Improving Process Model Precision by Loop Unrolling

David Sánchez-Charles[1], Marc Solé[1], Josep Carmona[2], Victor Muntés-Mulero[1]

[1] CA Strategic Research Labs, CA Technologies, Spain
David.Sanchez,Marc.SoleSimo,Victor.Muntes@ca.com
[2] Universitat Politècnica de Catalunya, Spain
jcarmona@cs.upc.edu

Abstract. Despite the advent of scalable process mining techniques that can handle both noisy and incomplete real-life event logs, there is a lack of scalable algorithms capable of handling a common cause of model underfitting: when the same activity in the log in fact behaves differently depending on the number of occurrences in a particular trace. This paper proposes a simple scalable technique to identify these cases and successfully mine better process models from event logs. The technique has been implemented and evaluated on well-known benchmarks in the literature.

1 Introduction

Process discovery techniques strive to derive models that are expected to be good under four quality dimensions: fitness, precision, generalization and simplicity [4]. Hence, these are multi-objective techniques that search in a large solution space, where typically not one but many optimal solutions exist. In practice, each discovery technique puts the emphasis in a proper subset of dimensions; for instance, techniques based in the *theory of regions* focus on deriving fitting and precise models, while simplicity and generalization is not always guaranteed. Another example is the recent block-based techniques that recently appeared [6, 7], where structured, fitting, generalized and simple process models are preferred.

The techniques from [6, 7] are the driving force of this work. On the one hand, they are among the few scalable process discovery technique that can derive structured process models. This has made [6, 7] one of the most popular techniques for process discovery nowadays. However, as mentioned in [3], these techniques can sacrifice precision significantly for the sake of deriving a fitting structured model (see the example in Section 1.1). The alternative offered in [3] is to use evolutionary techniques, which are far from scalable. Instead, the technique proposed in this paper represent a fresh look at this problem, amending (when possible) process models derived from the technique in [6, 7] as a simple post-processing step, based on unrolling loops in the model whenever the number of loop iterations found in the event log satisfy certain criteria. Next section illustrates the intuition behind the technique of this paper.

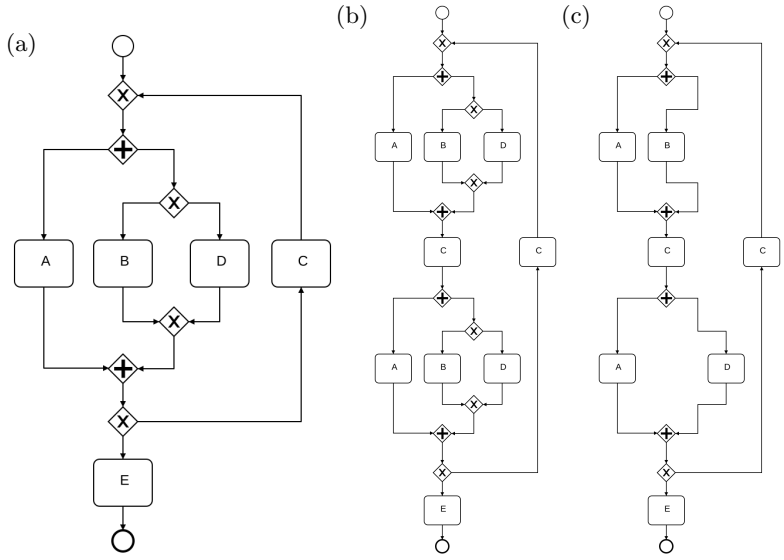

Fig. 1. A first model accepting traces such as $\sigma = ABCADCBACDACABCADCBACADE$. The second model is an unrolled version that only accepts executing twice the initial iterative behaviour. Finally, a repair of the model with respect to the trace σ highlighted that the second choice construct could be simplified to a simple sequence.

1.1 Label splitting as loop unrolling to improve precision

Consider the model Figure 1.a, which was discovered by considering the trace $\sigma = ABCADCBACDACABCADCBACADE$. It is hard to notice that the precision of this model could be improved: Activities A, B and D can be found in any ordering and hence the parallel construct is appropriate, and trace σ hints that the iterative approach might be a good candidate for describing such a process. Nevertheless, a further analysis shows that there is still place for improvement.

In this paper, we propose to *unroll* iterative parts of a process to check if there are hidden relations between the activities that are hindered by the limitation of only having one single copy of the activity in the model. See Figure 1.b for an example of such unrolling. In this particular case, we have chosen to repeat the iterative structure so we are forcing to execute its subprocess twice in each iteration. A replay of trace σ on this new process model highlights that activities B and D were never mutually exclusive. And hence, one could discover that the process model of Figure 1.c might be more precise in describing σ.

1.2 Related work

Different approaches exist in the literature for the problem of label splitting in the context of process mining. We will focus here in recent approaches, and will illustrate the different nature of the technique of this paper with respect to

them. The heuristic techniques in [10, 8] rely on a window local search approach to define the duplication of certain candidate activities. This process is done in the model itself ([10]) or as a refinement of the input log ([8]). By focusing on the loops of a process model, the technique of this paper complements these approaches.

Alternatively, global approaches can be found in [9, 5]. These global methods rely on the use of *unfolding* of the process model ([9]) and a later optimization technique to fold back activities, or search for special states of the underlying state space of the model ([5]), followed by a clustering strategy to merge them heuristically. By relying on complex representations and techniques (unfoldings or state spaces can be exponential on the size of the models), these approaches cannot be applicable for large inputs.

2 Definitions and notation

Definition 1. *A **process model**, or simply **process**, $N = (\mathcal{A}, \mathcal{C}, \mathcal{E})$ is a directed graph consisting of activities $\mathcal{A}$, control elements $\mathcal{C}$ and edges $\mathcal{E}$. Edges connect activities and control elements. Control elements define the behavior of the process model, and are of any of the following types start, finish, split-choice, join-choice spilt-parallel, join-parallel. The only condition over the graph structure is that all maximal paths must start and end with a start and a finish control flows. A **subprocess** of N is any valid process model $(\mathcal{A}', \mathcal{C}', \mathcal{E}')$, with $\mathcal{A} \supseteq \mathcal{A}'$, $\mathcal{C} \supseteq \mathcal{C}'$, in which $\mathcal{E}'$ is defined as all edges in $\mathcal{E}$ that connects any pair of elements in $\mathcal{A}'$ and/or $\mathcal{C}'$.*

Processes are a graph representation of a potentially infinite set of sequences of events, which is denoted by $\mathcal{L}(N)$. Generally, all paths from the initial *start* control element to the *end* traverse a sequence of activities. Such sequences are the elements of $\mathcal{L}(N)$. Although the graphical notation used for representing processes is irrelevant in terms of the results presented in this paper, Business Process Modelling Notation will be used to improve understandability.

Definition 2. ***Structured process** imposes extra conditions on the control elements of a process: all split-parallel nodes (resp. split-choice) must have a unique corresponding join-parallel node (resp. join-choice) such that all paths connecting these two nodes must visit zero or two of any other pair of control elements. This correspondence is unique in the sense that if two split nodes u and v have the same corresponding join node, then u and v are the same node.*

This definition allows us to consider structured processes as smaller subprocesses or individual activities that are interconnected via edges or control elements. Due to the soundness of structured processes, some notions can be easily described.

Definition 3. *An **iterative subprocess** or loop l is the combination of two subprocesses that describe a process that can be repeated. The **forward path***

of l (fwd(l)) is the subprocess that must be executed at least once during the execution l. Whereas the ***backward path*** *of l (back(l)) is the subprocess such that its execution enforces the loop to re-execute fwd(l).*

From now on we will consider all process models to be structured. Importantly, structured processes allow us to map particular events in the trace to a subprocess in the process model. Allowing us to define the following notions:

Definition 4. *Given a process model N with a loop l and a trace $\sigma \in \mathcal{L}(N)$, we define $E_l(\sigma)$ as the number of times fwd(l) is executed during the execution of σ.*

Definition 5. *Let l be a loop of a process model N and σ a trace accepted by N. We define the* ***projection of σ to*** **l** *(denoted by $\sigma|_l$) as the result of keeping the events that are mapped into activities contained in l after a replay of σ in N. Moreover, we define the* ***projection of σ to the exit condition of*** **l** *(denoted by $\sigma|_{Exit(l)}$) as keeping the events that are mapped into activities of N that cannot coexist with the execution of l. In particular, all activities contained in l and any other concurrent activity are erased by this projection.*

Considering again the process model of Figure 1.a and trace $\sigma = ABCAD$-$CBACDACABCADCBACADE$, we have a loop structure l consisting of: as the forward path, activities A, B and D that can be executed concurrently but B and D are mutually exclusive; and activity C as its backward path. The forward path was executed $E_l(\sigma) = 8$ times; $\sigma|_l = \sigma - \{E\}$ whereas $\sigma|_{Exit(l)} = E$. Any execution of activity E clearly indicates that any event after event E is not part of the loop.

Definition 6. *(Fitness and precision) Process mining techniques aim at extracting from a log L a process model N with the goal to elicit the real unknown process $\mathcal{S}$. By relating the behaviors of L, $\mathcal{L}(N)$ and $\mathcal{S}$, particular concepts can be defined [4]. A process model N* fits *log L if $L \subseteq \mathcal{L}(N)$. A process model is* precise *in describing a log L if $\mathcal{L}(N) \backslash L$ is small.*

Unless stated otherwise, we assume we deal with fitting process models. In case this condition is violated, we assume the process models are first aligned with current techniques to satisfy the fitness condition [1].

3 Label splitting with loop unrolling

Most discovery algorithms generate processes in which activities are not duplicated, forcing the algorithm to introduce loops when an activity is consistently occurring multiple times per trace. Unfortunately, this constraint may overgeneralize the resulting process model. Consider, for instance, trace $\sigma = ABCA$. A technique like the ones in [6, 7] will produce an iterative process model even though the trace is not showing so clearly that behavior.

First we will describe the unrolling algorithm for improving the precision of loops that are not included in any other iterative process. The main idea of this

algorithm is to create a process model that forces the re-execution of the loop and then filters out unused elements. Finally, we will extend this algorithm for the case of nested loops.

3.1 Simple case: Unrolling of individual loops

The first process model of Figure 1.a depicts a process describing the log consisting of the trace $ABCADCBACDACABCADCBACADE$. One may notice that, when replaying the log on the process model, the forward path (Activities A, B and D) is executed a multiple of two. Hence, we may force the process model to repeat the loop as in Figure 1.b. The unroll of a loop is precisely the process of making explicit this transformation.

Definition 7. *A k-unroll of a loop l is the process of substituting the loop for the subprocess defined by a loop structure with:*

- *A sequence of $k-1$ copies of the sequence fwd(l);back(l) as the forward path of the new loop structure,*
- *finishing the aforementioned sequence with another copy of fwd(l);*
- *The back(l) is maintained as the backward path of the new loop structure.*

In Figure 1.b, a 2-unroll of the iterative process is performed. In this case, the subprocess containing activities A, B and D is the forward path, whilst activity C is the backward path. And hence, its 2-unroll produces a loop structure with the sequence $AND(A\ OR(B, D))\ C\ AND(A\ OR(B, D))$ as the forward path and maintains C as the backward path.

Proposition 1. *Given a process model N describing the log L, a k-unrolling ($k > 1$) of a loop l increases the precision of the model.*

Besides, if the process model fits log L and k is a divisor of the greatest common divisor (gcd) of the number of executions per trace of the forward path of l, then the k-unrolled process also fits log L.

Proof. Let l be a loop of N and let N_k be a k-unroll of l with $k > 1$. By construction of the k-unroll, we can ensure that any trace is an element of the language of N such that the forward path of l is executed a multiple of k times. I.e.

$$\mathcal{L}(N_k) = \{\sigma \in \mathcal{L}(N) \,|\, E_l(\sigma) \text{ is divisible by } k\}$$

We will show that $\mathcal{L}(N_k) \subsetneq \mathcal{L}(N)$ and hence, based on Definition 6, N_k improves the precision of process model N. Let σ' be a trace accepted by the process N that visits exactly once the forward path of the loop l, and hence the backward path of l is never visited. Since 1 is not a multiple of k, we can ensure that σ' is not an element of $\mathcal{L}(N_k)$ and hence $\mathcal{L}(N_k) \setminus L$ is a non-trivial subset of $\mathcal{L}(N) \setminus L$ and therefore the precision of N' is bigger than the precision of N.

Besides, let k be a divisor of the greatest common divisor of the number of executions per trace of the forward path of l, N a process model that fits log L and N_k the k-unroll of the process model N. Let σ' be a trace of the log L. Since

N fits log L, the trace σ' is in the language of the process model. Moreover, the number of executions of the forward path of l is a multiple of k and hence the trace σ' is also an element of the language of L. Therefore, N_k fits log L. □

Once all loops have been unrolled, some activities and structures of the resulting process model may be redundant or unused and can be removed or simplified allowing for further improvement on the precision of the process model. The first process model of Figure 2 describes traces $ACADBCBD$, $BCBDACBD$ and $BCAD$. Such process may benefit from a 2-unroll as shown in the second process model. Besides, a replay of the three traces highlight that split choices between C and D are unnecessary: starting with C, activities C and D alternate in the execution of the process model. The last process model of Figure 2 depicts the process model after pruning unused paths.

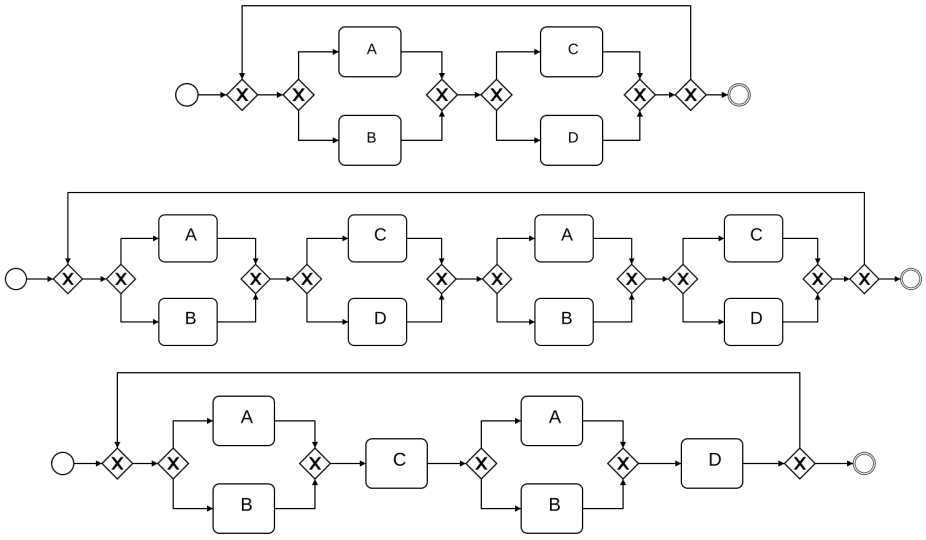

Fig. 2. A process model describing the traces $ACADBCBD$, $BCBDACBD$ and $BCAD$.

3.2 General case: Unrolling of nested loops

Structured subprocesses allow process models to have nested loops structures This poses a problem for deciding the number of unrolls, as the number of executions of the forward path per trace may be interleaved across embedded loops. The process model from Figure 3 depicts a process with a nested loop that accepts trace $ABBBABBB$. If we follow the count executions of the forward path, then activity B is recommended to be unrolled 6 times, even though it has never been executed 6 times in a row.

Instead of considering the number of executions per trace, we may count the number of consecutive executions of a forward path. In the particular case of trace $ABBBABBB$, the forward path B is consecutively executed 3 times at two different points in the trace, whilst the forward path consisting of activities A and B is consecutively executed 2 times. Definition 8 formalises this concept by counting the number of executions on maximal subtraces contained in the loop subprocess.

Definition 8. *Let l be a loop structure of the process model N, and σ a trace accepted by N. Then we define the* ***set of continuous executions*** *of the loop l in the trace σ as*

$$CE_l(\sigma) = \left\{ n \;\middle|\; \exists\, \sigma_1, \sigma', \sigma_2 \text{ s.t. } \begin{array}{r} \sigma = \sigma_1\sigma'\sigma_2 \\ \sigma'|_l \in \mathcal{L}(l) \\ \sigma'|_{Exit(l)} = \emptyset \\ E_l(\sigma'|_l) = n \\ \sigma' \text{ is maximal} \end{array} \right\}$$

Informally, the set $CE_l(\sigma)$ represents a set of numbers, each one denoting continuous executions of l in σ.

The combination of *no exit condition* and maximality of subtrace σ' in Definition 8 ensures that we are splitting the trace σ on chunks such that a continuous execution of the loop l is not separated in two different subtraces. Besides, non-consecutive executions of the loop cannot be included in the same group as this would have shown some activities incompatible with the execution of the loop. Notice that activities that are executed concurrently alongside the iterative subprocess l might be included in the subtrace σ', but they are removed during the projection to the iterative subprocess l and they are not part of the *exit condition*.

Consider trace $\sigma = ABBBABBB$ and the smaller loop, or B-loop, of process model 3. The exit condition of the B-loop is activity A, since any execution of that particular activity clearly shows that the execution is happening outside the B-loop. Hence, we may split σ in two instances of $\sigma' = BBB$. Notice that we cannot extend it since then we would include an exit condition, and σ' is accepted by the B-loop. And therefore, $CE_{B-loop}(\sigma) = 3$.

Similarly to the non-nested case, the language accepted by an unrolled process model can be described as a refinement on the language accepted by the original process model as depicted in Proposition 2.

Proposition 2. *Let N be a process model, and l a loop subprocess of N. Let N_k be any k-unroll of l. Then*

$$\mathcal{L}(N_k) = \{\sigma \in \mathcal{L}(N) \,|\, \forall n \in CE_l(\sigma), n \text{ is divisible by } k\}$$

Proof. The definition of the k-unroll of loop l ensures that any execution of the loop l executed a multiply of k times the forward path of l and hence any maximal subtrace $\sigma' \subseteq \sigma$ such that $\sigma'|_l \in \mathcal{L}(l)$, $\sigma'|_{Exit(l)} = \emptyset$ must satisfy that k divides $E_l(\sigma')$.

On the other hand, let σ be a trace in $\mathcal{L}(N)$ such that all continuous executions $CE_l(\sigma)$ are divisible by k. Then, σ is also an element of the language $\mathcal{L}(N_k)$. Suppose not, then the trace σ violates any behavioural relation between a set of activities or the iterative process must finish before repeating k times the forward path. Both cases are not possible. The former violates the fact that $\sigma \in \mathcal{L}(N)$ and the latter violates the fact that $CE_l(\sigma)$ only contains multiples of k. □

Proposition 3 is a direct consequence of Proposition 2, due to the hard constraint that k divides all $n \in CE(t, l)$.

Proposition 3 (Generalization of Proposition 1). *Given a process model N describing the log L, a k-unrolling ($k > 1$) of a loop l increases the precision of the process model. Besides, if the process model fits log L and k is a divisor of $CL(l, t)$ for all $t \in \mathcal{L}(N)$, then the k-unrolled process model also fits log L.*

Revisiting the example of trace *ABBBABBB* and the process model of Figure 3, which contains a nested loop, a replay of the trace contemplates that the big loop is executed 2 times and the smaller loop is executed 3 times on each execution. Hence, we could perform a 3-unroll on the latter and a 2-unroll on the former. Doing so, we discover the second process model of Figure 3, and a second replay highlights the possibility of removing the unnecessary loop structures as illustrated by Figure 3.

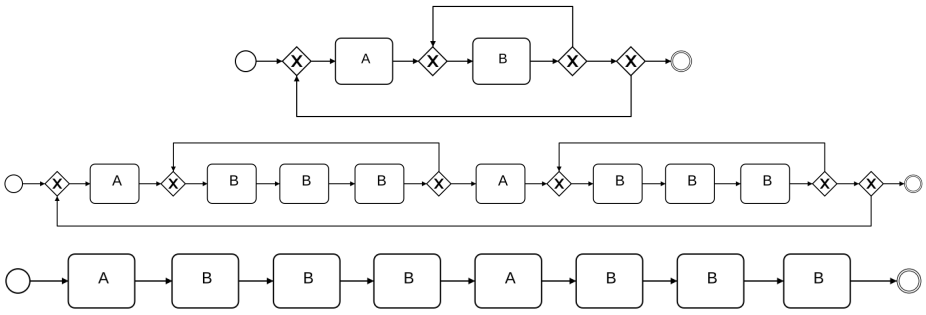

Fig. 3. Three process models describing the trace *ABBBABBB*.

4 Evaluation

To evaluate our duplicate technique, we reuse an existing dataset comprising 15 small logs [10] whose source processes are well-known and reproduce behavior commonly found in real-life scenarios. Besides, we also considered the BPI Challenge 2012 dataset. This real-life log contains events describing the application process for a personal loan, or overdraft, within a global financing organization.

From all these events, we have only selected the events starting with W as they show actions performed by workers of the organization, instead of states in a fixed sequential process.

	Inductive Miner (IM)					**IM + Unrolling**					**PNSimpl**
Model	P	T	τ	Precision	Fitness	P	T	τ	Precision	Fitness	Precision
alpha	11	17	6	0.6750	1.0	12	19	3	**0.7386**	1.0	0.70
betaSimpl	14	21	8	0.6216	1.0	14	15	2	0.9130	1.0	**0.93**
Fig5p1AND	9	8	3	0.8333	1.0	10	8	2	**1.0000**	1.0	**1.00**
Fig5p1OR	5	6	1	0.7000	1.0	6	6	0	**1.0000**	1.0	**1.00**
Fig5p19	9	14	6	0.6746	1.0	Not applicable					**0.85**
Fig6p9	10	15	8	0.6667	1.0	Not applicable					**0.83**
Fig6p10	15	24	13	0.6282	1.0	Not applicable					**0.76**
Fig6p25	22	35	14	0.7629	1.0	25	36	13	**0.8467**	1.0	**0.84**
Fig6p31	6	10	1	0.6250	1.0	7	10	0	0.90	1.0	**1.00**
Fig6p33	7	11	1	0.6667	1.0	8	11	0	0.90	1.0	**1.00**
Fig6p34	17	24	12	0.5785	1.0	Not applicable					**0.93**
Fig6p38	13	11	4	0.6212	1.0	13	10	2	**0.77**	1.0	0.65
Fig6p39	12	12	5	**0.8986**	1.0	Not applicable					**0.89**
Fig6p42	7	18	4	0.2284	1.0	Not applicable					**0.74**
RelProc	21	28	12	0.7143	1.0	Not applicable					**0.73**
BPIC2012	15	22	15	0.6195	1.0	16	23	15	0.6594	0.9893	**0.79**

Table 1. Comparison of the precision in selected process models discovered with Inductive Miner and then Unrolled. The simplicity of the processes is also depicted with the number of places P, transitions T and silent activities τ in the discovered Petri Nets. For some cases the unroll is not possible without sacrificing fitness.

Table 1 contains the precision obtained with the process model discovered with Inductive Miner (IM), the precision obtained after unrolling these process models and precision obtained by PNSimpl [5]. We have used the alignment-based precision metric [2] for this evaluation. 7 out of 15 processes do not show any improvements with our technique since it was not possible to perform an unroll without losing fitness. For the BPI Challenge 2012 log, it was also not possible to perform an unrolling without losing fitness. None of the iterative subprocesses was repeated a multiple of k times for any k. Nevertheless, for this dataset we followed another strategy: We choose k in order to minimize the loss in fitness. In this particular case, after performing a 2-unrolling on the activity *Calling to add missing information to the application*, 9% of the traces cannot be replayed by the unrolled process model, with a minimal impact on fitness, but its precision increases 5%.

Results indicate that precision gain is similar with both techniques, provided that unrolling is possible. Nevertheless, both approaches treat the initial process model differently: Our approach enhances the expressive power of the initial process, whilst PNSimpl rediscover the process after each label split and, hence, the final process might be significantly different. Besides, the ability of perform-

ing unrolls (by losing fitness) enables us to highlight interesting properties of the process. For instance, loop unrolling allowed us to check that the financing organization of the BPI Challenge 2012 usually had to call customers twice for getting the necessary information. Is there any reason one attempt is not enough?

In terms of complexity, the technique of this paper may be a light alternative for methods like [5], which require to iteratively apply agglomerative clustering for special sets in the state-space representation of the event log.

5 Conclusion

In this paper, we presented a method for improving the precision of structural subprocesses based on explicitly repeating iterative subprocesses and pruning unused constructs and activities. We have shown that this approach is applicable to simulations of real-life processes, and also it is applicable to real-life scenarios.

The presented approach is the first step on considering the unrolling of iterative processes. Results in Table 1 show several examples of how unrolling improve the precision of the process models, with minimal impact on their complexity. Nevertheless, bigger process models might be more difficult to understand and, hence, it remains to conduct expert reviews on readability and understandability of process models after unrolling. Besides, we have experienced on some datasets that some iterative processes can be explained as a few iterations are used for initialization, and then the real loop starts. We would also like to study how the k-unroll operation affects the precision of the process model for a particular precision metric. In particular, is it possible to establish a lower bound on the increase of the precision?

Acknowledgment

This work is funded by Secretaria de Universitats i Recerca of Generalitat de Catalunya, under the Industrial Doctorate Program 2013DI062, and the Spanish Ministry for Economy and Competitiveness, the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).

References

1. A. Adriansyah. *Aligning observed and modeled behavior*. PhD thesis, Technische Universiteit Eindhoven, 2014.
2. A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. Dongen, and W. M. Aalst. Measuring precision of modeled behavior. *Inf. Syst. E-bus. Manag.*, 13(1):37–67, Feb. 2015.
3. J. C. A. M. Buijs. *Flexible Evolutionary Algorithms for Mining Structured Process Models*. PhD thesis, Technische Universiteit Eindhoven, 2014.
4. J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014.

5. J. de San Pedro and J. Cortadella. Discovering duplicate tasks in transition systems for the simplification of process models. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 108–124, 2016.
6. S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Discovering block-structured process models from event logs - A constructive approach. In *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, pages 311–329, 2013.
7. S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Discovering block-structured process models from incomplete event logs. In *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings*, pages 91–110, 2014.
8. X. Lu, D. Fahland, F. J. H. M. van den Biggelaar, and W. M. P. van der Aalst. Handling duplicated tasks in process discovery by refining event labels. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 90–107, 2016.
9. H. Ponce de León, C. Rodríguez, J. Carmona, K. Heljanko, and S. Haar. Unfolding-based process discovery. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, pages 31–47, 2015.
10. B. Vázquez-Barreiros, M. Mucientes, and M. Lama. Mining duplicate tasks from discovered processes. In *Proceedings of the ATAED Workshop*, pages 78–82, 2015.