Know What You Stream: Generating Event Streams from CPN Models in ProM 6

S.J. van Zelst, B.F. van Dongen, and W.M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
{s.j.v.zelst,b.f.v.dongen}@tue.nl

Abstract. The field of process mining is concerned with supporting the analysis, improvement and understanding of business processes. A range of promising techniques have been proposed for process mining tasks such as process discovery and conformance checking. However there are challenges, originally stemming from the area of data mining, that have not been investigated extensively in context of process mining. In particular the incorporation of data stream mining techniques w.r.t. process mining has received little attention. In this paper, we present new developments that build on top of previous work related to the integration of data streams within the process mining framework ProM. We have developed means to use Coloured Petri Net (CPN) models as a basis for event-stream generation. The newly introduced functionality greatly enhances the use of event-streams in context of process mining as it allows us to be actively aware of the originating model of the event-stream under analysis.

Keywords: Process mining, event-streams, coloured Petri nets, ProM, CPN Tools

1 Introduction

We assume the reader to be familiar with the basics of process mining and refer to [1] for a detailed overview of the field.

Streams of events have rarely been studied within the field of process mining. Given the current state of art in information technology we are able to store huge quantities of information related to business process execution. However, classical analysis techniques are not able to cope with these quantities of data, i.e. within the ProM Framework [2][1] it is currently impossible to analyze an event log which size exceeds the computer's physical memory. Moreover, some business processes owners simply do not gain anything from static posteriori analysis. As an example, a chip manufacturer is interested in process deviation detection upon production of a batch of chips rather than after shipment of the batch to a customer. Thus, treating business process data as a dynamic sequence of events rather than a static event log is a natural next step.

In [3], we presented a standardized approach that extends the ProM framework with basic support for handling streaming data. The approach allows us to connect dynamic and volatile sources of data to the ProM framework. Although a preliminary implementation of a connection to an external source was provided, the framework has not

[1] http://www.promtools.org/

boosted the integration of stream based analysis within the process mining community. A potential cause of this could be explained by the lack of existing quality metrics for process models learned over streams. In this paper we present the integration of CPN Tools [4][2] with the existing stream framework within ProM. CPN Tools can be used to model, execute and analyze any type of discrete event systems in terms of Colored Petri Nets [5]. The connection to CPN Tools provides great flexibility w.r.t. generating streams, and, allows us to generate streams of which the actual process model is known. This in turn can greatly help in further development of established quality metrics for stream based process mining.

The remainder of this paper are organized as follows. In Section 2 we briefly touch upon the architecture and implementation of the newly developed integration. In Section 3 we demonstrate the use of the integration by means of an explanatory case study. Section 4 concludes the paper and provides pointers to interesting topics for future work. Additionally, we have recorded a screen-cast in which we discuss the integration in more detail, based on the case study [3].

2 Architecture and Implementation

The core of the integration of CPN Tools with the ProM stream framework is an *author entity* [3] that generates events and emits these onto a designated stream. The underlying connection to CPN Tools, i.e. for the purpose of simulation of a CPN Model, is handled by the Access/CPN framework [6, 7].

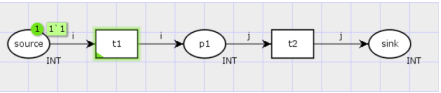
In order to generate events, the author entity needs a CPN model, an initial marking of the CPN model, a simulator object (SO) and a parameters object (PO). The PO specifies certain parameters of the author:

- The total number of times that the model should be executed starting from the initial marking, denoted by r_{max}.
- The maximum number of steps within a single execution, denoted by s_{max}.
- The emission rate of the author by specifying a delay in between the emission of two consecutive packets, denoted by e_r.
- The case identification technique. This property specifies which transitions will be emitted on the stream upon firing and how the corresponding events are identified. Currently, we have implemented two approaches being *repetition based* and *CPN variable based*.
- Event decoration. We can choose whether we want to emit all variables associated to the firing of a transition within a data packet or only the core elements, being the trace identifier and the event name.

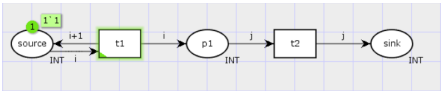
In the repetition based case, each repetition of an execution of the CPN Model is used as a basis for identifying a case. Thus all transitions fired in the first repetition will have *1* as a case identifier, all transitions fired in the second repartition will

[2] http://www.cpntools.org/

[3] https://svn.win.tue.nl/repos/prom/Packages/EventStream/Tags/publications/screen_captures/2015_bpm_demo_streams_cpn/2015_demo_cpn_tools_screen_capture.tar.gz



(a) CPN Model suitable for repetition based case identification technique.



(b) CPN Model suitable for CPN variable based case identification technique.

Fig. 1: Two CPN Model fragments representing different examples of case identification technique.

have *2* as a case identifier etc. In this identification technique, every transition that is fired will be emitted as an event where the transition name acts as an event name. As an example of a CPN Model suitable for a repetition based case identification technique, consider Figure 1a. Within the CPN model we have defined two variables of type `INT`, i.e. `var i,j: INT;`. An example stream originating from the CPN Model, where $r_{max}, s_{max} \geq 2$, including event decoration could be $S_r = \langle$ {`trace=1, concept:name=t1, i=1`}, {`trace=1, concept:name=t2, j=1`}, {`trace=2, concept:name=t1, i=1`}, {`trace=2, concept:name=t2, j=1`}, ...$\rangle$. Note that within the repetition based case, first all events related to trace *1* are emitted before events related to trace *2* are emitted, i.e. cases do not run concurrently.

In the CPN variable based approach, the user specifies a specific variable present within the CPN model to act as a case identifier. In this case, only those transitions that fire and that have the specified variable associated will be emitted to the event-stream. Consider Figure 1b which depicts a CPN model suitable for CPN variable based case identification. Again we have defined two variables, i.e. `var i,j: INT;`. If we define variable `i` as the trace identification variable, given $r_{max} \geq 1$, $s_{max} \geq 3$, a possible stream originating from the CPN Model could be $S_v = \langle$ {`trace=1, concept:name=t1, i=1`}, {`trace=2, concept:name=t1, i=2`}, {`trace=3, concept:name=t1, i=3`}, ...$\rangle$. Note that using CPN variable based case identification allows us to hide certain transitions present within the model from the stream, i.e., transition `t2` will never be emitted to the stream as it uses variable `j`.

All graphical components w.r.t. the author entity are inherited from the streaming framework presented in [3]. The only new graphical component of the Stream/CPN integration framework is the author configuration screen which provides means to select r_{max}, s_{max}, e_r, the case identification technique, and the event decoration. For an impression of the UI of the plug-in we refer to the screen-cast accompanying this paper.

3 Case Study

As an explanatory case we have designed a hierarchical CPN model that is used as a basis for stream generation. The model consists of one root model and two sub models. The root model is depicted in Figure 2[4]. The CPN model consists of two variables, i.e. `var trace, ignore: INT`. The initial marking of the root model is one token of

[4]The CPN model can be found at: `https://svn.win.tue.nl/repos/prom/Packages/EventStream/Tags/publications/data/2015_bpm_demo_streams_cpn/demo1.tar.gz`

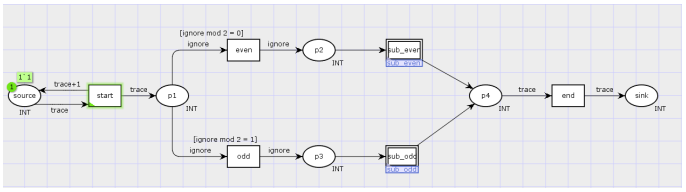
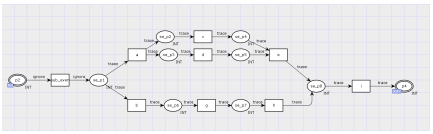
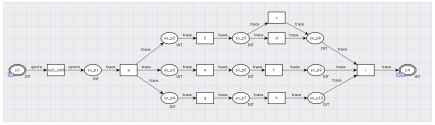


Fig. 2: Root CPN Model of the hierarchical model used within the case study



(a) CPN sub-model executed in case of a token with an even `INT` value.



(b) CPN sub-model executed in case of a token with an odd `INT` value.

Fig. 3: Two CPN sub-models used within the case study

`colset INT`, i.e. `1`1`, in place `source`. The transition labeled `start` is connected to place `source` and acts as a token generator. In its binding it uses the `trace` variable. If transition `start` fires, it produces a token with the value of `trace` in place `p1` and it produces a token with value `trace + 1` in place `source`. All tokens with an even `INT` value will be routed to the sub-model named `sub_even` whereas all tokens with an odd `INT` value will be routed to the sub-model named `sub_odd`. In routing to the sub-models the variable `ignore` is used. The two sub-models are depicted in Figure 3.

After importing the hierarchical model in the ProM framework, we configure an event-stream with the following parameters: $r_{max} = 1$, $s_{max} = \infty$, $e_r = 50\ ms.$, case identification = *CPN Variable* with value `trace` and event decoration is `true`. After the event-stream object is created we connect a stream-based implementation of the Inductive Miner [8]. After receiving a number of events, the stream-based Inductive Miner returns the Petri net depicted in Figure 4.

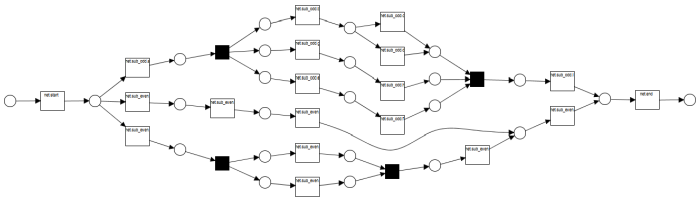


Fig. 4: Result of applying a stream-based implementation of the Inductive Miner to the event-stream generated by the hierarchical CPN model.

Although the stream-based miner is not able to discovery hierarchy the resulting model aligns rather acceptable with the input model, i.e., from a control-flow perspective it exactly describes all possible traces that are emitted onto the event-stream.

4 Conclusion

The newly presented CPN extension of the stream framework within ProM enhances researchers, business users, and developers to experiment with the concept of streaming data within a process mining context. The extension allows the user to import a CPN model, using any concept present within CPN tools, i.e. time, hierarchy etc., within ProM. The user is able to specify several parameters of the accompanying stream such as emission rates, event decoration and the trace identification technique.

An interesting direction for future work concerns support for the use of multiple case identification variables. This allows us to discover multiple perspectives of the model under study. Another interesting direction is the development of a stream evaluation framework which allows us to manipulate certain elements of the stream, e.g. case arrival rates, throughput time, etc., in order to investigate the impact of these parameters w.r.t. the stream-based algorithm under study.

References

1. Aalst, W.M.P.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer Publishing Company, Incorporated (2011)
2. Dongen, B.F.v., Alves de Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., Aalst, W.M.P.v.d.: The prom framework: A new era in process mining tool support. In Ciardo, G., Darondeau, P., eds.: Applications and Theory of Petri Nets 2005. Volume 3536 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 444–454
3. Zelst, S.J.v., Burattin, A., Dongen, B.F.v., Verbeek, H.M.W.: Data streams in prom 6: A single-node architecture. In: Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014. (2014) 81
4. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. STTT **9**(3-4) (2007) 213–254
5. Jensen, K., Kristensen, L.M.: Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer (2009)
6. Westergaard, M., Kristensen, L.M.: The access/cpn framework: A tool for interacting with the CPN tools simulator. In: Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, June 22-26, 2009. Proceedings. (2009) 313–322
7. Westergaard, M.: Access/cpn 2.0: A high-level interface to coloured Petri net models. In: Applications and Theory of Petri Nets - 32nd International Conference, PETRI NETS 2011, Newcastle, UK, June 20-24, 2011. Proceedings. (2011) 328–337
8. Leemans, S.J.J., Fahland, D., Aalst, W.M.P.v.d.: Discovering block-structured process models from event logs - A constructive approach. In: Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings. (2013) 311–329