Pattern-based Simplification of Process Models*

David Chapela-Campa, Manuel Mucientes, and Manuel Lama

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)
Universidade de Santiago de Compostela. Santiago de Compostela, Spain
{david.chapela, manuel.mucientes, manuel.lama}@usc.es

Abstract. Several simplification techniques have been proposed to deal with the understanding of complex process models, from the structural simplification of the model to the simplification of the log to discover simpler process models. But obtaining a comprehensible model explaining the behaviour of unstructured large processes is still an open challenge. In this paper, we present a novel algorithm to simplify process models by abstracting the infrequent behaviour in the logs.

Keywords: event abstraction, model simplification, process mining

1 Introduction

Process mining has emerged as a discipline focusing on techniques to discover, monitor and enhance real processes. One of the key areas of process mining is the discovery, whose objective is to generate a process model describing the behaviour of the event log of a process, to later analyze and enhance it. However, in scenarios where the quality of the discovered process model is too low —e.g. spaghetti models—, this analysis and enhancement becomes more difficult.

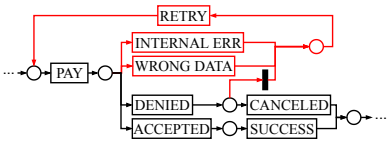
Different techniques have been developed to tackle with this problem: *i)* the simplification of the already discovered process models [1]; *ii)* the simplification of the log, to later discover an understandable process model, either by detecting outlier traces in the log [2], or by removing from the log the more chaotic activities [3]; and *iii)* the abstraction of frequent subprocesses in the log by replacing the execution of multiple activities with one [4].

A motivational example of an ideal abstraction is depicted in Fig. 1, where a sample of a log is shown in Fig. 1a, with its corresponding model in Fig. 1b. In this case, the frequent behaviour is related to the paths through `DENIED-CANCELED` and through `ACCEPTED-SUCCESS`. Fig. 1c and Fig. 1d show an abstraction where the infrequent behaviour of the paths going through the loop is encapsulated in one activity, `ERROR AND RETRY`, leaving the rest untouched.

* This research was funded by the Spanish Ministry of Economy and Competitiveness under grant TIN2017-84796-C2-1-R, and the Galician Ministry of Education, Culture and Universities under grant ED431G/08. These grants are co-funded by the European Regional Development Fund (ERDF/FEDER program). D. Chapela-Campa is supported by the Spanish Ministry of Education, under the FPU national plan (FPU16/04428).

Trace	
... PAY - DENIED - CANCELED ...	(×27)
... PAY - ACCEPTED - SUCCESS ...	(×43)
... PAY - INTERNAL ERR - RETRY - PAY ...	(×6)
... PAY - WRONG DATA - RETRY - PAY ...	(×7)
... PAY - DENIED - RETRY - PAY ...	(×8)

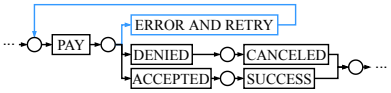
(a) Partial event log.



(b) Model for log in Fig. 1a.

Trace	
... PAY - DENIED - CANCELED ...	(×27)
... PAY - ACCEPTED - SUCCESS ...	(×43)
... PAY - ERROR AND RETRY - PAY ...	(×30)

(c) Abstracted event log.



(d) Model for log in Fig. 1c.

Fig. 1. Motivational example for the algorithm presented in this paper.

In this paper, we present an algorithm to simplify process models by abstracting the infrequent behaviour in the log and maintaining the more frequent one, allowing to discover a simpler process model. The main novelty of our approach is that it first detects the frequent behaviour of the process —using the frequent patterns extracted by WoMine [5]— and then abstracts the infrequent behaviour. The use of WoMine to detect frequent behaviour allows our technique to retain not only frequent activities, but frequent subprocesses, abstracting the infrequent behaviour which obfuscates the understanding of the overall process.

2 Abstraction Algorithm

The approach presented in this paper (Alg. 1) takes as input an event log, a process model and a frequency threshold, and returns the event log with the abstraction of the infrequent behaviour. The execution of a discovery algorithm over that abstracted log allows to obtain a more precise and simpler process model, keeping a good fitness.

The first step of our proposal is to identify the frequent behaviour to be kept in the log. For this purpose WoMine [5] is used (Alg. 1: 2), extracting from the process model a set of behavioural patterns executed in a percentage of the traces of the log frequent w.r.t. the defined threshold. As an example, the frequent patterns obtained for the log in Fig. 1a with a threshold of 25% cover the behaviour going through `DENIED - CANCELED`, through `ACCEPTED - SUCCESS`, and the executions of `PAY`.

Later, the algorithm builds, for each trace, the abstractions of the events not covered by the frequent patterns (Alg. 1: 4-7). For this, each trace is analyzed marking the events which are not executed in any frequent pattern. Then, the marked events of each trace connected between them in the model are grouped to be replaced with the same abstracted event. In Fig. 1a, the marked events are those depicted in red. Then, in each trace a single connected group is formed.

Once the abstracted events of each trace are known, it is neccesary to assign the same activity to the abstracted events modeling the same behaviour

Algorithm 1. Abstraction algorithm.

Input: An event log $L = [\tau_1, ..., \tau_m]$, a process model M, and a threshold t.
Output: An abstracted event log $L' = [\tau'_1, ..., \tau'_m]$.

```
1  Algorithm AbstractInfrequentBehaviour(L, M, t)
2      P ← getFrequentPatterns(L, M, t) // using algorithm in [5]
3      Λ ← ∅
4      forall the τ ∈ L do
5          Λ_τ ← obtainAbstractions(τ, P)
6          Λ ← Λ ∪ Λ_τ
7      end
8      Λ ← assignAbstractedEvents(Λ)
9      L' ← abstractLog(L, Λ)
10     return L'
```

(Alg. 1: 8). For this purpose, a new abstracted activity is assigned to the set of groups with the same input activities in the model, or with the same output activities. In the example in Fig. 1a, all groups have as input the activity `PAY`. Thus, the same activity —`ERROR AND RETRY`— is assigned to the abstracted events replacing these groups.

Finally, the log is abstracted with function `abstractLog` (Alg. 1: 9) by replacing, in each trace, the event groups formed with the corresponding abstracted events. The result of this process in Fig. 1 is shown in Fig. 1c, allowing to discover the model depicted in Fig. 1d.

3 Results

The proposed algorithm has been tested against state of the art techniques in 11 logs from Business Process Management Challenges and real processes. Results show that, in logs where the complexity of the process is high, our technique allows to obtain better process models —in terms of F-score and simplicity w.r.t. the original model— than the state of the art techniques.

References

1. de San Pedro, J., Carmona, J., Cortadella, J.: Log-based simplification of process models. In Motahari-Nezhad, H.R., Recker, J., Weidlich, M., eds.: BPM 2015. Volume 9253 of LNCS., Springer (2015) 457–474
2. Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Trans. Knowl. Data Eng. **29**(2) (2017) 300–314
3. Tax, N., Sidorova, N., van der Aalst, W.M.P.: Discovering more precise process models from event logs by filtering out chaotic activities. J. Intell. Inf. Syst. **52**(1) (2019) 107–139
4. Mannhardt, F., Tax, N.: Unsupervised event abstraction using pattern abstraction and local process models. In Gulden, J., Nurcan, S., et al., eds.: BPMDS 2017. Volume 1859 of CEUR Workshop Proceedings., CEUR-WS.org (2017) 55–63
5. Chapela-Campa, D., Mucientes, M., Lama, M.: Mining frequent patterns in process models. Inf. Sci. **472** (2019) 235–257