Multi-paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences (Short Paper)

Johannes De Smedt*, Jochen De Weerdt, and Jan Vanthienen

KU Leuven Faculty of Economics and Business
Department of Decision Sciences and Information Management
Naamsestraat 69 B-3000 Leuven, Belgium
`firstname.lastname@kuleuven.be`

Abstract. Business process mining is a well-established field of research which focuses on the automatic retrieval and analysis of process flows. The discovery and representation of these models is based on techniques that come in all shapes and forms. Most notably, procedurally-based algorithms such as Heuristics Miner have been used successfully for this purpose. Also, declarative process model miners have been proposed, which give other insights into the model by generating rules that apply on the activities. This paper proposes an integrated approach to combining these paradigms to discover process models that contain best of both worlds to enrich insights into the event logs under scrutiny.

Keywords: Business Process mining, Mixed-Paradigm mining, Causal Nets, Declare.

1 Introduction

Over the last decade, the field of process mining has gained a lot of traction. Its main focus lies on the automatic retrieval and subsequent analysis of business process models and insights from data logs containing events [1]. The three pillars of process mining focus on process discovery, enhancement and conformance checking. The former can be considered the primordial task in a process mining exercise, with the two latter pillars typically building on it. The goal of process discovery is to learn a process model from data in the form of an event log in the most comprehensive, comprehensible and correct way. In order to do so, many mining techniques have been proposed, including, amongst others, Alpha Miner and Heuristics Miner [1]. The representation form used by these models are procedural models.

More recently, declarative process modeling and mining has gained popularity and several discovery techniques such as, e.g., Declare Miner [2] have been

* Corresponding author.

implemented for discovery purposes. These miners retrieve rules from process logs to create models with a more flexible view on the information contained in the log, as any behavior that is not strictly forbidden is allowed.

This paper presents an algorithm for combining procedural and declarative constraints in one map and shows results that indeed provide a new way of looking at mined processes. Both process mining approaches offer different characteristics that each enlightens certain aspects of an event log, but in the literature, the combination of both paradigms into one discovery algorithm has received very little attention so far. Nonetheless, the prospect of learning richer, multi-paradigm process models seems promising, especially in the context of semi- or unstructured processes. Our proposed approach has been implemented in ProM[1] as a plug-in which builds on the two most frequently used process mining techniques for each paradigm, (Flexible) Heuristics Miner and Declare Miner.

The remainder of this paper is structured as follows. The second section covers the related work, followed by a running example which uncovers some issues found for current approaches in section three. Section 4 contains a comparison of and opportunities for combined mining approaches. Next, section five provides an overview of the implementation, followed by an evaluation section with results for the running example. The last section concludes the paper with a discussion and future work.

2 Related Work

Within the field of process mining, a strong emphasis is put on the automatic retrieval of business process models from event logs. Numerous techniques have been proposed, such as Alpha Miner, Heuristics Miner, Fuzzy Miner and a Genetic Miner [1]. These algorithms offer mining solutions that deal with aspects such as the trade-off between recall, precision, generalization, and the presence of noise in the event log. Initially, procedural process mining approaches were put forward. They capture sequence constraints and parallelism by incorporating information supporting adjacency and (direct) succession in a process log, extended with (X)OR- and AND-split and -join information. The techniques represent their outcomes often in process models such as Petri nets [3].

More recently, the modeling and mining of flexible process models has gained popularity among researchers. The most prominent declarative control flow modeling frameworks are DecSerFlow [4] and its successor Declare [5], which offer a set of Linear Temporal Logic (LTL)-based constraint templates for modeling and rule verification purposes, bundled in the ConDec language. Many declarative process discovery algorithms have been developed, such as [2,6,7,8].

A real mixed-paradigm outcome, however, has not been pursued yet as such, with the very recent exception of [9] in which the authors break down the event log into a hierarchy and mine the different subprocesses according to the appropriate paradigm. While the authors propose an approach based on counting

[1] `http://www.processmining.org/`

predecessors and successors which is somewhat comparable to direct succession in Section 4, applying a threshold on the exact number of predecessors and successors seems rather coarse for deciding on the structured versus unstructured nature of activities, especially for smaller logs. Therefore, our approach includes a configurable threshold based on the concept of entropy, which allows for making a more versatile trade-off between structured and unstructured behavior. In addition, the mandatory hierarchical structure of the mined hybrid model puts a limitation on its application to event logs where structured and unstructured behavior are much more intertwined. The approach presented in this paper does not presume such a hierarchical structure. Finally, the visualization in [9] seems strongly disjoint while our technique is capable of representing any mixture of procedural and declarative constraints in a single mined process model.

3 Running Example

As a running example, we provide the simple multi-paradigm process model in Figure 1. Both declarative ConDec constraints and more sequentially-based Petri nets are combined to resemble the progress of a PhD student throughout his career, which contains the strict order of a first and second seminar followed by the defence. Meanwhile, he/she creates content which is subsequently published in journals or presented at a conference, resembled by the *Alternate Precedence* constraints. This constraint expresses that both *Journal Paper* and *Conference* can happen after *Content Creation*, and again only after the next occurrence of the *Content Creation* activity. The first seminar cannot happen before a first contribution to a conference and the second seminar has to be preceded by a journal publication. Note that, while the multi-paradigm model is fairly simple and understandable, the simulated event log presents characteristics that are typically found in real-life logs originating from complex processes.

In order to assess the ability of procedural miners to retrieve the complex, multi-paradigm, and flexible relations between these activities, we enacted the model and mined the simulation log with (Flexible) Heuristic Miner. The algorithm is unable to retrieve the exact position and relation of the three activities *Content Creation*, *Conference* and *Journal Paper* as shown in Figure 2. While Heuristics Miner captures loops and invisible events to support the quite random appearance of *Content Creation*, it fails to capture the relation of *Journal Paper* and *Second Seminar*. The model does not support the *Precedence* constraint and leaves no room to repeat the process after firing *Journal Paper*. Furthermore, the model is cumbersome to read due to the large number of invisible tasks needed to express the flexible nature of the relations. Note that for Heuristics Miner it is possible to use other configurations, e.g., one with a lower dependency threshold. This would result in a model that better captures the behavior in the event log, but this solution would include a very generic model in which every transition can be executed in any order.

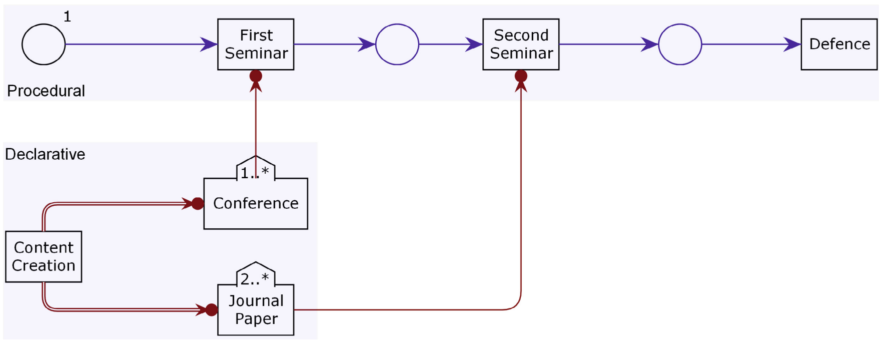

Fig. 1. A multi-paradigm process model representing a PhD student's progress flow. The model contains procedural behavior in the form of Petri net fragments, supplemented with Declare constraints.

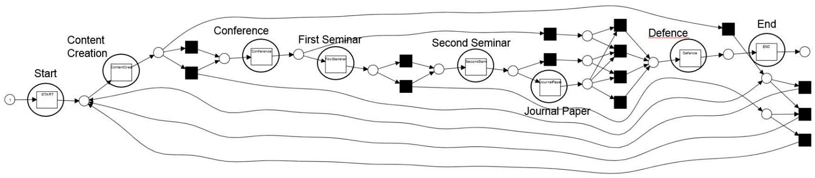

Fig. 2. A Petri net retrieved from the mined causal net produced by Flexible Heuristics Miner (default settings)

4 Advantages of a Combined Mining Approach

When comparing approaches for both paradigms, one of the main differences can be found in the granularity of capturing relations between activities. While procedural miners' outcomes rely mainly on rather local information, declarative constraints capture rules supported on trace level, which can be interpreted as rather global information of the process. Both types of results carry different control flow knowledge, which can be combined to gain different insights. Consider this simple example: trace (a, b, c, a, a, b) fulfills both a *Succession(a,b)* and a local direct succession constraint ($a \to b$ or $a > b$, as often used in procedural process mining algorithms such as Heuristics Miner) between a and b, while (b, c, a, b, c, a, a, b) only fulfills the local constraint. Either the constraint is too restrictive for (the log which contains) this trace, or the model can be pruned by using the *Succession* template as counter evidence for the direct succession. A procedural miner would include the relation between a and b as $a \to b$, while a declarative miner would derive a *Response* constraint which lacks the *Precedence* part of a *Succession*. As such, it is better capable of deriving different types of relations, and is more expressive in this respect.

The main benefits of mining a multi-paradigm process model can be summarized as follows. On the one hand, a procedural model can benefit from the addition of declarative constraints in order to uncover relations between activities that previously remained hidden. In this sense, the declarative constraints enhance the model which can better fit the parts of the log that were previously hard to capture. Since rules are defined over the full execution path, they are also better suited to represent, amongst others, duplicate tasks and long-distance dependencies. Furthermore, flexible parts of a log that are not captured (well) by procedural models (as they remained either too restrictive or too general) can be represented with declarative constraints to retrieve them in a more correct and readable way. Although capturing flexible behavior might be possible with procedural models, the sequential information would end up in a very convoluted and unstructured graph of loops, splits and joins, and arrows pointing every direction due to the ad-hoc appearance of activities as can be seen in Figure 2. Since most Declare rules represent behavior that can be labeled as non-trivial token games, they are better able to retrieve such parts of an event log. For example, expressing *Alternate Precedence* in a Petri net is a challenging task, leading to the usage of artificial model constructs to approximate the same state space.

On the other hand, declarative process models can benefit from the structuring and representation that procedural model discoverers offer, thus making represented flows more readable and more defined where no flexibility is needed, i.e. for a very fixed process sequence. In order to address these synergies between both paradigms, our approach starts from two sets of activities. The activities A in log L are divided in sets R and P, the declarative and procedural activities respectively. Note that $A = R \cup P$. By searching for Declare rules and procedural relations between both sets, the algorithm proposed in Section 5 captures the information that resides in the log by using both mining approaches.

5 Multi-paradigm Miner

The Multi-Paradigm Miner implementation[2] is based on the combination of Flexible Heuristics Miner (the Causal Net Retrieval implementation) and Declare Miner. Starting from the dependency graph, it identifies activities that are connected to a relatively higher number of other activities in the graph, as these can be considered a causal factor of the increase in potential process behavior. As such, we target them for inclusion in the set of activities that are subject to Declare mining in the second part of the discovery process. Finally, the mining result is displayed in a model which contains all behavior, which can be pruned optionally.

The Algorithm. By analyzing the strength of the direct succession metric (used in Heuristics Miner, which employs a certain threshold d called Dependency threshold for this purpose) between activities, one can retrieve the activities most closely related in a small window. Activities that have a lot of other

[2] The implementation and high resolution figures can be found at `j.processmining.be/multiparadigmminer`.

activities connected or are somewhat but not strongly connected, are candidates to be placed in the set of declarative activities $R \subseteq A$. Others that have few but strong connections to neighboring activities, are candidates to remain in the procedural basis of the model $P \subseteq A$. Phrased differently, we target activities with unclear direct succession relations, which can be an indication of the ad-hoc all-over-the-place occurrence of this activity, which results in non-structured and cluttered up sequential process models. Note that this approach also often captures the activities that cannot be fitted into the model and thus puts tasks that are connected only when the "All activities connected" option is chosen in Heuristics Miner in R.

To check for such activities we propose a metric called Activity Entropy (AE) which captures the average of the direct succession (DS) metrics between an activity and the others in the log where the dependency threshold d is not met. In other words, it captures weak dependencies. Procedural activities in a log will have a very low activity entropy, as most of the connections will be either strong ($> d$) or non-existing (close or equal to zero). Based on a given threshold e which is an input parameter of Multi-Paradigm Miner, a proportion of the log is withheld. The different values AE_i are ranked and $\lfloor |L|(1-e) \rfloor$ activities are kept in the sorted set E. Furthermore, if there is a gap of $1/e$ between the values for AE of two activities in E, the activities ranked below the gap are removed. This procedure is established to avoid introducing too many activities in R and as a consequence possibly too many rules between them. Note, however, that a fully declarative model can be obtained by using 1 for e.

6 Evaluation

In this section, the results of our experiments are presented as to evaluate the capabilities of our approach. The following representation is used in the mixed Declare and Causal Graph figures:

- The full (blue) arcs represent the procedural behavior as introduced by Heuristics Miner. They form the Causal net of the model.
- The checkered activities exceed the entropy threshold.
- The dark activities (filled in red) fulfill the *Exactly1* constraint.
- The light activities (filled in gray) fulfill the *Existence* constraints.
- The striped arrows represent ConDec constraints, which are labelled.

In short, all checkered model constructs are the outcome of the Declare mining, while the full lines and activity borders are part of the Causal net.

The PhD Process. Figures 3 and 4 show the discovered multi-paradigm models in ProM. Even for a small entropy value (Figure 3), the activity *Content Creation* becomes subject to Declare constraint mining. By its constant enabledness it can appear anywhere in the workflow and clutter up a sequential process. By retrieving a few rules for the activity, we are able to represent it in a sense-making way in a mixed model. The constraints for single activities are always applied, as they can only improve the understanding of the model. Since we use a simulated example, we apply a rule support of 100%. The model is already capable of capturing the initial model more correctly, as the relationships between *ContentCreation* and the other activities are correct. The arc between *SecondSeminar* and *JournalPaper* is still incorrect.

By raising the entropy level (Figure 4), more activities are added to the declarative set R, in this case *Conference* and *Journal Paper*. This makes sense given the model. Only constrained by the appearance of *Content Creation*, these activities are also rather unpredictable. Note that the procedural and Petri net part of the model is becoming smaller and smaller, while the Declare constraints offer the same behavior and more. Hence, a trade-off between precision and generalization exists.

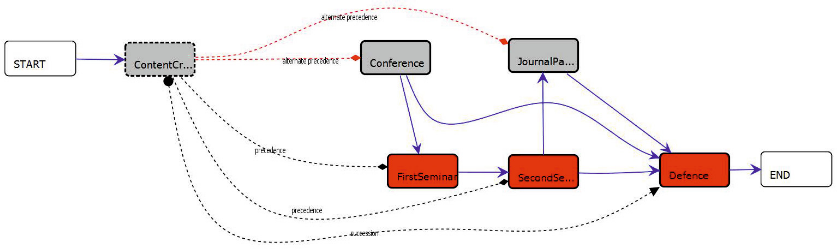

Fig. 3. Result of Multi-Paradigm Miner with $e = 0.2$

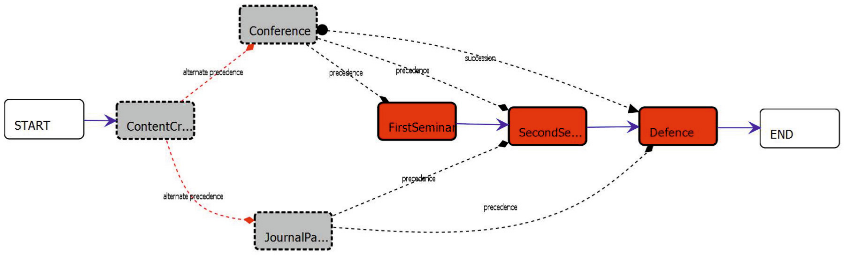

Fig. 4. Result of Multi-Paradigm Miner with $e = 0.5$

7 Conclusions and Future Work

This paper presents Multi-Paradigm miner, a process discovery technique that is capable of combining procedural and declarative process constructs into one discovery result. Hence, structured and more flexible parts of a workflow can be represented with the most appropriate constructs, thus allowing to find a trade-off between recall, precision, generalization, and simplicity in a more versatile way. Hereto, we introduce the notion of a mixed-paradigm model. Initial experimental evaluations show that our technique can indeed discovery multi-paradigm models that are better and more useful than single-paradigm models.

However, to fully assess the power of the technique, the notion of precision, fitness and generalization needs to be introduced in future work. For both paradigms there already exist techniques such as [10,11] to evaluate fitness, which can serve as a starting point. An initial approach could include checking traces for Declare violations as proposed by [10] and use the move on model outcome to replay the corrected traces on, e.g., Petri nets.

Acknowledgments. This research is funded by FWO (Fonds voor Wetenschappelijk Onderzoek) Project G0804 13N.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
2. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 270–285. Springer, Heidelberg (2012)
3. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
4. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
5. Pesic, M., Schonenberg, H., van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2007, pp. 287–287. IEEE (2007)
6. Di Ciccio, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 135–142. IEEE (2013)
7. Westergaard, M., Stahl, C., Reijers, H.A.: Unconstrainedminer: Efficient discovery of generalized declarative process models
8. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 278–295. Springer, Heidelberg (2009)
9. Maggi, F.M., Slaats, T., Reijers, H.A.: The automated discovery of hybrid processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 392–399. Springer, Heidelberg (2014)
10. de Leoni, M., Maggi, F.M., van der Aalst, W.M.: An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. Information Systems (2014)
11. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Conformance checking using cost-based fitness analysis. In: 2011 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 55–64. IEEE (2011)