# Orientation and conformance: A HMM-based approach to online conformance checking

Wai Lam Jonathan Lee [a,1,*], Andrea Burattin[b], Jorge Munoz-Gama[a], Marcos Sepúlveda[a]

[a]*Department of Computer Science, School of Engineering, Pontificia Universidad Católica de Chile, Vicuña Mackenna 4860, Macul, Santiago, Chile*
[b]*Department of Applied Mathematics and Computer Science, Technical University of Denmark*

## Abstract

*Online conformance checking* comes with new challenges, especially in terms of time and space constraints. One fundamental challenge of explaining the conformance of a running case is in balancing between making sense at the process level as the case reaches completion and putting emphasis on the current information at the same time. In this paper, we propose an online conformance checking framework that tackles this problem by incorporating the step of estimating the "location" of the case within the scope of the modeled process before conformance computation. This means that conformance checking is broken down into two steps: orientation and conformance. The two steps are related: knowing "where" the case is with respect to the process allows a conformance explanation that is more accurate and coherent at the process level and such conformance information in turn allows better orientations. Based on Hidden Markov Models (HMM), the approach works by alternating between orienting the running case within the process and conformance computation. An implementation is available as a Python package and experimental results show that the approach yields results that correlate with prefix alignment costs under both conforming and non-conforming scenarios while maintaining constant time and space complexity per event.

*Keywords:* Conformance checking, online processing, Business Process

[*]Corresponding author
*Email addresses:* walee@uc.cl (Wai Lam Jonathan Lee ), andbur@dtu.dk (Andrea Burattin), jmun@uc.cl (Jorge Munoz-Gama), marcos@ing.puc.cl (Marcos Sepúlveda)
[1]Majority of the work was completed during his research stay at DTU

## 1. Introduction

Process mining is a research discipline that sits between Business Process Management (BPM) and Data Science [1]. Traditionally, BPM focuses on process models and the various aspects, e.g., design, execution, and optimization, of business processes rather than dealing with the generated event data [2]. In contrast, data-oriented analysis tends to look at particular decisions or patterns and typically does not consider end-to-end processes [1]. Process mining connects the two disciplines by adapting data-oriented analysis to improve processes. Event data from the event logs of information systems, such as ERP (Enterprise Resource Planning) systems (SAP, Oracle, etc.) and BPM (Business Process Management) systems (Pegasystems, Bizagi, Appian, BPM BPM, etc.), is used in process mining as a starting point to "discover, monitor, and improve real processes" [1].

There are three main areas in process mining [1]. First, *process discovery* takes an event log and produces a model through different discovery algorithms [3]. Second, *conformance checking* relates a process model with event data of the same process to identify commonalities and discrepancies [4]. Third, *enhancement* enriches an existing process model with information extracted from event data [5]. In this paper, we focus on **conformance checking**. There are many reasons for performing conformance checking. For example, conformance checking can support the audit process of organizations so that they can strive towards Continuous Auditing [6]. Moreover, organizations might also want to know whether their set procedures are meeting the requirements of reality. Through identifying discrepancies between the recorded event data and the process model, conformance checking can aid process re-design and modification of the BPM lifecycle [2]. Lastly, conformance checking plays a key role in the other process mining areas. For example, conformance checking is used to compare existing discovery algorithms [7], to recommend discovery techniques [8], and to support discovery algorithms [3, 9].

However, most existing conformance checking techniques require the trace of events to correspond to a completed case. This means that these techniques target offline scenarios and do not typically cater for online contexts where it is desirable to raise alerts as soon as a significant deviation is observed for cases that have not reached completion. Moreover, due to the continuous increase in recorded data,

it can be infeasible for organizations to store data for offline processing. For example, Wal-Mart is estimated to collect more than 2.5 petabytes of data every hour from its customer transactions [10]. As such, in recent years, a new set of algorithms [11, 12, 13] has been proposed for *online* scenarios in which we assume to have an *event stream* as input so that each item relates to an observed event for a case. In this paper, we propose a novel online approach which performs conformance checking on an event stream with constraints on memory and time.

There are several works on online conformance checking [11, 12, 13], but there still exists areas for improvement. For example, prefix alignments [11] and a similar approach based on enriching a transition system using alignment concepts [12] have difficulties handling warm start scenarios. Another approach [13] that performs conformance checking on behavioral patterns can lose information due to its abstraction.

In this paper, we present a framework based on Hidden Markov Models (HMM) that balances between making sense at the process level as the case reaches completion and putting emphasis on the current information at the same time. As new events come in for running cases, the model alternates between localizing the running case within the reference model using the observed event and computing conformance from such estimated position. Different to the assumption of the standard HMM, both the previous state and observation can influence the next state due to non-conformance. This is modeled by conditioning state transition and observation probabilities by both the previous state and observation. Furthermore, rather than deciding beforehand the effects of non-conformance, an Expectation-Maximization (EM) algorithm is applied to compute the parameters from past data.

The remainder of the paper is structured as follows: Section 2 motivates the need for the proposed framework. Section 3 presents the preliminaries of the paper. Section 4 presents the proposed technique. Section 5 details the parameter computation and estimation of the proposed technique. Section 6 presents the experimental evaluation of the proposed technique. Section 7 illustrates the application of the proposed technique on a real-life dataset. Section 8 presents the related work. Finally, Section 9 concludes the paper.

## 2. Motivation

Consider the running example log and model in Figure 2 and Figure 1. It presents the billing process of a hospital based on a real-life dataset [14]. As shown by the process model, ideally, each instance of the process corresponds

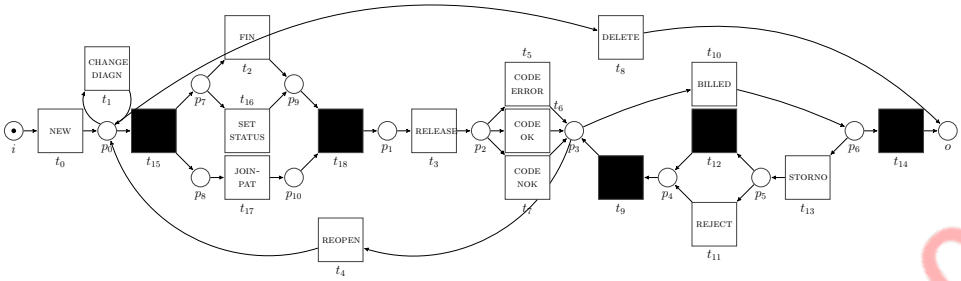



Figure 1: Running example: Petri net model

$\sigma_0 = \langle \text{NEW}, \text{FIN}, \text{JOIN-PAT}, \text{RELEASE}, \text{CODE OK}, \text{BILLED} \rangle$,
$\sigma_1 = \langle \text{NEW}, \text{FIN}, \text{JOIN-PAT}, \text{RELEASE}, \text{CODE OK}, \text{BILLED}, \text{STORNO}, \text{REJECT}, \text{BILLED} \rangle$,
$\sigma_2 = \langle \text{NEW}, \text{CHANGE DIAGN}, \text{FIN}, \text{JOIN-PAT}, \text{RELEASE}, \text{CODE NOK}, \text{CODE OK}, \text{BILLED} \rangle$,
$\sigma_3 = \langle \text{NEW}, \text{FIN}, \text{JOIN-PAT}, \text{RELEASE}, \text{CODE NOK}, \text{REOPEN} \rangle$,
$\sigma_4 = \langle \text{NEW}, \text{CHANGE DIAGN}, \text{FIN}, \text{RELEASE}, \text{CODE OK}, \text{MANUAL}, \text{RELEASE}, \text{CODE OK}, \text{REOPEN}, \text{DELETE} \rangle$,
$\sigma_5 = \langle \text{RELEASE}, \text{CODE OK}, \text{BILLED} \rangle$,
$\sigma_6 = \langle \text{NEW}, \text{FIN}, \text{RELEASE}, \text{CODE OK}, \text{BILLED} \rangle$,
$\sigma_7 = \langle \text{NEW}, \text{FIN}, \text{JOIN-PAT}, \text{SET STATUS}, \text{JOIN-PAT}, \text{RELEASE}, \text{CODE OK}, \text{BILLED} \rangle$

Figure 2: Running example: Traces

to the billing process of a particular patient. For example, trace $\sigma_0$ in Figure 2 illustrates that after undergoing different medical services, these services are collected in a billing package and the package is released so that the patient can be billed. However, different scenarios can potentially occur during the process. For example, an invoice has to be sent to the insurance company of the patient and the invoice can potentially be rejected (REJECT) as shown by trace $\sigma_1$.

Trace $\sigma_5$ illustrates an example for which the trace is fully compliant with the model if one considers that it is starting from the middle of the process at the release of the billing package to the insurance company (RELEASE). This is known as a warm start scenario in which we are observing an already ongoing process instance. As mentioned in [13], alignment-based techniques such as prefix alignments [11] would simply decrease the conformance values.

The behavioral pattern based approach presented in [13] is able to handle such warm start scenarios. However, the proposed instantiation of using weak ordering relations leads to a less robust approximation that might not detect some

conformance issues. Trace $\sigma_7$ presents conformance issues with activities FIN, JOIN-PAT, and SET STATUS. However, the online framework using weak ordering relations would observe the following weak ordering relations: $(\prec, \text{NEW}, \text{FIN})$, $(\prec, \text{FIN}, \text{JOIN-PAT})$, $(\prec, \text{JOIN-PAT}, \text{SET-STATUS})$, $(\prec, \text{SET-STATUS}, \text{JOIN-PAT})$, $(\prec, \text{JOIN-PAT}, \text{RELEASE})$, $(\prec, \text{RELEASE}, \text{CODE OK})$ and $(\prec, \text{CODE OK}, \text{BILLED})$. Since all of the observed relations are compliant with the modeled weak ordering relations, the behavioral pattern based approach would yield perfect conformance, completeness and confidence.

As such, while (prefix) alignment techniques are robust when evaluating full traces and represents how conformance is understood in the literature, they cannot handle the warm start scenario. Furthermore, their time and memory complexity often exceed the requirement for stream processing.

To address both the need of putting emphasis in the current information and being robust enough to detect conformance issues at the process level, our proposed approach alternates between orienting the running process instance within the process and conformance computation. As shown later in the paper, the approach represents process instances by a probabilistic estimation of their locations via states within the process model. This allows the framework to meet the requirement for stream processing, handle warm start scenarios and detect conformance issues that require evaluating the instance at the process level. For trace $\sigma_5$, our proposed approach would first orientate the trace to the middle of the process with the observation of RELEASE so that conformance is computed with respect to the updated location. Since the estimated location of a process instance is updated with the incoming events, the previous execution of FIN and JOIN-PAT in trace $\sigma_7$ would update the estimated location of the trace so that the execution of SET STATUS and an additional JOIN-PAT would not be compliant with the model.

## 3. Preliminaries

This section introduces basic concepts related to process models and event stream. Processes are depicted using process models. There are many different process modeling languages, e.g., the Business Process Modeling Notation (BPMN), Event-Driven Process Chains (EPCs), Unified Modeling Language (UML) Activity diagrams, Yet Another Workflow Language (YAWL) and others [2]. In this paper, we assume that the model can be represented as a bounded Petri net [15], possibly through a translation from other process modeling languages such as transforming a BPMN model into Petri nets [16].

**Definition 1 (Petri net).** *A Petri net is a tuple $N = (P,T,F)$ with $P$ the set of places, $T$ the set of transitions, $P \cap T = \emptyset$ and $F = (P \times T) \cup (T \times P)$ the set of arcs, which is sometimes referred to as the flow relation.*

In the field of process mining, often times labeled Petri nets are used where there is an additional labeling function $l \in T \nrightarrow A$ that maps transitions to activity labels $a \in A$. Figure 1 illustrates a labeled Petri net. As shown, transitions can be either visible or invisible. Visible transitions are ones that can be mapped to an activity label while invisible transitions cannot be mapped to any activity label and is often simply mapped a $\tau$ label. A fundamental concept of the Petri net notation is the idea of markings which denotes the state of a Petri net. A marking $M$ is a multiset of places, i.e., $M \in \mathscr{B}(P)$. For example, the Petri net in Figure 1 has a marking of $[i]$, and is visualized by a single token in the place $i$. Furthermore, firing enabled transitions can bring a Petri net from one marking to another. A transition $t \in T$ is *enabled* by a marking $M$ if and only if each of its input places contains at least one token in $M$. An enabled transition may *fire* by removing one token from each of the input places and producing one token at each of the output places. For example, firing the enabled transition $t_0$ from marking $[i]$ would yield marking $[p_0]$.

This paper works on the problem of performing conformance checking on streams of event data. Similar to [13], each unit of the stream corresponds to an observable unit from which related process information can be extracted.

**Definition 2 (Observable unit).** *Let $\mathscr{C}$ denote the set of case ids, and let $\mathscr{A}$ denote the set of activities. An observable unit $o = (c,a) \in \mathscr{C} \times \mathscr{A}$ is a pair describing an activity $a$ observed in context of case id $c$. The universe of all possible observable units is defined as $\mathscr{O} = \mathscr{C} \times \mathscr{A}$.*

The activities of observable units correspond to observations of fired transitions in the corresponding Petri net model. Projection operators can be used to extract the case id and the activity, i.e., given $o = (c,a)$, $o \restriction_c = c$ and $o \restriction_a = a$.

**Definition 3 (Event stream).** *Given the universe of observable units $\mathscr{O} = \mathscr{C} \times \mathscr{A}$, an event stream is defined as an infinite sequence of observable units: $S : \mathbb{N}_{\geq 0} \rightarrow \mathscr{O}$.*

As such, an event stream can be seen as an unbounded sequence of observable units for which the sequence order corresponds to the chronological order of the corresponding events. For the rest of the paper, where it is clear that we are referring to an event stream of only one case, we will directly refer to the corresponding activities rather than apply the projection operators for each observable unit.

## 4. Proposed technique

The proposed technique is based on representing the conformance checking scenario using a modified Hidden Markov Model (HMM). Given a process model and a stream of events, the approach represents process instances by a probabilistic estimation of their locations (state) within the process model and performs two tasks per event: orientation and conformance. This online procedure is supported by an offline component that is performed over past data as illustrated by the diagram in Figure 3. In this section, we present the online procedure which is initiated by an observable unit from the event stream, denoted by the grayed box, and then passed onto the two tasks of orientation and conformance, denoted by the subsequent three boxes.

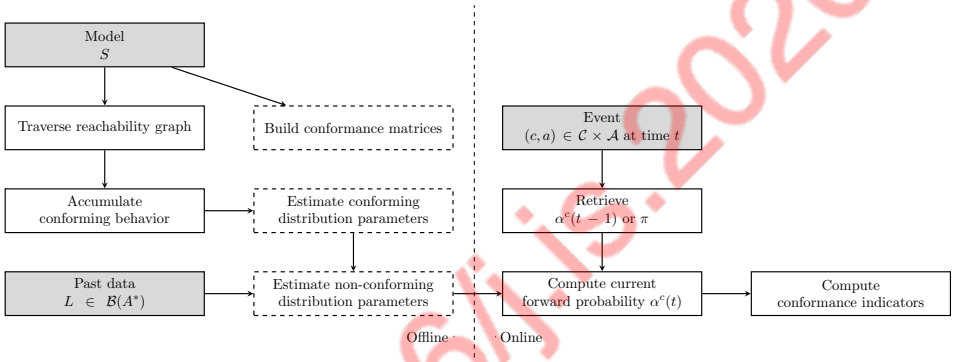



Figure 3: Overview of the proposed approach. The online component is presented in Section 4 and the offline component is presented in Section 5.

Given the natural connection between Petri nets and HMMs, we use markings as possible states of a process instance. However, note that we can take further approximations by partitioning the set of possible markings into a smaller set of classes.

### *4.1. Overview*

Algorithm 1 presents an overview of the online procedure. When a new observable unit comes in from the event stream (Line 2), it first goes through the orientation phase. In this phase, either the previous state estimation or an initial state estimation of the case is retrieved depending on whether the corresponding case has been previously observed (Line 3). Then, state estimation is performed,

taking into account the new information given by the new observable unit (Line 4). Afterwards, it moves onto the conformance phase where various conformance indicators are computed (Line 5). Lines 2 – 5 correspond to the four boxes in the previous flow diagram in Figure 3.

**Algorithm 1:** Overview of online conformance computation

```
   Input: S: stream of observable units
1  forever do
2      (c,a) ← observe(S) ;                    // New caseid-activity pair from the stream
       // Phase 1: Orientation
3      state_c ← get_either_previous_or_initial_state_estimation(c);
4      state_c ← update_state_estimation(state_c, a);
       // Phase 2: Compute online conformance values
5      compute_conformance_indicators(state_c, a);
```

### *4.2. Walk-through of an example*

Here we walk through a case corresponding to trace $\sigma_6$ in Figure 2 to give some intuition. At the start of the case, one could assume that the case would be at the initial marking ($[i]$) so that its initial state estimation is a one-hot vector $\mathbf{1}_{[i]}$ with $n$ components where $n$ corresponds to the number of possible locations. Then, suppose we observe the first event (NEW). Clearly, this is described by the model and corresponds to firing transition $t_0$. This yields a perfect conformance value. Similarly, the next event (FIN) puts the case at the state estimation of $\mathbf{1}_{[p_0]}$ (marking $[p_0]$ eventually enables activity FIN by firing invisible transitions) with perfect conformance value.

Suppose we observe the next event (RELEASE). Clearly, this is not conforming to the modeled behavior. To yield a plausible explanation for the non-conforming behavior, we perform two tasks to estimate the current state. First, given the previous event (FIN) and the assumption that it was perfectly conforming, the case must be currently at marking $[p_8, p_9]$. Second, checking the possible observations at marking $[p_8, p_9]$ would tell us that RELEASE does not correspond to an enabled transition. Incorporating both the state transition and the current event, we would estimate that the current state to be at $\mathbf{1}_{[p_8,p_9]}$ and that the event is completely non-conforming to the model. However, since the current observation is not described by the modeled behavior, we can no longer rely on the model to estimate the next state following the event (RELEASE).

Suppose we observe the next event (CODE OK). Similar to the previous event, we first estimate its current state given its previous event (RELEASE) and conformance. Since the previous conformance shows to be completely non-conforming

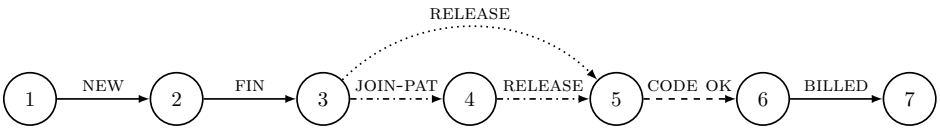



Figure 4: State estimation taken throughout trace $\sigma_6$ in Figure 2. Line style indicates the conformance explanation of the corresponding execution where a solid line indicates complete conformance, a dotted line indicates complete lack of conformance, a dashed line indicates moderate conformance, and a dash-dotted line indicates a possible model execution that non-conforming observation might be referring to.

to the modeled behavior, we cannot use the model as the basis for the state estimation. Instead, we assume to have a categorical distribution that estimates the next state given the previous event and state. This distribution is learnt from past data using a procedure that is presented later in the paper. Suppose that in the past, there are similar cases which later turned out to have skipped the execution of activity JOIN-PAT. This would encourage us to estimate the current state to be at marking $[p_2]$. Second, we incorporate the current event (CODE OK) in our estimation. Since the transition corresponding to activity CODE OK is in fact enabled at marking $[p_2]$, observing CODE OK reinforces our estimation of the current state to be at marking $[p_2]$. In contrary, if the observed event corresponds to activity RELEASE, then we might reallocate some probability mass from marking $[p_2]$ to marking $[p_1]$ rather than reinforcing our estimation on marking $[p_2]$. In this case, reinforcing our state estimation on marking $[p_2]$ yields a high conformance on the current event (CODE OK). In fact, the following event BILLED will further confirm our estimation.

Figure 4 illustrates the state estimations taken throughout the case. The line style of the arrows indicates the conformance explanation of the corresponding execution. For example, since we assume that the initial state is $\mathbf{1}_{[i]}$, the first event (NEW) and the second event (FIN) is completely conforming with the modeled behavior. In contrary, the third event (RELEASE) is completely non-conforming since it skipped over the activity JOIN-PAT. As previously explained, the event RELEASE brings the state estimation to allocate a high probability on marking $[p_2]$ where the transition corresponding to activity CODE OK is enabled. This leads to a value in between complete conformance and non-conformance. The last event has a high conformance value since the corresponding transition is enabled at the previous state estimation.

*4.3. HMM-based conformance checking*

As previously explained, the current event and an estimation of the case's current state are required to compute the conformance of the current observation. We define a conformance function as a mapping from a tuple of the current state estimation and observation to a value between 0 and 1.

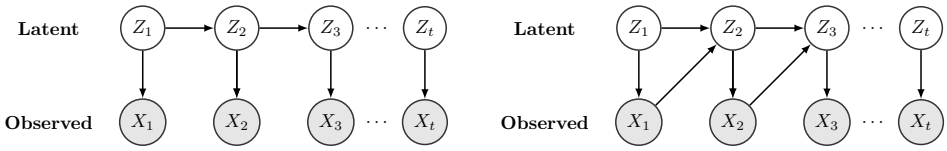
**Definition 4 (Conformance function).** *Let $Z$ denote the set of possible states. $conf : \mathbb{R}^{|Z|} \times A \to [0,1]$ is a function that maps a state estimation and an activity execution to a value between 0 and 1 so that a value near 0 indicates complete non-conformance and a value near 1 indicates complete conformance.*

For the model in Figure 1, $conf(\mathbf{1}_{[i]}, \text{NEW}) = 1$[2] yields complete conformance on the one-hot vector with 1 at the initial marking since transition $t_0$ is enabled.

As recalled, we perform a state estimation each time we observe a new event. Moreover, under perfect conformance, the current state is dependent solely on the previous state. We extend this assumption to the scenario of non-perfect conformance to meet the computational constraints of online processing. This means that the chain of state estimations corresponds to a Markov chain. However, under the scenario of non-perfect conformance, one cannot directly observe a case's state (hence the need for *state estimations*). Instead, the case's state is hidden or so-called *latent* so that we have to infer it from the observed events. The discrete progression of cases and the dependence between states of different time steps yields a modified HMM.

**Definition 5 (HMM for conformance checking (HMMConf)).** *Let $Z$ be a set of latent states and let $X$ be a set of observations. Let $W_{x_k} \in \mathbb{R}^{|Z| \times |Z|}$ be a state-transition probability matrix given observation $x_k \in X$ where $1 \leq k \leq |X|$, $V \in \mathbb{R}^{|Z| \times |X|}$ be an emission probability matrix under conforming conditions and let $W^d_{x_k}$ and $V^d$ be their counterparts under non-conforming conditions. Let $conf : \mathbb{R}^{|Z|} \times \mathscr{A} \to [0,1]$ be a conformance function and let $\pi \in \mathbb{R}^{|Z|}$ be the initial state distribution.*

[2] $\mathbf{1}_{[i]}$ is a one-hot vector with $n$ components where $n = 7$ is the number of states in the reachability graph of the Petri net in Figure 1. $[i]$ is a multiset that denotes the state with 1 in $\mathbf{1}_{[i]}$; all other states has 0.




(a) HMM (b) HMMConf

Figure 5: Graphical representation of a standard HMM and the proposed HMMConf

$(W_{x_1}, \ldots, W_{x_{|X|}}, W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V, V^d, conf, \pi)$ *is a modified hidden Markov model with states dependent on previous observations [17] so that:*

$$P(x_t \,|\, z_t, \ldots, z_1, x_{t-1}, \ldots, x_1) = P(x_t \,|\, z_t), 1 \leq t \leq T \tag{1}$$

$$P(z_t \,|\, z_{t-1}, \ldots, z_1, x_{t-1}, \ldots, x_1) = P(z_t \,|\, z_{t-1}, x_{t-1}), 2 \leq t \leq T \tag{2}$$

*where $T$ denotes the largest possible time unit which can be unbounded in the context of an event stream.*

Figure 5b presents a graphical representation of HMMConf. Compared with the graphical representation of a standard HMM in Figure 5a, one can observe that the difference between the two is in the extra dependency between the current state $Z_t$ and the previous observation $X_{t-1}$. In a standard HMM, the current state $Z_t$ only depends on the previous state $Z_{t-1}$. In contrast, the current state $Z_t$ in the proposed HMMConf depends on both the previous state $Z_{t-1}$ *and* the previous observation $X_{t-1}$. Referring to trace $\sigma_6$ in the running example in Figure 2, suppose that $X_1 = \text{NEW}$ and $z_1$ as the corresponding state, then the state $Z_2$ can computed directly using $z_1$ for the standard HMM. However, as shown in more detail later in the paper, calculating the state $Z_2$ requires both the previous state $z_1$ and the previous observation $X_1$ for the proposed HMMConf. As previously explained, the latent states are the possible markings in the Petri net model and the observations are the activities. We now present the state-transition probability (Eq. 2) and the observation probability (Eq. 1).

**Definition 6 (Conformance dependent state-transition probability).** *Let $W_{x_k}$ be the state-transition probability matrix given observation $x_k \in X$ where $1 \leq k \leq |X|$, $W^d_{x_k}$ be the probability matrix for the deviating behavior, and $\hat{\mathbf{z}}_{t-1}$ be an estimation of the state at time $t-1$. Let $i, j \in Z$ be latent states and let $1 \leq k \leq |X|$ indicate*

*an observation.*

$$w_{i,j}(k) = P(Z_t = j \,|\, Z_{t-1} = i, X_{t-1} = x_k)$$
$$\approx conf(\hat{\mathbf{z}}_{t-1}, x_k) W_{x_k,i,j} + [1 - conf(\hat{\mathbf{z}}_{t-1}, x_k)] W^d_{x_k,i,j}$$

This is an approximation of the latent state conditional probability because $W_{x_k}$ is a substochastic matrix; the final marking of a Petri net corresponds to an absorbing state that does not transition to other states once it is reached. To simplify the notations, we assume that the latent state and observation sets are ordered so that members can be referenced by their indices. Also, $z_{1:t} \equiv z_1, z_2, \ldots, z_{t-1}, z_t$. Component $i$ of the estimation of the previous state $\hat{\mathbf{z}}_{t-1}$ is computed as $P(Z_{t-1} = i \,|\, X_{1:t-1} = x_{1:t-1}) = \frac{\alpha_i(t-1)}{\sum_k \alpha_k(t-1)}$ where $\alpha_i(t-1) = P(X_{1:t-1} = x_{1:t-1}, Z_{t-1} = i)$ is the forward probability.

**Definition 7 (Conformance dependent observation probability).** *Let $V$ be the observation probability matrix, $V^d$ be the probability matrix for the deviating behavior, $j \in Z$ be a latent state, $x_k \in X$ be an observation such that $1 \leq k \leq |X|$, and $\hat{\mathbf{z}}_t$ be an estimation of the current state.*

$$v_j(k) = P(X_t = x_k \,|\, Z_t = j)$$
$$\approx conf(\hat{\mathbf{z}}_t, x_k) V_{j,k} + [1 - conf(\hat{\mathbf{z}}_t, x_k)] V^d_{j,k}$$

Similar to the state-transition probability, an absorbing state does not emit any observation so that typically not all the rows of $V$ sum to 1. Component $j$ of the estimation of the current state $\hat{\mathbf{z}}_t$ is computed as $P(Z_t = j \,|\, X_{1:t-1} = x_{1:t-1}) = \frac{\sum_i w_{i,j}(x_{t-1}) \alpha_i(t-1)}{\sum_k \alpha_k(t-1)}$.

### *4.4. Forward probability*

To compute the previous and current state estimations, Definition 6 and Definition 7 make use of the forward probability $\alpha_i(t) = P(X_{1:t} = x_{1:t}, Z_t = i)$. This is the joint probability of the latent state being $i$ and having observed $x_{1:t}$. Similar to traditional HMMs, this can be computed recursively by taking advantage of conditional independence using the conformance dependent state-transition probability, i.e., $P(Z_t | Z_{t-1}, X_{t-1})$, the conformance dependent observation probability, i.e., $P(X_t | Z_t)$, and the forward probability from the previous time step, i.e., $P(X_{1:t-1}, Z_{t-1})$ as

$$P(X_{1:t}, Z_t) = P(X_t | Z_t) \sum_{z \in Z} P(X_{1:t-1}, Z_{t-1} = z) P(Z_t | Z_{t-1} = z, X_{t-1})$$

Specifically, this is achieved by iterating over the all possible values of the previous state $Z_{t-1}$ and then using the chain rule to expand $P(X_{1:t}, Z_t, Z_{t-1} = z)$ so that

$$\begin{aligned}
P(X_{1:t}, Z_t) &= \sum_{z \in Z} P(X_{1:t}, Z_t, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(X_t, Z_t | X_{1:t-1}, Z_{t-1} = z) P(X_{1:t-1}, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(X_t | Z_t, X_{1:t-1}, Z_{t-1} = z) P(Z_t | X_{1:t-1}, Z_{t-1} = z) P(X_{1:t-1}, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(X_t | Z_t) P(Z_t | X_{1:t-1}, Z_{t-1} = z) P(X_{1:t-1}, Z_{t-1} = z) \\
&= P(X_t | Z_t) \sum_{z \in Z} P(Z_t | X_{1:t-1}, Z_{t-1} = z) P(X_{1:t-1}, Z_{t-1} = z)
\end{aligned}$$

Note that $P(X_t | Z_t, X_{1:t-1}, Z_{t-1} = z) = P(X_t | Z_t)$ because of the conditional independence as defined in Definition 5. This allows its extraction outside of the summation since it is not dependent on the values of the previous state $Z_{t-1}$. The initial forward probability $P(X_1 = x_1, Z_1 = j) = \pi_j v_j(x_1)$ is computed by multiplying an initial state estimation $\pi_j$ with the conformance dependent observation probability of $x_1$ given the initial state.

As previously mentioned, in online conformance checking, most cases might not have reached completion. This means that monitoring their conformance does not give a full picture. Similar to previous work [13], we include the concept of *completeness* to indicate whether the entire trace has been observed since the beginning.

### *4.5. Conformance metrics*

Figure 6 graphically illustrate the two distinct aspects of conformance that our proposed technique measures. *Conformance* is between the current observation and the state estimation of the corresponding case. *Completeness* indicates whether we are observing the complete trace by looking at previous observable units and the total injected distance [13]. Injected distance refers to the number of states that are skipped so that a running case can be brought from its last observed state to its updated state throughout its execution. This means that completeness is inversely correlated with the total amount of injected distance.

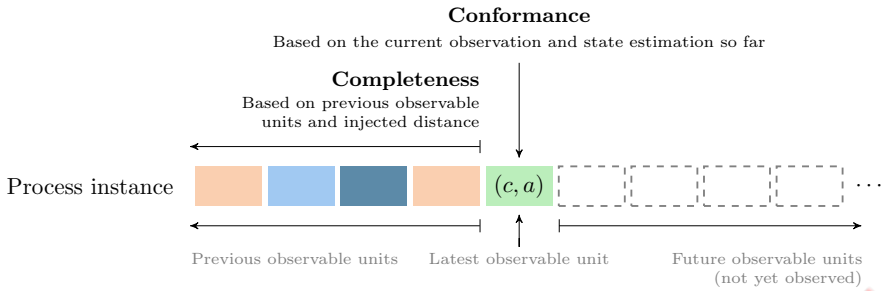



Figure 6: General idea of the two conformance indicators based on a running process instance: *conformance, completeness* (based on a similar diagram in [13])

For example, trace $\sigma_0$ in Figure 2 would have both perfect conformance and completeness since it corresponds to a complete model trace in the model in Figure 1. In contrary, trace $\sigma_5$ would yield a high conformance value but a low completeness since it corresponds to a partial model trace that is missing in the prefix of at least three activities (NEW, FIN and JOIN-PAT).

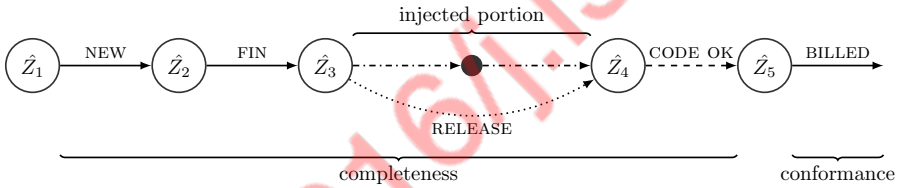



Figure 7: Metric breakdown projected onto the evaluation of trace $\sigma_6$ in Figure 4. Same as Figure 4, line style indicates various conformance explanations.

We refer to trace $\sigma_6$ of the running example in Figure 2 to provide some more intuition. Figure 7 enriches Figure 4 with a breakdown of the different conformance metrics. We can see that each state node is denoted by $\hat{Z}_i$ and corresponds to the state estimation at each time step. As before, suppose that we assume that all cases should start at the initial marking, the state estimation $\hat{Z}_1 = \mathbf{1}_{[i]}$ would concentrate all the probability mass on marking $[i]$. Similarly, $\hat{Z}_2 = \mathbf{1}_{[p_0]}$ and $\hat{Z}_3 = \mathbf{1}_{[p_8,p_9]}$. Due to the non-conforming event (RELEASE) and the following event (CODE OK), the state estimation $\hat{Z}_4$ concentrates all the probability mass on marking $[p_2]$. Inspecting the reachability graph of the model would tell us that

the markings $[p_8, p_9]$ and $[p_2]$ are not adjacent and are separated by another node. This "injected distance" indicates that the observed sequence of events does not correspond to a complete model trace. Moreover, for this particular state representation of markings, an event should not bring the corresponding case's state to a non-adjacent state, the total injected distance can be normalized and inverted as the completeness metric.

### *4.6. Algorithm for online processing*

The procedure for online conformance computation is presented in Algorithm 2. The algorithm requires a stream of observable units (cf. Definition 2), the HMM-based model (cf. Definition 5), and a state distance matrix as input.

The algorithm has an infinite loop to process a stream of observations (lines 1 and 2). The whole procedure can be split into three phases: 1) updating the state estimation of a case upon a new event, 2) computing conformance, and 3) entry removals if the number of tracked cases is reaching maximum capacity.

For the first phase (Lines 3 – 10), we update the discrete time step of the case, i.e., the case length, the forward probability, and the state estimation. Forward probability is computed with respect to each latent state. If the new event is the first observed event of the case, then the forward probability just corresponds to updating the initial distribution using the observation probability. Otherwise, we also need to account for the state-transition probability from each state of the previous forward probability. The update state estimation then corresponds to the normalized forward probability. For the second phase (Lines 11 – 18), we compute three conformance metrics. The *conformance* of the observed event with respect to its estimated state is computed in Line 11. Then, we use the modes of estimated previous and current state (which are categorical distributions) to update the *total injected distance* (Line 12 – 17). We assume that in a conforming scenario, the two states should have a distance of 1 so that the observed event corresponds to the firing of a transition that progresses the previous marking to an adjacent marking in the reachability graph. To convert the total injected distance into a *completeness* metric, we assume that the sum of the total injected distance and the case length corresponds to the number of latent states traveled across in the most likely latent state sequence so that completeness corresponds to the proportion of latent states that can be mapped to observations (Line 18). The third phase of the algorithm (Lines 19 – 20) removes the oldest entries due to the finite amount of memory to cater possibly an infinite number of cases.

**Suitability for online settings.** The computational complexity of the infinite loop is linear with respect to the stream size given the reference model as input. In

**Algorithm 2:** Online conformance computation

**Input:** $S$: stream of observable units
$M = (W_{x_1}, \ldots, W_{x_{|X|}}, W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V, V^d, conf, \pi)$: HMMConf
$D \in \mathbb{R}^{|Z| \times |Z|}$: state distance matrix

**forever do**
  // New caseid-activity pair from the stream
  $(c, a) \leftarrow observe(S)$ ;
  // Phase 1: update forward probability $P(Z_t = i, X_{1:t} = x_{1:t})$
  $time(c) \leftarrow time(c) + 1$ ; // $time(c) = 0$ if $a$ is the first event
  **for** $j \in Z$ **do**
    **if** $time(c) = 0$ **then**
      $\alpha^c_j(time(c)) \leftarrow \pi_j v_j(a)$ ; // $c$ denotes the caseid. See Definition 7
    **else**
      // $a_{time(c)-1}$ is the activity observed at $time(c) - 1$
      $\alpha^c_j(time(c)) \leftarrow \sum_{i \in Z} v_j(a) w_{i,j}(a_{time(c)-1}) \alpha^c_i(time(c) - 1)$ ; // See Definition 6 and 7
  **for** $i \in Z$ **do**
    // $P(Z_t = i \mid X_{1:t} = x_{1:t}) = \frac{P(Z_t = i, X_{1:t} = x_{1:t})}{\sum_{k \in Z} P(Z_t = k, X_{1:t} = x_{1:t})}$
    $state(c, i) \leftarrow \frac{\alpha^c_i(time(c))}{\sum_{k \in Z} \alpha^c_k(time(c))}$ ;
  // Phase 2: compute online conformance values
  $conformance(c) \leftarrow conf(state(c), a)$ ; // See Definition 4
  // Modes are used to compute injected distance
  **if** $time(c) = 1$ **then**
    $\hat{z}_{time(c)-1} \leftarrow \text{argmax}_{i \in Z}\ \pi_i$ ;
  **else**
    $\hat{z}_{time(c)-1} \leftarrow \text{argmax}_{i \in Z}\ \alpha^c_i(time(c) - 1)$ ;
  $\hat{z}_{time(c)} \leftarrow \text{argmax}_{i \in Z}\ \alpha^c_i(time(c))$ ;
  // $inj(c) = 0$ if $a$ is the case's first event
  $inj(c) \leftarrow inj(c) + \max\{0, D_{\hat{z}_{time(c)-1}, \hat{z}_{time(c)}} - 1\}$ ;
  $completeness(c) \leftarrow \frac{time(c)}{inj(c) + time(c)}$ ;
  // Phase 3: cleanup
  **if** *size of* $\alpha$ *and state is close to max capacity* **then**
    Remove entries of oldest cases

phase 1, both the forward probability and state estimation corresponds to matrix operations of vectors and matrices that have a fixed size given the reference model. This means they can be computed in constant time for each event. Similarly, all computations in phase 2 can be done in constant time given the reference model. Phase 3 can also be done in constant time using data structures like LinkedHashMaps. The space required by the algorithm is bounded by the maximum number of tracked cases. For each case, a vector of estimated state and several metric values are stored. Since processing an event takes a constant amount of time and space, the algorithm is suitable for online processing. Next, we turn to the task of computing and estimating the model parameters of the proposed technique.

## 5. Parameter computation and estimation

In the previous section, we presented the proposed technique and explained how online conformance checking can be performed. This section presents the parameter computation and estimation that are done offline on past data.

As previously presented, Figure 3 illustrates the dichotomy of the entire procedure where grayed boxes corresponds to data sources, dash lined boxes corresponds to parameter estimations, and solid lined boxes corresponds to other computations. In the following, we present the various details of the offline component, starting with the parameter estimation aspect. We note that this is for the presented instantiation of the proposed technique where we use Petri net markings to represent a case's state in the process. The conformance matrix that is used for the conformance function (cf. Definition 4) is computed by traversing the Petri net model.

**Computation of conformance matrix.** By traversing the reachability graph $G = (V,E)$, we compute a matrix $(c_{ij}) \in \mathbb{R}^{|V|\times|A|}$ so that $c_{m,a} = 1$ iff it is possible to observe activity $a$ because either the corresponding transition is enabled at marking $m$ or if there is a sequence of enabled invisible transitions whose firings would lead to a marking $m'$ that enables the corresponding transition. Formally, $\forall_{m_i \in V}\ \forall_{t\in\{t'\in T | l(t')\in A\}} c_{m_i,l(t)} = 1$ iff $\exists_{j>i} \langle m_i,\ldots,m_j\rangle$ s.t. $(m_{j-1},m_j) = t \wedge \forall_{i\leq q<j-1}\ l(m_q,m_{q+1}) = \tau$.

**Computation of distance matrix.** Similar to the computation of the conformance matrix, we traverse the reachability graph $G = (V,E)$ to compute a distance matrix $(d_{ij}) \in \mathbb{R}^{|V|\times|V|}$ so that $d_{ij}$ corresponds to the shortest path distance between nodes $v_i$ and $v_j$ in $G$. Moreover, since invisible transitions are not observable in the event log, edges $e \in E$ that correspond to invisible transitions have weight 0. Lastly,

there can exist node pairs that do not have a directed path from one to the other. For these node pairs, we compute the shortest undirected path. Removing the edge direction from $G$ yields a connected graph since all markings are reachable from the initial marking.

**Parameters of conforming probability distributions.** In the case where a case is perfectly fitting with respect to the Petri net model, the corresponding marking sequence is not hidden, i.e., the parameters can be directly estimated rather than in an iterative manner using the EM algorithm. Standard replay techniques can be used to yield compute the marking sequence. The remaining issue is therefore on the firing of invisible transitions which cannot be mapped to an observed event. For this, we assume that all firings of invisible transitions from the last recorded marking is related to the current observation; this gives a consistent interpretation even in the case of requiring the firing of invisible transitions at the initial marking.

As such, both the state-transition and emission probability distributions are categorical distributions that describe the probability of transitioning to different next states given the current state and the probability of observing a particular activity given the current state respectively. The parameters can then be estimated by normalizing the respective counts from the replay of the cases in the training sets.

However, it is possible that not all of the modeled behavior has been observed in the training set. We take a Bayesian approach to incorporate the knowledge of all possible modeled behavior captured from the Petri net model into the distributions. Given that both distributions are categorical distributions, modeled behavior can be added as pseudo counts by using a Dirichlet prior [18]. To accumulate the pseudo counts, we traverse the reachability graph in the same manner as for the conformance matrix. For simplicity and computation speed, rather than using the full posterior distribution of the parameters, the expected values are used as point estimates of the parameters. This corresponds to normalizing the parameters of the Dirichlet posterior.

**Parameters of non-conforming probability distributions.** Following the parameter estimation of the conforming probability distributions, we now turn to the more difficult task of estimating the parameters of the non-conforming state-transition and emission probability distributions. This corresponds to finding parameters that maximize the likelihood of the observations given the parameters:

$$W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V^d = \text{argmax}_{W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V^d} P(x_{1:T}; W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V^d)$$

This is difficult to directly optimize because both the parameters and the corre-

sponding latent states of the observations are free parameters with dependence. Therefore, rather than direct optimization, the EM algorithm [19] is used. The EM algorithm alternates between estimating the latent states (Expectation step) and the matrix parameters (Maximization step) until the convergence threshold is met.

At the Expectation step, we fix the current parameter estimates and together with the observations, we compute the conditional probability of each observation being at the different latent states as $Q(z_{1:T}) = P(z_{1:T} \,|\, x_{1:T}; W_{x_1}, \ldots, W_{x_{|X|}}, V, W^d_{x_1}, \ldots, W^d_{x_{|X|}}, V^d)$. Then, at the Maximization step, we find new parameter estimates that maximize the conditional expected log likelihood of both the observations and their latent states. Similar to a conventional HMM, we can set up and derive closed form updates for the parameter estimates as follows:

$$\begin{aligned}
\log P(x_{1:T}) &= \sum_{z_{1:T}} \log P(x_{1:T}, z_{1:T}) \\
&\geq \sum_{z_{1:T}} Q(z_{1:T}) \log \left[ \frac{P(x_{1:T}, z_{1:T})}{Q(z_{1:T})} \right] \\
&= \sum_{z_{1:T}} Q(z_{1:T}) \left[ \log \pi_{z_1} + \underbrace{\sum_{t=2}^{T} \log w_{z_{t-1}, z_t}(x_{t-1})}_{1} + \underbrace{\sum_{t=1}^{T} \log v_{z_t}(x_t)}_{2} \right] \qquad (3)
\end{aligned}$$

We can separately estimate the two variables of interest by focusing on the labeled parts of Equation 3:

1. State-transition probability matrix of deviating behavior $W^d_a$
2. Observation probability matrix of deviating behavior $V^d$

$$W^d_{a,i,j} = \frac{\sum_{t=2}^{T} \alpha_i(t-1) w_{i,j}(z_{t-1}) v_j(x_t) \beta_j(t) \; 1\{x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\}}{\sum_{j=1}^{|Z|} \sum_{t=2}^{T} \alpha_i(t-1) w_{i,j}(z_{t-1}) v_i(x_t) \beta_j(t) \; 1\{x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\}}$$

$$V^d_{j,a} = \frac{\sum_{t=1}^{T} 1\{x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \alpha_j(t) \beta_j(t)}{\sum_{t=1}^{T} 1\{conf(\boldsymbol{z_t}, x_t) < 1\} \alpha_j(t) \beta_j(t)}$$

$\beta_j(t) = P(X_{t+1:T} = x_{t+1:T} \,|\, Z_t = j, X_t = x_t)$ is the backward probability for state $j$ at time $t$ and is computed as $\beta_j(t) = \sum_{k \in Z} v_k(x_{t+1}) w_{j,k}(x_t) \beta_k(x_{t+1})$ with base case $\beta_j(T-1) = \sum_{k \in Z} v_k(x_T) w_{j,k}(x_{T-1})$.

Next, we present the experimental evaluation of the proposed technique where we performed a stress test and correlation test on existing dataset for comparability with state of the art techniques.

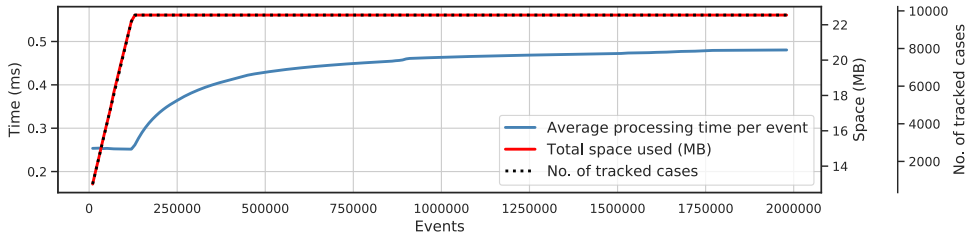



Figure 8: Performance during a stress test of ∼2 million events (see colored version online)

## 6. Experimental evaluation

For the sake of space and scope, we focus the evaluation on the conformance checking results of the proposed model rather than its predictive capability. The proposed approach is implemented in Python and can be found in the GitHub repository[3], along with further detail for reproduction. The experimental results are open and publicly available [4].

### *6.1. Stress test*

Similar to [13], we performed a stress test of our approach using the dataset from the mentioned work so that the results are comparable. However, preprocessing was necessary to filter out noisy events with randomly generated activity names. After filtering, the event stream has 1,985,744 events[5]. The test was performed on a standard machine, equipped with Python 3.6, an Intel Core i7-4700MQ 2.40GHz CPU and 12GB of RAM. We allowed the model to track 10,000 cases at most.

Figure 8 presents the results. Space is measured as the object size of the model. We can see that space is directly correlated with the number of cases after adding on a fixed space for the parameters. Both the number of cases and total space used reach a peak and remain stable at around 125k events when all 9,778 cases are tracked. Since the state transition probability is not involved for the first event of a case, the average processing time is particularly low for the first 125k events. Then, time eventually stabilizes at ∼0.49 ms after 1M events. This test demonstrates that the model can sustain a high load of events. As expected, the results

[3] `https://github.com/jwllee/HMMConf/tree/Computing2019`

[4] DOI will be requested after the acceptance of the manuscript. Meanwhile, they can be accessed at `https://drive.google.com/open?id=1w_Lt6aPmbwHFgP6BPpy3Mel0_AGhGDK7`

[5] Models and streams available in `https://doi.org/10.5281/zenodo.1194057`

suggest that both processing time and memory usage are non-increasing with respect to the number of events after reaching stability. However, it is significantly slower than the average processing time of below 0.009 ms/event reported in [13].

While the computation time performance of the offline training does not impact the online stream processing, we also performed a training time evaluation of our approach using a training set consisting of 1000 randomly sampled cases (193,712 events) from the same dataset. The EM algorithm was applied over the training set for 10 iterations and the experiment was repeated 3 times. The training took on average $2334 \pm 6$ (SD) seconds or $\approx 39$ mins. The significant amount of time required for training is expected since each EM iteration requires passing through all the training data to update the parameter estimations.

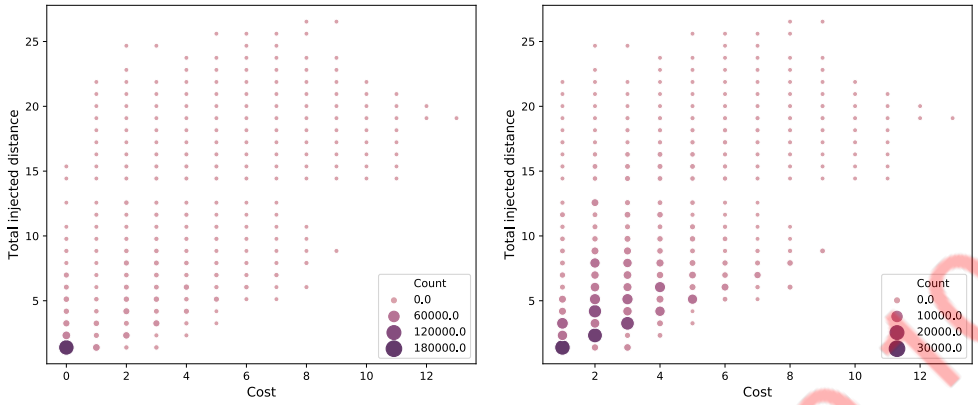
### *6.2. Correlation with alternative conformance metrics*

In this section, the cost based prefix alignment technique described in [11] is used as a baseline for comparison. Since this is a more informative and well-established technique, it is used as a baseline to evaluate the correlation of our proposed approach with the conformance as understood in the literature. Moreover, it also provides a way to evaluate our approach against the behavioral pattern based technique [13] since this technique also takes an approximation of the conformance as understood in the literature.

Similar to before, we make use of the dataset generated for the correlation test in [13]. Given that the prefix alignment results were not available in the open source dataset, we implemented prefix alignments using the Alignment package[6] in ProM6 [20]. The alignments are then computed using the standard cost function [21]. For each Petri net model, a 5-fold cross validation is performed on all the traces so that the mean measurement is taken for each event. Moreover, for the EM parameter estimation, we set the convergence condition as a tolerance threshold of 5 in log probability difference or a maximum of 10 iterations. Since we know that the traces can be non-conforming, an epsilon of $\sim$ 1e-5 (0.001%) is added to all states of the initial distribution.

We compare the two techniques under the context of all results and only non-conforming results as determined by alignment costs. Overall, we find that total injected distance is conceptually more similar to alignment costs. Figure 9 presents bubble plots where total injected distances are binned so that the y-axis values are of the intervals' mid value. Moreover, Figure 10 presents the Spear-

[6] `https://svn.win.tue.nl/trac/prom/browser/Packages/Alignment`




(a) All results (b) Non-conforming results

Figure 9: Bubble plots of total injection distance (with epsilon mass at initial distribution) versus incremental alignment costs

man's rank correlation coefficient ($\rho$-value) between costs and the proposed metrics. Given that conformance computed by our approach is between the estimated state and one event, we use the mean conformance for comparison. This means that for a case of length $k$, the mean conformance is computed from $k$ values.

| | Spearman correlation coefficient[a] | |
|---|---|---|
| | All | Non-conforming |
| Behavioral pattern based [13] | | |
| Conformance | 0.953 | 0.295 |
| HMMConf | | |
| Mean conformance | -0.470 | 0.252 |
| Total injected distance | 0.697 | 0.665 |
| Completeness | -0.712 | -0.519 |

[a] All results are statistically significant with two-sided p-value $<0.001$

Figure 10: Statistics comparing prefix alignment costs and three metrics

As shown in Figure 9a, the dataset is predominantly conforming. In fact, as we will later show in the confusion matrix analysis, according to the prefix alignment technique, only $\sim 20\%$ of the case prefixes are non-conforming. Referring to Figure 10, there is a $\rho$-value of 0.697 between total injected distance and cost. The moderate positive correlation between the two is expected since higher costs imply that a larger number of consecutive latent states are likely to be not adjacent in

the reachability graph. In addition, the correlations between costs and mean conformance (-0.470) and completeness (-0.712) are also within expectations. For the non-conforming results, visually, Figure 9b suggests that higher costs correlates with larger total injected distance and this is supported by a $\rho$-value of 0.665. We observe a similar result with the moderate negative correlation between costs and completeness (-0.519). There is a low positive correlation between mean conformance and costs (0.252). This is surprising as one would expect conformance to be negatively correlated with costs. However, it is likely that the two metrics are simply measuring different conformance qualities. The proposed approach's conformance metric measures whether the observed event is conforming with respect to the local position of the case while costs takes a global perspective in measuring the cost of aligning the observed trace in an optimal manner. In comparison to the previous work on a behavioral pattern based approach [13] which yielded a $\rho$-value of -0.954 for the whole dataset and a $\rho$-value of -0.295 for non-conforming results, the results suggest that our approach yields conformance results that are closer to those provided by prefix alignments than the behavioral pattern based approach under non-conforming scenarios. One of the reasons for the lower correlation when considering the whole dateset is due to the differences at the lower cost region. This is likely to be due to the fact that our approach handles warm start scenarios by quickly orienting the case at the corresponding location in the model rather than classifying events as log moves.

### *6.3. Conformance classification performance analysis*

An important scenario in conformance checking is the classification of whether a case is conforming or not. For this, prefix alignments can be treated as the ground truth so that the conformance classification performance of alternative metrics can be evaluated. Similar to the previous correlation analysis, we emphasize that optimal prefix alignments are used as a baseline to evaluate the proposed approach with respect to the conformance as understood in the literature. Figure 11 shows the confusion matrix of the proposed HMMConf's performance. A case is deemed to be conforming if the conformance of the current event $>$0.99 (to account for float imprecision) and has a total injected distance of 0. We find that our approach has a precision of 0.992 and recall is at 0.863. This gives a F1-score of 0.923, which is much better than 0.838, the F1-score of a stratified dummy classifier.

In conclusion, we find that our approach yields results that correlate with prefix alignments. In particular, the total injected distance metric has a moderate

| | Non-conforming | Conforming [a] |
|---|---|---|
| cost $> 0$ | 450794 | 17763 |
| cost $= 0$ | 333356 | 2093647 |

[a] conformance $>0.99$ and total injection distance is 0

Figure 11: Confusion matrices of the behavioral pattern based approach (Pattern) [13] and the proposed approach (HMMConf)

correlation with alignment costs under both conforming and non-conforming scenarios. Moreover, differences between the two techniques can be adequately explained by the differences in treatment of both conforming and non-conforming scenarios, as well as the limitations of our approach.

## 7. Real-life dataset evaluation

We also perform an evaluation on a real-life dataset - hospital billing event log [14]. One focus is to illustrate its applicability in the current real life context where often times there is no available normative process model corresponding to the event data.

**Preprocessing and model construction.** Given that the event log spans over four years from 2012 to 2016, during which it can be shown that the process went through concept drifts, we filtered the event log to only contain cases that started at 01-01-2013 and after. Moreover, since our focus is on tracking the conformance of cases over the course of their executions, it is easier to highlight this aspect by looking at cases that have more than just a few events. We perform our final filter to only include cases that have 10 or more events. The final event log contains 2992 cases and 630 trace variants. As previously mentioned, it can be difficult to find a normative process model in the current real life context. One possible solution is to construct a process model from the observed data as a proxy of the underlying process. Here we assume that the underlying process is composed of the most frequent trace variants. Figure 13 shows that most of the cases belong to only a few trace variants. In fact, there are only 10 trace variants with more than 50 cases and in total there are 1717 cases associated with these 10 trace variants, i.e., they make up for more than 50% of the event log. We then created a handmade model that permits the 10 trace variants as shown in Figure 12. Overall, there are two main parts to the process: the billing goes through an approval process to yield a code before getting billed.

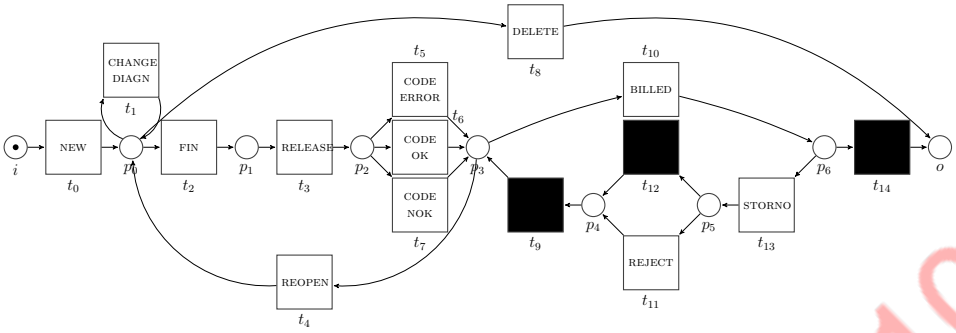



Figure 12: Petri net model extracted from 10 most frequent trace variants

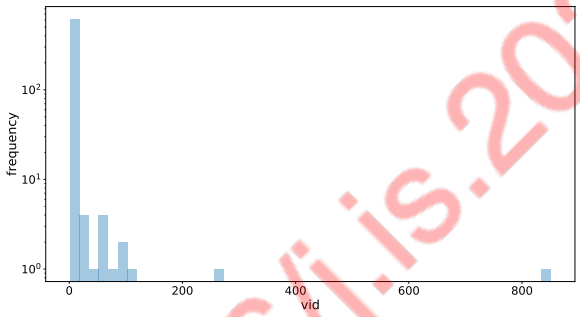


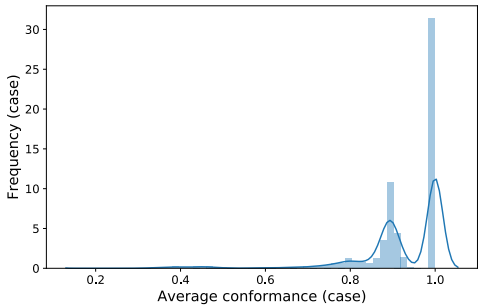

Figure 13: Distribution plot showing concentration of cases on a few trace variants

**Experiment setup.** We used a $k$-fold cross validation approach to evaluate the model where $k = 5$ and stratified sampling is used over the variant category of the cases so that the training set has a similar distribution as the overall dataset. For the training of the proposed model, a maximum of 10 iterations is set for the EM algorithm.

One interesting challenge is that the process model only has 12 visible transitions and does not include all 17 possible activities since some of the activities are not included in any of the 10 trace variants. We assumed that in real life we are able to know the set of possible observable activities beforehand so that these unmodeled activities can be incorporated into the observation variable set of the probability distributions and conformance matrix. Note that an even weaker assumption of not knowing the set of possible activities can be taken by adding a

wild card activity into the observation variable set so that all unmapped activities are mapped to this wild card activity.

**Result analysis.** Overall, we find that the data have high conformance. This is expected since the model captures $> 50\%$ of the observed behavior on a trace level. Inspecting the results on an activity level identifies specific conformance problems.




(a) Distribution plot of average conformance per case

| Fold no. | Conformance | | Total injected distance | | Completeness | |
|---|---|---|---|---|---|---|
| | mean | std | mean | std | mean | std |
| 1 | 0.930 | 0.233 | 0.172 | 0.528 | 0.985 | 0.045 |
| 2 | 0.947 | 0.205 | 0.119 | 0.420 | 0.990 | 0.035 |
| 3 | 0.951 | 0.198 | 0.108 | 0.389 | 0.991 | 0.033 |
| 4 | 0.834 | 0.330 | 0.475 | 1.153 | 0.966 | 0.075 |
| 5 | 0.944 | 0.211 | 0.145 | 0.480 | 0.988 | 0.040 |
| All | 0.901 | 0.269 | 0.267 | 0.817 | 0.980 | 0.057 |

(b) Test set statistics of conformance metrics

Figure 14: Experiment results on case level

Table 14b presents the mean and standard deviations of the test set conformance results. We can see that with the exception of the fourth test fold to some extent, all of the results are quite similar. This is expected since a stratified sampling approach was taken to create the cross validation data partition. The results show that generally the data is conforming and the injected distance is quite low. However, the standard deviation is somewhat high. Figure 14a shows the distribution plot of the average conformance per case. It shows two pronounced peaks: one at 1.0 and the other at 0.9. In fact, 1534 cases resulted with an avg. conformance of large or equal to 0.99 and 1173 cases resulted with an avg. conformance within the interval of $[0.80, 0.99)$.

Next, we analyze the results from the activity perspective. Figure 15 shows a violin plot of events which have a conformance value lower than 0.99, i.e., non-conforming events. It shows that the conformance values are distributed with multiple modes. This is actually due to the nature of the conformance issues in the data. For most of the non-conforming cases, the problem is in having "extra" activity executions that are not modeled. This means that either an activity is observed while the case is at the "wrong" location within the process or that a series of previous non-conforming observations had caused uncertainty in the

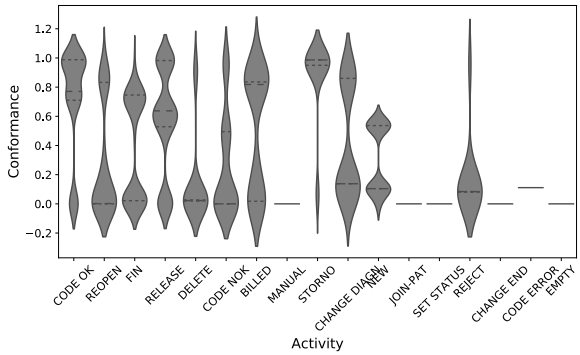



Figure 15: Violin plots of the conformance per activity for non-conforming events

state estimation but most of the probability mass is at the "correct" location for the current observation. Analyzing the specific shapes of the plots can even point out the particular conformance problem. For example, looking at the plot shapes of the activities CODE OK and CODE NOK shows that for the non-conforming observations, observations of CODE OK generally have high conformance and observations of CODE NOK generally have low conformance. Inspecting the cases in the event log shows that there are 77 cases where there is a string of one or more events with CODE NOK that ends with CODE OK before moving onto a different part of the process. In contrary, there is only 1 case where an event of CODE OK is followed by CODE NOK and even in this case, the event corresponding to CODE NOK ultimately ends with CODE OK.

## 8. Related work

There are many offline conformance checking techniques, e.g., alignment-based techniques [21, 22, 23, 24, 25, 26, 27, 28, 29], behavioral profile techniques [30], token replay based techniques [31, 32], and HMM techniques [33]. This work differs from [33] in that the state-space of the model is used as the latent variable set and both the previous state and observation are used to estimate the next state. In addition, the EM algorithm is used to estimate the state-transition and observation probabilities under non-conforming scenarios. For online conformance checking, we discuss three recent works. Prefix alignments are presented in [11] to provide alignment explanations for possibly ongoing cases. However, alignment computations can take a long time and also the technique cannot handle warm start scenarios. Another approach is to pre-compute possible deviations on

top of the model behavior [12]. There is some similarity between the proposed approach and [12] in that we try to keep track of the state of a running case. But rather than deciding how non-conformance should be handled beforehand, the EM algorithm is used to estimate the necessary parameters. Lastly, [13] transforms an event stream into a stream of behavioral patterns and checks whether the observed patterns are conforming. This approach takes a strong abstraction to trade for run time efficiency. It also proposed a breakdown of conformance in an online context into three aspects: conformance, completeness, and confidence. This idea was brought into our approach as: conformance and total injected distance.

## 9. Conclusion and future work

In this paper we presented an approach to perform conformance checking on a stream of events against a reference model. The approach alternates between updating the state estimation of a running case and computing its conformance with respect to the updated state. To model the behavior of a process under both conforming and non-conforming scenarios, the approach modifies a conventional HMM so that both the state estimation and observations are used to estimate the next state. Similar to a recent work [13], to measure different aspects of conformance, the approach includes three different metrics: conformance, total injected distance, and completeness. The proposed approach is implemented as a Python package and has been verified through a stress test, a comparison with prefix alignments, and a real-life dataset. As future work we plan to address the limitations identified in the paper as well as to further develop the approach. This includes investigating ways to abstract from using markings as latent states for efficiency and other constructs to model behavior in the presence of non-conformance, e.g., decomposition techniques [34]. Moreover, we plan to conduct further experimental evaluation of the proposed approach against other state of the art techniques. For example, while the proposed approach is compared with the behavioral pattern based approach through correlation analysis, it would be interesting to extend the conformance classification performance analysis to include the behavioral pattern based approach. Furthermore, we plan to investigate and evaluate the effects brought by the parameter approximation of the conforming probability distribution using expected values rather than the full posterior distribution.

## Acknowledgments

This work is supported by *CONICYT-PCHA / Doctorado Nacional / 2017-21170612*, *CONICYT FONDECYT Iniciación 11170092*, by *CONICYT REDI 170136*.

## Appendix A. Hospital billing example

Table A.1: Activity description of the hospital billing event log taken from [35]

| Activity | Description |
|---|---|
| NEW | A new billing package is created. |
| FIN | The billing package is closed, i.e., it may not be changed anymore. |
| RELEASE | The billing package is released to be sent to the insurance company. |
| CODE OK | A declaration code was successfully obtained. |
| BILLED | The billing package has been billed, i.e., the invoice is sent out. |
| CHANGE DIAGN | The diagnosis that the billing package is based on was changed. |
| DELETE | The billing package was deleted. |
| REOPEN | The billing package was reopened, i.e., additional medical services may be added or existing services removed. |
| CODE NOK | The declaration code was obtained with an error message. |
| STORNO | The billing package was canceled. |
| REJECT | The invoice sent to the insurance company was rejected. |
| SET STATUS | The status (i.e., new, closed, etc.) was manually changed. |
| EMPTY | The billing package is declared empty. |
| MANUAL | The billing package was manually changed from a non-standard system. |
| JOIN-PAT | The billing package was joined together since they refer to the same patient. |
| CODE ERROR | The declaration code could not be obtained. |
| CHANGE END | The projected end date of the billing package was changed. |

## Appendix B. Parameter estimation

Here we detail various parts of the proposed approach.

### *Appendix B.1. Forward probability (prior to observation update)*

Let $P(X_{1:t-1}, Z_t)$ be the desired forward probability prior to update from the current observation $X_t$ at time $t$. We show that it can be computed using the conformance dependent state-transition probability, i.e., $P(Z_t|Z_{t-1}, X_{t-1})$, and the forward probability from the previous time step, i.e., $P(X_{1:t-1}, Z_{t-1})$, as

$$P(X_{1:t-1}, Z_t) = \sum_{z \in Z} P(X_{1:t-1}, Z_{t-1} = z) P(Z_t|Z_{t-1} = z, X_{t-1}) \tag{B.1}$$

$$\begin{aligned}
P(X_{1:t-1}, Z_t) &= \sum_{z \in Z} P(X_{1:t-1}, Z_t, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(X_{1:t-2}, X_{t-1}, Z_t, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(Z_t|X_{1:t-2}, X_{t-1}, Z_{t-1} = z) P(X_{1:t-2}, X_{t-1}, Z_{t-1} = z) \\
&= \sum_{z \in Z} P(Z_t|X_{t-1}, Z_{t-1} = z) P(X_{1:t-1}, Z_{t-1} = z)
\end{aligned}$$

*Appendix B.2. State-transition probability matrix*

Here we detail the closed form update of the non-conforming state-transition probability matrices. As recalled from Section 5, the goal is to find parameters that maximizes the expected log likelihood with respect to all latent state sequences with a constraint on their sum.

$$\begin{aligned} \underset{W^d_{a,i,j}}{\text{maximize}} \quad & \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=2}^{T} \log w_{z_{t-1},z_t}(x_{t-1}) \right] \\ \text{subject to} \quad & \sum_{j=1}^{|Z|} W^d_{a,i,j} = 1 \end{aligned}$$

While the matrix is substochastic where a case does not transition to another state once the final state is reached, deviating behavior might actually transition a case from the final state. We construct the Lagrangian:

$$\begin{aligned} \mathscr{L}(W^d_{a,i,j}) = \sum_{z_{1:T}} Q(z_{1:T}) \Bigg[ & \sum_{t=2}^{T} \sum_{i=1}^{|Z|} \sum_{j=1}^{|Z|} \sum_{a=1}^{|A|} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\} \\ & \log w_{z_{t-1},z_t}(x_{t-1}) \Bigg] + \sum_{a=1}^{|A|} \sum_{i=1}^{|Z|} \lambda_{a,i} (1 - \sum_{j=1}^{|Z|} W^d_{a,i,j}) \end{aligned}$$

Similar to before, taking the partial derivatives and setting them to zero with respect to $W^d_{a,i,j}$ and $\lambda_{a,i}$ yield the parameter estimation:

$$\begin{aligned} & \nabla_{W^d_{a,i,j}} \mathscr{L}(W^d_{a,i,j}) \\ & = \nabla_{W^d_{a,i,j}} \sum_{z_{1:T}} Q(z_{1:T}) \Bigg\{ \sum_{t=2}^{T} \sum_{i=1}^{|Z|} \sum_{j=1}^{|Z|} \sum_{a=1}^{|A|} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\} \\ & \quad \log \left[ conf(\mathbf{1}_i, a) W_{a,i,j} + (1 - conf(\mathbf{1}_i, a)) W^d_{a,i,j} \right] \Bigg\} + \sum_{a=1}^{|A|} \sum_{i=1}^{|Z|} \lambda_{a,i} (1 - \sum_{j=1}^{|Z|} W^d_{a,i,j}) \\ & = \sum_{z_{1:T}} Q(z_{1:T}) \Bigg[ \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\} \\ & \quad \frac{1 - conf(\mathbf{1}_i, a)}{conf(\mathbf{1}_i, a) W_{a,i,j} + (1 - conf(\mathbf{1}_i, a)) W^d_{a,i,j}} \Bigg] - \lambda_{a,i} = 0 \end{aligned}$$

We only need to update $W^d_{a,i,j}$ when $conf(\mathbf{1}_i, a) = 0$ (note that an activity is either conforming or non-conforming to a given state), in which case:

$$= \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\} \frac{1}{W^d_{a,i,j}} \right] - \lambda_{a,i} = 0$$

$$W^d_{a,i,j} = \frac{1}{\lambda_{a,i}} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\} \tag{B.2}$$

The partial derivative of the Lagrangian can then be computed as:

$$\nabla_{\lambda_{a,i}} \mathscr{L}(W^d_{a,i,j}) = 1 - \sum_{j=1}^{|Z|} W^d_{a,i,j} = 0$$

$$= 1 - \sum_{j=1}^{|Z|} \frac{1}{\lambda_{a,i}} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\} = 0$$

Then, the Lagrange multiplier can be derived so that it can be substituted into the update of the parameter.

$$\lambda_{a,i} = \sum_{j=1}^{|Z|} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\}$$

$$= \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\} \tag{B.3}$$

$$W^d_{a,i,j} = \frac{\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\}}{\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge x_{t-1} = a \wedge conf(\mathbf{z}_{t-1}, x_{t-1}) < 1\}} \tag{B.4}$$

Similar to before, we can use the forward and backward probabilities to effi-

ciently compute $\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\}$:

$$
\begin{aligned}
&\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\} \\
&= \sum_{z_{1:T}} P(Z_{1:T} = z_{1:T} \,|\, X_{1:T} = x_{1:T}) \sum_{t=2}^{T} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\} \\
&= \sum_{t=2}^{T} \sum_{z_{1:T}} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\} P(Z_{1:T} = z_{1:T} \,|\, X_{1:T} = x_{1:T}) \\
&= \sum_{t=2}^{T} \sum_{z_{1:T}} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\} \frac{P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T})}{P(X_{1:T} = x_{1:T})} \\
&= \frac{1}{P(X_{1:T} = x_{1:T})} \sum_{t=2}^{T} \sum_{z_{1:T}} 1\{z_{t-1} = i \wedge z_t = j \wedge x_{t-1} = a\} P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \\
&= \frac{1}{P(X_{1:T} = x_{1:T})} \sum_{t=2}^{T} \alpha_i(t-1) w_{i,j}(a) v_j(x_t) \beta_j(t)
\end{aligned}
$$

This means that to update the weights for the non-conforming transition matrices:

$$
W^d_{a,i,j} = \frac{\sum_{t=2}^{T} \alpha_i(t-1) w_{i,j}(a) v_j(x_t) \beta_j(t)\ 1\{x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\}}{\sum_{j=1}^{|Z|} \sum_{t=2}^{T} \alpha_i(t-1) w_{i,j}(a) v_j(x_t) \beta_j(t)\ 1\{x_{t-1} = a \wedge conf(\boldsymbol{z}_{t-1}, x_{t-1}) < 1\}} \tag{B.5}
$$

*Appendix B.3. Emission probability matrix*

Same as for the state-transition probability matrix, the goal is to find parameters that maximizes the expected log likelihood with respect to all latent state sequences with a constraint on their sum.

$$
\begin{aligned}
\underset{V^d_{k,j}}{\text{maximize}} \quad & \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=1}^{T} \log v_{z_t}(x_t) \right] \\
\text{subject to} \quad & \sum_{a=1}^{|A|} V^d_{a,j} = 1
\end{aligned}
$$

Similar to the state-transition matrix, for conforming behavior, the matrix is sub-stochastic where a case at the final state would not emit any activity observation.

However, deviating behavior might mean that a case can emit an activity observation despite being in the final state. We construct the Lagrangian:

$$\mathscr{L}(V^d_{a,j}) = \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=1}^{T} \sum_{j=1}^{|Z|} \sum_{a=1}^{|A|} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1)\} \log v_{z_t}(x_t) \right] + \lambda_j (1 - \sum_{a=1}^{|A|} V^d_{a,j})$$

Taking the partial derivatives and setting them to zero with respect to $V^d_{j,a}$ and $\lambda_{j,a}$ yield the parameter estimation:

$$\nabla_{V^d_{a,j}} \mathscr{L}(V^d_{a,j}) = \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \frac{1 - conf(\mathbf{1_j}, a)}{conf(\mathbf{1_j}, a) V_{a,j} + (1 - conf(\mathbf{1_j}, a)) V^d_{a,j}} \right] - \lambda_j = 0$$

The non-conforming observation matrices $V^d_{a,j}$ would be updated when $conf(\mathbf{1_j}, a) = 0$,

$$= \sum_{z_{1:T}} Q(z_{1:T}) \left[ \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \frac{1}{V^d_{a,j}} \right] - \lambda_j = 0$$

$$V^d_{a,j} = \frac{1}{\lambda_j} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \tag{B.6}$$

$$\nabla_{\lambda_j} \mathscr{L}(V^d_{a,j}) = 1 - \sum_{a=1}^{|A|} V^d_{a,j} = 0$$

$$= 1 - \sum_{a=1}^{|A|} \frac{1}{\lambda_j} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} = 0$$

$$\lambda_j = \sum_{a=1}^{|A|} \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\}$$

$$= \sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \tag{B.7}$$

$$V^d_{a,j} = \frac{\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\}}{\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge conf(\boldsymbol{z_t}, x_t) < 1\}} \tag{B.8}$$

Similar to before, we can use the forward and backward probabilities to efficiently compute $\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\}$:

$$
\begin{aligned}
&\sum_{z_{1:T}} Q(z_{1:T}) \sum_{t=1}^{T} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \\
&= \sum_{t=1}^{T} \sum_{z_{1:T}} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} Q(z_{1:T}) \\
&= \sum_{t=1}^{T} \sum_{z_{1:T}} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} P(Z_{1:T} = z_{1:T} \,|\, X_{1:T} = x_{1:T}) \\
&= \frac{1}{P(X_{1:T} = x_{1:T})} \sum_{t=1}^{T} \sum_{z_{1:T}} 1\{z_t = j \wedge x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) \\
&= \frac{1}{P(X_{1:T} = x_{1:T})} \sum_{t=1}^{T} 1\{x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \alpha_j(t) \beta_j(t)
\end{aligned}
$$

This means that

$$
V^d_{a,j} = \frac{\sum_{t=1}^{T} 1\{x_t = a \wedge conf(\boldsymbol{z_t}, x_t) < 1\} \alpha_j(t) \beta_j(t)}{\sum_{t=1}^{T} 1\{conf(\boldsymbol{z_t}, x_t) < 1\} \alpha_j(t) \beta_j(t)} \tag{B.9}
$$

## References


[1] W. M. P. van der Aalst, Process Mining - Data Science in Action, Springer, 2016.

[2] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Second Edition, Springer, 2018. doi:10.1007/978-3-662-56509-4.
URL https://doi.org/10.1007/978-3-662-56509-4

[3] B. Vázquez-Barreiros, M. Mucientes, M. Lama, Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm, Inf. Sci. 294 (2015) 315–333.

[4] J. Carmona, B. F. van Dongen, A. Solti, M. Weidlich, Conformance Checking - Relating Processes and Models, Springer, 2018.

[5] B. Vázquez-Barreiros, M. Mucientes, M. Lama, Enhancing discovered processes with duplicate tasks, Inf. Sci. 373 (2016) 369–387.

[6] M. Jans, M. Hosseinpour, How active learning and process mining can act as continuous auditing catalyst, Int. J. Accounting Inf. Systems 32 (2019) 44–58.

[7] J. D. Weerdt, M. D. Backer, J. Vanthienen, B. Baesens, A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs, Inf. Syst. 37 (7) (2012) 654–676.

[8] J. Ribeiro, J. Carmona, M. Misir, M. Sebag, A recommender system for process discovery, in: Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings, 2014, pp. 67–83.

[9] A. K. A. de Medeiros, A. J. M. M. Weijters, W. M. P. van der Aalst, Genetic process mining: an experimental evaluation, Data Min. Knowl. Discov. 14 (2) (2007) 245–304.

[10] D. Gaurav, J. K. P. Singh Yadav, R. K. Kaliyar, A. Goyal, An outline on big data and big data analytics, in: 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 74–79. doi:10.1109/ICACCCN.2018.8748683.

[11] S. J. van Zelst, A. Bolt, M. Hassani, B. F. van Dongen, W. M. P. van der Aalst, Online conformance checking: relating event streams to process models using prefix-alignments, International Journal of Data Science and Analytics (Oct 2017).

[12] A. Burattin, J. Carmona, A framework for online conformance checking, in: Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, 2017, pp. 165–177.

[13] A. Burattin, S. J. van Zelst, A. Armas-Cervantes, B. F. van Dongen, J. Carmona, Online conformance checking using behavioural patterns, in: Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, Proceedings, 2018, pp. 250–267.

[14] Mannhardt, F. (Felix), Hospital billing - event log (2017). doi:10.4121/UUID:76C46B83-C930-4798-A1C9-4BE94DFEB741.
URL https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

[15] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (4) (1989) 541–580.

[16] R. M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Information & Software Technology 50 (12) (2008) 1281–1294.

[17] Y. Li, Hidden markov models with states depending on observations, Pattern Recognition Letters 26 (7) (2005) 977–984.

[18] K. P. Murphy, Machine learning - a probabilistic perspective, Adaptive computation and machine learning series, MIT Press, 2012.

[19] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1) 1–22.

[20] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. v. Dongen, W. M. P. v. d. Aalst, ProM 6: The Process Mining Toolkit, in: M. La Rosa (Ed.), Proc. of BPM Demonstration Track 2010, Vol. 615 of CEUR Workshop Proceedings, CEUR-WS.org, Hoboken, USA, 2010, pp. 34–39.

[21] A. Adriansyah, Aligning Observed and Modeled Behavior, Ph.D. thesis, Technische Universiteit Eindhoven (2014).

[22] F. Taymouri, J. Carmona, An evolutionary technique to approximate multiple optimal alignments, in: Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, 2018, pp. 215–232.

[23] B. F. van Dongen, Efficiently computing alignments - using the extended marking equation, in: Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, 2018, pp. 197–214.

[24] D. Reißner, R. Conforti, M. Dumas, M. L. Rosa, A. Armas-Cervantes, Scalable conformance checking of business processes, in: On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I, 2017, pp. 607–627.

[25] H. V. D. Aa, H. Leopold, H. Reijers, Efficient process conformance checking on the basis of uncertain event-to-activity mappings, IEEE Transactions on Knowledge and Data Engineering (2019) 1–1.

[26] A. Armas-Cervantes, P. Baldan, M. Dumas, L. García-Bañuelos, Diagnosing behavioral differences between business process models: An approach based on event structures, Inf. Syst. 56 (2016) 304–325.

[27] F. Taymouri, J. Carmona, An evolutionary technique to approximate multiple optimal alignments, in: Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, 2018, pp. 215–232.

[28] B. F. van Dongen, J. Carmona, T. Chatain, F. Taymouri, Aligning modeled and observed behavior: A compromise between computation complexity and quality, in: Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, 2017, pp. 94–109.

[29] F. Taymouri, J. Carmona, A recursive paradigm for aligning observed behavior of large structured process models, in: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, 2016, pp. 197–214.

[30] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, M. Weske, Process compliance analysis based on behavioural profiles, Inf. Syst. 36 (7) (2011) 1009–1025.

[31] A. Rozinat, W. M. P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Inf. Syst. 33 (1) (2008) 64–95.

[32] S. K. L. M. vanden Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, J. Vanthienen, Event-based real-time decomposed conformance analysis, in: On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings, 2014, pp. 345–363.

[33] A. Rozinat, M. M. Veloso, W. M. P. van der Aalst, Using Hidden Markov Models to Evaluate the Quality of Discovered Process Models, BPM Center Report BPM-08-10 (2008).

[34] W. M. P. van der Aalst, Decomposing Petri nets for process mining: A generic approach, Distributed and Parallel Databases 31 (4) (2013) 471–507.

[35] Mannhardt, F. (Felix), Multi-perspective Process Mining, Ph.D. thesis, Technische Universiteit Eindhoven (2018).