Process Discovery and Exploration

Sander J.J. Leemans(✉)

Eindhoven University of Technology, Eindhoven, The Netherlands
s.j.j.leemans@tue.nl

1 Introduction

Process mining, and in particular process discovery, have gained traction as a technique for analysing actual process executions from event data recorded in event logs. Process discovery aims to automatically derive a model of the process. Current process discovery techniques either do not provide executable semantics, do not guarantee to return models without deadlocks, or do not achieve a right balance between quality criteria.

A process model can be used in for instance automatic enactment of models in systems [10], in automatic prediction [13] and in compliance checking [11]. For all these purposes, the model needs to have executable semantics and must be sound, i.e. must be free of deadlocks and other anomalies.

Process Discovery. Figure 1 shows the context of process discovery. Traditionally, models have been evaluated using the event log, by means of fitness, precision and generalisation [4]: *fitness* measures what part of the event log is described by the model, *precision* measures what part of the model is present in the event log, and *generalisation* 'measures' what part of future behaviour is present in the model. Although fitness, precision, and generalisation are intuitively clear, different formal definitions are possible [5]. Measuring the quality of a discovered model with respect to its event log might be useful, but whether the best model for the event log is the best model for the system is not captured by these measures, while ultimately, process mining aims to learn information about the system and how it is used in practise by its users (we summarise this to just *system*). Therefore, we propose to (continue) study rediscoverability; if a process discovery technique has *rediscoverability*, it is able to discover models that have the same language as the system by which the log was produced [1,2,6], given some assumptions on the event log and model.

Several process discovery techniques have been proposed; for instance the α algorithm [1], the Integer Linear Programming miner (ILP) [12], and the Evolutionary Tree Miner (ETM) [3]. All these discovery techniques return process models having executable semantics. Of all process discovery techniques we encountered, only ETM guarantees to return sound models, which it achieves by using a hierarchical abstract view of block-structured workflow nets, called *process trees*. Rediscoverability has been proven for α [2], and likely ILP also provides some form of it. However, there is no process discovery technique available that both guarantees sound models and provides rediscoverability. Part of the project is to *introduce techniques that fulfil these basic requirements*.

F. Fournier and J. Mendling (Eds.): BPM 2014 Workshops, LNBIP 202, pp. 582–585, 2015.
DOI: 10.1007/978-3-319-15895-2_52

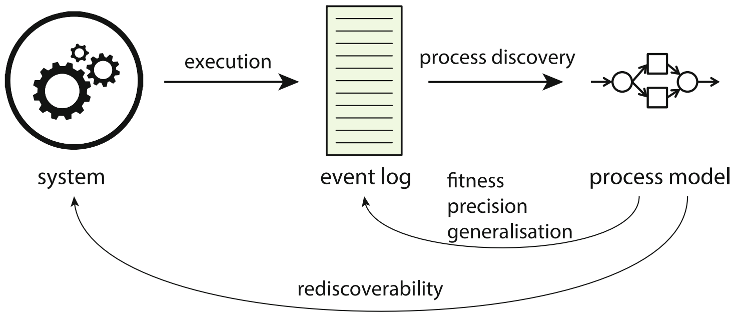

Fig. 1. Process discovery and quality metrics.

Process Exploration. As fitness, precision and generalisation compete [4], there is usually no model that is perfect according to these and other quality criteria. Thus, no discovery algorithm can be expected to result in the best model in all circumstances. That means that, unless a user knows the ins and outs of the event log and the parameters of the discovery technique, obtaining the best model for the job is a repetitive process of choosing a discovery technique, setting its parameters and evaluating the resulting model, until a satisfying model is found (for a reliable evaluation, the model should have executable semantics and be sound as well [9]). Academic tools usually provide executable semantics, but the repetitive nature of process exploration makes them tedious to use and most tools lack features like animation and seamless zooming. Commercial tools produce models that lack semantics, thereby disabling reliable evaluation and further automated use. Part of the project is to provide a methodology that allows process exploration tools to use sophisticated process mining and process discovery techniques, as these are currently used only in academic prototypes, while academic tools use simpler techniques with less extensibility.

2 Inductive Miner Family

The solution we propose to process discovery starts with process trees. This guarantees that the models returned have executable semantics and are sound. ETM uses a genetic approach to search for a process tree that is optimal with respect to several quality criteria, but is slow [7] and cannot guarantee rediscoverability.

Given the hierarchical nature of process trees, we have shown that can be exploited in a divide-and-conquer strategy to process discovery. The *Inductive Miner* (IM) applies such a strategy and performs three steps: (1) it searches for a partition of the process activities, (2) splits the event log and (3) recurses until it hits a base case. In [6], we have formally proven that IM provides rediscoverability and guarantees perfect fitness, i.e. the model describes all behaviour that is in the event log.

However, as process exploration requires a trade-off between quality criteria, perfect fitness is not always desired, for instance if the event log contains noise or infrequent behaviour and the user wants a global overview of the process. Therefore, we introduced the *Inductive Miner - infrequent* (IMi) [7], that closely resembles IM but applies log filtering in all three steps of IM. A case study showed that IMi trades fitness for precision, and discovers sound process models fast.

Both IM and IMi require the event log to contain a minimum amount of information, in particular in presence of concurrency. The *Inductive Miner - incompleteness* (IMin) [8] focuses on rediscoverability when facing logs containing incomplete information. It keeps the divide-and-conquer strategy of IM and IMi, but replaces the activity partitioning (1) step with an optimisation problem to find the best partition. A formal proof showed rediscoverability, a case study showed that IMin needs less information for rediscoverability than IM and therefore generalises better.

Using the members of the IM family, it became possible to introduce a process exploration tool that combines the executable semantics and guarantees of academic tools with the ease-of-use and features of commercial tools. The *Inductive visual Miner* (IvM) [9] uses a variant of IMi to discover a process model. It shows that to the user, extended with a visualisation of the deviations between log and model. A user can change some parameters and immediately gets a new model, enabling the repetitive nature of process exploration. Moreover, it animates the event log on the process model and provides several filtering options (although not as extensive as some commercial tools). IvM is the first process exploration tool of which users can verify the outcome and use the discovered model in automated use cases.

3 Future Research

First, work on the IM family has not finished yet: IMin is able to handle incompleteness and even a bit of noise, at the expense of run time. Moreover, several restrictions apply, such as that each process activity can only be present once and that so-called short loops cannot occur. Without these restrictions, rediscoverability cannot be guaranteed without making unrealistic assumptions about the information in the log: rediscovering arbitrary process trees requires infinite event logs. We would like to know what restrictions could be dropped without stretching the assumptions on the information in the log too far.

Second, we would like to further improve on noise filtering. Ideally, these new approaches would be evaluated using real-life event logs with known systems. In case of absence of these systems, approaches should be (and have been) compared to one another using fitness, precision and generalisation. Automated comparisons are challenging: reliable evaluation requires sound process models, which other approaches usually do not guarantee. These established criteria measure properties of the model with respect to the event log, while the more interesting (but not always answerable) question is how the models relate to the system. In particular, generalisation intuitively describes the probability that future behaviour will be included in the model. As the future is unknown, in the current practise in which the model is evaluated using training data, *any measure* *of generalisation is nothing more than a wild guess* (note that generalisation is still useful and valid to steer genetic algorithms). We would like to take steps towards an evaluation framework that circumvents these concerns, for instance using k-fold cross validation.

References

1. van der Aalst, W.M.P., Weijters, A., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
2. Badouel, E.: On the α-reconstructibility of workflow nets. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 128–147. Springer, Heidelberg (2012)
3. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: A genetic algorithm for discovering process trees. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
4. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 305–322. Springer, Heidelberg (2012)
5. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf. Syst. **37**, 654–676 (2012)
6. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013)
7. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013 Workshops. LNBIP, vol. 171, pp. 66–78. Springer, Heidelberg (2014)
8. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 91–110. Springer, Heidelberg (2014)
9. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Exploring processes and deviations. In: Fournier, F., Mendling, J. (eds.) BPM 2014 Workshops. LNBIP, vol. 202, pp. 302–314. Springer, Heidelberg (2015)
10. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 171–186. Springer, Heidelberg (2013)
11. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did I misbehave? Diagnostic information in compliance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 262–278. Springer, Heidelberg (2012)
12. van der Werf, J., van Dongen, B., Hurkens, C., Serebrenik, A.: Process discovery using integer linear programming. Fundam. Inform. **94**(3–4), 387–412 (2009)
13. Wynn, M., Rozinat, A., van der Aalst, W.M.P., ter Hofstede, A., Fidge, C.: Process mining and simulation. In: Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N. (eds.) Modern Business Process Automation, pp. 437–457. Springer, Heidelberg (2010)