RWTH AACHEN UNIVERSITY

This work was submitted to:

Chair of Process and Data Science (PADS - Informatik 9), RWTH Aachen University

---

Translucify: Tackling the Translucent Log Generation Problem (Abstract)

Bachelor's Thesis

Author: **Geonho Yun**

Student ID: **422305**

Supervisors: Harry H. Beyel
Christopher T. Schwanen

Examiners: Prof. Wil M. P. van der Aalst
Prof. Two

Registration Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Contents

Chapter 1

Introduction

1.1 Motivation

Business processes are seldom linear. Instead, they are usually a messy, chaotic, intertangled mash of activities where its golden path is meticulously hidden. On top of that, event logs have no guarantee of completeness. The quality of the process model therefore is entirely dependent on the quality of its event log.

An ideal event log would be *transparent*, containing metadata of structural properties of the corresponding process model, e.g. state information in a Petri net setting. Event logs, however, are usually *opaque* - one cannot identify the underlying process model straight away by solely looking at the log. One must instead utilize process discovery algorithms to generate corresponding models. By its nature, process discovery algorithms primarily focus on what *happened*. However, they often do not take into consideration what *could have happened* instead. An event log is *translucent* if the log contains the information which potential, alternative realities of the past could have occurred instead of the activity occurred in the real world. Logs of this nature are called *translucent event logs.* and are extremely beneficial to enhance the quality of existing process discovery algorithms.

Let us consider a small example as motivation. Suppose the underlying model of our business process is represented as the petri net below.

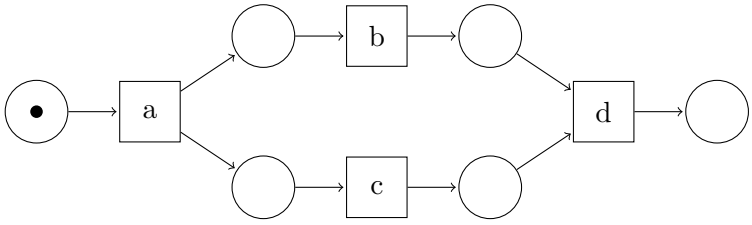

Figure 1.1: Example business model represented as a petri net.

We play-out the model fifty times and retrieve the event log $\mathcal{L}_1 = [\langle a, b, c, d\rangle^{48}, \langle a, c, b, d\rangle^2]$ in the process. We can then leverage widely used process discovery algorithms to rediscover the process model described in Figure 1.1.

This works under the assumption of log completeness, but what would happen if the

Figure 1.2: Translucent Log Generation Problem.

trace $\langle a, c, b, d \rangle$ occurs so rarely that we weren't able to capture the behavior? Given the incomplete log $\mathcal{L}_1 = [\langle a, b, c, d \rangle^{50}]$, every process discovery algorithm will return a linear process model. The translucent variant would look like $\mathcal{L}_2 = [\langle \underline{a}, \underline{b}c, \underline{c}, \underline{d} \rangle^{50}]$, where all enabled activities are listed and the executed activity is underscored. Here, the choice situation between activities b and c is clearly visible, thus preventing the linear modelling of our process.

Despite its benefits, lucent models and translucent event logs are relatively new concepts and are therefore scarcely researched. As a result, translucent event logs are hardly available in real-life process logs. This motivates us to devise novel methods to generate translucent event logs from a non-translucent variant.

1.2 Problem Statement

In order to look deeper into the main problem of this thesis, we first formally define our problem as described below.

Definition 1.1 (Translucent Log Generation Problem)**.** Given an event log $\mathcal{L}$ as input, produce a translucent event log $\mathcal{L}'$ where the set of enabled activities are added as attributes.

There is a variant of the *Translucent Log Generation Problem*, where a process model is provided along with the event log.

Definition 1.2 (Translucent Log Generation Problem - Process Model Variant)**.** Given an event log $\mathcal{L}$ and a process model $\mathcal{M}$ as inputs, produce a translucent event log $\mathcal{L}'$ where the set of enabled activities are added as attributes.

The second variant differs from the first, as the set of enabled activities is constrained by the process model. Note that the function of $\mathcal{M}$ is to provide an upper bound on the set of enabled activities, and the log is there to provide further constraints to enrich the model. Of particular interest are parallel and choice situations, since we are able to deduce supplementary patterns not demonstrated in the process model using log data. We name the first variant defined in Definition 1.1 as *Bottom-up Translucent Log Generation Problem*, whereas the second variant in Definition 1.2 is named as *Top-down Translucent Log Generation Problem.*

1.3 Research Questions

In the scope of the thesis, we define the five research questions below:

(RQ1.) Which techniques can be derived to tackle the *Translucent Log Generation Problem*?

(RQ2.) How can we guarantee the accuracy of predictions made by our algorithm?

(RQ3.) How much value do the novel techniques add compared to the state-of-the-art process discovery algorithms?

(RQ4.) How can the data-centered aspect be incorportaed in our method?

(RQ4.) How should we design and implement an intuitive, user-friendly tool to demonstrate our results?

1.4 Research Goals

The principal research goal is to develop a framework dedicated to solve the *Translucent Log Generation Problem* with the following properties:

1. Accurately reflects the enabled activities
2. Performs well regarding time complexity, particularly with large event logs
3. Easily operable for end-users

By building a meaningful framework, our framework should add value to enhancing existing process discovery algorithms with more accurate models.

Chapter 2

Related Work

2.1 Data-Aware Process Mining

Conventional process discovery algorithms only consider the control-flow aspect of the process, i.e. the activity attribute of the event log, thereby ignoring the data attributes the event log provides. Data-aware process mining, on the other hand, attempts to incorporate both the control-flow and data attributes of the event log. The prevalent repretoire is to discover decision points in the model and to annotate them with guard functions, which in turn are discovered with decision trees. This method of process enhancement is called *decision mining* [1–3], and is a well-estalished field of research within process mining. De Leoni et al. [3] proposes a Petri net with data *(DPN-net)* setting to expand the notation of Petri nets. While previous papers worked with the assumption of deterministic, mutual exclusive transition behavior in decision points, they do not take into account how certain decisions cannot be modeled dichotomously. Often, one needs a softer classification assumption, stating that data attributes affect the decision probabilistically. Mannhardt et al. [4] further extends the concept of data-annotated Petri nets by integrating stochastic information into the model. In the paper, the authors introduce the concept of stochastic labeled data Petri nets *SLDPNs* and propose a method to generate an SLDPN from a Petri net and an event log. Each transition will be mapped with its own weight function learned with the activiation instances of individual transitions using logistic regression.

2.2 Translucent Event Logs

Little research has been performed on the topic of translucent event logs. Being a relatively young concept in the field of process mining, they were first hinted in 2018 by van der Aalst [5], where a possible event log revealing the set of enabled activities is mentioned. [6] formally introduces and defines translucent event logs and relates concepts of lucency and translucency by showing that a lucent process model can be rediscovered by using a translucent event log retrieved from the model. Methods of creating translucent event logs are first discussed in Bayel et al. [7]. Here, a system's screenshot is matched with the labelled activity pattern and annotated with the corresponding activities. Furthermore, a model-based approach is introduced by replaying the event log on the model and annotating enabled activities. [8] formally introduces a precision measure between a Petri

net and a translucent event log by comparing log-enabled activities and model-enabled activities.

2.3 Predictive Process Monitoring

- Explain the term and list previous differnet approaches
- Lin et al. [9]: Next event and next data attribute prediction using RNNs
- Gunnarsson et al. [10]: Remaining trace and runtime prediction using LSTMs
- Bukhsh et al. [11]: Next activity, next event time, and remaining time prediction using transformer architecture
- General overview of the field is given in [12].

Chapter 3

Translucent Log Generation

3.1 Framework Overview

- The program should accept an event log and an optional process model, e.g. a Petri net, as inputs and should return a corresponding translucent event log as output. Users should have the option to select from various methods of log generation, will will be specified below. Mainly, these methods can be classified in two categories: top-down approaches which require a Petri net, and bottom-up approaches which solely need the event log.

3.2 Top-down Approaches

- Given a Petri net and an event log, the program utiltizes the alignment-based appraoch presented in [7] as its baseline algorithm. After computing the alignment for each trace, the program replays the alignment on the reachabiliy graph trace by trace in order to circumvent the silent transitions. For each activity, the algorithm then augments the event log with the corresponding transitions situated in outgoing arcs of the current state. After creating a basic translucent event log, the log can be refined by further algorithms.
- A method to incorporate the data attributes is to implement multivariate regression. Similar to the setting in [4], we construct a training data set for each transition of a Petri net consisting of data attributes and a boolean label indicating whether the transition was executed given the data attributes as input. We then perform a regression analysis on the training data set for each transition. The resulting dictionary of transitions and regression functions can be employed to filter out transitions lying below a certain probability threshold p in each decision point during replay.

3.3 Bottom-up Approaches

- Given an event log, we compute the set of unique activities and generate a next-activity matrix $\mathcal{A}$, where the entry $\mathcal{A}_{ij}$ represents the number of times activity j follows activity i. We can then easily transform $\mathcal{A}$ into a probability matrix $\mathcal{A}'$ by dividing each row by the sum of the row. We then filter out the results by a certain

threshold p and add the entries surviving the threshold to each event trace. This algorithm can serve as baseline for further extensions.

- Furthermore, we can utilitze a deep-learning based black-box approach. The issue with implementing supervised learning algorithms is that we need a labeled training dataset. In our case, this would be a preexisting translucent event log, which is unavailable in our setting due to missing enabled activities data. We can cirucmvent the problem by training the model using the next activity information as label, as this information is available in every event log. Since most learning algorithms would not return a single value but an underlying probability distribution of possible outcomes, we can substitute the final *argmax* operation with selecting a threshold p and returning all labels lying above it.

Note that this is not the final list as we are still in the process of selecting new methods. The final list of methods will be updated in the final version of the thesis paper.

Chapter 4

Evaluation

- Limitations: Using real-life translucent event logs is often implausible for our scenario due to the fact that most real-life logs do not contain the set of enabled activities. Therefore, direct evaluation by receiving a real-life translucent event log as input, stripping away the enabled activities column, inserting the log in our algorithm as input then comparing the result with the original translucent event log is not possible.
- Instead, we can use artificial process models. The evaluation process works like the following:
  1. We generate random process models, e.g. data Petri nets.
  2. We then play-out the model randomly and extract 1. a normal log and 2. a translucent event log.
  3. We then use the normal log as input to our TLG program and compare the result with the original translucent log.
- On top of that, we can also evaluate its versatility by directly comparing models generated using translucent event logs and the usual state-of-the-art process discovery algorithms. The evaluation process works like the following:
  1. Given a normal process log, we use state-of-the-art process discovery algorithms to generate process models.
  2. We use the log as input to our program and generate a translucent event log.
  3. We then generate a process model based on translucent-log based process discovery algorithms.
  4. We then compare the models based on their performance measures.
- In the model annotation setting, we can evaluate the performance of the model extension algorithm by comparing the stochastic precision of the annotated model with the original model. The evaluation process works like the following:
  1. We iterate over each trace and replay it on two models: The original model received as input and the annotated model.

2. For each transition, we calculate the stochastic precision by computing the product of the transition probabilities in each transition step.
3. We then add up the stochastic precision score for each trace and divide it by the total number of traces to get the average stochastic precision score.

Note that this is different from the translucent precision score defined in [8], since we need a precision measure comparable and applicable to both of the original log-model-pair and the translucent-log-annotated-model pair.

- As we are not presenting a single generation method, it will be necessary to compare and evaluate each method separately using the process described above.

Bibliography

[1] A. Rozinat and Wil M. P. van der Aalst. Decision mining in prom. In *Business Process Management*, pages 420–425, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-38903-3.

[2] A. Rozinat and Wil M. P. van der Aalst. *Decision mining in business processes.* BETA publicatie : working papers. Technische Universiteit Eindhoven, 2006. ISBN 90-386-0685-0.

[3] Massimiliano de Leoni and Wil M. P. van der Aalst. Data-aware process mining: discovering decisions in processes using alignments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, page 1454–1461, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450316569. doi: 10.1145/2480362.2480633. URL `https://doi.org/10.1145/2480362.2480633`.

[4] Felix Mannhardt, Sander J. J. Leemans, Christopher T. Schwanen, and Massimiliano de Leoni. Modelling data-aware stochastic processes - discovery and conformance checking. In Luis Gomes and Robert Lorenz, editors, *Application and Theory of Petri Nets and Concurrency*, pages 77–98, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-33620-1.

[5] Wil M. P. van der Aalst. Markings in perpetual free-choice nets are fully characterized by their enabled transitions. *CoRR*, abs/1801.04315, 2018. URL `http://arxiv.org/abs/1801.04315`.

[6] Wil M .P. van der Aalst, Victor Khomenko, Jetty Kleijn, Wojciech Penczek, and Olivier H. Roux. Lucent process models and translucent event logs. *Fundam. Inf.*, 169(1–2):151–177, jan 2019. ISSN 0169-2968. doi: 10.3233/FI-2019-1842. URL `https://doi.org/10.3233/FI-2019-1842`.

[7] Harry H. Beyel and Wil M. P. van der Aalst. Creating translucent event logs to improve process discovery. In Marco Montali, Arik Senderovich, and Matthias Weidlich, editors, *Process Mining Workshops*, pages 435–447, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-27815-0.

[8] Harry H. Beyel and Wil M. P. van der Aalst. Translucent precision: Exploiting enabling information to evaluate the quality of process models. In João Araújo, Jose Luis de la Vara, Maribel Yasmina Santos, and Saïd Assar, editors, *Research Challenges in Information Science*, pages 29–37, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-59468-7.

[9] Li Lin, Lijie Wen, and Jianmin Wang. *MM-Pred: A Deep Predictive Model for Multi-*

attribute Event Sequence, pages 118–126. 2019. doi: 10.1137/1.9781611975673.14. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975673.14.

[10] B. Gunnarsson, S. Broucke, and J. De Weerdt. A direct data aware lstm neural network architecture for complete remaining trace and runtime prediction. *IEEE Transactions on Services Computing*, 16(04):2330–2342, jul 2023. ISSN 1939-1374. doi: 10.1109/TSC.2023.3245726.

[11] Zaharah A. Bukhsh, Aaqib Saeed, and Remco M. Dijkman. Processtransformer: Predictive business process monitoring with transformer network, 2021. URL https://arxiv.org/abs/2104.00721.

[12] Chiara Di Francescomarino and Chiara Ghidini. *Predictive Process Monitoring*, pages 320–346. Springer International Publishing, Cham, 2022. ISBN 978-3-031-08848-3. doi: 10.1007/978-3-031-08848-3_10. URL https://doi.org/10.1007/978-3-031-08848-3_10.