Online Conformance Checking Using Behavioural Patterns

Andrea Burattin[1], Sebastiaan J. van Zelst[2], Abel Armas-Cervantes[3], Boudewijn F. van Dongen[2], and Josep Carmona[4]

[1] Technical University of Denmark, Kgs. Lyngby, Denmark
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
[3] The University of Melbourne, Melbourne, Australia
[4] Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. New and compelling regulations (e.g., the GDPR in Europe) impose tremendous pressure on organizations, in order to adhere to standard procedures, processes, and practices. The field of conformance checking aims to quantify the extent to which the execution of a process, captured within recorded corresponding event data, conforms to a given reference process model. Existing techniques assume a *post-mortem* scenario, i.e. they detect deviations based on *complete* executions of the process. This limits their applicability in an online setting. In such context, we aim to to detect deviations online (i.e., *in-vivo*), in order to provide recovery possibilities before the execution of a process instance is completed. Also, current techniques assume cases to start from the initial stage of the process, whereas this assumption is not feasible in online settings. In this paper, we present a generic framework for online conformance checking, in which the underlying process is represented in terms of behavioural patterns and no assumption on the starting point of cases is needed. We instantiate the framework on the basis of Petri nets, with an accompanying new unfolding technique. The approach is implemented in the process mining tool ProM, and evaluated by means of several experiments including a stress-test and a comparison with a similar technique.

Keywords: Conformance checking; online processing; behavioural patterns; stream processing; Petri nets; unfoldings.

1 Introduction

Organizations are facing challenges that arisie by digital transformation. Important concerns to face are the way processes are managed, their strategic alignment w.r.t. the organization's goals and their compliance with respect to applicable regulations. An example of these challenges is the compliance with the new regulations on the protection of data in Europe, i.e. *GDPR*[1], where unprecedented requirements on the use of data of EU citizens by organizations will be applicable from May 2018. Are current business processes in organizations aligned with these new regulations?

Conformance checking is acknowledged as one of the key enabling technologies for verifying compliance monitoring of regulations [11]. It compares (prescriptive) process models to the actual execution of a process, and allows us to pinpoint deviations. The

[1] See `http://eur-lex.europa.eu/eli/reg/2016/679/oj`.

Burattin, A., Zelst, S., Armas, A., Dongen, B., Carmona, J. Online conformance checking using behavioural patterns. A: International Conference on Business Process Management. "Business Process Management,16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018: proceedings". Berlín: Springer, 2018, p. 250-267.
The final authenticated version is available online at https://doi.org/10.1007/978-3-319-98648-7_15

detection of compliance problems can be narrowed to the set of detected deviations [1]. In spite of being a powerful aid, a rigid exploration of conformance checking techniques has only been performed relatively recently [2, 4, 8, 14, 17, 18, 20–22].

A widespread application mode of conformance in literature is *post-mortem*: the relation between the model and the observed behaviour is computed, assuming that traces of observed process behaviour are complete. Such analysis, though meaningful and accurate, only allows us to detect deviations *after they occurred*, which, in some contexts, is too late. For example, consider the case where a trace of process behaviour represents the treatment of a patient during her life, and the model encompasses the clinical guidelines to follow for a given disease.

In contrast, *online conformance checking* techniques consider a live, real-time stream of events as input, where every event belongs to a particular case, i.e. process instance. As such, several different unfinished (running) cases at any position in the stream need to be considered [4, 22]. Moreover, in real scenarios cases may start at different points in the process, not necessarily in its initial stage, e.g. a patient process being monitored in the middle of her clinical life. Such *warm start* mode of online conformance checking allows us to not only analyze cases from which the full history is available, but also those cases that lack historical process information.

In this paper, we present a novel framework, accompanied with a corresponding instantiation that builds on top of the notion of Petri net unfoldings [13], that enables the application of online conformance checking in warm start settings. To the best of our knowledge, this is the first solution for this important problem. We present a framework that relies on the notion of *behavioural patterns*, i.e., relations between process activities. In particular, for each possible behavioural pattern, the number of different behavioural patterns preceding/following it for a case is assumed to be known. Subsequently, the approach assesses *compliance* by checking whether the expected behavioural patterns are either observed or violated. Additionally, *completeness* (is the running case expected to be complete?) and *confidence* (is the compliance metric reliable?) values provide a more holistic view on the compliance of running cases.

We provide an instance of the framework based on *weak order relations*, accompanied by an implementation in the process mining framework ProM [19]. We validate the approach by means of a synthetic data set containing models and traces of varying sizes and a data set containing cases that start in different stages of the process (warm start). Furthermore we assess the applicability of the approach on a real data set. We also asses the correlation of the technique w.r.t. the technique presented in [22], which confirms that our framework provides a good estimation of conformance.

The remainder of the paper is structured as follows. In Sec. 2, we motivate the need for an online conformance checking technique capable of handling the warm start scenario. In Sec. 3, we present related work. In Sec. 4, we briefly present background terminology. In Sec. 5, the general framework is described, which is instantiated in Sec. 6 for weak order relations. We evaluate the instantiation in Sec. 7. In Sec. 8 we discuss limitations of the work, whereas Sec. 9 concludes this paper.

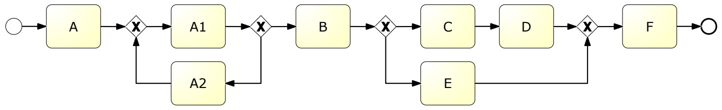


Fig. 1: Running example considered throughout this paper.

Table 1: Comparison of offline ([2]) and online conformance values (as proposed in this paper) based on the process model in Fig. 1.

Trace	Offline Conformance	Online Conformance	Online Completeness	Online Confidence
$t_1 = \langle A, A1, B, E, F\rangle$	1.00	1.00	1.00	1.00
$t_2 = \langle B, C, D, F\rangle$	0.78	1.00	0.60	1.00
$t_3 = \langle A, A1, A2, A1, B\rangle$	0.80	1.00	1.00	0.50
$t_4 = \langle B, C, D\rangle$	0.62	1.00	0.50	0.75

2 Motivation

Consider the process model reported in Fig. 1. Furthermore, consider some possible executions of such process and their corresponding conformance values as reported in Tab. 1. Trace t_1 conforms w.r.t. the model: it represents a possible complete execution of the process. This information is properly captured by both the offline technique [2] and our online approach. Execution t_2, on the other hand, is compliant with the process but just from activity B onward, i.e. assuming that the initial activity A was executed yet not observed. Such case is known as a warm start scenario: we start monitoring ongoing process instances rather than processes started after monitoring. Our approach is explicitly designed to deal with this problem by additionally quantifying the *completeness of the execution*. Note that, as Tab. 1 reports, offline approaches do not capture the notion of completeness, and thus, in case of warm start, the final conformance value is simply decreased. Trace t_3 suffers from the opposite problem: it conforms to the process model only up-until activity B, i.e., we expect to observe future behaviour. If we do not assume to be in a *post-mortem* scenario, this trace has no conformance problem, but is simply *partial*. Our approach is designed to explicitly handle this problem by quantifying the *confidence* of the execution, i.e. the degree of reliability of the reported conformance metric. Again, Tab. 1 shows that offline techniques cannot handle this situation. The combination of the last two problems is present in trace t_4, i.e. the trace captures an intermediate execution of the process which conforms the model but lacks initial and final parts of the execution. The offline approach reports a conformance of 0.62, whereas the online approach indicates that, subject to incompleteness and a little lack of confidence, the behaviour as seen conforms to the model.

3 Related Work

Until recently, conformance checking has only focused on relating modeled and observed behaviour in a post-mortem fashion. Techniques for this task have been proposed, with different assumptions and guarantees. Among existing techniques, we observe: rule-based- [14, 20], token replay-based- [14], and alignment-based techniques. [2, 3, 8, 17, 18, 21]. The work presented in this paper can be seen as an evolution of the rule-based approaches, where important new features, i.e. from offline to online and the warm start capability, have been properly incorporated.

For online conformance checking, we identify two research lines. In [22] the authors propose to compute *prefix-alignments*, i.e. providing explanations for prefixes of complete behaviour. Unfortunately the complexity requirements are high and the technique is unable to handle the warm start scenario. An alternative approach is presented in [4], where all the possible deviations are pre-computed on top of the model behaviour, which is used to walk through the input stream.

4 Background

4.1 Process Models and behavioural Patterns

We do not assume a specific process modelling formalism, yet we do assume process models to be defined in context of collections of activities. As such, we assume a process model to constrain the relative ordering of its activities, e.g. reconsider the BPMN diagram in Fig. 1, which specifies that we are able to execute activity A prior to activity $A1$, yet the reverse is not the case. We furthermore assume the execution process activities to be atomic. A model M is potentially an imperative model, e.g. BPMN, Petri net or EPC. The only requirement we impose on the considered model(s) is the fact that we are able to deduce a language in terms of the activities it is defined upon.

Given a process model, with a corresponding language and relative ordering on its activities, we assume that we are able to derive more advanced behavioural relations, i.e. *behavioural patterns*, such as weak ordering, parallelism, causality and conflict. Given two activities part of a process model, formally, we define a behavioural pattern as a relation that the process imposes on them. As an example, consider the model in Fig. 1, which dictates that activity A is *always followed* by activity $A1$.

Definition 1 (Behavioural Pattern). *Given a set of activities $\mathcal{A}$ and a set of possible control-flow relations $\mathcal{R}$, a* behavioural pattern *is defined as $b(a_1, a_2)$ where $a_1, a_2 \in \mathcal{A}$ are activities and $b \in \mathcal{R}$ represents a control-flow relation. An alternative writing of $b(a_1, a_2)$ is $a_1\ b\ a_2$.*

Using the notion of behavioural patterns, we formalize process models as follows.

Definition 2 (Process Model). *A* process model *B is the set of all behavioural patterns prescribed by the process, such that $B \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{A}$, where $\mathcal{A}$ is the set of activities and $\mathcal{R}$ is the set of possible control-flow relations.*

In context of this paper, we are primarily interested in behavioural patterns induced by the possible sequential ordering of activities, i.e. we take a *control-flow perspective*. As such we assume the existence of a universe of *control-flow relations* $\mathcal{R}$ that allow us to induce behavioural patterns. Examples of control-flow relations present in $\mathcal{R}$ are defined in [15]. Consider for example the *weak order relation*. Let's assume the existence of two activities a_1 and a_2. They are in weak order relation, expressed as $a_1 \prec a_2$, if there exists an execution of the process where a_1 occurs before a_2. Such relations are used not only for the formal definition of the process, but also for the definition of our observations: instances of these relations represent the observable units against which we want to compute the conformance. For example, consider the BPMN model in Fig. 1. Based on the semantics of BPMN, we deduce, for the control-flow relation $\prec$ (weak order relation), to have $\{(\prec, A, A1), (\prec, A1, B), \ldots, (\prec, D, F), (\prec, E, F)\}$.

4.2 Data Streams

A data stream is typically defined as an infinite sequence of data items. As such, we define a sequence over set X of length n as a function $\sigma\colon \{1, ..., n\} \to X$, and an infinite sequence as $\sigma\colon \mathbb{N}^+ \to X$. We also refer to a sequence using string representation: $\sigma = \langle x_1, x_2, \ldots, x_n \rangle$ where $x_i = \sigma(i) \in X$. In context of this paper, the streams we observe refer to executions of a certain behavioural pattern. Therefore, we define an *observable unit* as a behavioural pattern which is observed in a process instance.

Definition 3 (Observable Unit). *Let $\mathcal{C}$ denote the set of case ids, let $\mathcal{R}$ denote the set of control-flow relations and let $\mathcal{A}$ denote the set of activities. Let $b \in \mathcal{R} \times \mathcal{A} \times \mathcal{A}$ denote a behavioural pattern. An observable unit $o = (c, b) \in \mathcal{C} \times \mathcal{R} \times \mathcal{A} \times \mathcal{A}$ is a tuple describing a behavioural pattern $b \in B$ that is observed in context of case id c. The universe of all possible observable units is defined as $\mathcal{O} = \mathcal{C} \times \mathcal{R} \times \mathcal{A} \times \mathcal{A}$.*

For each observable unit we assume to have projection operators to extract the case id and the pattern i.e. given $o = (c, b)$, $\pi_c(o) = c$ and $\pi_b(o) = b$.

Definition 4 (Stream of Behavioural Patterns). *Given the universe of observable units $\mathcal{O} = \mathcal{C} \times \mathcal{R} \times \mathcal{A} \times \mathcal{A}$, a* stream of behavioural patterns *is defined as an infinite sequence of observable units: $S : \mathbb{N}^+ \to \mathcal{O}$.*

A stream of behavioural patterns can be seen as an unbounded sequence of observable units where their ordering complies with the time order of the observable units, as defined by the underlying execution time of the corresponding activities. Note that, a stream of behavioural patterns refers to information at a high level of abstraction, i.e. when compared to the commonly used stream of executed process events [5]. However, under specific circumstances, e.g. the behavioural pattern considered in Sec. 6 (we consider a stream of direct follows relations), a stream of behavioural patterns is easily extracted from a stream of simple events. We refer to [5, 6], where techniques to convert a stream of events to a stream of behavioural patterns are described.

5 Online Conformance Checking using behavioural Patterns

In this section we present conformance checking in terms of behavioural patterns. We first present the envisioned requirements for an online conformance checking approach after which we propose a generic framework that fulfills these requirements.

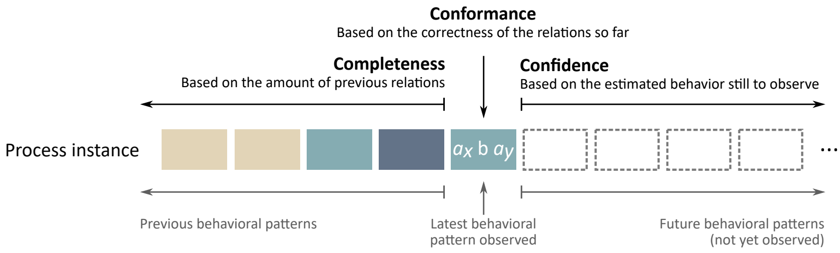

Conformance
Based on the correctness of the relations so far
Completeness
Based on the amount of previous relations
Confidence
Based on the estimated behavior still to observe
Process instance
a_x b a_y
...
Previous behavioral patterns
Latest behavioral pattern observed
Future behavioral patterns (not yet observed)

Fig. 2: General idea of the 3 conformance measures computed based on a partially observed process instance: *conformance*, *completeness*, and *confidence*.

5.1 Problem Statement

Existing conformance checking techniques quantify conformance using one specific metric, typically in terms of compliance or deviation costs. In online settings however, we suffer both from the fact that we perform in-vivo analysis, i.e. new event data is likely to be observed in the future, as well as the warm start scenario. Using only one metric to express conformance, therefore, leads to misleading results, i.e. cases that already started and/or that are not finished yet get falsely penalized for this. To solve these issues, we propose a breakdown of conformance in:

1. *Conformance*: Indicating the amount of correct behaviour observed thus-far;
2. *Completeness*: Indicating whether the entire trace is observed since the beginning.
3. *Confidence*: Indicating the possibility that the conformance score remains stable.

Consider Fig. 2 in which we graphically illustrate the proposed conformance metrics. *Conformance* is based on the current knowledge of a case, witnessed by the observed behaviour. *Completeness* indicates the degree to what behaviour is potentially missed for a case. *Confidence* signifies to what degree we are able to trust the conformance metric, i.e. if more behaviour is expected in the future, deviations may occur later as well.

5.2 Process Representation

The foundation of our online conformance checking technique is the notion of behavioural pattern. Hence, we need a model capturing the following information:

1. The set of behavioural patterns prescribed by the model;
2. For each behavioural pattern, the minimum and maximum number of distinct prescribed patterns that must be *observed before*, since the beginning of the case;
3. For each behavioural pattern, the minimum number of distinct patterns *still to observe* in order to reach the end of the process (as prescribed by the reference model).

We formalize such (process) model as follows.

Definition 5 (Process Model for Online Conformance (PMOC)). *A* process model for online conformance *(PMOC) $M = (B, P, F)$ is defined as a triplet containing the set of prescribed behavioural patterns B. Each pattern is defined according to Def. 1.*

Algorithm 1: Online conformance computation

```
   Input: $S$: stream of behavioural patterns
          $M = (B, P, F)$: process model for online conformance
1  Initialize map obs          // Maps case ids to (finite) set of observed prescribed patterns
2  Initialize map inc                                              // Maps case ids to integers
3  forever do
4      $(c, b, t) \leftarrow observe(S)$                    // New observable unit from the stream
       // Step 1: update internal data structures
5      if $b \in B$ then
6          $\mathsf{obs}(c) \leftarrow \mathsf{obs}(c) \cup \{b\}$             // If $b$ already in $\mathsf{obs}(c)$, then no effect
7      else
8          $\mathsf{inc}(c) \leftarrow \mathsf{inc}(c) + 1$
       // Step 2: compute online conformance values
9      $\mathsf{conformance}(c) \leftarrow \frac{|\mathsf{obs}(c)|}{|\mathsf{obs}(c)| + \mathsf{inc}(c)}$
10     Notify new value of $\mathsf{conformance}(c)$
11     if $b \in B$ then
12         if $P_{\min}(b) \leq |obs(c)| \leq P_{\max}(b)$ then
13             $\mathsf{completeness}(c) \leftarrow 1$
14         else
15             $\mathsf{completeness}(c) \leftarrow \min\left\{1, \frac{|\mathsf{obs}(c)|}{P_{\min}(b) + 1}\right\}$
16         $\mathsf{confidence}(c) \leftarrow 1 - \frac{F(b)}{\max_{b' \in B} F(b')}$
17         Notify new values of $\mathsf{completeness}(c)$ and $\mathsf{confidence}(c)$
       // Step 3: cleanup
18     if size of obs and inc is close to max capacity then
19         Remove oldest entries from obs and inc
```

P contains, for each behavioural pattern $b \in B$, the pair of minimum and maximum number distinct prescribed patterns (i.e., B) to be seen before b. We refer to these values as $P_{\min}(b)$ and $P_{\max}(b)$. Finally, for each pattern $b \in B$, $F(b)$ refers to the minimum number of distinct patterns (i.e., B) required to reach the end of the process from b.

5.3 Computing Online Conformance Metrics

The procedure for the online computation of the conformance checking is reported in Alg. 1. The algorithm requires a stream of behavioural patterns (cf. Def. 4) and a PMOC (cf. Def. 5) as input. The algorithm initializes two maps/functions: obs and inc (lines 1-2). Given a case id as key, these maps store the set of observed prescribed behavioural patterns and the number of observed patterns not prescribed. Note that, for each case, the amount of data to store is bounded by the model, and thus, constant w.r.t. the stream.

The online conformance procedure has an infinite loop to process the unbounded stream of behavioural relations (lines 3 and 4). The procedure is then split into 3 steps: *(i)* updating the maps; *(ii)* computing the conformance; and *(iii)* housekeeping. In the first step (lines 5-8) the obs and inc data structures are updated with the new observation: if the pattern refers to prescribed relation, then it is added to the $\mathsf{obs}(c)$ set[2]. Otherwise, the value of incorrect observations is incremented.

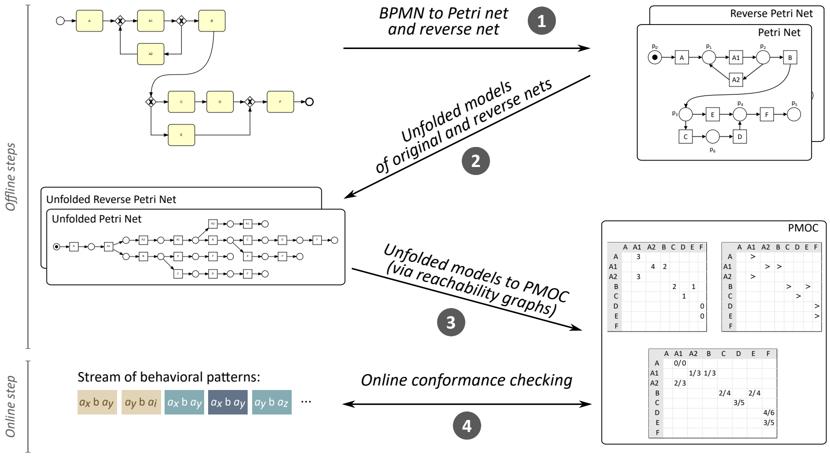
[2] If obs has no key c, $\mathsf{obs}(c)$ returns the empty set. If inc has no key c then $\mathsf{inc}(c)$ returns 0.

The second step of the algorithm (lines 9-17) computes the actual conformance. The *conformance* for a (partial) process instance c is calculated in line 9: the number of distinct observed prescribed patterns in c (i.e., $|\mathsf{obs}(c)|$) divided by the sum of the number of prescribed observed patterns and the incorrect patterns (i.e., $|\mathsf{obs}(c)| + \mathsf{inc}(c)$). We quantify, in the interval $[0, 1]$, the correct behaviour observed, where 1 indicates full conformance (i.e., no incorrect behaviour) and 0 indicates no conformance at all (i.e., only incorrect behaviour). Completeness and confidence are updated only when a prescribed behavioural pattern is observed (line 11) since they require to locate the pattern itself in the process. Specifically, the *completeness* of process instance c is calculated in lines 12-15. It depends on whether the number of distinct behavioural patterns observed so far is within the expected interval for current pattern b (i.e., $P_{\min}(b) \leq |\mathsf{obs}(c)| \leq P_{\max}(b)^3$) or not. In the former case, we assume completeness is perfect (therefore value 1). In the latter case, the problem could be due to two reasons: we observe less patterns than expected ($|\mathsf{obs}(c)| < P_{\min}(b)$) and in this case we have the ratio of observed pattern over the minimum expected. Alternatively we observe more behavioural patterns than expected ($|\mathsf{obs}(c)| > P_{\max}(b)$) and in this case we assume a completeness value of 1. Note that this last case could represent a "false positive": we count the number of observed correct patterns without checking which exact patters we are dealing with. This approximation is imposed by online processing constraints. Finally, the *confidence* of case c is calculated in line 16 as 1 minus the ratio of patterns still to observe (i.e., $F(b)$) and the overall maximum number of future patterns (i.e., $\max_{b' \in B} F(b')$). Confidence also ranges in $[0, 1]$: 1 indicates strong confidence (i.e., the execution reached the end of the process), 0 means low confidence (i.e., the execution is still far from completion, therefore there is room for changes). Observe that, the metrics computed by the algorithm implement the metrics described in the problem statement section (cf. subsection 5.1).

The third step of the algorithm (lines 18, 19) consists of cleanup operations. Specifically, only a finite amount of memory is available: we can store only some process instances. This step of the algorithm takes care of that: once the size of obs and inc reaches the memory limit, oldest entries are removed. For the sake of readability, we do not focus on the actual procedures to achieve that (cf. [5, 6] for possible solutions).

Suitability of the Algorithm for Online Settings. The computational complexity of the main loop of the algorithm is constant for each event (given the reference model as input). Specifically, step 1 (lines 5-8) updates hash maps in constant time. All computations in step 2 (lines 9-17) require constant time complexity (note that $\max_{b' \in B} F(b')$ depends just on the model and can be pre-computed in advance). Finally, step 3 (lines 18, 19), can be realized to require constant time complexity (e.g., using `LinkedHashMaps`). The space required by the procedure is bounded by an imposed maximum number of keys in obs and inc. Then, since obs stores sets of prescribed behavioural patterns (which are finite) and inc stores just one integer, the whole memory can not grow above the imposed threshold. Since processing a single event takes a constant amount of time and fixed amount of space, the procedure is suitable for online processing.

[3] $P_{\min}(b)$ and $P_{\max}(b)$ refer to the min./max. number of distinct patterns to be seen before b.


Offline steps
Online step
BPMN to Petri net and reverse net
1
Reverse Petri Net
Petri Net
p0
A
p1
A1
p2
B
A2
p3
E
p4
F
p5
C
p6
D
Unfolded models of original and reverse nets
2
Unfolded Reverse Petri Net
Unfolded Petri Net
Unfolded models to PMOC (via reachability graphs)
3
PMOC
Stream of behavioral patterns:
a_x b a_y
a_y b a_i
a_x b a_y
a_x b a_y
a_y b a_z
...
Online conformance checking
4

	A	A1	A2	B	C	D	E	F
A		3						
A1			4	2				
A2		3						
B					2		1	
C						1		
D								0
E								0
F								

	A	A1	A2	B	C	D	E	F
A		>						
A1			>	>				
A2		>						
B					>		>	
C						>		
D								>
E								>
F								

	A	A1	A2	B	C	D	E	F
A		0/0						
A1			1/3	1/3				
A2		2/3						
B					2/4		2/4	
C						3/5		
D								4/6
E								3/5
F								

Fig. 3: General idea of the approach presented in this paper. Steps 1-3 are performed once, offline. Step 4 is the only online activity.

6 Online Conformance Checking using Weak Ordering Relations

In this section, we present an instantiation of the framework proposed in this paper. We do so by computing three matrices out of the original model before the actual online analysis. These matrices contain the information about the possible relations between pairs of activities in the process model (behavioral patterns) that is needed by PMOC (cf. Def. 5). In particular, for each possible behavioural pattern, we compute the (min. and max.) number of different behavioural patterns preceding/following it for each case in the model. The computation of these matrices allows us to retrieve information online in constant time. The roadmap for the computation of the three matrices out of a process model is shown in Fig. 3, while each of the steps is described in more detail in the remaining of the section.

Step ①: Input Process Models

As mentioned in Sec. 4, we do not assume a specific process modelling formalism. However, in the context of this particular instantiation, we assume that the model can be represented as a Petri net, possibly through a transformation from other process modelling languages (e.g., transforming BPMN into Petri nets [7]). For instance, Fig. 4 shows the Petri net system representation of the BPMN process in Fig. 1, where transitions, places, arcs and tokens are represented as squares, circles, directed black arrows and black dots, respectively.

Given a (transformed) Petri net, an additional *reverse* net is computed. The reverse net is a net with the same set of places and transitions as the original one, but where the direction of the edges is inverted. The use of this additional net is made clear in Step ③. Some notions used later in this section relate to the execution semantics of Petri nets, which we briefly/informally introduce here. A transition t is enabled iff there

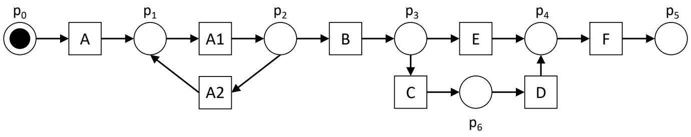


Fig. 4: Labeled net system of the model in Fig. 1.

is at least one token in each place in the preset of t. An enabled transition t can be fired and, as a consequence, modifies the distribution of tokens over the net, thus producing a new marking. The firing of a transition t removes one token from each place in its preset and puts one token in each place in its poset. Finally, a marking is *reachable* if it is produced by the firing of a sequence of transitions. We restrict to Petri nets systems whose reachable markings contain up to 1 token in every place, i.e. *safe Petri nets*.

Step ②: Finite representation of Process Model Behavior Through Unfoldings

The information about the behavioral patterns required by our framework can be extracted by analyzing the state space (markings) of the Petri net. Specifically, at each marking, the number of behavioral patterns are computed and counted, and the number of different behavioural patterns preceding/following the last observed pattern is stored. Nevertheless, if a net is cyclic then the number of behavioural patterns it can produce is infinite. Several authors have proposed techniques for computing finite Petri net representations of the behavior of a net known as *complete prefix of an unfolding*. For instance, [13] introduces a way to truncate the unfolding of a net at a finite level, while keeping a representation of any reachable marking. Then, a framework for constructing a *canonical unfolding prefix*, complete with respect to a suitable property, not limited to reachability, was proposed in [10]. Our own work relies on such a framework, i.e. we compute a finite fragment of the unfolding capturing enough information about the distinct behavioral patterns in a net.

The new unfolding, specially developed for this instantiation, analyses each reachable marking at every possible case and computes the set of behavioral patterns between the transitions (activities) that were fired to reach such marking. The idea of this new unfolding is to keep firing transitions in the original net and create new instances of places and transitions whenever they are fired, in the case of transitions, or visited by a token, in the case of places. Then, the unfolding stops once it finds information that has been observed before. As a concrete example, consider the *weak order relation* between activities. Fig. 5 shows the complete prefix unfolding for the running example (unfolding of the net shown in Fig. 4). Observe that p_2', p_2'' and p_2''' are instances of the place p_2. However, the unfolding stopped at p_2''' because the weak order relations are the same as those captured at the marking in p_2''. In [10], the necessary conditions that a notion of equivalence between execution states shall satisfy to guarantee that the complete prefix unfolding is canonical and finite are defined. In our case, a pair of markings are equivalent if they have *(i)* the same places, *(ii)* the same relations (i.e., weak order) between activities executed to reach such marking, and *(iii)* the same set of activities that were lastly executed for reaching such markings. These conditions allow to prove canonicity and finiteness of the new complete prefix unfolding.

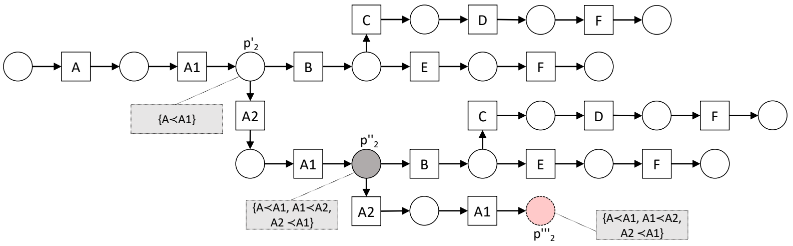


Fig. 5: Weak order relation preserving unfolding computed by our new unfolding technique. The unfolding stops when reaching p_2''' since the induced weak order relations are the same as those observed in p_2''.

Step ③: Computation of PMOC's Data Structures via Reachability Graphs

Given the complete prefix unfolding described above, different ways to compute the weak order relations can be envisioned. For simplicity, in our implementation we construct the corresponding reachability graph $TS = (S, TR, s_0)$. Such graph, which is always finite, is used to derive the set of allowed weak order relations: for each state $s \in S$ it is possible to compute the set $t_{in}^{s} \subset TR$ with all non-silent transitions immediately leading to s, and the set $t_{out}^{s} \subset TR$ with all non-silent transitions immediately leaving s. In case there is a silent transition connected to s it is necessary to recursively follow it and retrieve all incoming/outgoing transitions which will be part of t_{in}/t_{out}. The set $\bigcup_{s \in S}\{x \prec y \mid x \in t_{in}^{s}, y \in t_{out}^{s}\}$ represents all weak order relations that can be extracted from TS. A weak order relation $x \prec y$, defined as x entering $s \in S$ and y leaving s, might appear several time in TS. By finding the longest and shortest paths from s_0 to all occurrences of s, and converting these paths into distinct weak order relations, it is possible to identify the minimum and maximum number of weak order relations preceding $x \prec y$.

For the purpose of this paper we do not only require the minimum and maximum number of relations preceding a given one, but also the minimum number of relations required to reach the end of the model. Thus, we use the reverse net for computing such information by computing the complete prefix of the reverse net (reusing the methodology in Step ②), and then counting the distinct relations over the corresponding reachability graph.[4] Observe that by inverting the direction of the weak order relations in the reverse net, we obtain information referring to the end of the model: the distances now refer to the minimum/maximum number of relations to reach the *end*. The techniques described in this section allow the computation of the information needed to have a proper process model abstraction for online conformance checking (cf. Def. 5).

[4] In general, not all Petri nets can be reversed for computing the minimal number of relations to reach the *end*. Hence, for computing confidence, we assume in the realization of the framework presented in Sec. 6 a proper subclass, i.e., sound workflow nets.

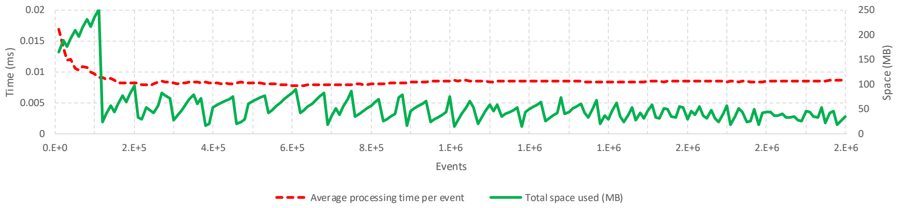


Fig. 6: Performance of the system during a stress test involving 2 million events.

7 Experimental Evaluation

In this section, we present an experimental evaluation of the proposed techniques in terms of performance, as well as its indicative power of conformance. We additionally compare our technique against an alternative, state-of-the-art online conformance checking technique. The proposed technique is available as ProM plugin.[5]

7.1 Stress Test

We performed a stress test of our prototype. We randomly generated a BPMN model containing 64 activities and 26 gateways. The model was then used to simulate an event stream of 2 million events[6]. The test was performed on a standard machine, equipped with Java 1.8(TM) SE Runtime Environment on Windows 10 64bit, an Intel Core i7-7500U 2.70GHz CPU and 16GB of RAM. Results of the test are reported in Fig. 6. After an initial phase, when the constructed data structures were still in memory, the Java Virtual Machine was able to remove these unreferenced objects. This explains the drop in the memory and the stabilization of the processing time, after about 100k event. From that moment on, the memory used remained permanently around 100MB and the average processing time persisted below 0.009 ms/event.

This test shows that the implemented prototype is capable of sustaining a high load of events on a standard laptop machine. Moreover, we observe that both the processing time and memory usage show a relatively stable, non-increasing trend. This aligns well with our expectations and the general requirements of data stream analysis.

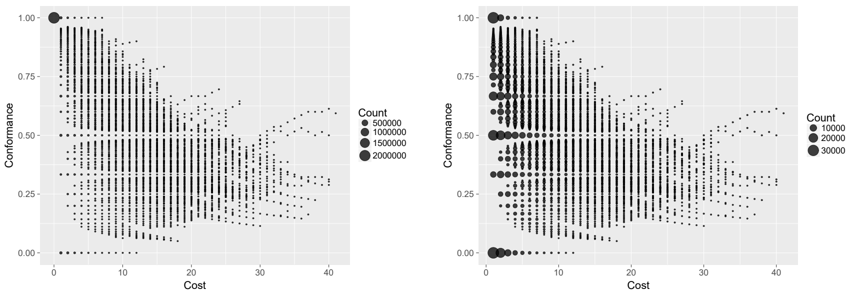
7.2 Correlation with Alternative Conformance Metrics

In this section, we examine the correlation of the proposed metrics with the alternative described in [22], which reports a potential deviation in terms of *costs*, rather than a conformance metric. Hence, the higher the cost of deviation, the less conformance. As the metric in [22] is a more informed technique (at the expense of using more memory) than the one proposed in this paper, a correlation between both metrics shows that our technique reflects online conformance well.

We generated 12 random process models [9] with number of activities according to a triangular distribution with lower bound 10, mode 20, and upper bound 30. We

[5] See `https://svn.win.tue.nl/repos/prom/Packages/StreamConformance/`.

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



(a) All results.

(b) Non-conforming results.

Fig. 7: Scatter plots of conformance metric versus incremental alignment costs [22].

Table 2: Correlation matrix (ρ-values, Spearman) for non-conforming results (cf. Fig. 7b), showing the conformance metrics of this paper and costs as defined in [22].

		Metrics from this paper			**Cost** [22]
		Conformance	Completeness	Confidence	
Metrics from this paper	Conformance		0.52282662	0.3862707	-0.29513342
	Completeness			0.1851850	-0.02546182
	Confidence				0.25104526

did not include duplicate labels, a probability of 0.2 for addition of silent activities, moreover, the probability of control-flow operator insertion was: 0.45 for sequence, 0.2 for parallel and xor-split operators, 0.05 for an inclusive-or operator and 0.1 for loop constructs. From these models a collection of event logs has been created (each log contains 1000 traces), subsequently treated as streams by both techniques. Incremental noise levels (both on a trace- and event-level) were introduced in the logs. Probability of trace- and event-level noises ranged from 0.1 to 0.5 with steps of 0.1. In order to compute the conformance, the technique presented in this paper needs, at least, two events. Hence for a fair comparison, we only consider conformance values from the second event onward, yielding a total of $2,977,744$ analyzed events[6].

In Fig. 7 we present a scatter-plot of the conformance metric (this paper) versus the incremental alignment-based costs (alternative approach). Fig. 7a plots all results, i.e. all events, where the size of the dot indicates the number of instances for the specific value combination. *Spearman's rank correlation coefficient* for the whole data set (ρ-value) is -0.9538502. As the chart reports, coordinate $(0, 1.0)$ dominates the data (in 73.4% of cases both techniques agree on no deviation). Hence, the data is extremely skewed (vast majority of results at coordinate $(0, 1.0)$) which explains the strong negative correlation. Nonetheless, the result shows that the two metrics generally agree when no deviations occur. In Fig. 7b, we present the same results but only for combinations in which at least one of the techniques identifies a deviation. In general, when alignment costs increases, the conformance metric decreases. However, we observe that the conformance values are spread around, i.e. we do not observe a clear linear trend. This is supported by the corresponding ρ-value of -0.2951334, presented in Tab. 2, which shows a correlation matrix for non-conforming results (cf. Fig. 7b) of the conformance metrics presented in the paper and the costs as defined in [22]. Correlations among the

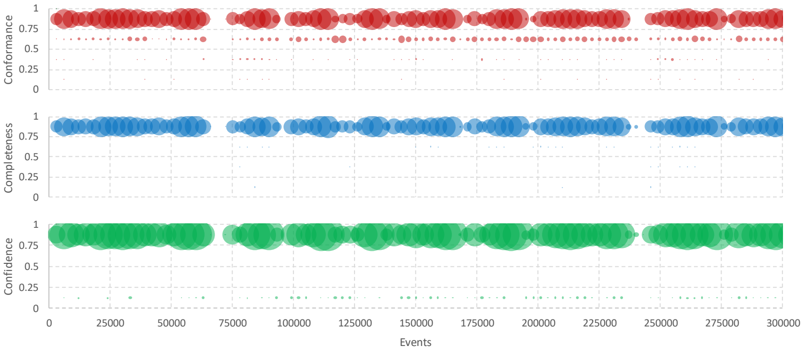


Fig. 8: Online conformance on the road traffic fines log.

metrics presented in this paper are the strongest. The fact that completeness and conformance depict the strongest correlation is explained by the fact that the data set in general contains complete cases. For confidence, weaker correlation is found. Based on the data used, it is expected that once a case matures, relatively more correct behaviour is observed than incorrect behaviour. The correlation between costs and completeness is negligible. For costs and confidence we observe a weak positive correlation: towards the end of a trace, the likelihood of having observed noise, and thus costs, goes up.

We conclude that the two metrics largely agree when no noise is present (with a minor number of outliers). When both methods observe deviations, corresponding quantifications do not clearly correlate. This is partly due to the fact that the alignment based approach always explains observations in terms of the model, whereas the approach in this paper does not. Secondly, the use of weak order relation as a behavioural pattern leads to the use of a strong abstraction of the model: this representational bias seems not in-line with the deviation approximation of the alternative approach.

7.3 Real-World Event Data Test

Finally, we investigated the real event log of an information system managing road traffic fines for the Italian police [12]. This log has a reference model, designed with the help of domain experts and regulators [12]. To avoid the state explosion problem during the computation of the matrices, we removed self-loops from the model[7]. Additionally, to focus on most relevant traces, we discarded all process instances with just one or two activities. The resulting log contains $316\,868$ events, over $83\,614$ cases. The processing of the log, (excluded the offline computations, and with support for up to $10\,000$ process instance in parallel) took $44\,967$ms (0.14 ms/event). Fig. 8 contains the binned results of the analysis. The x axis reports the different events (by time, grouped in bins of $3\,000$ events). The y axes of the charts report conformance, completeness and confidence levels (grouped in bins of 0.25). Each point represents several events (bubble size proportional to number of events) but different process instances can be intertwined. Therefore, two consecutive events could refer to cases with very different conformance levels (this explains fluctuations). We can see that the conformance values are mostly at

[7] This limitation only affects Sec. 6: it is possible to manually define the behavioural patterns.

1: only few events deviated from the reference model (93.5% of the events have conformance 1 and 99.6% of events have conformance ≥ 0.5). Average conformance value is 0.97, suggesting very high conformance in general. 99.8% of the events have completeness of 1: most executions actually started from the beginning with just sporadic warm starts. Finally, confidence has mostly value 1 (99.4% of events). This is due to the specific behaviour of the process which allows immediate termination of the execution right after the execution of the first activity.

8 Discussion

The approach presented in this paper can be used to monitor any set of behavioural patterns, i.e. we represent processes models as sets of prescribed behavioural patterns and streams as infinite sequences of behavioural patterns. Because of this, the framework is rather abstract and allows us to monitor any possible set of relations. Note that we could also use the *organizational perspective*, rather than the control-flow perspective. An example relation which might be relevant for monitoring purposes is whenever pairs of activities have to be performed collaboratively and simultaneously (i.e., *cooperation* [16]). The provided instantiation automatically extract instances of weak order behavioural patterns out of a Petri net and an event stream. We focus on weak order relations since they are widely used and relatively easy to deduce. Clearly, using more advanced behavioural patterns such as causality, parallism and/or different perspectives, i.e. organizational, requires a corresponding algorithmic design to deduce such patterns from the process model and/or stream under study.

9 Conclusions and Future Work

In this paper we present a generic approach to compute the conformance of data streams against a reference process model. In order to cope with all possible scenarios, the approach decomposes the actual conformance into 3 metrics: the actual *conformance*, the *completeness* and the *confidence*. Thus, the technique can be used on partial executions and on traces already running (i.e., warm start). Moreover, we provide an instantiation of the generic approach for the case of weak order relations, which is based on a new unfolding technique. This instantiation is implemented and available in ProM and it has been verified on large dataset for stress test, on a real dataset, and it has also been compared against a prefix-alignment based approach. As future work we plan to investigate further realizations of the framework, including declarative models, to understand which behavioural patterns are useful in order to converge towards optimal approaches bearing in mind that, being online, approximations must be in place.

Acknowledgments This work has been partially supported by MINECO and FEDER funds under grant TIN2017-86727-C2-1-R.

Bibliography

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

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

[3] vanden Broucke, S., Munoz-Gama, J., Carmona, J., Baesens, B., Vanthienen, J.: Event-based Real-time Decomposed Conformance Analysis. In: Proceedings of Confederated International Conferences: CoopIS. pp. 345–363 (2014)

[4] Burattin, A., Carmona, J.: A Framework for Online Conformance Checking. In: Business Process Management Workshops - BPM. pp. 165–177 (2017)

[5] Burattin, A., Cimitile, M., Maggi, F.M., Sperduti, A.: Online Discovery of Declarative Process Models from Event Streams. IEEE TSC 8(6), 833–846 (nov 2015)

[6] Burattin, A., Sperduti, A., van der Aalst, W.M.: Control-flow Discovery from Event Streams. In: Proceedings of the IEEE CEC. pp. 2420–2427 (2014)

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

[8] van Dongen, B.F., Carmona, J., Chatain, T., Taymouri, F.: Aligning Modeled and Observed Behavior: A Compromise Between Computation Complexity and Quality. In: Proceedings of CAiSE. pp. 94–109 (2017)

[9] Jouck, T. and Depaire, B.: PTandLogGenerator: A Generator for Artificial Event Data. In: Proceedings of the BPM Demo Track. pp. 23–27 (2016)

[10] Khomenko, V., Koutny, M., Vogler, W.: Canonical Prefixes of Petri Net Unfoldings. Acta Informatica 40(2), 95–118 (2003)

[11] Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Inf. Syst. 54, 209–234 (2015)

[12] Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing 98(4), 407–437 (2016)

[13] McMillan, K.L., Probst, D.K.: A Technique of State Space Search Based on Unfolding. Formal Methods in System Design 6(1), 45–65 (1995)

[14] Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Inf. Syst. 33(1), 64–95 (2008)

[15] Smirnov, S., Weidlich, M., Mendling, J.: Business Process Model Abstraction Based on Behavioral Profiles. In: Proceedings of ICSOC. Springer (2010)

[16] Song, M.: Organizational mining in business process management. Ph.D. thesis, Pohang University of Science and Technology, Pohang, South Korea (2006)

[17] Taymouri, F., Carmona, J.: A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models. In: Proceedings of BPM. pp. 197–214 (2016)

[18] Taymouri, F., Carmona, J.: Model and Event Log Reductions to Boost the Computation of Alignments. In: Proceedings of SIMPDA. pp. 50–62 (2016)

[19] Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Proceedings of CAiSE Forum. pp. 60–75 (2010)

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

[21] van Zelst, S.J., Bolt, A., van Dongen, B.F.: Tuning Alignment Computation: An Experimental Evaluation. In: Proceedings of ATAED. pp. 6–20 (2017)

[22] van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.P.: Online Conformance Checking: Relating Event Streams to Process Models using Prefix-Alignments. Int J Data Sci Anal (Oct 2017)