KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Multi-paradigm process mining: retrieving better models by combining rules and sequences

De Smedt J, De Weerdt J, Vanthienen J.

KBI_1413

Multi-Paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences

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, extracted from event logs which are the outcome of today's omnipresent information systems. 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. The findings have been implemented as a ProM plug-in that is based on Heuristics Miner and Declare miner.

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 has become the main part of the mining process on which the two other ones build. Its goal is to represent the information on processes in the most comprehensive, comprehensible and correct way. In order to do so, many mining techniques have been proposed, including, amongst others, Alpha Miner [2], Heuristics Miner [3] and Fuzzy Miner [4]. The representation forms used for these models are rather procedural. More recently, declarative process modeling and mining has gained popularity and numerous discovery techniques such as Declare Miner [5], MinerFul++ [6] and UnconstrainedMiner [7] have been implemented for discovery purposes. These miners retrieve rules from process

logs to create models with a more descriptive rather than prescriptive view on the information contained in the log.

In accordance with the "maps" view on process models proposed by [8], one retrieves different information by mining for different paradigms. Similar to reading maps with different perspectives which cover multiple layers of an area, it is possible to retrieve different paradigms at once to gain complementary insights from the information retrieved from the log. For example, combining street maps with altitude information can provide a deeper understanding of the explored area.

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 enlighten certain aspects of an event log, but these aspects have not been merged yet to form a richer process model. The main challenges arise when the two types of maps need to be merged and made compatible in a sense-making and understandable way. The findings have 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 first section covers a running example which uncovers current problematic behavior and contains a comparison of mining approaches for both paradigms. Next, a hybrid model for mining is proposed. Section four provides an overview of the implementation, followed by the last section with conclusions and future work.

2 Procedural and Declarative Process Mining Techniques

2.1 Running Example

As a running example, we provide the simple model in Figure 1. Both declarative ConDec constraints [9] and more sequentially-based Petri nets are added to resemble the possible 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 to journals or seminars at a conference, resembled by the *Alternate Precedence* constraints. 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 the Petri net could have been replaced by two *Succession* constraints. Inspired by the approach proposed by [10] for creating test logs in CPN Tools, we enacted the model. If one mines the simulation log for a procedural model with (Flexible) Heuristic Miner or Alpha Miner, the algorithms are unable to retrieve the exact position and relation of the three activities *Content Creation*, *Conference* and *Journal Paper* as shown in Figure 2 and 3. The results by Declare Miner are shown in Figure 4. There are many constraints, even for a support of 100 %, due to the search for rules between every activity and the others, but the overall

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

process is hard to discern. This paper tries to offer an integrated approach like stated above, which combines the insights of both the procedural and declarative process mining approaches to establish better models.

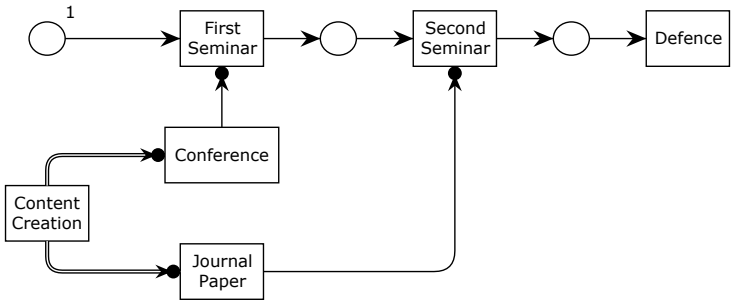

Fig. 1: A rather declarative process model representing a PhD's progress flow.

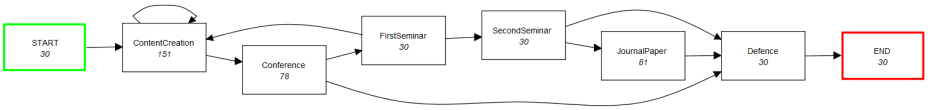

Fig. 2: A dependency graph retrieved from the mined causal net produced by Flexible Heuristics Miner (default settings). The mining algorithm is unable to correctly identify the relationships between, e.g., *Second Seminar* and *Journal Paper*.

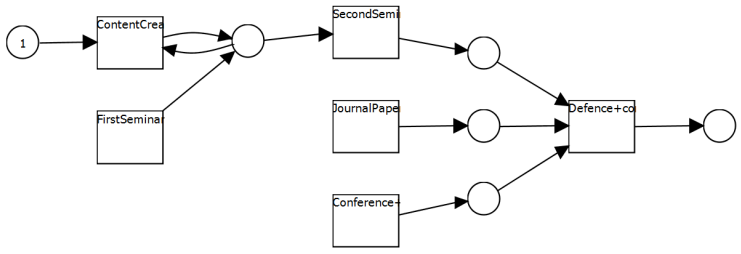

Fig. 3: A Petri net retrieved by Alpha Miner (ProM 6 implementation). The mining algorithm cannot derive the relationships between, amongst others, *Content Creation* and *JournalPaper* or *Conference* correctly.

2.2 Current Mining Approaches

Procedural process miners capture sequence constraints and parallelism by incorporating information supporting adjacency and direct succession in a process

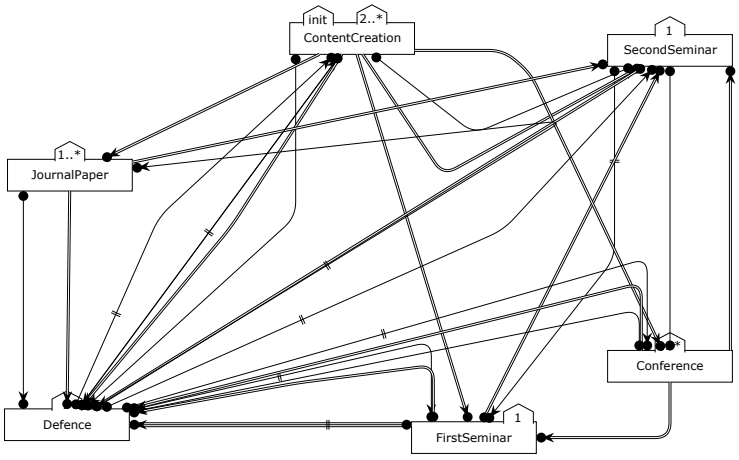

Fig. 4: A Declare model retrieved by Declare Miner (alpha = 0, support = 100%)

log, extended with (X)OR- and AND-split and -join information. Most of the constraints are locally defined (an exception is Fuzzy Miner, which uses an optional distance metric for deriving sequence constraints) but calculated on the log level. Examples of implementations are included in the introduction.

Declarative miners are usually based on ConDec rules expressed in Linear Temporal Logic (LTL) [11] such as *Alternate Precedence* or *Succession*. The latter two approaches in the introduction convert the templates into regular expressions to derive the rules in a faster fashion. Afterwards, the different constraints are checked for the log on trace level and the results are displayed on a diagram which contains the rules between different activities.

The granularity of capturing relations between activities is different as local and global (trace-based) constraints carry different information. By combining the information of both, it is already possible to gain different insights. Consider this simple example: trace (a, b, c, a, a, b) fulfills both an *(Alternate) Precedence(a,b)* and a local direct succession constraint 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 a certain log, or the model can be pruned by using the *Precedence* constraint as counter evidence for the sequence constraint.

2.3 Advantages of a Combined Mining Approach

The benefits of mining a multi-paradigm process model included in this paper 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 transform the log into a richer 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 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. 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. a very fixed process sequence.

3 Hybrid Process Models for Mining

In order to retrieve hybrid models for mining practices, we build a definition for hybrid process models based on causal graphs [12] and ConDec constraints. The choice is founded on the usage of these representation forms in Heuristics Miner and Declare Miner. For modeling purposes, mixed-paradigm approaches have already been proposed for, amongst others, YAWL and Declare [13] and Petri nets and Declare [14].

We use the definition of causal nets for the procedural model (PM) as follows: Let PM be a tuple $PM = (A, a_i, a_o, D, I, O)$ with A the finite set of activities, $a_i \in A$ the start activity, $a_o \in A$ the end activity, $D \subseteq A \times A$ the dependency relations, $I \in A \to AS$ the set of possible input bindings and $O \in A \to AS$ the set of possible output bindings with $AS = \{X \subseteq P(A) \mid X = \emptyset \vee \emptyset \notin X\}$. $P(A)$ represents the powerset of A. Causal nets represent markings over a directed graph to express sequential and parallel behavior in a process model by allowing transitions through the different input and output bindings. By nature, these graphs are already more declarative than, for example, Petri nets, but can be transformed to one. We have included this transformation for the examples in section 4 for illustration purposes.

We define declarative, rule-based models as follows: Let DM be a tuple $DM = (A, \pi)$ with π the rules over the activities in set A, or $\pi \mapsto A^n$. π can express LTL rules over A, e.g. $\pi(a, b) = \Box(a \to \diamond\, b) = \mathit{Response}(a, b)$.

The different behavior of the models can be represented as proposed by [13]. Figure 5a shows the declarative behavior $\pi(A)$ in green and the procedural behavior (A, a_i, a_o, D, I, O) in yellow. While the graph was proposed for modeling, it is now applied to mining. Therefore, the behavior contained in the log should be included. It is represented in orange.

We propose a hybrid model as follows: Let HM be a tuple $HM = (A, a_i, a_o, D, I, O, \pi)$. As such, we define both rules and explicit sequences over the activities. The outcome of the model is then any part of $PM \cup DM = HM$, which constricts the behavior of A in different ways:

- **A more narrow result:** $PM \cap DM$: by taking the intersection of the behavior allowed by both models, it becomes possible to more strictly describe the process flow. This is represented by Figure 5b.
- **A broader result:** $PM \cup DM$: by taking the union of all behavior, it becomes possible to capture behavior in the log that previously remained undiscovered or was too broadly captured by either model type separately. By taking subsets of this union, it becomes possible to more closely retrieve the behavior in the log. This is shown in Figure 5c.
- **A different result:** if $PM \subseteq DM$ or vice versa: by choosing the best representation form for the mined area, it becomes possible to represent processes in an event log in the most informative way. In other words, where declarative and procedural behavior overlap, one can choose the representation form most suited for clarification. For example, strict processes can be captured by causal graphs, while cluttered areas are more conveniently approached by some rules.

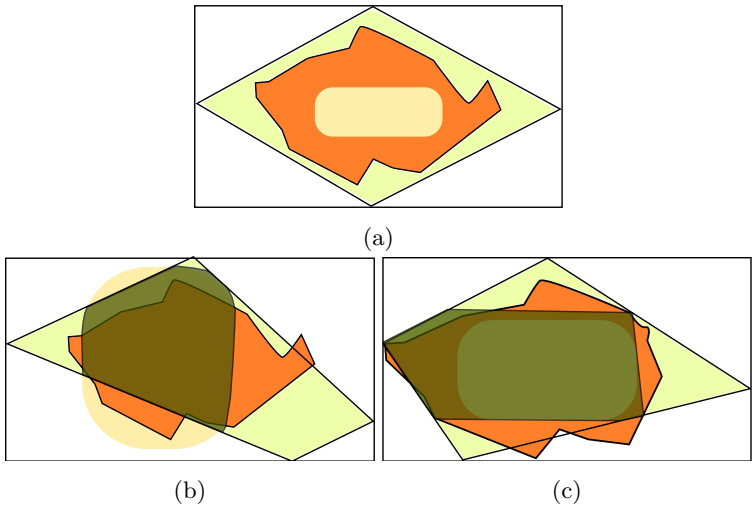

Fig. 5: Graphical representation of the behavior allowed by the procedural model (yellow), the declarative model (green) and the behavior contained in the log (orange)

4 Hybrid Miner

4.1 Introduction

The Hybrid 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 captures activities that introduce a lot of behavior in the log, which indicates that they play a key role in the structuring of flows and the model overall. Especially the causing of a lot of non-conclusive behavior indicates the more flexible nature of their existence. As such, we target them for inclusion in the set of activities that are subject to Declare mining in the second part of the mining process. Finally, the information is displayed in a model which contains all mined behavior, which can be pruned optionally.

4.2 The algorithm

By analyzing the strength of the direct succession metric between activities, one can retrieve the activities most closely related in a small window. Activities that have a lot of other 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 $P \subseteq A$. Note that $P \cup R = A$. Also, the occurrence of numerous level-two loops, much like the unclear direct succession count, can indicate the ad-hoc all-over-the-place occurrence of activities, 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 contains tasks that are connected only when the "All activities connected" option is chosen in Heuristics Miner.
To check for such activities we propose a metric called Activity Entropy (AE):

- $AE_i = \sum_j^n DS_{ij}, \qquad \forall\, i, j\, (i \neq j), DS_{ij} < d$, $n = |L|$ with L the log.
- e, a configurable threshold.
- $|E| = |A|(1 - e)$, with $|A|$ the number of activities in the log.

AE captures the average of the direct succession (DS) metrics between the 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, a proportion of the log is withheld. The different values AE_i are ranked and $|A|(1 - e)$ activities are kept in the sorted set E. Furthermore, if there is a gap of $1/e$ between two activities in E, the activities are removed. This procedure is installed to avoid introducing too many activities and possibly too many rules between them. Note, however, that a fully declarative model can be obtained by using 1 for e. ConDec rules mined for single activities are always included in the log. As such $\pi \rightarrow A$.

[2] The implementation can be found at j.processmining.be/hybridminer/.

4.3 Pruning and constraint choice

After the results of both miners are retrieved, an optional manipulation of the outcome is performed which cuts and adds some constraints, extending the approach proposed by [15]. E.g., *Precedence* and *Response* constraints are transformed into *Succession* constraints when the antecedents and consequents involved only appear once, and *Co-existence* is removed when both activities appear at least once. After this pruning phase, a check for transitivity is performed, which can now include extra *Succession* constraints.

In Hybrid Miner, the Declare model that is mixed with the procedural one is derived in a more straightforward and sequential fashion, which offers an alternative to the approach in Figure 4. This greatly improves the readability of the resulting model. Furthermore, it is possible to substitute the behavior of the procedural part with the declarative constraints between two activities, providing the possibility to either retrieve a more loosely defined model, or a stricter model on which both paradigms are applied, as illustrated in Figure 5.

Finally, since the ConDec rule set contains over 20 entries and we combine this set with a procedural model, some constraints become obsolete or less relevant:

- *(Strong) Init()* and *Last()* constraints are captured by the procedural model by start and end activities (a_i, a_o).
- The *Chain Response/Precedence/Succession()* constraints are capture by the direct succession constraints in the procedural model.
- The *Choice()* constraints are captured by the XOR- and AND-joins and -splits in the procedural model.
- Negative constraints are also left out of scope.

4.4 Results

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.
- The checkered activities exceed the entropy threshold.
- The red activities fulfill the *Exactly1* constraint.
- The green activities fulfill the *Existence* constraints.
- The striped arrows represent ConDec constraints, which are labelled.

The PhD process The running example is used to demonstrate the capabilities of Hybrid Miner. In Figures 6 through 9 the results in ProM and the corresponding Petri nets are compared for different threshold levels for e. Even for a small entropy value (Figures 6 and 7), 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 can use 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 *Journal-Paper* is still incorrect.

By raising the entropy level (Figures 8 and 9), 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.

If we position this approach in Figure 5, it would be categorized as an attempt to cover behavior that is not mined by the procedural technique, depicted in Figure 5c. The rules are applied as follows, $\pi \rightarrow R \times A$, while $D \subseteq P \times P$ and $I, O \in P$.

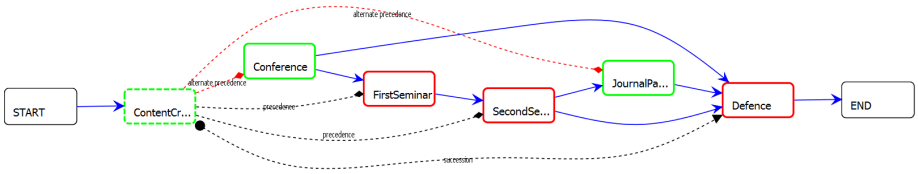

Fig. 6: Result of Hybrid Miner with $e = 0.2$

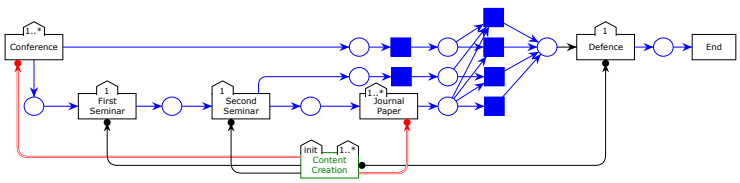

Fig. 7: Corresponding Petri net for $e = 0.2$

Real-life incident management log A real-life log was used that originated from an incident management system. The log contains 20 activities and quite a lot of procedural behavior. By applying Hybrid Miner, we obtain the process model in Figure 10. The activity entropy is kept quite low (0.4), the dependency threshold is 0.9 and the Declare rule support 80%.

By interpreting the Declare constraints, it becomes possible to prune sequences for a certain support from the process model as illustrated in Figure 5b. By choosing a certain path between, e.g., activities *Status As...* and *Status W...*,

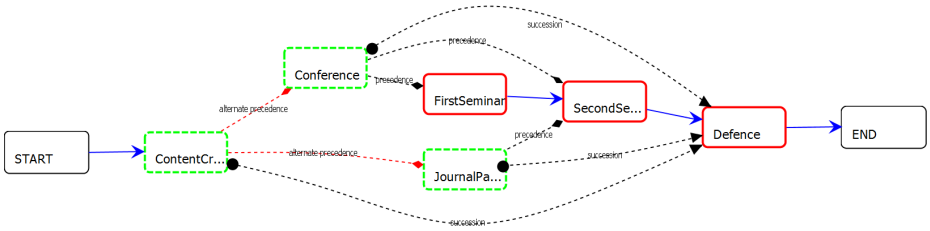

Fig. 8: Result of Hybrid Miner with $e = 0.6$

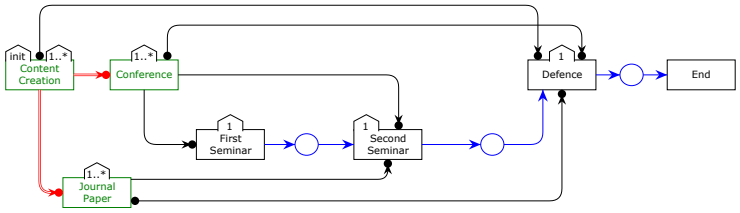

Fig. 9: Corresponding Petri net for $e = 0.6$

other paths become invalid because of the *Exactly1* constraints. For example, the direct arrow between *Status As...* and *Group OR...* (the uppermost activity) is not supported by the rules, nor are any incoming arcs for the first activity. Another example of a rule which restricts behavior is the *Succession* between *Individual* and *Status Clo...*. This prevents going straight to the end when *Individual* is enabled. Note that the *Exactly1* and *Succession* constraints reinforced each other in the constraint manipulation phase.

While there is an overlap between the two paradigms in this example, e.g. the connection between *Status As...* and *Group OR...* and the *Succession* constraint, they contain different information. For example, a *Succession* constraint enforces the appearance of a certain activity, while a regular sequence constraint does not. Note that by adding the rules, it becomes obsolete to add some extra semantics, such as splits and joins for the input and output bindings. By knowing in which order activities appear from the rules, combined with the cardinality, one can derive which sequences are possible and which ones are not compatible with the rules that are derived on a trace-based (global) level.

The more straightforward part of the log, which is partially hidden behind the super-figure in Figure 10, is not included in R and contains very sequential behavior. This illustrates again that the algorithm is capable of discerning the two types of activities and does not necessarily include too many rules.

If we position this approach in Figure 5, it would be categorized as an attempt to cover behavior that is mined by a procedural miner, which is then further restricted by adding rules. The rules are applied as follows, $\pi \rightarrow R \times A$, while $D \subseteq A \times A$ and $I, O \in A$. Furthermore, when the same behavior is captured and displayed, one can choose to either read the rules or interpret the input and

output bindings, hence choosing which representation form is most suited for the situation.

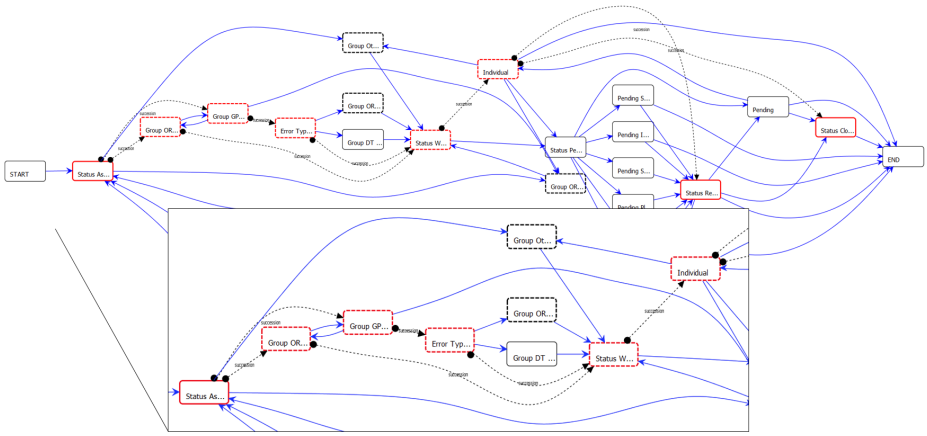

Fig. 10: Result of Hybrid Miner for a real-life log with following settings: dependency threshold 0.9, Declare support 80%, and activity entropy 0.4.

These examples have shown the potential of Hybrid Miner; while the former shows how new relations are uncovered, the latter presents a way of restricting behavior even further for a certain support, or visualize it in a different way. As the information retrieved by Declare rules is positioned at trace level, it becomes possible to extract knowledge that is supported for a fixed part of the log, cutting, e.g., infrequent behavior.

Hybrid Miner can also be used for an initial log discovery to search for the amount of different behavior as well. As such, it can serve the purpose of checking the level of flexibility contained in the event log.

5 Conclusions and Future Work

In this paper we proposed a first attempt towards hybrid process mining. By combining principles of miners for both the procedural and declarative paradigm, we extract models that can capture fixed paths mixed with flexible parts in the workflow. Hence, it becomes possible to exploit the gap between over- and underfitting in a more versatile way. However, to fully asses 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 to evaluate fitness which can serve as a starting point. Furthermore, compliance and region-based techniques can be used to extend the proposed approach to more precisely locate flexible or non-fitting behavior in the log. Also, to improve performance of the rule mining, faster techniques such as UnconstrainedMiner will be explored as the current implementation is restricted by the speed of Declare Miner.

Also, the investigation of representing and using mixed models should be further pursued.

Other research directions include the discovery of mixed-paradigm hierarchies to distinguish for example a declarative super-process with procedural subprocesses or the other way around.

References

1. Van der Aalst, W.M.: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on **16**(9) (2004) 1128–1142
3. Weijters, A., van der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP **166** (2006)
4. Günther, C.W., Van Der Aalst, W.M.: Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In: Business Process Management. Springer (2007) 328–343
5. Maggi, F.M., Bose, R.J.C., van der Aalst, W.M.: Efficient discovery of understandable declarative process models from event logs. In: Advanced Information Systems Engineering, Springer (2012) 270–285
6. Di Ciccio, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on, IEEE (2013) 135–142
7. Westergaard, M., Stahl, C., Reijers, H.A.: Unconstrainedminer: Efficient discovery of generalized declarative process models
8. van der Aalst, W.M.: What makes a good process model? Software & Systems Modeling **11**(4) (2012) 557–569
9. Pesic, M., van der Aalst, W.M.: A declarative approach for flexible business processes management. In: Business Process Management Workshops, Springer (2006) 169–180
10. De Medeiros, A.A., Günther, C.W.: Process mining: Using cpn tools to create test logs for mining algorithms. In: Proceedings of the sixth workshop on the practical use of coloured Petri nets and CPN tools (CPN 2005). Volume 576. (2005)
11. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.: User-guided discovery of declarative process models. In: Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on, IEEE (2011) 192–199
12. Van Der Aalst, W., Adriansyah, A., Van Dongen, B.: Causal nets: a modeling language tailored towards process discovery. In: CONCUR 2011–Concurrency Theory. Springer (2011) 28–42
13. Pesic, M., Schonenberg, H., van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International, IEEE (2007) 287–287
14. Westergaard, M., Slaats, T.: Mixing paradigms for more comprehensible models. In: Business Process Management. Springer (2013) 283–290
15. Maggi, F.M., Bose, R.J.C., van der Aalst, W.M.: A knowledge-based integrated approach for discovering and repairing declare maps. In: Advanced Information Systems Engineering, Springer (2013) 433–448

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be