No Journal Yet manuscript No.
(will be inserted by the editor)

Conformance Checking

Jorge Munoz-Gama

Received: date / Accepted: date

1 Synonyms

Business process conformance checking

2 Definition

Given an event log and a process model from the same process, conformance checking compares the recorded event data with the model to identify commonalities and discrepancies. The conformance between a log and model can be quantified with respect to different quality dimensions: *fitness*, *precision*, and *generalization*.

3 Cross-references

Automated Process Discovery
Event Logs and Visualization
Process Model Repair

4 Overview

Conformance checking compares an event log with a process model of the same process [Munoz-Gama, 2016]. An event log is composed of a series of log traces where each log trace relates to the sequence of observed events of a process instance, i.e., a case. An event can be related to a particular activity in the process, but can also record many other process information such as timestamp, resource and cost. In a real-life context, event logs can be extracted from Process Aware Information

Jorge Munoz-Gama
Department of Computer Science, School of Engineering,
Pontificia Universidad Católica de Chile
E-mail: jmun@uc.cl (✉)

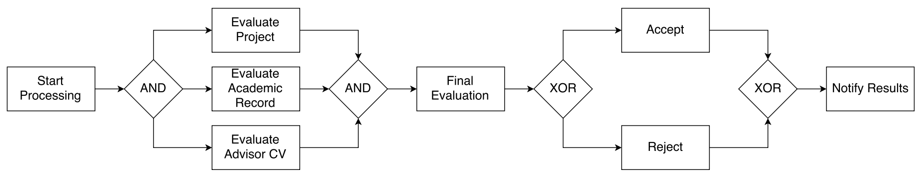


Fig. 1 Informal process model of a university scholarship process

Systems (PAIS) such as Workflow Management (WFM) systems, Business Process Management (BPM) systems, or typical relational databases, such as SAP database. Similarly, process models can often be extracted from the organization's information systems. These can be normative models that the organization uses to manage their process, or descriptive, created by hand or automatically discovered go gain insight into their processes [van der Aalst, 2013].

Depending on the nature of the model, discrepancies between the log and model can have different interpretations [van der Aalst, 2016]. For a normative model, deviations indicate violations of imposed constraints. For example, a banking process may require the processing and approval of a loan to be done by different employees to avoid the risk of misconduct (four-eyes principle). Clearly, conformance checking between an event log of the handled loan applications and the process model can be applied to assess compliance. On the other hand, for a descriptive model, deviations indicate that the model is not fully capturing all the observed behavior in the log. For example, process analysts might perform conformance checking on the models discovered by different process discovery algorithms before selecting the ones that are of sufficient quality for further analysis.

To illustrate conformance checking, a simple process is introduced. Figure 1 shows a doctoral scholarship application process in an informal modeling notation. This process consists of eight activities: *Start Processing*, *Evaluate Project*, *Evaluate Academic Record*, *Evaluate Advisor CV*, *Final Evaluation*, *Accept*, *Reject*, *Notify Results*. To begin the process, an applicant has to submit their academic record, their advisor's CV, and a description of their proposed project. Once the required documents are received, the committee would begin by evaluating the submitted documents. As shown by the AND gateway, the committee can choose to evaluate the three documents in any order. Following the preliminary evaluation, a final evaluation is done to consolidate the previous results. This leads to either the acceptance or rejection of the application. Finally, the applicant is notified of the result. An example of log trace corresponding to an accepted application could be ⟨ *Start Processing*, *Evaluate Project*, *Evaluate Academic Record*, *Evaluate Advisor CV*, *Final Evaluation*, *Accept*, *Notify Results* ⟩.

4.1 Dimensions of Conformance

Through conformance checking, commonalities and discrepancies between a log and model are quantified. One simple idea would be to consider that a log and model are conforming with each other if the observed behavior in the log is captured by the model. This means that a log and model are perfectly conforming if all the log traces can be *fitted* to the model. However, this can be easily achieved

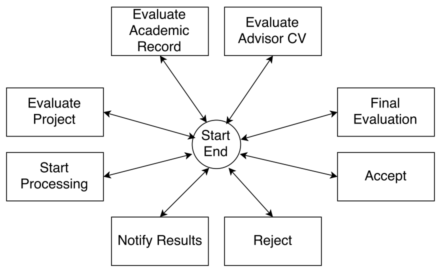


Fig. 2 Imprecise flower model of the doctoral scholarship process

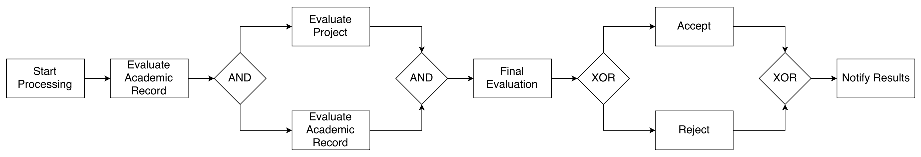


Fig. 3 Precise but unfitting model of the doctoral scholarship process

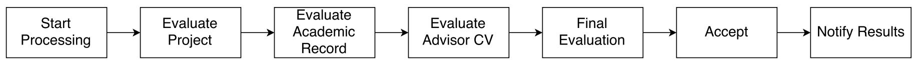


Fig. 4 Model that overfits one particular possible execution of the doctoral scholarship process

with a model that allows any behavior. Such models do not provide much information to the data analyst about the process. This shows that there is a need to consider conformance with respect to different dimensions.

Currently, conformance is generally considered with respect to three dimensions – *fitness*, *precision*, and *generalization*.

Fitness relates to how well a model and log fit each other. A log trace perfectly *fits* the model if it can be replayed onto the model and corresponds to a complete model trace. For example, ⟨ *Start Processing*, *Evaluate Project*, *Evaluate Academic Record*, *Evaluate Advisor CV*, *Final Evaluation*, *Accept*, *Notify Results* ⟩ perfectly fits the model in Figure 1 since each of the observed steps can be sequentially replayed at the model and the trace corresponds to a particular possible way to execute the process model. However, the trace ⟨ *Start Processing*, *Evaluate Project*, *Evaluate Academic Record*, *Final Evaluation*, *Reject*, *Notify Results* ⟩ does not fit the model because the advisor's CV (*Evaluate Advisor CV*) is never evaluated. This suggests that the corresponding application has been rejected without proper evaluation.

Precision relates to a model's ability to capture the observed behavior without allowing unseen behavior. It is not enough to have a model that is perfectly fitting with the log since this can be easily achieved with a model that permits any behavior. Consider the "flower" model in Figure 2, it consists of all the transitions attached to a state that corresponds to both the start and end state. This means any sequence involving the connected transitions is permissible by the model. Though perfectly fitting with the log, such underfitting model does not convey much useful information to the user. In contrary, the process model illustrated in Figure 3 is much more precise than the flower model.

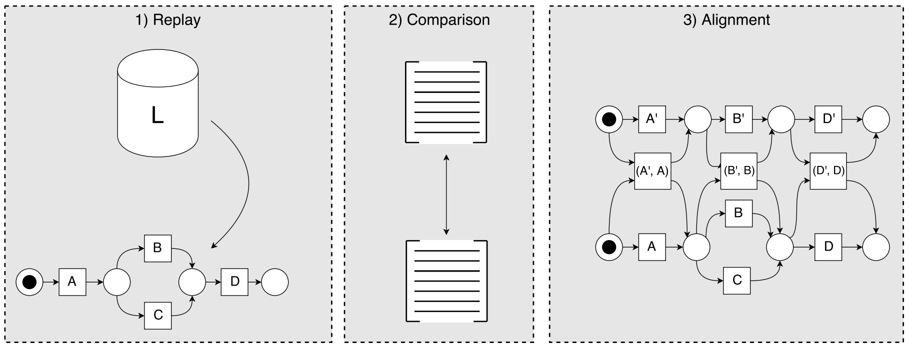


Fig. 5 Three main types of conformance checking approaches

Generalization relates to a model's ability to account for yet to be observed behavior. Typically, an event log only represents a small fraction of the possible behavior in the process. As such, a good model must be generalizing enough so that unobserved but possible behavior is described. For example, if the model in Figure 4 was discovered from an event log that contains only the trace ⟨ *Start Processing*, *Evaluate Project*, *Evaluate Academic Record*, *Evaluate Advisor CV*, *Final Evaluation*, *Accept*, *Notify Results* ⟩, then the model would be both perfectly fitting and precise since all observed behavior is captured by the model and that it does not allow any unseen behavior. Clearly, there is much unseen behavior that is very likely to occur in the future, e.g., the rejection of an application. This shows that while it is important to have precise models, it is also important to avoid overfitting the observed behavior.

Some authors consider a fourth dimension called *simplicity*, relating to the model complexity, i.e., simple models should be preferred over complex models if both describe the same behavior. However this quality dimension relates only to the model and therefore is not normally measured by conformance checking techniques. This dimension is covered in the *Automated Process Discovery* entry of this encyclopedia.

Overall, while the three quality dimensions are orthogonal to each other, in a real life context, one is unlikely to find a pair of log and model that are in perfect conformance (i.e., perfectly fitting, precise, and generalizing). Often times, different scenarios may require different conformance levels and prioritization of the quality dimensions. For example, to analyze the well established execution paths of a process, an analyst might prioritize fitness over the other dimensions. On the other hand, if an event log only contains a small number of cases, generalization would likely to be prioritized over the other dimensions to account for possible future behavior.

4.2 Types of Conformance

Conformance checking techniques can be applied to understand and quantify these relationships between a log and model. There is a large collection of approaches and metrics that are based on different ways to compare a log and model.

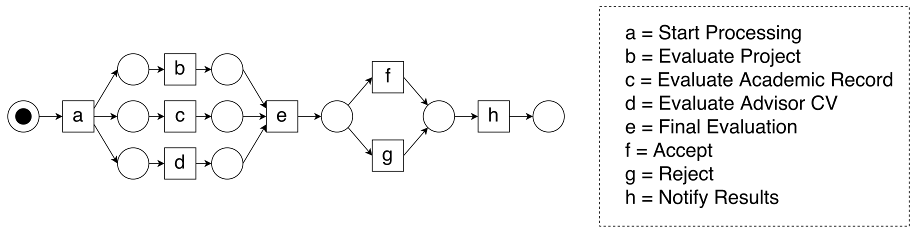


Fig. 6 Model M_1 of the university doctoral scholarship process denoted in Petri net notation

Figure 5 shows that there are three main groups of conformance checking approaches – *replay*, *comparison* and *alignment*. Replay based approaches replay log traces onto the model and record information about the conformance during the replay. Process models can be denoted in different modeling notations, e.g., Business Process Modeling Notation (BPMN), Petri nets and Process Trees, and each representation bias has distinct characteristics, e.g., formalism, and determinism. However, a proper process model is typically executable so that log traces can be re-executed stepwise by the model. Comparison based approaches convert both the log and model into a common representation so that the log and model are directly comparable. Last but not least, alignment based approaches seek to explain observed behavior in terms of modeled behavior by aligning log traces with the model. This brings conformance checking to the level of events and can offer detailed diagnosis on conformance issues.

5 Key Research Findings

In this section, two conformance checking approaches and three conformance metrics are presented.

$$\begin{aligned} L_1 = [t_1 &= \langle a, b, c, d, e, f, h \rangle, \\ t_2 &= \langle a, c, b, e, f, h \rangle, \\ t_3 &= \langle a, b, d, c, g, e, h \rangle] \end{aligned}$$

Fig. 7 Running example: Event log L_1

5.1 Token Replay

Token replay is a replay based conformance checking approach that measures the fitness between a log and model by replaying log traces onto process models denoted in the Petri net notation [Rozinat & van der Aalst, 2008].

Consider model M_1 in Figure 6 and log L_1 in Figure 7. Model M_1 is denoted in Petri net notation so that the squares correspond to the activities in the process,

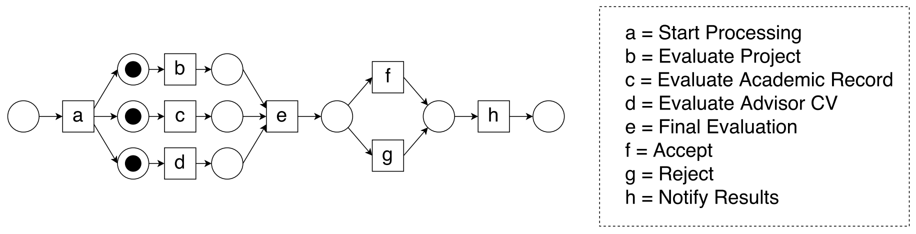


Fig. 8 Model M_1 after firing activity a

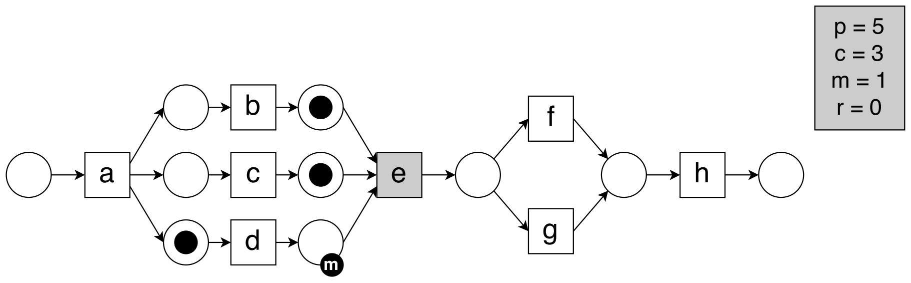


Fig. 9 Missing token to fire activity e in token replay of trace $t_2 = \langle a, c, b, e, f, h \rangle$

filled circles correspond to tokens that mark the state of a process instance as activities get executed and empty circles correspond to places that hold tokens. To execute an activity, all its input places (i.e., all the places connected by an incoming arrow to the activity) must have at least one token. This means that the activity is *enabled* and can be *fired*. When an activity is fired, the activity *consumes* a token from each of its input places before *producing* one token at each of its output places (i.e., all the places connected by an outgoing arrow from the activity). For example, activity a in model M_1 in Figure 6 is currently enabled. If the activity is fired, it would consume the token of its input place and produce three tokens at each of its three output places as illustrated in Figure 8. As such, an instance of the process can be recorded by successively firing enabled activities until no activities are enabled. For a valid Petri net model, an instance is initiated by having a token at each of the source places (i.e., places without any incoming arrows) and is deemed to be completed by firing activities until there is only one token at each of the sink places (i.e., places without any outgoing arrows) and none at any other places. The sequence of fired activities corresponds to a complete model trace, i.e., a possible execution of the model.

Log traces can be replayed onto the model by successively firing the activities related to each event in the log trace at an initiated Petri net model. If the log trace is perfectly fitting with the model, there should not be any problem with the replay since the log trace corresponds to a complete model trace. However, for deviating traces, replay would not be successful due to missing or redundant tokens. An activity might be marked to be fired in the log trace but is not enabled in the model due to missing tokens at its input places. Consider the replay of trace t_2 in L_1. Starting from the initial state of model M_1 as shown in Figure 6, the

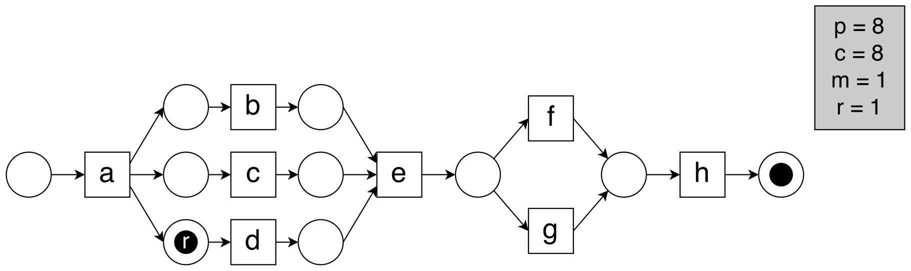


Fig. 10 Remaining token in token replay of trace $t_2 = \langle a, c, b, e, f, h \rangle$

firing of activity a, b and c would consume three tokens and produce five tokens in the process. Figure 9 shows the state of model M_1 after the firing of the first three activities and records the number of consumed and produced tokens. According to trace t_2, the next activity to be fired is activity e. However, this is not possible since one of the input places of activity e does not have any token, i.e., activity e is not enabled. To continue the replay, the missing token is artificially added into the empty input place and the number of missing token is incremented. The rest of trace t_2 (activity f and g) can be replayed successively. Figure 10 shows that after firing activity h, there is a remaining token in the input place of activity d since this activity was not fired in the replay of trace t_2. As recalled, a process instance is only completed when there is only one token at each of the sink places and none at any other places. To complete the replay, the remaining token is removed artificially and the number of remaining tokens is incremented.

Based on the count of each token types consumed, produced, missing, and remaining (p = 8, c = 8, m = 1, r = 1) the fitness between model M_1 and trace t_2 can be computed as:

$$\begin{aligned} \mathit{fitness}(t_2, M) &= \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p}) \\ &= \frac{1}{2}(1 - \frac{1}{5}) + \frac{1}{2}(1 - \frac{1}{5}) = 0.8 \end{aligned}$$

This fitness metric can be extended to the log level by considering the number of produced, consumed, missing and remaining tokens from the token replay of all log traces.

5.2 Cost-based Alignment

The token replay approach can easily identify deviating traces in an event log. Moreover, the deviation severity can be compared using a fitness metric computed from the number of produced, consumed, missing and remaining tokens. However, the token replay approach is prone to creating too many tokens for highly deviating traces so that any behavior is allowed. This can lead to an overestimation of the fitness. The approach is also specific to the Petri net notation.

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|} a & b & d & c & g & e & \gg & h \\ \hline a & b & d & c & \gg & e & g & h \end{array} \qquad \gamma_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|} a & b & d & c & \gg & g & e & \gg & h \\ \hline a & b & \gg & c & d & \gg & e & g & h \end{array}$$

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|} a & b & d & c & \gg & g & e & h \\ \hline a & b & d & c & e & g & \gg & h \end{array}$$

Fig. 11 Possible alignments between trace $t_3 = \langle a, b, d, c, g, e, h\rangle$ in L_1 and model M_1 in Figure 6

More importantly, in the case of a deviating trace, the approach does not provide a model explanation of the log trace. For example, the deviations in trace $t_2 = \langle a, c, b, e, f, h\rangle$ can be explained if it was considered with respect to the complete model trace $\langle a, c, b, d, e, f, h\rangle$. From this mapping, it is clear that the log trace is missing the execution of activity d (Evaluate Advisor CV). These mappings from log traces to model traces were introduced as *alignments* to address this limitation [van der Aalst et al., 2012].

Alignments are tables of two rows where the top row corresponds to the observed behavior (i.e., log projection) and the bottom row corresponds to the modeled behavior (i.e., model projection). Each column is therefore a move in the alignment where the observed behavior is aligned with the modeled behavior. Consider alignment γ_1 in Figure 11. This alignment aligns trace $t_3 = \langle a, b, d, c, g, e, h\rangle$ in L_1 and model M_1 in Figure 6. The top row (ignoring $\gg$) yields the trace $t_3 = \langle a, b, d, c, g, e, h\rangle$ and the bottom row (ignoring $\gg$) yields a complete model trace $\langle a, b, d, c, e, g, h\rangle$. For each move in alignment γ_1, the top row matches the bottom row if the step in the log trace matches the step in the model trace. This is called a *synchronous move*. In the case of deviations, a no-move symbol $\gg$ is placed in the bottom row if there is a step in the log trace that cannot be mimicked by the model trace. For example, activity g is executed before activity e in trace t_3 but model M_1 requires activity e to be fired before activity g. Hence, a *log move* is put where the top row has activity g and the bottom row has a no-move $\gg$. Similarly, a no-move symbol $\gg$ is placed in the top row if there is a step in the model trace that cannot be mimicked by the log trace. For example, activity g is executed after activity e in the model trace according to model M_1. Therefore, a *model move* is added where the top row has a no-move $\gg$ and the bottom row has activity g. It is also possible that there are invisible transitions in the model which are not observable in the log. Similar to a model move, there would be a no-move in the top row and an invisible transition label in the bottom row. In total, there are four types of *legal moves* in an alignment: synchronous move, log move, model move and invisible move.

For a particular log trace and model, there could be many possible alignments where each represents a different explanation of the observed behavior in terms of modeled behavior. For example, Figure 11 shows three possible alignments between trace t_3 and model M_1 in Figure 6. Clearly, alignment γ_1 and γ_3 are better alignments of trace t_3 and model M_1 than alignment γ_2 since they provide closer

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
a & b & d & c & g & e & h & \gg & \gg & \gg & \gg & \gg & \gg & \gg \\
\hline
\gg & \gg & \gg & \gg & \gg & \gg & \gg & a & b & d & c & e & g & h
\end{array}$$

Fig. 12 Default alignment between trace t_3 in log L_1 and model M_1 in Figure 6

explanations with less log moves and model moves. The quality of an alignment can be quantified by assigning costs to moves. In general, model moves and log moves are assigned higher costs than synchronous moves because they represent deviations between modeled behavior and observed behavior. A standard cost assignment could be that all model moves and log moves are assigned a cost of 1 and synchronous moves and invisible moves are assigned a cost of 0. Invisible moves are normally assigned zero costs as they are related to invisible routing transitions in the model that are not observable in the log. Under the standard cost assignment, the costs of the alignments in Figure 11 can be computed as follows:

$$\begin{aligned}
cost(\gamma_1) &= 0+0+0+0+1+0+1+0 = 2\\
cost(\gamma_2) &= 0+0+1+0+1+1+0+1+0 = 4\\
cost(\gamma_3) &= 0+0+0+0+1+0+1+0 = 2
\end{aligned}$$

This confirms the previous intuition that alignment γ_1 and γ_3 are better alignments than alignment γ_2. Alignments with the minimal costs correspond to *optimal alignments* that give the closest explanations of log traces in terms of modeled behavior. Note that there could be multiple optimal alignments for a particular log trace. For example, alignment γ_1 and γ_3 are both optimal alignments of trace t_3 under the standard cost assignment. Furthermore, optimal alignments are only optimal with respect to the given cost assignment. For example, alignment γ_1 would cease to be the optimal alignment if model moves and log moves of activity g are assigned a cost of 2 (i.e., $cost(\gamma_1) = 4$) to reflect that having deviations at the decision part of the process is quite severe. In practise, optimal alignments can be automatically found by finding the cheapest complete model trace in the synchronous product of the log trace and model using heuristic algorithms with proven optimality guarantees, e.g., the A* algorithm [van der Aalst et al., 2012].

Alignments can also be used to compute conformance metrics with respect to the different quality dimensions.

5.3 Cost-based fitness metric

The fitness of a log trace and a model can be quantified by comparing the cost of an optimal alignment with the worst case scenario cost [Adriansyah, 2014]. In the worst scenario, the log trace is completely unfitting with the model. A default alignment between the two can be computed by assigning all the steps in the log trace as log moves and all the steps in the complete model trace as model moves.

$$\gamma_5 = \begin{array}{|c|c|c|c|c|c|c|} a & b & c & d & e & f & h \\ \hline a & b & c & d & e & f & h \end{array} \qquad \gamma_6 = \begin{array}{|c|c|c|c|c|c|c|} a & c & b & \gg & e & f & h \\ \hline a & c & b & d & e & f & h \end{array}$$

$$\gamma_7 = \begin{array}{|c|c|c|c|c|c|c|c|} a & b & d & c & \gg & g & e & h \\ \hline a & b & d & c & e & g & \gg & h \end{array}$$

Fig. 13 Optimal alignments between trace t_1, t_2, t_3 in log L_1 and model M_1 in Figure 6

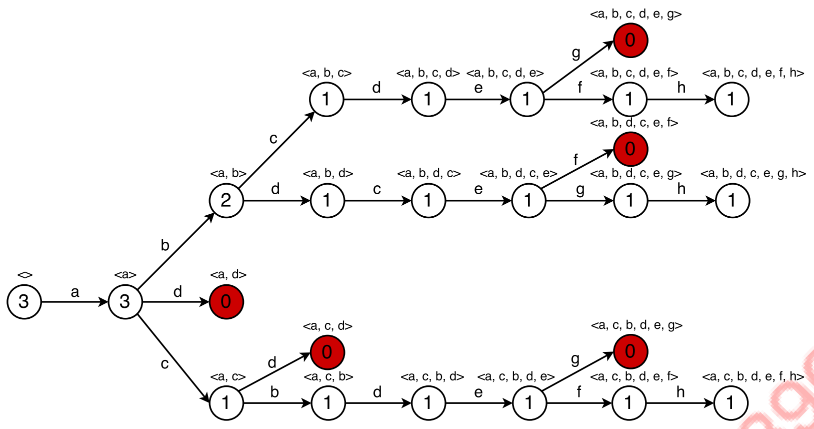


Fig. 14 Prefix automaton $\mathcal{A}_1$ of alignments between log L_1 and model M_1 enhanced with model behavior

Since the optimal alignment minimizes the total alignment cost, the least costly complete model trace is used. Figure 12 shows the default alignment between trace t_3 and model M_1 under the standard cost assignment. The top row (ignoring $\gg$) yields trace t_3 and the bottom row (ignoring $\gg$) yields a complete model trace $\langle a, b, d, c, e, g, h \rangle$. The cost-based fitness of trace t_3 can be computed as:

$$\begin{aligned} \mathit{fitness}(t_3, M) &= 1 - \frac{\mathit{cost}(\mathit{align}(t_3, M))}{\mathit{cost}(\mathit{align}_{\mathrm{default}}(t_3, M))} \\ &= 1 - \frac{0+0+0+0+1+0+1+0}{1+1+1+1+1+1+1+1+1+1+1+1+1+1} \\ &= 1 - \frac{2}{14} = 0.857 \end{aligned}$$

where a fitness value of 1.0 means that the model and log trace are perfectly fitting.

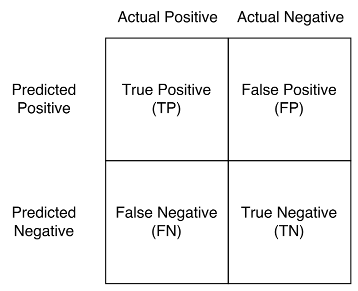
5.4 Escaping arc precision

Precision based on escaping arcs can also be computed using alignments [Adriansyah et al., 2012]. As previously mentioned, an imprecise model allows unobserved behavior, i.e., underfitting. For example, consider the Petri net model M_1 in Figure 6 and the optimal alignments (under the standard cost assignment) between model M_1 and log L_1 in Figure 13. Clearly, model M_1 is not perfectly precise as it allows for behavior that is not observed in log L_1. According to model M_1, activity b, c, and d can be executed in parallel following the execution of activity a. However, none of the log traces execute activity d after activity a. This imprecision in the model can be quantified by constructing a prefix automaton using the model projection of the alignments, i.e., the bottom row of the alignments. As previously presented, model projections of alignments explain potentially unfitting log traces in terms of modeled behavior so that they can be replayed on the process model. Figure 14 illustrates the constructed prefix automaton $\mathcal{A}_1$ for the alignments between log L_1 and model M_1 (ignoring the circles highlighted in red for now). Each prefix of the model projections of the alignments identifies a state (represented as circles) and the number in the states corresponds to the weight. For example, the state $\langle a \rangle$ has a weight of 3 because it appears three times in the model projections (all three alignments start with activity a). On the other hand, the state $\langle a, c, b \rangle$ is only present in alignment γ_6 and therefore has a weight of 1. The states of automaton $\mathcal{A}_1$ represent states reached by the model during the execution of the log. For any particular state in automaton $\mathcal{A}_1$, there might be activities that are enabled by the model but are not observed in the log execution. These activities indicate imprecisions of the model and are called *escaping arcs* of the model. Escaping arc states (represented as circles highlighted in red) are added to automaton $\mathcal{A}_1$ by replaying the automaton onto the model and checking for enabled activities at each state. For example, at state $\langle a \rangle$ (i.e., after firing activity a), activity b, c, and d are enabled as shown in Figure 8. However, the prefix $\langle a, d \rangle$ was not observed in the construction of automaton $\mathcal{A}_1$ using log L_1. This means that there is an escaping arc from state $\langle a \rangle$ to state $\langle a, d \rangle$ and this is added to the automaton by the state highlighted in red. The rest of the escaping arcs can be added in a similar way.

With the constructed prefix automaton, escaping arc precision can be computed by comparing the number of escaping arcs with the number of allowed arcs for all states:

$$\begin{aligned} \mathit{precision}(\mathcal{A}_1) &= 1 - \frac{\sum_{s \in S} \omega(s) \cdot |\mathit{esc}(s)|}{\sum_{s \in S} \omega(s) \cdot |\mathit{mod}(s)|} \\ &= 1 - \frac{3 \cdot 0 + 3 \cdot 1 + 2 \cdot 0 + \ldots + 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0}{3 \cdot 1 + 3 \cdot 3 + 2 \cdot 2 + \ldots + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 1} \\ &= 1 - \frac{6}{36} = 0.833 \end{aligned}$$

where S is the set of states in automaton $\mathcal{A}_1$, $\omega(\cdot)$ maps a state $s \in S$ to its weight, $\mathit{esc}(\cdot)$ maps a state $s \in S$ to its set of escaping arcs states, and $\mathit{mod}(\cdot)$ maps a state $s \in S$ to its set of allowed states. A precision value of 1.0 indicates perfect precision, i.e., the model only allows observed behavior and nothing else.



	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Fig. 15 Confusion matrix

5.5 Artificial negative events

Another approach to measure precision is through artificial negative events. Artificial negative events are induced by observing events that did not occur in the event log. These unobserved events (i.e., negative events) give information about things that are not allowed to occur in the process. Assuming that the event log gives a complete view of the process (i.e., a log completeness assumption), the precision of the process model can be computed using artificial negative events and the concepts of precision and recall in data mining.

Artificial negative events can be induced by grouping similar traces and then observing the events that did not occur for every event in the traces. Under the log completeness assumption, this means that these unobserved events are negative events that are not allowed to happen by the process [Goedertier et al., 2009].

The process model can then be compared with the log by treating the model as a predictive model. For a given incomplete event sequence (i.e., an unfinished process instance), activities that are permitted by the model and observed in the log are classified as true positives (TP). Activities that are permitted by the model but are induced as negative events from the log are classified as false positive (FP). Activities that are not permitted by the model but observed in the log are classified as false negative (FN). Finally, activities that are not permitted by the model and are induced as negative events from the log are classified as true negative (TN). As shown in Figure 15, precision and recall can be computed using a confusion matrix. Specifically, the precision of the positive class can be computed as:

$$\mathit{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

The computed precision value corresponds to the precision of the three quality dimensions in process mining since it refers to the proportion of modeled behavior that is observed in the log.

The use of artificial negative events can also be extended to quantify generalization and to compute a precision metric that is more robust against less complete event logs. This is achieved by extending the artificial negative event induction strategy to assign weights to the induced negative events [vanden Broucke et al., 2014].

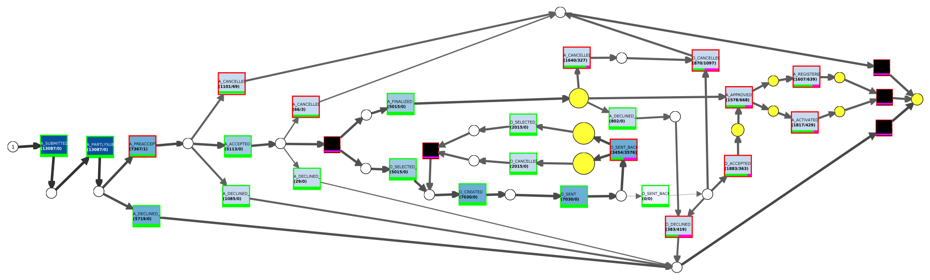


Fig. 16 Process model projected with alignment results

6 Examples of Application

All of the conformance checking techniques presented in the previous section have been implemented and are applicable to most real-life scenarios. In the following, the A* cost based alignment technique is applied to a real life dataset to illustrate how conformance checking can be applied to gain insights about a process. This example utilizes the data presented in the ”Conformance Checking: What does your process do when you are not watching?” tutorial by de Leoni, van Dongen, and Munoz-Gama at the ”15th International Conference on Business Process Management (BPM17)”.

As previously presented, a process model and an event log are required to perform conformance checking. The real-life event log is taken from a Dutch Financial institute and is of an application process for a personal loan or overdraft within a global financial organization [van Dongen, 2012]. This means that each case in the log records the occurred events of a particular loan application. The log contains some 262,200 events in 13,087 cases. Apart from some anonymization, the data is presented as it is recorded in the financial institute. The log is merged from three intertwined sub processes so that the originating sub process of each event can be identified by the first letter of the activity recorded by the event. In this example, the log is filtered so that it only contains events from two of the sub processes: the process which records the state of the application (identifiable by “A_”) and the process which records the state of an offer communicated to the applicant (identifiable by “O_”). The model has been created with the help of domain experts and can be assumed to be a realistic representation of the underlying process.

Figure 16 shows the process model projected with the computed alignment results to allow a visual diagnosis of the conformance results. For each transition, there is an error bar to show the distribution of synchronous moves (green) and model moves (pink) for the transition. For example, there are 383 synchronous moves and 419 model moves related to transition *O_DECLINED*. The occurrence and amount of log moves are indicated by highlighting places in yellow and the size of the highlighted places. Observing the model, one can note that transition *O_SENT_BACK* is associated with a large amount of model moves. This transition is quite an important part of the process as it corresponds to the event where the financial institute receives a reply from the applicant after a loan offer is made. A model move of *O_SENT_BACK* in a log trace means that the system did not register a reply from the applicant regarding a made offer as required by the process for the corresponding loan application. Upon investigation of cases with a model move in *O_SENT_BACK* (e.g., the case with caseId 174036) would show that there are cases for which an offer was created, sent and accepted without having received a reply from the corresponding applicant. Whether it was due to a system error, an employee's mistake or at worst a fraudulent case, clearly it is in the financial institute's best interest to investigate the root cause of this conformance issue.

7 Future Directions for Research

While there have been significant advances in the research of conformance checking over the recent years, there are still many open challenges and research opportunities. Some of them include:

7.1 Conformance Dimensions

The proposed three quality dimensions (fitness, precision, and generalization) have been widely accepted but there is still a need for further understanding on how to interpret and quantify them through metrics. Furthermore, conformance can be extended beyond the current three dimensions, e.g., log completeness to quantify whether if the observed event data gives the full picture of the underlying process [Janssenswillen et al., 2017].

7.2 Big Data and Real Time

Process mining techniques and tools are getting applied to larger and more complex processes. This means that they have to be scalable to handle the increased size and complexity. In fact, much of the recent research efforts in conformance checking have been focused on this issue. Related research lines include decomposed conformance checking [Munoz-Gama, 2016] and online conformance checking for event streams [Burattin, 2015].

7.3 Conformance Diagnosis and Process Model Repair

It is not enough to just identify conformance issues; good diagnostic and visualization tools are crucial in helping the analyst identify and understand the root causes of the conformance issues. While there has been work done in this aspect of conformance checking, e.g., [Buijs & Reijers, 2014,Munoz-Gama et al., 2014], there is much more to be done to provide better conformance diagnosis technology, e.g., new techniques and user study. Finally, once the differences between the model and the log have been diagnosed, the user may wish to repair the model in order to fix such differences, and achieve a model that better describes the real process executed. This topic is extensively covered in the *Process Model Repair* entry of this encyclopedia.

References

Adriansyah, 2014. Adriansyah, Arya 2014. Aligning Observed and Modeled Behavior. PhD thesis, Eindhoven University of Technology.

Adriansyah et al., 2012. Adriansyah, Arya, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, & Wil M. P. van der Aalst 2012. Alignment Based Precision Checking. In Rosa, Marcello La, & Pnina Soffer (eds), Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers, volume 132 of *Lecture Notes in Business Information Processing*, pages 137–149. Springer.

Buijs & Reijers, 2014. Buijs, Joos C. A. M., & Hajo A. Reijers 2014. Comparing Business Process Variants Using Models and Event Logs. In Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings, pages 154–168.

Burattin, 2015. Burattin, Andrea 2015. Process Mining Techniques in Business Environments - Theoretical Aspects, Algorithms, Techniques and Open Challenges in Process Mining, volume 207 of *Lecture Notes in Business Information Processing*. Springer.

Goedertier et al., 2009. Goedertier, Stijn, David Martens, Jan Vanthienen, & Bart Baesens 2009. Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research, 10:1305–1340.

Janssenswillen et al., 2017. Janssenswillen, Gert, Niels Donders, Toon Jouck, & Benoît Depaire 2017. A comparative study of existing quality measures for process discovery. Inf. Syst., 71:1–15.

Munoz-Gama, 2016. Munoz-Gama, Jorge 2016. Conformance Checking and Diagnosis in Process Mining - Comparing Observed and Modeled Processes, volume 270 of *Lecture Notes in Business Information Processing*. Springer.

Munoz-Gama et al., 2014. Munoz-Gama, Jorge, Josep Carmona, & Wil M. P. van der Aalst 2014. Single-Entry Single-Exit decomposed conformance checking. Inf. Syst., 46:102–122.

Rozinat & van der Aalst, 2008. Rozinat, Anne, & Wil M. P. van der Aalst 2008. Conformance checking of processes based on monitoring real behavior. Inf. Syst., 33(1):64–95.

van der Aalst, 2013. van der Aalst, Wil M. P. 2013. Mediating between modeled and observed behavior: The quest for the "right" process: Keynote. In RCIS, pages 1–12. IEEE.

van der Aalst, 2016. van der Aalst, Wil M. P. 2016. Process Mining - Data Science in Action. Springer.

van der Aalst et al., 2012. van der Aalst, Wil M. P., Arya Adriansyah, & Boudewijn F. van Dongen 2012. Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery, 2(2):182–192.

van Dongen, 2012. van Dongen, B.F. 2012. BPI Challenge 2012.

vanden Broucke et al., 2014. vanden Broucke, Seppe K. L. M., Jochen De Weerdt, Jan Vanthienen, & Bart Baesens 2014. Determining Process Model Precision and Generalization with Weighted Artificial Negative Events. IEEE Trans. Knowl. Data Eng., 26(8):1877–1889.