Online Conformance Checking for Petri Nets and Event Streams

Burattin, Andrea

Published in:
Online Proceedings of the BPM Demo Track 2017

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Online Conformance Checking for Petri Nets and Event Streams

Andrea Burattin
University of Innsbruck, Austria; Technical University of Denmark, Denmark
andbur@dtu.dk

Abstract. Within process mining, we can identify conformance checking as the task of computing the extent to which executions of a process model are in line with the reference behavior. Most approaches currently available in the literature (for imperative models, such as Petri nets) perform just a-posteriori analyses. This means that the amount of non-conformant behavior is quantified after the completion of the current execution. The tool presented in this paper, instead, proposes an approach for online conformance checking: not only it is capable of quantifying the deviating behavior on the fly, but the computation complexity is also restricted to a constant complexity per event analyzed. This enables the online analysis of an infinite stream of events. The tool is implemented as a package of the ProM framework and promising results have been obtained and are presented in this paper.

Keywords: online process mining, conformance checking, event stream.

1 Introduction

The analysis of data referring to business process is typically referred to as “process mining” [1]. This academic and industrial research topic achieved important reputation as an effective way of extracting knowledge out of event logs. In process mining, it is possible to identify several subproblems, such as control-flow discovery, conformance checking and extension. The work presented in this paper belongs to the conformance checking domain as it requires, as input, a reference process model and execution traces and computes the extent to which the performed actions actually conform the given model.

The most popular conformance checking techniques currently available in the literature [6] require a whole trace (i.e., a complete trace) in order to compute corresponding conformance value. Such requirement, from a business point of view, represents a limitation since it is necessary to wait the process instance to complete before computing its actual conformance. The technique presented in this paper, instead, drops such requirement and allows the computation of conformance values for running process instances. As a consequence, if a deviation is observed the system can immediately notify the process owner who can enact proper countermeasures. When several of such deviations are occurring within the same process instance, then corresponding seriousness value is increased, thus providing filtering techniques to the administrator (e.g., to first focus on
most critical instances). [7] proposes a RESTful service for conformance checking, performing token replay with a pull mechanism. The approach, however, provides no constant time/memory boundaries. Please note that, in the context of this work, with the term online we refer to the type of input of our technique: we assume to have an event stream which, basically, is a data stream of events.

The implemented components of the tool presented in this paper are reported in Fig. 1. The inputs of the first component (which is “Transition System Deviations Extender”) consist of a reference model and some conformance parameters (i.e., costs for specific types of deviations). Internally, this component converts the Petri net into a transition system which is then extended with additional connections in order to create an OCTS (i.e., an “Online Conformance Transition System”). This is then used by the actual conformance checker to validate the event stream provided as additional input in a later stage.

2 Online conformance checking

In this paper we would like not to stress the actual online conformance checking technique, which is described in details in [4]. Instead, we just want to sketch the general idea behind the approach.

The fundamental idea is that, given a process model, the system analyzes an event stream, to discover and notify the use about process instances deviating from the expected behavior (which is reported in the model). This tool assumes the model to be represented using Petri net. Standard approaches for conformance checking on Petri nets compute the optimal alignment between an observed trace and the most similar trace allowed by the model. However, when moving into the online scenario, due to the strictness of the rules applied, we know in advance the impossibility to achieve the same goal (i.e., find the optimal alignment). This is due to the lack of time available for backtracking operations.
while analyzing an event of the stream (in online context just constant time
operations are allowed for each event processed).

To solve this issue, we devised a two-step approach. Before starting the actual
procedure, we convert the Petri net into a transition system. To do that we
decided to generate the coverability graph.\footnote{The \textit{coverability graph},
actually, does not represent a good transition system for conformance purposes,
as it allows more behavior with respect to the original Petri
nets. Therefore, in this paper, we assume that the given Petri net is bounded.
This assumption is typically fulfilled in many real-world applications.} By restricting ourselves to bounded
Petri nets, the coverability graph corresponds to the reachability graph. Given
such transition system, we extend it with several arcs, in order to allow the
execution of every possible activity from each state. The newly introduced arcs
(i.e., arcs dealing with behavior not available in the original model) are associated
with a cost larger than 0. The details regarding the actual policy on how to
extend the set of arcs are reported in \cite{4}. All the operations described so far
are executed before the online analysis (i.e., “offline”), when the computational
complexity is not an issue yet. The resulting model, called Online Conformance
Transition System (OCTS), is used for the second phase, which is the actual
online conformance checking. In this case, each event is replayed on top of the
OCTS and the costs of the executed arc is summed to the total cost of the
running process instance. Traces with cost 0 have no deviations, whereas traces
with cost larger than 0 must contain deviations.

The entire approach is implemented in two ProM plugins\footnote{The implementation is available at \url{https://andrea.burattin.net/public-files/online-conformance/source.zip}.}. The first plugin
is in charge of extending a transition system with all possible deviations. This
is a straight implementation of the technique described in \cite{4}, which actually
leverages the plugins already available for the creation of coverability graphs.

The second plugin, instead, is in charge of the actual online conformance. The
event stream is assumed to come from a TCP/IP connection, where each event
is actually a small event log with one trace containing the actual event. This
approach is common for online process mining techniques as reported in \cite{2, 5}.
The plugin itself is actually composed of two parts. The first is a “dashboard”,
depicted in Fig. 2. Such dashboard allows to immediately grasp what is hap-
pening in the system right now and in the recent past. It is composed of two
parts: on the left-hand side running process instances are listed. They are also
color-coded by severity (i.e., cost) and can be sorted based on the update time
(most recently updated) or by severity (process instances with most errors) to
provide a snapshot of the problematic process instances. Double clicking each
instance opens a new tab, with the details of the trace (e.g., deviations). The
right-hand side of the dashboard panel contains some status charts with general
information, such as the evolution of number of errors observed every 5 seconds
(on top), the evolution of number of events per second (in the middle), the num-
ber of traces currently in memory (bottom left), and the memory consumption
of the plugin (bottom right). The second component, depicted in Fig. 3, reports

---

1 The \textit{coverability graph}, actually, does not represent a good transition system for conformance purposes, as it allows more behavior with respect to the original Petri nets. Therefore, in this paper, we assume that the given Petri net is bounded. This assumption is typically fulfilled in many real-world applications.

2 The implementation is available at \url{https://andrea.burattin.net/public-files/online-conformance/source.zip}.
the list of the most frequent deviations. By selecting an error, it is possible to see the behavioral model (not the OCTS) with the deviating paths highlighted. A similar visualization is opened when, from the main dashboard, an instance is selected. In this second case, however, only the errors referring to that process instance are reported.

In order to analyze the quality of our implementation, we simulated the stream of a process model composed of 26 tasks and 20 gateways, by using PLG2 [3]. We then simulated the model producing an unlimited event stream. Additional, we configured PLG to generate about 90 events per second. With such event stream, we tested the capabilities of our conformance checking implementation, by running the conformance checker for more than 1 hour.

---

3 The BPMN model of the process is available at https://andrea.burattin.net/public-files/online-conformance/model.pdf.
The results of the experiment are reported in Fig. 4: in total 256110 events were generated. The simulator, however, was not able to keep the configured generation pace (we used a standard office laptop machine), and the system stabilized in producing about 65 events per second. As the picture clearly shows, all generated events were processed in time by our prototype with no processing queuing.

Fig. 4: Events generated vs events processed

3 Conclusion

This paper presents a tool for the computation of conformance checking in online scenarios. The tool is actually implemented in the ProM framework as a pair of plugins. Specifically, given a process model (represented as Petri net) the tool is capable of analyzing an event stream by connecting, via TCP/IP, to the event source. The tool, then, provides a general dashboard where it is possible to analyze the current “status” of the system via general indicators (i.e., number of events per seconds collected over time, number of errors over time, which traces are the most problematic, etc.). Moreover, the tool also supports the detailed analysis of specific process instances in order to understand which problems occurred to cause the actual deviation.

A screencast showing the capabilities of the plugin is available at https://youtu.be/cPxyENKqmK4 (enable the subtitles for live description).

References