Robust Process Mining with Guarantees

Sander J.J. Leemans

Queensland University of Technology, Brisbane, Australia

Organisations store a lot of data from their business processes nowadays. This execution data is often stored in an event log. In many cases, the inner workings of the process are not known to management, or, for instance, only vaguely resemble their three-year old PowerPoint design. Gaining insights into the process as it is executed using only an event log is the aim of process mining. In this thesis, we address three aspects of process mining: process discovery, conformance checking and enhancement.

Process Discovery

Once an event log has been obtained, process discovery is an essential next step in many process mining projects. Process discovery aims to automatically discover a process model from the event log, such that this model describes the underlying (unknown) process well. As we do not want to assume that all possible behaviour is included in the event log, process discovery algorithms inherently make a trade-off between several quality criteria. For instance, fitness denotes the fraction of the event log is described in the model, and log precision denotes the fraction of the model is described in the event log. Simplicity denotes whether the model needs few and simple constructs to express its behaviour. For some event logs, process models that score well on fitness, log precision and simplicity might not exist in the representational bias of the algorithm. Other quality criteria of algorithms include how well a discovered model resembles the running process that underlies the event log. Even though the process is typically unknown, studying under which conditions a discovery algorithm rediscovers the process allows us to compare discovery algorithms. In this thesis, we argue that process discovery algorithms should provide several guarantees, such as that the discovered models are free of deadlocks and other anomalies, that, depending on the use case, fitness is perfect, and that the processes underlying the event logs are rediscovered (the algorithm should provide *rediscoverability*). With these guarantees, algorithms become more robust to real-life behaviour and larger event logs.

In order to ease generalising over the behaviour in an event log, many process discovery algorithms use an abstraction of the behaviour in the event log, such as directly follows graphs. We conducted a systematic study of four of these abstractions and identified classes of models for which the abstractions uniquely represent systems and logs. That is, we proved that any difference in behaviour is detectable in the abstraction. The four abstractions studied were the directly-follows relation, the minimum self-distance relation, the concurrent-optional-or relation and the co-occurrence relation.

Use cases might require several different process discovery techniques, but these techniques might offer similar guarantees and require similar proofs. Therefore, we introduced the Inductive Miner (IM) framework, which aids algorithms in providing several guarantees and enables algorithm designers to consider only the *most important* behaviour in an event log, instead of *all behaviour*. The IM framework searches recursively for the most important behaviour in an event log. That is, the framework searches for the combination of the root process tree operator and a proper division of the activities in the event log (a *cut*). If such a cut can be found, the event log is split into several sublogs and the framework is applied to each sublog recursively until a base case (for instance, an event log containing only a single activity) is encountered. If no such cut can be found, a fall through (a model allowing for a superset of the behaviour in the event log) is returned. An algorithm that uses the IM framework only needs to provide implementations for these four steps. The guarantees aided by the IM framework are soundness, which is guaranteed for any algorithm by the use of process trees, and fitness, log-precision and rediscoverability, for which proof obligations have been expressed as local properties.

Using the IM framework and the studied abstractions, we introduced several discovery algorithms: we introduced a fitness-guaranteeing basic algorithm, a deviating- and infrequent-behaviour handling algorithm and an incomplete-behaviour handling algorithm. Deviating behaviour should not be possible in the process but appears in the event log, while infrequent behaviour denotes little-used parts of the process. Such behaviour needs to be filtered to avoid complex models. An event log that does not fully witness a language abstraction is incomplete, and this challenges discovery algorithms. However, we show that in some cases guarantees can still be given. Furthermore, we introduced algorithms to handle more process tree constructs, such as inclusive choices (OR) and skipping (τ) constructs. These algorithms highlight the flexibility of the IM framework: one can take an existing algorithm and improve it locally with little-impacting changes, and likely rediscoverability and perhaps fitness guarantees will be preserved. For all these algorithms, we proved rediscoverability, using the proof obligations identified for the IM framework.

Some event logs contain non-atomic executions of activities: activities take time, as the event contains the starts and ends of activity executions. To handle such event logs, we introduced a family of algorithms that uses knowledge of non-atomicity in the construction of a directly follows graph, but further resembles the other algorithms: a fitness-guaranteeing basic algorithm, a deviating and infrequent behaviour handling algorithm and an incomplete behaviour handling algorithm. Also for these algorithms, we proved rediscoverability.

Most of the proposed algorithms have a run time that is polynomial in the number of activities and run quick on real-life event logs. However, the IM framework requires the event log to be copied for every recursion, thus larger event logs might be problematic. To handle larger event logs, that is, logs containing tens of millions of events and thousands of activities, we adapted the IM framework to recurse on a directly follows abstraction instead of on event logs, such that in each recursion, only a directly follows abstraction needs to be copied instead of an event log. This framework, the *IM directly follows based* (IMd) framework, was instantiated in three algorithms: a basic algorithm, a infrequent and deviating behaviour handling algorithm and an incompleteness handling algorithm.

Even though the algorithms presented in this thesis apply different strategies, instead of searching for the *entire* behaviour while worrying about soundness, the Inductive Miner framework allowed us to focus on strategies to find the *most important* behaviour in an event log (that is, the root operator and root activity partition) and have soundness and, in some cases, fitness guaranteed. By using the IM framework in different settings and for different algorithms, we have shown that it, and the proofs for guarantees, can be reused. Therefore, the IM framework can be seen as a starting point for more algorithms that leverage the algorithms and proofs we provided.

Conformance Checking

While discovering a model, process discovery algorithms might need to exclude behaviour of the event log from the model, or include behaviour that is not in the event log into the model, in order to obtain a model with the "right" balance. Therefore, discovered models should be evaluated before further usage, for which a conformance checking technique could be used. Two types of conformance checking techniques were addressed in this thesis: log-conformance checking, which compares a process model to an event log and advises on their differences, and model-conformance checking, which compares two process models. For instance, the discovered model and another model representing a reference implementation, representing a different geographical area or representing a different time period could be compared. We identified three levels on which conformance checking techniques provide information about the correspondence between logs and models: a summarised measure (e.g. a fitness or precision number), information on the model level, and information on the log level. Many existing conformance checking techniques are either not robust enough to deal with the complexity of real-life event logs and the models discovered from these logs, do not support all features of such discovered models, or use an abstraction that is too coarse to capture the behaviour of logs and models well.

Therefore, we introduced a new robust approach to conformance checking: the *Projected Conformance Checking* (PCC) framework. This framework is applicable to compare event logs to models and models to models, and checks for conformance by constructing the language of these logs and models explicitly in DFAs and compares their behaviour to measure fitness and precision. Thus, the PCC framework supports all regular languages, regardless of the model formalism used. To avoid constructing the entire state space and consequently take a lot of time, the PCC framework constructs DFAs of all *subsets* of activities of a user-specified length in the logs and the model. This allows the PCC framework to consider behaviour on a scale from fine-grained to very coarse, depending on the size of the subsets. These partial measures provide insight in the locations of deviations between the logs and models, that is average fitness and precision can be computed for each activity and this can be visualised on the model. Furthermore, the partial measures can be averaged over all subsets of activities to provide a summarised fitness and precision measure.

Evaluation

To evaluate the introduced process discovery and conformance checking techniques, we performed several experiments, using both real-life and synthetic data.

The new process discovery techniques were compared to 10 existing process discovery techniques: techniques that guarantee soundness and other techniques. First of all, we performed a scalability experiment, using several artificially generated logs of increasing size and complexity in a limited-RAM environment, which showed that most existing techniques cannot handle the complexity and size of logs that the algorithms of the IM framework can handle. This experiment did not challenge the directly follows-based IMd framework, as the disk space to generate the event logs became the bottleneck.

In a second discovery experiment, the techniques were applied to 9 real-life public event logs, and were compared both quantitatively and qualitatively. We found that of the 9 real-life logs, an algorithm of the IMd framework was pareto optimal for all 9 logs, and other Inductive Miner algorithms were pareto optimal for 8 logs. Existing techniques achieving pareto optimality were the Evolutionary Tree Miner [2] (8 times pareto optimal, however many activities were left out of the models) and the Structured Miner [1] (2 times).

In a third discovery experiment, we evaluated the new techniques on rediscoverability challenges: we applied the algorithms to artificially generated event logs that contained several challenges: deviating, incomplete and infrequent behaviour. We found that the introduced algorithms targeting these challenges handled them better than other algorithms. For instance, the algorithms designed to be robust against incompleteness of information required only half of the information as other techniques to rediscover the intended behaviour.

Furthermore, we evaluated the PCC framework on real-life event logs and discovered models, and found that it is applicable to large event logs that cannot be handled by current conformance checking techniques. Furthermore, we found that in many cases, the measures of the PCC framework rank process models discovered by discovery techniques similar to existing alignment-based techniques.

Enhancement & Tool Support

Given a discovered process model and the result of a log-conformance checking technique, a process model and an event log can be enhanced with additional information. We addressed four types of information to enhance models and event logs: deviations, frequency, performance and animation. Performance information, for instance the sojourn, waiting, queueing and service times, enables analysts to discover time-consuming activities in the process. For log animation, the event log is visually replayed on the model: each case can be visually tracked as it traverses the process, which enables the detection of changes in the process (concept drift) and bottlenecks. Queues might determine the majority of waiting times in a business process, so analysing queues might reveal bottlenecks. Deviations between log and model, that is, log moves and model moves, are essential to evaluate a model and should be considered before drawing conclusions about a process using a model.

We introduced a software tool, the Inductive visual Miner (IvM), which performs several steps, all fully automated. First, IvM discovers a process model using one of the newly introduced process discovery algorithms of the IM framework. Second, it performs conformance checking, that is, computes an alignment, between the event log and the discovered model. Third, it enhances the model based on this alignment with the four described types of information: options include performance information on the model and the event log, animation on the model, and deviations projected on both event log and model.

The IvM combines the strong points of commercial products with strong points of academic software. For instance, commercial products offer ease-of-use and practical applicability, while academic software provides reliability and semantic results that allow an analyst to validate any gained insights. Using IvM, analysts can explore the event log by repeatedly discovering a model (which is guaranteed sound, and potentially is fitting and language equivalent to the system), evaluate this model to ensure its validity, filter the event log and enhance it with performance information. Inductive visual Miner shows that it is possible to use powerful techniques with formal guarantees in a user-friendly package, and we hope that IvM will inspire commercial vendors to consider models with executable semantics and support deviation analysis.[1]

References

1. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Bruno, G.: Automated discovery of structured process models: Discover structured vs. discover and structure. In: Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings. pp. 313–329 (2016)
2. Buijs, J.C.A.M.: Flexible Evolutionary Algorithms for Mining Structured Process Models. Ph.D. thesis, Eindhoven University of Technology (2014)

[1] Update April 2018: commercial parties, such as Celonis, are starting to support deviation analyses.