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

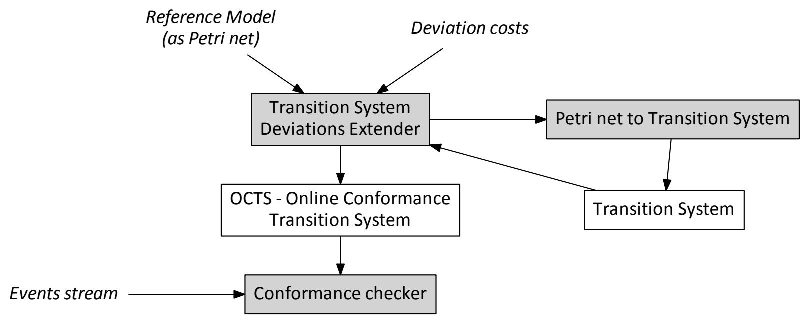

Reference Model
(as Petri net)
Deviation costs
Transition System Deviations Extender
Petri net to Transition System
OCTS - Online Conformance Transition System
Transition System
Events stream
Conformance checker

Fig. 1: Components of the online conformance checking ProM plugin. Gray boxes represent ProM plugins, white boxes represent intermediate objects, item with no border represent input elements.

most critical instances). [7] proposes a RESTful service for conformance checking, performing token replay with at *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).

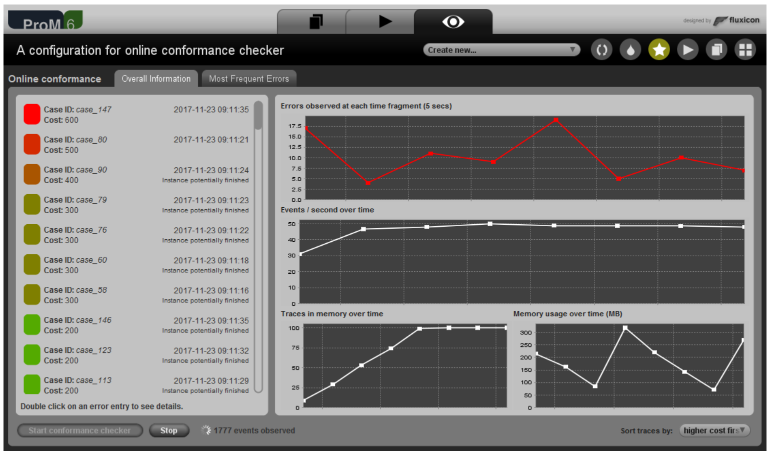
To solve this issue, we devised a two-step approach. Before starting the actual procedure, we convert the Petri net into a transition system. To do that we decided to generate the coverability graph.[1] By restricting ourselves to bounded Petri nets, the coverability graph corresponds to the reachability 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 [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[2]. 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 [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 [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 happening 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 number 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 *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 `https://andrea.burattin.net/public-files/online-conformance/source.zip`.

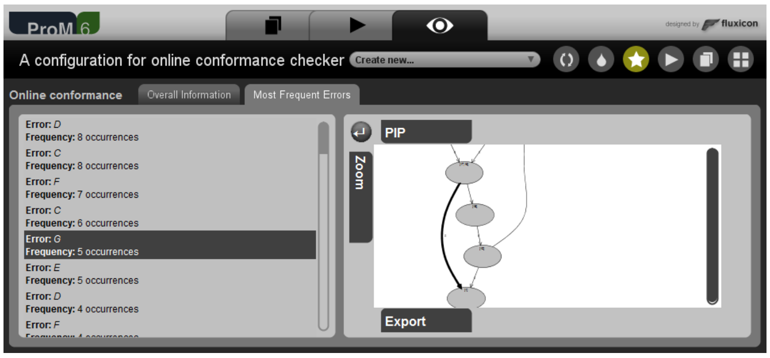

ProM 6
A configuration for online conformance checker
Create new...
Online conformance
Overall Information
Most Frequent Errors
Errors observed at each time fragment (5 secs)
Events / second over time
Traces in memory over time
Memory usage over time (MB)
Start conformance checker
Stop
Sort traces by:
higher cost first
designed by fluxicon

Case ID: *case_147* Cost: 600 — 2017-11-23 09:11:35
Case ID: *case_80* Cost: 500 — 2017-11-23 09:11:21
Case ID: *case_90* Cost: 400 — 2017-11-23 09:11:24 Instance potentially finished
Case ID: *case_79* Cost: 300 — 2017-11-23 09:11:23 Instance potentially finished
Case ID: *case_76* Cost: 300 — 2017-11-23 09:11:22 Instance potentially finished
Case ID: *case_60* Cost: 300 — 2017-11-23 09:11:18 Instance potentially finished
Case ID: *case_58* Cost: 300 — 2017-11-23 09:11:16 Instance potentially finished
Case ID: *case_146* Cost: 200 — 2017-11-23 09:11:35 Instance potentially finished
Case ID: *case_123* Cost: 200 — 2017-11-23 09:11:32 Instance potentially finished
Case ID: *case_113* Cost: 200 — 2017-11-23 09:11:29 Instance potentially finished

Double click on an error entry to see details.

1777 events observed

Fig. 2: Screenshots of the ProM plugin with the main dashboard depicted



Error: *D* **Frequency:** 8 occurrences
Error: *C* **Frequency:** 8 occurrences
Error: *F* **Frequency:** 7 occurrences
Error: *C* **Frequency:** 6 occurrences
Error: *G* **Frequency:** 5 occurrences
Error: *E* **Frequency:** 5 occurrences
Error: *D* **Frequency:** 4 occurrences
Error: *F* **Frequency:** 4 occurrences

Fig. 3: Screenshots of the ProM plugin with the routing of frequent errors

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][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.

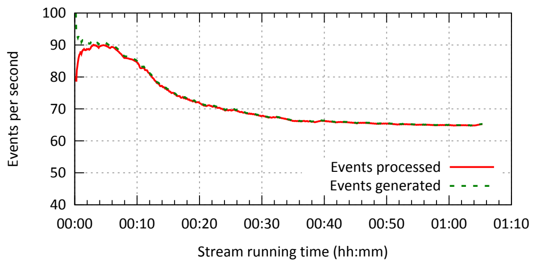


Fig. 4: Events generated vs events processed

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

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Burattin, A.: Process Mining Techniques in Business Environments, LNBIP, vol. 207. Springer (2015)
3. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: Proceedings of the BPM Demo Track 2016. pp. 1–6 (2016)
4. Burattin, A., Carmona, J.: A Framework for Online Conformance Checking (manuscript submitted to BPI 2017), `https://andrea.burattin.net/public-files/online-conformance/paper.pdf`
5. Burattin, A., Cimitile, M., Maggi, F.M., Sperduti, A.: Online discovery of declarative process models from event streams. IEEE Trans. Serv. Comput. 8(6), 833–846 (2015)
6. Munoz-Gama, J.: Conformance Checking and Diagnosis in Process Mining - Comparing Observed and Modeled Processes, Lecture Notes in Business Information Processing, vol. 270. Springer (2016)
7. Weber, I., Rogge-Solti, A., Li, C., Mendling, J.: CCaaS: Online conformance checking as a service. In: BPM Demo. pp. 45–49. Innsbruck, Austria (2015)