [13]; *the minimal instances per decision-tree leaf* parameter was set to 125 to avoid over-fitting;

[6] `http://dx.doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5`

Table 2. Precision and fitness scores for the normative and discovered process models of a process managing road traffic fines enacted in an Italian municipality

Process Model	**Precision**	**Fitness**
A: Inductive Miner	0.298	1
B: Inductive Miner with discovered rules	0.362	0.932
C: Normative Model without guards	0.639	0.997
D: Normative Model	0.714	0.974
E: Normative Model with discovered rules	0.831	0.986

Model C, which is the normative model, shown in [10], but without any guards;
Model D, the normative model from [10] again, yet including all those guards that concern attributes available in the public event log;
Model E, which extends Model C with the guards discovered with the DTM (using the same as for model B).[7]

Table 2 shows the precision and fitness scores for the described process models. Intuitively, **Model A** should be the least precise process model. This model does not constrain the allowed behavior with any data rules. Also, the *IM*, set to guarantee perfect fitness, is unlikely to discover a precise model for this event log, which includes infrequent behavior. Indeed, Model A scores a low precision of 0.298. Model B (shown in Fig. 6) has the same control-flow as Model A, but additional guards based on discovered rules. As expected, the discovered rules in Model B result in an improved precision of 0.362 and a lower fitness. Fig. 6 shows the precision measurement for **Model B** as it is returned by the ProM plug-in. The coloring of the places allows to locate the effect of the discovered rules on precision. It shows that the data rule added for *Send for Credit Collection* results in a perfect precision in that part of the model, i.e., the rule added to this transition is mutually exclusive with the rule added for the alternative transition τ_4. Still, Model B arguably allows for too much behavior. The normative model without data rules, **Model C**, is more precise than the models discovered with the IM; it precision is 0.639. As expected, adding the normative data rules shown in [10] to arrive at **Model D** will increase its precision to 0.714. However, adding those data rules has an impact on the fitness of model D (-0.023). As reported in [10], the event log shows that the rules are not always respected. Finally, we applied the DTM on the normative model, which resulted in **Model E**. It scores best on precision (0.831), and better on fitness than model D.

Completely in line with expectations, Model E scores better in fitness, because it discovers the as-is rules rather than the to-be rules. A cursory glance on the results may invoke surprise that the precision of model E is higher than the precision of model D. However, this can be expected: The guards are discovered so as to maximize fitness and precision. Therefore, the rules added in model D only reflect constraints on the process from a compliance perspective, e.g. `Send of Credit Collection` should only be executed if the fine is not yet fully paid. By contrast, the DTM strives for discovery of mutually-exclusive rules that describe the real behavior as observed in the event log. Being based on the real

[7] All models can be retrieved from `http://purl.tue.nl/726309911741849`

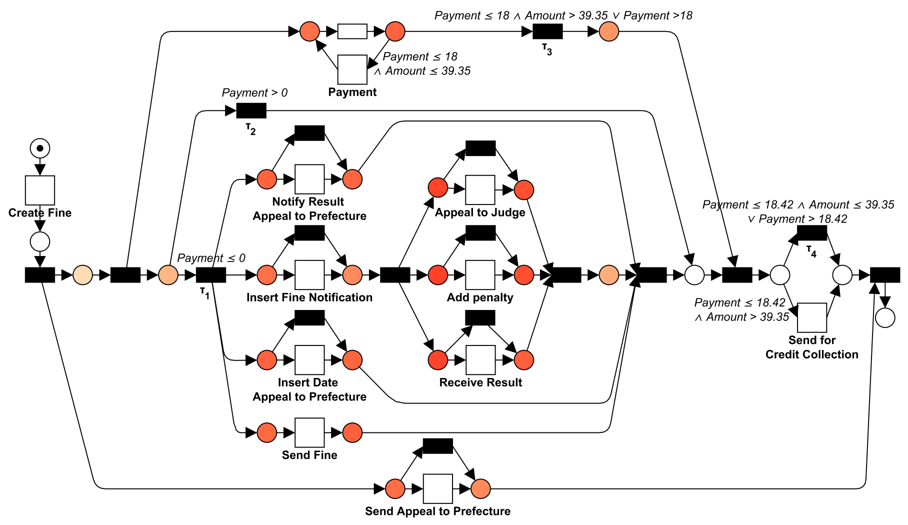


Fig. 6. The visualization in ProM provides a "helicopter view" of the precision measurement of Model B. Black rectangles depict invisible routing transitions. Write operations are omitted and the figure was redrawn to improve the readability on paper

process executions, these rules may violate the normative rules and provide logic that has no business relevance. This case study shows that our way of computing precision is applicable to evaluate the quality of multi-perspective process models and provides intuitive results.

4 Conclusion

In this paper, we proposed a new measure for the *precision* of multi-perspective process models in relation to behavior that is described in the form of an event log. Whereas process modeling languages commonly used in practice (e.g. BPMN) allows one to specify data-driven rules to model choices, existing approaches to measure the precision of a process model ignore data-related aspects. The precision of a process model can be seen as the fraction of the possible behavior allowed by the model in relation to what has actually been observed, as recorded in the event log. This paper reports on the *first proposal* to measure precision for multi-perspective process models.

As future work, we aim to put our technique to the test in several real-life case studies. In particular, we want to perform an end-user evaluation with business analysts to verify whether our notion of precision is in line with their expectations. Our preliminary results make us believe that will be the case. In particular, given an event log of a certain process and a number of models for the same process, we are able to determine which model scores higher on precision. For each model, the respective precision score can be combined with the fitness score, which, for instance, can be computed using the approach reported in [10]. In many cases, higher values of precision are associated with lower values of fitness, and vice versa. By finding the right trade-off between these two quantities, we can determine which model provides a better representation of the process in question. Last but not least, we aim to employ the precision measure to improve the discovery of guards. The approach proposed in [13] only allows for discovering mutually exclusive guards at decision points. Often, guards at decision points are not mutually exclusive: in the same process state, multiple alternatives need to be enabled. We want to allow for multiple alternatives (to increase fitness), but not for too many (which would reduce precision).

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Mining expressive process models by clustering workflow traces. In: Advances in Knowledge Discovery and Data Mining. Volume 3056 of LNCS. Springer (2004) 52–62
3. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Min Knowl Discov **14**(2) (2007) 245–304
4. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf Syst **33**(1) (2008) 64–95
5. Munoz-Gama, J., Carmona, J.: A general framework for precision checking. Int J Innov Comput I **8**(7(B)) (2012) 5317–5339
6. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Measuring precision of modeled behavior. Inf Syst E-Bus Manage **13**(1) (2015) 37–67
7. van den Broucke, S., De Weerdt, J., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Trans Knowledge Data Eng **26**(8) (2014) 1877–1889
8. Maggi, F.M., Dumas, M., García-Bañuelos, L., Montali, M.: Discovering data-aware declarative process models from event logs. In: BPM'13. Volume 8094 of LNCS. Springer (2013) 81–96
9. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdiscip Rev Data Min Knowl Discov **2**(2) (2012) 182–192
10. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing (2015)
11. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995)
12. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Application and Theory of Petri Nets and Concurrency. Volume 7927 of LNCS. Springer (2013) 311–329
13. de Leoni, M., van der Aalst, W.M.P.: Data-Aware Process Mining: Discovering Decisions in Processes Using Alignments. In: SAC'13, ACM (2013) 1454–1461