μSystems Research Group
School of Electrical and Electronic Engineering
Merz Court
Newcastle University
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

Process Mining using Parameterised Graphs

Andrey Mokhov, Josep Carmona

andrey.mokhov@ncl.ac.uk, jcarmona@cs.upc.edu

NCL-EEE-MICRO-MEMO-2014-009

August 2014

Abstract

This is a brief report on Andrey Mokhov's visit to Josep Carmona's research lab in Universitat Politècnica de Catalunya. The report summarises potential benefits that Parameterised Graphs (PGs) [1] can bring to the domain of process mining [2] and outlines important challenges to be addressed in future research. The authors intend to elaborate on the presented ideas in follow-up peer reviewed publications.

1 Motivation for using PGs in process mining

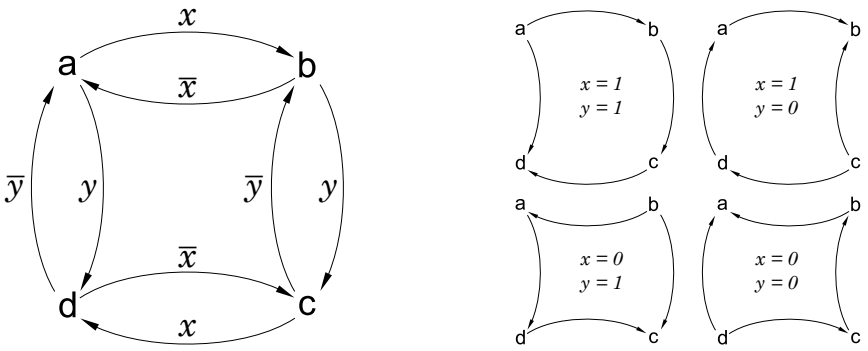
In this section we discuss key reasons that motivate us to study the application of PGs in process mining, namely: (i) their ability to compactly represent complex behaviours without excessive over-generalisation, and (ii) the possibility of capturing event log meta data as part of PG representations and taking advantage of the data for the purpose of explaining the processes under observation.

Note that in this report we work with PGs whose equivalence is considered up to the transitive closure. Such PGs have historically been studied under the name of *Conditional Partial Order Graphs* [3].

1.1 Compact representation

We have performed a number of experiments with state-of-the-art process mining software and found out that the existing solutions often produce too general solutions for event logs corresponding to processes with non-trivial mix of concurrency, causilty and choice.

One example of such an event log is $L = \{abcd, cdab, badc, dcba\}$. One can notice that in this example the order between events a and b always coincides with the order between events c and d, which we consider an important piece of information about the process. The existing process mining methods, however, fail to capture this information and produce the most general explanation for the process in which all four events are concurrent.



(a) PG representation (b) Four projections of the PG

Figure 1: Exact PG representation of log $L = \{abcd, cdab, badc, dcba\}$

One reason for this could be the fact that constructing both compact and exact representation of L in models like Petri Nets leads to very complex and practically incomprehensible results. PGs, however, can represent L exactly in a very compact form, as shown in Fig. 1(a). One can easily verify that this representation is exact by trying all possible assignments of variables x and y and checking that they generate traces $\{abcd, cdab, badc, dcba\}$ as expected. See Fig. 1(b) for the corresponding illustration. The compactness of the exact PG representation of L is not just a lucky accident; we have consistently observed such results during our experiments. Therefore, we believe that this property of PGs is a strong motivation for their further study in the process mining context.

It is worth mentioning that PGs allows us to recognise *second order relations* between events. These are relations that are not relating events themselves, but are relating relations between events: indeed, the PG in

Fig. 1(a) clearly shows that the relation between a and b is equal to the relation between c and d, and the same holds for pairs (a,d) and (b,c). In principle, one can go even further and consider third order relations and so forth. The practical use of such a relation hierarchy is that it may help to extract an event hierarchy from event logs, thereby simplifying the resulting representation even further. The current example, however, demonstrates that such an event hierarchy is not always unique. Indeed, it is possible to come up with two reasonable event hierarchies: (i) grouping events $\{a,c\}$ and $\{b,d\}$ into *second order events* that are related according to variable x, and (ii) grouping events $\{a,b\}$ and $\{c,d\}$ that are related according to variable y.

One may be unsatisfied by the PG representation in Fig. 1(a) due to the use of 'artificial' variables x and y. Where do they come from and what exactly do they correspond to in the process? We found out that meta data which is often present in event logs can be used to answer such questions. In fact, as we will show in the next subsection, it may be possible to use easy-to-understand predicates constructed from the data instead of rather 'opaque' variables.

1.2 Capturing data

Event logs used in process mining often come with meta data attached to individual events or whole traces. For example, a particular event in a paper reviewing process can be tagged by the identity of a person who was responsible for it, e.g., 'paper submitted (authors: Alice and Bob)', 'review received (reviewer: Carl)', etc.

Coming back to our previous example, consider the same log L but augmented with temperature data attached to traces:

- $abcd$, $t = 25^\circ$
- $cdab$, $t = 30^\circ$
- $badc$, $t = 22^\circ$
- $dcba$, $t = 23^\circ$

With this information at hand we can now explain what variable x means. In other words, we can open the previously 'opaque' variable x by expressing it as a predicate involving parameter t:

$$x = t \geq 25^\circ.$$

One can easily verify that this equation is consistent with the above event log and its PG representation in Fig. 1(a). One can subsequently drop x completely from the PG by using conditions $t \geq 25^\circ$ and $t < 25^\circ$ in place of x and $\bar{x}$, respectively.

We have preliminary ideas on how to derive PGs with 'transparent' conditions using data embedded in given event logs, as outlined in the next subsection. This is one of the most intriguing directions of the future research.

1.3 Using meta data in PG mining

The simplest approach to handling meta data is to consider each data 'tag' as an event, which is unordered with respect to other events. In other words, we are only concerned with the fact of occurrence of such an event, but not with its order. By applying this approach to the above example, we get the PG shown in Fig. 2. In addition

to normal events $\{a,b,c,d\}$ the PG contains new *data events* $\{t = 22^\circ, t = 23^\circ, t = 25^\circ, t = 30^\circ\}$ that are not connected to any other events, but are labelled with Boolean conditions. Notice that conditions x and $\bar{x}$ have been replaced with $x_3 \vee x_4$ and $x_1 \vee x_2$, respectively, in other words we have explained the meaning of variable x by using meta data from the log.

The final simplification step is to realise that it is possible to use a threshold value (e.g., 25°) to express conditions $x_1 \vee x_2$ and $x_3 \vee x_4$ in terms of parameter t directly, without involving intermediate Boolean variables $x_1..x_4$. This leads to simple conditions $t \geq 25^\circ$ and $t < 25^\circ$. We believe it is possible to automate this procedure both for continuous and discrete data parameters.

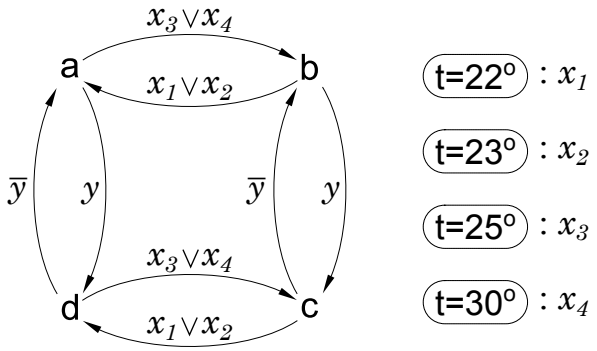


Figure 2: PG mining using meta data

2 Extracting exact concurrency relation

One of the most important steps in process mining is extraction of the concurrency relation from a given event log. The classic methods, such as the α-algorithm [2], often extract non-exact concurrency relation aiming to simplify the resulting representation, but also potentially losing important information about the process, as has been demonstrated in the previous section. The compactness of PG representations motivates us to study more exact methods of concurrency extraction. In particular, one idea is based on a generalisation of a so-called *reveals relation*, which was introduced in [4].

Fig. 3 briefly illustrates the idea. The PG shown in the top left box shows two pairs of possibly concurrent events: $\{a,b\}$ and $\{c,d\}$. The classic methods of concurrency extraction would consider them concurrent. However, we point out that occurrence of event e reveals the order in pairs $\{a,b\}$ and $\{c,d\}$, in particular, e occurs only in those traces where event b precedes event a and event d precedes event c since $x \wedge y \Rightarrow x$ and $x \wedge y \Rightarrow y$. The bottom left box shows a different situation, in which the order between events a and b cannot be revealed, therefore one can safely assume that these events are concurrent. Events c and d, however, cannot be made concurrent, because their order is revealed by both events e and f.

2.1 Generating negative examples

The reveals relation can be used for generating negative examples of traces, i.e., such traces that should never occur in the process according to a given log. Such negative examples may be useful for other process mining methods for improving their accuracy. For example, trace *abcde* would contradict the PG representation shown

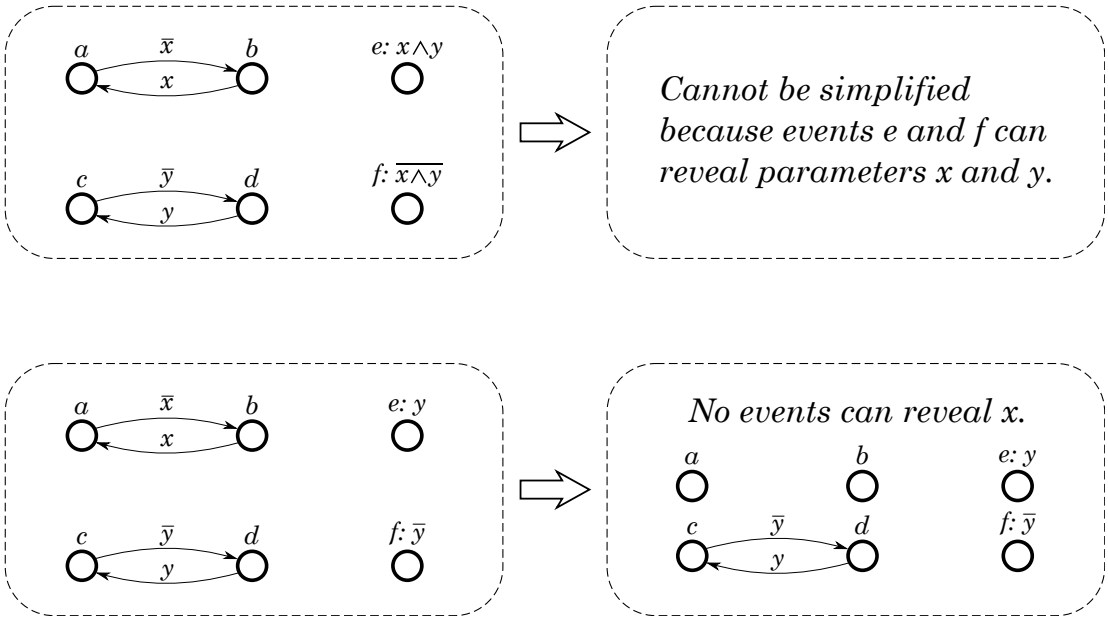


Figure 3: Revealing absence of concurrency

in the top left box of Fig. 3, because event e can occur only if b precedes a and d precedes c.

3 Challenges

There are still many challenges to be solved before PG-based process mining can become a viable alternative to established methods. Below we list some of them together with possible solutions.

3.1 Dealing with cyclic behaviour

PGs are particularly well suited for compact representation of partial orders, which are acyclic objects. Unfortunately most processes are cyclic in nature which makes it problematic to model them with PGs. We have identified several possible workarounds for this problem, for example, use of 'tandem-repeats' that can be used to identify cycles at the pre-processing stage of process mining and removing them from traces. The obtained PG representation can later be annotated with the information about cycles.

Other workarounds include the use of dynamic PGs where events are allowed to change the variables [5], as well as the introduction of cut-off events in PGs similar to Petri Net unfoldings.

3.2 Scalability

The current PG encoding algorithms cannot handle large event logs, therefore new algorithms will need to be developed with scalability in mind. We have identified a promising direction for overcoming this challenge that is based on reducing the PG encoding problem to the problem of FSM synthesis [6].

3.3 Generating BPMN representations

BPMN is a widely adopted standard for process description. It is our aim to support BPMN representations by deriving them from PGs. This may be implemented by using one-hot encoding for identification of inclusive (OR) and mutually exclusive (XOR) blocks of behaviour. In addition to one-hot literals one can also allow their conjunction to be used in PG conditions in order to express hierarchy of BPMN constructs. Finally, we believe that meta data can help generate efficient PG encodings: one can create encoding variables directly from meta data as discussed in Section 1.2.

4 Conclusion

This is the first attempt to apply Parameterised Graphs in the field of process mining and there is still a lot of work to be done. This brief report is by no means complete; our intention was merely to summarise our early findings on this topic in hope that a reader might find them interesting. We will be happy to receive comments, suggestions and criticism – please use the email addresses provided on the title page of this report.

Acknowledgements

We would like to thank Jordi Cortadella for the insightful discussion on the PG encoding problem and for his help in arranging Andrey Mokhov's visit to Universitat Politècnica de Catalunya.

References

[1] A. Mokhov and V. Khomenko. Algebra of Parameterised Graphs. *ACM Transactions on Embedded Computing*, 2014 (in print).

[2] Wil van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[3] A. Mokhov and A. Yakovlev. Conditional Partial Order Graphs: Model, Synthesis and Application. *IEEE Transactions on Computers*, 59(11):1480–1493, 2010.

[4] Stefan Haar, Christian Kern, and Stefan Schwoon. Computing the reveals relation in occurrence nets. *Theor. Comput. Sci.*, 493:66–79, 2013.

[5] A. Mokhov, D. Sokolov, and A. Yakovlev. Adapting Asynchronous Circuits to Operating Conditions by Logic Parameterisation. *Int'l Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 17–24, 2012.

[6] Tiziano Villa, Timothy Kam, Robert K Brayton, and Alberto L Sangiovanni-Vincentelli. *Synthesis of finite state machines: logic optimization*. Springer Publishing Company, Incorporated, 2012.