Mining with User Interaction

Robin Bergenthum, Sebastian Mauser

Department of Software Engineering, FernUniversität in Hagen
{robin.bergenthum,sebastian.mauser}@fernuni-hagen.de

Abstract. In this short paper we present an interactive mining approach which is based on net synthesis. First, a net is generated from a log file by a liberal mining algorithm such as the alpha-algorithm. Then, using concepts from the theory of regions, runs of this net which are not included in the log are calculated and feedbacked to a user who has to decide whether they are valid runs of the process or not. Finally, a net having the runs of the log and the additionally specified runs is synthesized.

1 Introduction

Many of today's information systems record information about performed activities of processes in so called event logs. Process mining techniques attempt to extract useful, structured information from such logs. In this paper we focus on the problem of constructing a process model which matches the actual workflow of the recorded information system, called process discovery. There are many process discovery techniques in the literature (e.g. [1]), often implemented in the ProM framework (www.promtools.org/prom6).

One main difficulty of process discovery is that a typical log contains only example runs of the recorded process (we do not discuss the problem of noise here), i.e. logs are incomplete. Therefore, precise mining algorithms based on net synthesis which exactly reproduce a log (see e.g. [2]) are often not appropriate. Consequently, most mining algorithms try to generate a process model which includes the recorded example runs and also allows for some additional behavior – they overapproximate the given event log. Since there is no information on runs missing in the log, overapproximation is a heuristic approach. When a mining algorithm allows a run which is not given in the log, it is not clear (without additional information) whether the added run (1) is a run of the process which coincidentally has not been observed in the log or (2) is no possible run of the process and for this reason is of course not included in the log. The only way to solve this problem is to ask a user whether (1) or (2) is true. Therefore, we here suggest an interactive mining approach where overapproximation is explicitly determined by a user (first ideas in this direction have been developed in [3]). Note that a crucial assumption of this approach is the existence of a process owner having enough insights to decide if a feedbacked run is a run of the process or not.

The crucial challenge of this approach is to choose an appropriate set of runs for user feedback. Of course it is not viable to feedback each run which is not given in the log. It is important to only consider such runs which have a "high probability" for being runs of the process. The basic idea of our approach is to feedback the difference between a liberally mined net and an exactly synthesized net.

Starting with the given log, we first use a liberal mining algorithm to generate a net representing an upper bound. Using the theory of regions we compute the runs which are allowed by this net but not included in the log. For each such run, a user has to decide whether it represents valid behavior of the process or not. Thus, the log is completed according to the feedback. The resulting log is then used as the input for an exact synthesis algorithm, i.e. a model is generated which allows the recorded runs and the runs explicitly specified by the user. In this way overapproximation becomes controllable. In particular, the conflict between underfitting, i.e. too much overapproximation, and overfitting, i.e. not enough overapproximation, can be solved.

2 Interactive Mining Algorithm

In this section we explain the interactive mining approach in detail. As a preparative step for our algorithm, we have to choose a mining algorithm (e.g. the alpha-algorithm [1]) which we use as a reference for the overapproximation performed by our approach. In the field of conformance checking there exist the notions of recall and precision. Our interactive algorithm heavily depends on the recall and precision of the underlying mining algorithm. While our approach works with any mining algorithm, an algorithm having a high recall is preferred, since our interactive approach always generates a net which allows all the behavior given in the log (we assume that there is no noise in the given log file). The precision of the underlying mining algorithm determines the search space of the interactive approach. The larger the search space the more queries are necessary.

Having chosen a reference mining algorithm, the interactive mining algorithm is as follows (see Figure 1):

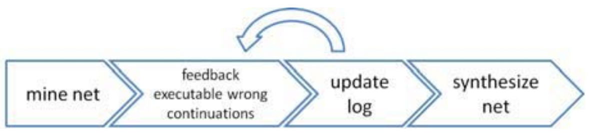


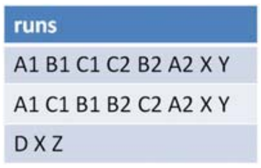
Fig. 1. Interactive mining algorithm.

- The starting point is a typical event log as described e.g. in [1, 2] which defines a set of runs. As a first step, the chosen mining algorithm is applied to the log generating a net which represents an upper bound for user feedback. The idea behind this approach is that a liberal mining algorithm performs a “reasonable” overapproximation but has a tendency to introduce “too much” overapproximation.
- The second step is to compute runs to be feedbacked to the user. For this purpose, we consider so called wrong continuations [2] of the given log. A wrong continuation extends a run (or more precisely the Parikh-vector of a run) of the log by one event such that the resulting run is not specified. That means, only minimal non-specified behavior is considered for feedback. Still the set of all wrong continuations is too large for practical purposes. Therefore, we only consider wrong

continuations which are allowed by our upper bound, i.e. which are executable w.r.t. the net mined by the chosen liberal mining algorithm. This set of wrong continuations is presented to the user for feedback. For each such run, the user then has to decide whether it is a run of the underlying process or not. For this purpose, the runs are illustrated in a list, and for each run there is a respective checkbox.

- As a third step, the log is updated by adding the runs which have been classified by the user as valid runs of the process. For all these new runs, new wrong continuations appear which have not yet been feedbacked. Therefore, there is a choice for the user now: He can decide to either proceed with the next final step (fourth step) or to repeat the feedback steps (second and third step) with the new wrong continuations. The latter choice means that the new wrong continuations which are executable w.r.t. the initially mined net are again feedbacked to the user and possibly added to the log. The second and third step can be iterated until no more new valid runs are found or the upper bound is reached.
- The fourth step consists of using an exact synthesis algorithm on the updated log, i.e. a net reproducing the log which has been completed by the user in the previous steps is generated. We here apply the mining algorithm based on regions of languages from [2, 4] which is implemented in the tool VipTool.

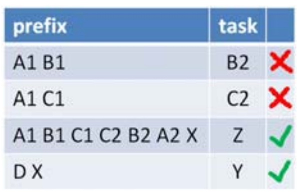
We now illustrate this procedure by a small example log which defines the three runs shown in Figure 2.



runs
A1 B1 C1 C2 B2 A2 X Y
A1 C1 B1 B2 C2 A2 X Y
D X Z

Fig. 2. Runs of the example log.

As a reference mining algorithm we use the alpha-algorithm for this example. The alpha-algorithm tries to extract from a log the direct dependencies of activities and translates them into places of a workflow net. Figure 4 shows a ProM screenshot of the net mined from the example log with the alpha-algorithm. This net overapproximates the log. It allows the three runs of the log and eleven further runs. The exact synthesis algorithm of VipTool generates the net shown in Figure 5 from the example log. It coincidentally has the same places as the net from Figure 4 and some additional places which are shown in grey (in general such a relation does not hold). Note that, since the log cannot precisely be represented by a Petri net, this net not only allows the three runs of the log but also two additional runs. The idea of our mining approach is now to construct a net in between the liberally mined net from Figure 4 and the synthesized net from Figure 5 by interacting with a user. That means, on the net level in this example the question is whether the grey places of Figure 5 should be included in the net or not.



prefix	task	
A1 B1	B2	✗
A1 C1	C2	✗
A1 B1 C1 C2 B2 A2 X	Z	✓
D X	Y	✓

Fig. 3. Feedback of wrong continuations.

Algorithmically, in the first step of our algorithm we mine the net from Figure 4. Then, in the second step first the set of all wrong continuations of the log is computed. For instance, A1 B1 is a prefix of a run of the log (see Figure 2). Since A1 B1 B2 is not contained in the log and extends the previous run by one event, it is a wrong continuation. Altogether, there are 97 wrong continuations in our example. However, only four of the 97 wrong continuations are enabled in the net from Figure 4. That means, for feedback we only consider these four runs which are given in Figure 3.

For each of these four wrong continuations the user has to decide whether in the process the last task can occur after the occurrence of the given prefix. In the context of the first two wrong continuations the question is if B2 (resp. C2) can immediately occur after B1 (resp. C1) or if B2 (resp. C2) has to be ordered behind C1 (resp. B1). For our example let us assume that the second case is true, i.e. the two wrong continuations are marked to be no runs of our process. In the context of the third and fourth wrong continuation the question is if task Z (resp. Y) can freely be chosen at the end of the process also in the case that the part of the net including A1, B1, C1, C2, B2, A2 (resp. D) has been chosen at the beginning of the process or if the choice of Z (resp. Y) depends on the choice at the beginning of the process. Here we assume that the first case is true, i.e. the two wrong continuations are marked as runs of our process. After this user feedback, in the third step of our algorithm the two positively evaluated wrong continuations are added to the log. In the following, a repetition of the feedback steps is not necessary in our example, since all wrong continuations of the two newly added runs are not enabled in the net from Figure 4, i.e. the upper bound is reached. Thus, in the fourth step, the VipTool synthesis algorithm is applied to the completed log with five runs. Since we use an exact synthesis method, for this step we only need the original log and the positively evaluated runs here. The result is the net from Figure 5 without the two grey places connected with the tasks Y and Z, i.e. there is a free choice between Y and Z.

3 Alternative Approach

In the described approach, the liberally mined net is just used as the upper bound for feedback when completing the log. Afterwards, this net is dropped and the final net is constructed from scratch by using synthesis methods. However, as it is also the case for our example, the net mined by a typical liberal mining approach is often nicely readable. The places generated by such mining algorithms are mostly introduced according to simple rules which in a natural way reflect the dependencies given in the log, i.e. they often nicely reflect the intuition of users of the business process. Thus, an interesting idea for the construction of the final net in our interactive mining approach is to keep the initially mined places and to only complement them by newly synthesized places as far as necessary. In this way, we support the generation of an intuitive and readable net.

Still, keeping the initially mined places causes the following problem: For most mining algorithms it is possible that so called unfeasible places which prohibit runs given in the log are generated. These places have to be deleted for enabling the approach of this section.

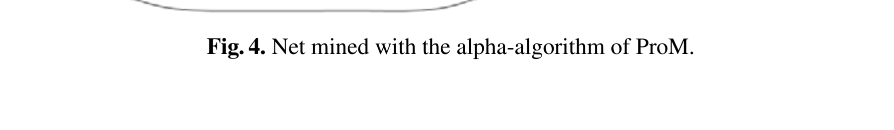


Fig. 4. Net mined with the alpha-algorithm of ProM.

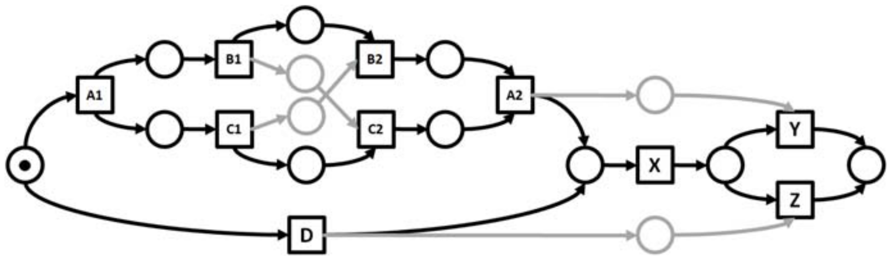


Fig. 5. Net synthesized with VipTool.

Altogether this alternative approach yields the algorithm shown in Figure 6. The difference to the algorithm in Figure 1 is the additional step "delete unfeasible places" and the changed last step. In this new last step, the runs which have explicitly been specified by the user to be no runs of the process are considered. For this set of runs, places which prohibit this set but are feasible w.r.t. the updated log (i.e. they allow the runs of the log) are computed and added to the initially mined net.

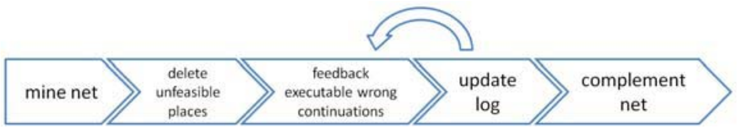


Fig. 6. Alternative mining algorithm.

For the example log of Figure 2, again first the net given in Figure 4 is generated using the alpha-algorithm of ProM. In this example, all places are feasible, i.e. all runs of the log are executable in the initially mined net. Thus, the same set of wrong continuations as in the first presented algorithm (Figure 3) are feedbacked to the user in a first feedback step. We assume that the user makes the same choices as in the last section. Consequently, the last two wrong continuations are added to the log and the first two wrong continuations have to be stored to be prohibited later on. As before, after the first feedback step the upper bound is reached. Therefore, it is proceeded with the last step of the algorithm. With standard synthesis methods, the algorithm computes two regions, each prohibiting one of the two stored wrong continuations. The initially mined net is then extended by the two places corresponding to the two regions yielding the same net as in the case of the first interactive approach.

When comparing the two approaches, it is a coincidence that we get the same result in our example. In particular, it was a coincidence of the first approach that the resulting synthesized net included the liberally mined net from Figure 4. In this alternative approach, it is the central idea to keep the initially mined net and to extend it with additional places to always get such a nice result. Moreover, we do not anymore focus on completing the log but on identifying runs to be prohibited. As a consequence, in the case a user decides to stop the feedback phase, i.e. to not finish the completion of the log, there is an important difference to the algorithm presented in the last section. While the algorithm of the last section tries to prohibit all the behavior which has not been feedbacked anymore in such case, the algorithm of this section keeps the liberal control flow of the initially mined net.

Finally, it remains to mention that processes with loops are problematic for the presented basic interactive mining approaches. If a loop is included in the liberally mined net, it is possible to repeat the feedback steps of the interactive approaches arbitrarily often and thus the user has to cancel the iterated feedback at some point. In the case of the first algorithm, the precise mining approach of VipTool then tries to detect such loops from the extended log as described in [4]. In the case of the second algorithm, the loop is simply kept from the liberally mined net. But, if it is not detected by the approach from [4], it is still possible that the loop is coincidentally prohibited by the additional places. In this context several improvements are possible, in particular mechanisms to automatically stop the feedback steps in the case of loops and improved methods for loop detection.

4 Conclusion

In this paper, we tackle the problem of incomplete logs by applying mining with user interaction. We introduce overapproximation in a user controlled way. The starting point is a net generated by a liberal mining algorithm. The behavior of this net is then restricted by applying concepts from the theory of regions. We presented two different approaches implementing this idea.

As future work it is important to both evaluate the quality of our interactive mining approach and the practicability of the approach for users.

References

1. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Trans. Knowl. Data Eng. **16**(9) (2004) 1128–1142
2. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining Based on Regions of Languages. In: BPM 2007, LNCS 4714, Springer (2007) 375–383
3. Esparza, J., Leucker, M., Schlund, M.: Learning Workflow Petri Nets. In: Petri Nets 2010, LNCS 6128, Springer (2010) 206–225
4. Bergenthum, R., Desel, J., Kölbl, C., Mauser, S.: Experimental Results on Process Mining Based on Regions of Languages. In: Workshop CHINA, Petri Nets 2008, X'ian (2008)