# Exploratory Process Discovery for Petri Nets

Jakub Kovář[1][0000−0002−7775−3698] and Robin Bergenthum[2][0000−0003−0464−8843]

[1] Lehrgebiet Programmiersysteme, FernUniversität in Hagen, Germany
[2] Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Germany

The goal of process discovery is to automatically build a process model from an event log [1,3]. In applications, business processes become more complex and process discovery often generates spaghetti-like models [2]. This is why exploratory discovery enables a user to interactively filter the input and thus, alter the generated discovery result.

Existing interactive discovery approaches generate either directly-follows models [8], or process trees [9]. In these approaches, we order traces of an event log by their frequencies and move a slider to add or remove traces to the model. Thus, we can "zoom-in" and "zoom-out" of frequent behaviour. This feature is so helpful that almost every commercial process mining tool provides a similar feature to visualise an event log.

Using interactive discovery, we can discover both directly-follows models and process trees very efficiently and, thus, organise and replay different cases of a workflow. But if we only focus on the directly-follows relation, we generate models with a high level of generalisation and a low level of precision. We argue, in an interactive and exploratory setting, where we hand-pick a set of traces, the resulting model should be precise. If we want to dig deep and analyse and explore the control flow of a business process faithfully, we must use Petri net like models.

In this work, we outline the key sub-tasks of exploratory discovery for Petri nets. Such discovery has not been introduced yet, because it is hard to modify the behaviour of a Petri net interactively. As a proof-of-concept, we present a possible implementation, the 🦓 Miner for Petri nets, available online at www.fernuni-hagen.de/ilovepetrinets/zebra. We discuss the different sub-tasks of exploratory discovery and compare our implementation to the state-of-the-art exploratory discovery Directly-Follows Visual Miner [8]. The steps of interactive discovery for Petri nets are as follows.

In a first step of exploratory discovery, we *select* a subset of traces and *discover* our initial model. For a directly-follows based approach, this operation is inexpensive. We argue, if we *select* and hand-pick a sub-set of traces in an exploratory setting, we are looking for a precise approach. In regular discovery, being too precise and generating over-fitting models is a major disadvantage. If we only consider a small subset of traces of some event log, the high run-time of highly precise region-based synthesis approaches may not be so problematic.

In a second step, we *validate* if the process model is a precise representation of the selected set of traces. A synthesised Petri net model is the best upper approximation of the selected traces, therefore it may also contain traces of the input log that were not selected by the user. This weakens the relation between

the set of selected traces and the current model and can cause confusion. We replay all non-selected traces by firing, as presented in [4], in the synthesised model. We highlight the set of already enabled traces of the event log, as this provides important information about the quality of the current model.

In a third step, we *rate* the current selection of traces using the current process model as a reference. For directly-follows based models, we look for our preferred balance between fitness and simplicity of the resulting model. For Petri net models, we can additionally use structural and behavioural properties of the model to help us judge, how well the selected traces align. For example, a precise discovery algorithm will often produce unsound Petri net models. These unsound models provide very good feedback on the currently selected set of traces. They indicate that there may be other, not yet selected traces of the event log, which complete the current, incomplete model. If a precise algorithm produces a sound model, the set of selected traces fits together very well. Identifying these subsets of traces in an interactive setting provides an in-depth insight into the structure of the recorded behaviour.

The fourth and final step restarts the iterative exploratory procedure. We *adjust* the current selection of traces based on information from the previous step and *update* the current process model accordingly. Directly-follows based approaches can *update* the model very efficiently. For classical Petri net discovery approaches, if we change the set of input traces, we must restart the synthesis procedure. This adds a huge run-time cost and sabotages the interactivity of exploratory discovery. For this reason, the Miner implements *discover* and *update* using a new region-based discovery technique based on the token trail semantics for Petri nets [5,7]. We outline this Petri net region theory in [6]. Petri net regions can synthesise a Petri net model from a set of labelled nets. Roughly speaking, we can take two models and synthesise them into the most precise upper approximation of their combined behaviour. Furthermore, we store intermediate results to be able to remove traces later at no cost. This incremental synthesis uses smaller inputs and so has a more favourable run-time. Our preliminary experiments show it is possible to use this technique in an interactive setting.

Adjusting the selection of traces, together with the current model as a reference, we can find our preferred balance between fitness and simplicity of the resulting model. Furthermore, using an exploratory approach, we learn about frequent and infrequent behaviour recorded in the event log. We either "zoom-in" on a precise, hopefully readable model of only the frequent behaviour, or "zoom-out" to a more general, possibly more complex view. We want to generate precise models for a selected subset of traces, to uncover and explore the structure of the recorded behaviour. A perfect use case for Petri net models.

## References

1. Van der Aalst, W.M.P.: Process Mining. Springer Berlin, Heidelberg (2011)
2. Van der Aalst, W.M.P.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Computer Science **164**, 321–328 (2019)
3. Van der Aalst, W.M.P., Carmona, J.: Process mining handbook. Springer Nature (2022)
4. Bergenthum, R.: Firing Partial Orders in a Petri Net. In: Proceedings of PETRI NETS 2021. pp. 399–419. Springer (2021)
5. Bergenthum, R., Folz-Weinstein, S., Kovář, J.: Token Trail Semantics - Modeling Behavior of Petri Nets with Labeled Petri Nets. In: Proceedings of PETRI NETS 2023. pp. 286–306. Springer (2023)
6. Bergenthum, R., Kovář, J.: A First Glimpse at Petri Net Regions. In: Proceedings of ATAED 2022. pp. 60–68. CEUR Workshop Proceedings 3167 (2022)
7. Kovář, J., Bergenthum, R.: Token Trail Semantics II - Petri Nets And Their Net Language. In: Proceedings of PETRI NETS 2024. pp. 175–196. Springer Nature Switzerland (2024)
8. Leemans, S.J., Poppe, E., Wynn, M.T.: Directly Follows-Based Process Mining: Exploration & a Case Study. In: Proceedings of ICPM 2019. pp. 25–32 (2019)
9. Schuster, D., van Zelst, S.J., Van der Aalst, W.M.P.: Cortado — an interactive tool for data-driven process discovery and modeling. In: Proceedings of PETRI NETS 2021. pp. 465–475. Springer (2021)