

Discover Context-Rich Local Process Models (Extended Abstract)

Mitchel Brunings¹, Dirk Fahland¹ and Eric Verbeek¹

¹Eindhoven University of Technology, Mathematics and Computer Science, Eindhoven, the Netherlands

Abstract

We introduce a new ProM plugin called *Discover Context-Rich LPMs* which mines a log for large local process models (LPMs) based on *supported words*. The main advantage of this plugin is that it produces much larger and much fewer LPMs than other tools. The plugin is packaged with an additional plugin called *Generate HTML coverage report* which calculates the coverage of LPMs along with several other quality measures. This extra plugin is useful to select and improve a *set of LPMs*.

Keywords

Sets of LPMs, coverage, ProM, process mining, process analytics, interactive process discovery

1. Introduction

Discovering a single process model from a large and complex process often yields an undesirable model. Instead, one could discover a set of local process models (LPMs). By limiting the scope of individual LPMs to different smaller (local) sections of the process, they end up smaller and therefore more comprehensible. A complete set of LPMs can in turn give insights into the behavior of the entire process.

Several tools exist to discover LPMs from event data [1, 2]. These tools generate 1000s of LPMs of limited size; the largest LPMs produced by [1] have about 4 transitions, and the LPMs produced by [2] are limited to 10 transitions by their method's window size. In [3] we argue that we need fewer and larger LPMs for humans to make sense of a set of LPMs. We present a ProM implementation of an alternative approach [4] to discovering LPMs resulting in sets of fewer but larger LPMs in line with [3].

We implemented two ProM plugins: *Discover Context-Rich LPMs* and *Generate HTML coverage report* with the in- and outputs sketched in Fig. 1.

2. Discover Context-Rich LPMs

We first discover context-rich LPMs using the following 4 steps, described in detail in [4].


1) *Mine supported words*. This step is based on an algorithm developed in [5]. It efficiently finds all gapped subsequences (words) that occur in a log at least as often as a user-defined

ICPM 2022 Doctoral Consortium and Tool Demonstration Track

✉ m.d.brunings@tue.nl (M. Brunings); d.fahland@tue.nl (D. Fahland); h.m.w.verbeek@tue.nl (E. Verbeek)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

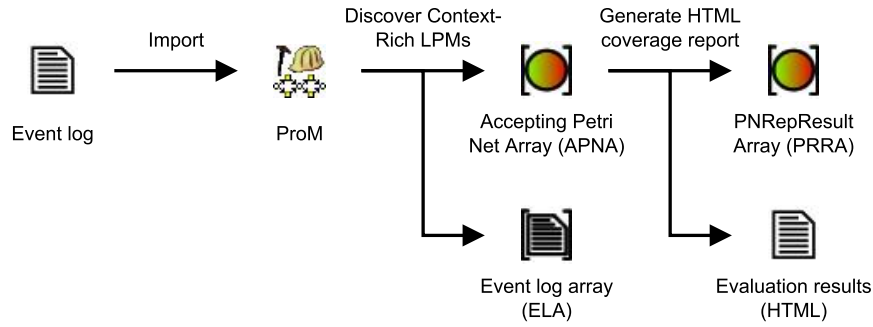


Figure 1: The process of generating LPMs and the coverage report from a log in ProM.

threshold t . For our implementation, we set t to 10% of the number of occurrences of the most frequent event class.

2) *Group by prefix of subsequence.* Next, we group the supported words by the first n events in each word. The default value $n = 1$ groups words by their starting activity only, values $n > 1$ distinguish words by a longer prefix. This allows grouping all frequent sub-behaviors of a process that begin in the same way.

3) *Filter words.* Then, we look at each group of words. There may be pairs of words in a group where the longer word contains the shorter. It is impossible for short words to occur less often than long words that contain them, but they may occur more often. When we see such a long and short word occur (almost) equally often, the short word does not add (much) information. If the short word occurs much more often, then the part we only see in the long word can apparently be skipped. We use a support delta d (by default equal to t) to determine how much more often we need to see a short word to keep it. Thus, for each pair of words where one contains the other, we discard the shorter of the two if it does not occur at least d more times than the longer one. What remains will be a set of words that we know are each important to represent in an LPM for this (local) part of the process.

4) *Discover models.* We now treat each set of filtered supported words with the same prefix as a local log and apply a process discovery algorithm to obtain an LPM of this local log. We already did all selection and filtering in the previous steps, so we should use an algorithm that guarantees 100% fitness. Therefore, we choose to use the basic Inductive Miner [6].

Results. Using this plugin on the BPIC2012 log [7] returned a set of 26 LPMs (Accepting Petri net Array) in about 8 minutes on a laptop with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz and with 4.0 GB RAM @ 2400 MHz allocated to ProM. The mean number of labeled transitions for these LPMs is 9.2, the median is 7.5, the largest is 24. This shows that we can find a much more reasonable number of LPMs (10s instead of 1000s) and much larger LPMs (more than twice as large) in roughly equal running time as the other tools [1, 2].

3. Selecting subset of desired LPMs

In [3] we show that a desired property of a set of LPMs is that it describes all events in the data, i.e., all events are covered. We implemented “Generate HTML coverage report” to calculate

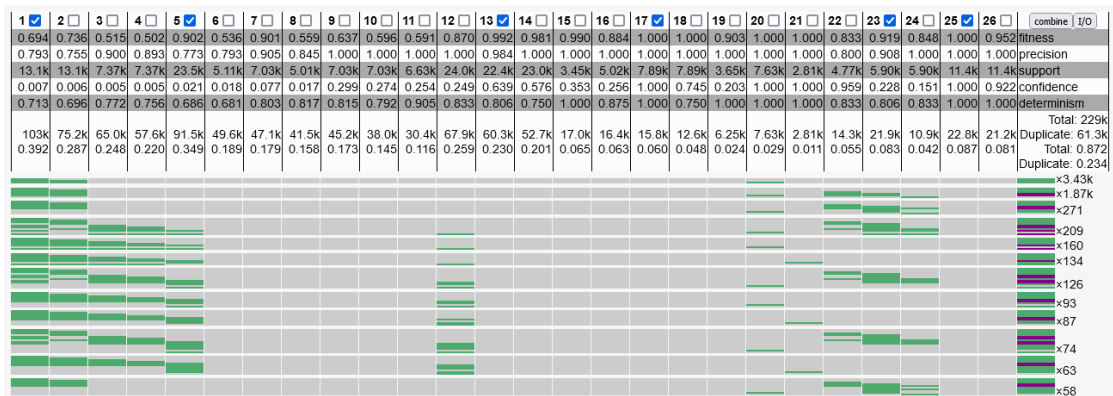


Figure 2: A fragment from the top of the coverage report for the LPMs discovered from BPIC2012 [7].

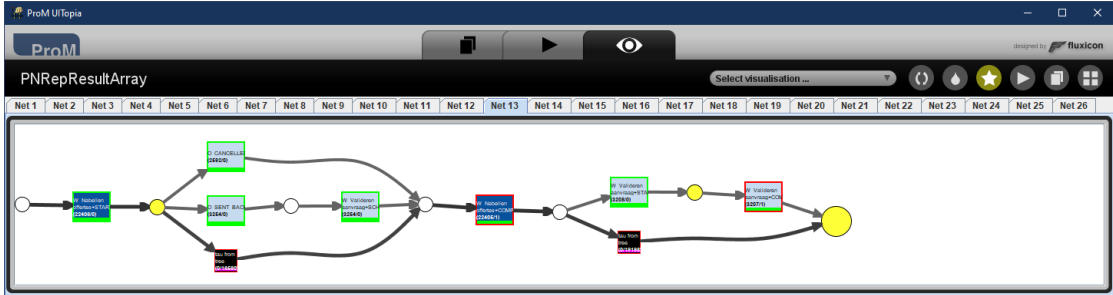


Figure 3: LPM 13 of 26 after replaying the set of trace fragments from BPIC2012 [7].

the coverage of LPMs along with several other quality measures. It first evaluates each model individually and then combines the results into a single report. For each LPM, we split the original log into subsequences bounded by the first and last activity of the LPM, and project onto the activities present in the LPM. We treat this set of subsequences as a log which we replay on the LPM. From the replay we extract measures such as fitness and precision. The plugin then overlays each discovered LPM with replay information as shown in Fig. 3 and opens a local browser to show a coverage report as shown in Fig. 2. This plugin took about 1 minute on the LPMs discovered from the BPIC2012 log with the technique of the previous section.

The coverage report shows one column for each LPM, in the same order as in ProM. The top rows provide statistics per LPM and a selection box to pick LPMs to combine into a set. The coverage of each LPM over each trace variant is shown below by colored rectangles. Each LPM (column) either covers (green) or does not cover (gray) events (2 pixels high) in the trace variants (rows, with frequency on the right). The right-most column shows the combined coverage where gray means no LPM in the selected set covers that event, green means exactly one LPM covers that event, and purple means more than one covers it. The row right above the coverage visualization provides coverage statistics as both absolute numbers and ratio of the total events in the log.

Fig. 2 shows that selecting the LPMs with ticked boxes gives a coverage of 0.872 with 0.234

duplicate. The analyst can (de-)select LPMs to maximize total coverage and minimize duplicate coverage; selecting all LPMs on BPIC2012 has 0.951/0.815 total/duplicate coverage.

4. Links

Downloads and installation instructions, screencast, and source code can be found here: <https://svn.win.tue.nl/repos/prom/Documentation/LPMSupportedWords/index.html>

5. Conclusion

The presented tool provides a first step for interactively constructing meaningful *sets of LPMs* [3] by discovering far fewer but larger LPMs from which an analyst can choose relevant LPMs based on various quality criteria, in particular coverage. This is similar in a sense to *sub-model freezing* [8], as the user is in control of what the final model looks like. After using the tool, a user should still perform manual selection and cleanup of LPMs.

Further developments should aid the user by suggesting “optimal” sets of LPMs. However, determining “optimal” requires further research on quality measures for LPM (c.f. [3]). We also argue the need for notation describing how LPMs in a set relate to each other globally.

References

- [1] N. Tax, N. Sidorova, R. Haakma, W. M. van der Aalst, Mining local process models, *Journal of Innovation in Digital Ecosystems* 3 (2016) 183–196.
- [2] V. Peeva, L. L. Mannel, W. M. van der Aalst, From place nets to local process models, in: *Petri Nets*, Springer, 2022, pp. 346–368.
- [3] M. D. Brunings, D. Fahland, B. F. van Dongen, Defining meaningful local process models, in: *ToPNoC XVI*, Springer, 2022, pp. 24–48.
- [4] M. D. Brunings, *Discovering Context-Rich Local Process Models*, Master’s thesis, Eindhoven University of Technology, 2018.
- [5] D. Fahland, D. Lo, S. Maoz, Mining branching-time scenarios, in: *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2013, pp. 443–453.
- [6] S. J. Leemans, D. Fahland, W. M. van der Aalst, Discovering block-structured process models from event logs—a constructive approach, in: *Petri Nets*, Springer, 2013, pp. 311–329.
- [7] B. F. van Dongen, *BPI Challenge 2012*, 2012.
- [8] D. Schuster, S. J. van Zelst, W. M. van der Aalst, *Sub-model freezing during incremental process discovery in cortado*, 2021.