

Improving the eST-Miner Models by Replacing Imprecise Structures Using Place Projection

Christian Rennert*, Lisa L. Mannel* and Wil M. P. van der Aalst

Chair of Process and Data Science (PADS), RWTH Aachen University, Germany

Abstract

In process discovery, the goal is to find, for a given event log, the model describing the underlying process. While process models can be represented in a variety of ways, Petri nets form a theoretically well-explored description language. A Petri net describes the underlying process well if it allows for all relevant behavior (fitness) and, at the same time, restricts other process executions not contained in the event log (precision). A process discovery algorithm aiming to balance these often conflicting requirements is the eST-Miner. To this end, the approach employs a user-definable parameter which indicates a lower bound for the fraction of behavior in the event log that each place in the Petri net must be able to replay. As the eST-Miner is restricted to uniquely labeled transitions, i.e., it does not return silent or duplicate transitions, there are inputs where precision is decreased for the purpose of guaranteeing minimal fitness. An example is the situation where part of the process is optional and can be skipped, which is usually modeled by using a silent transition. Being constraint to uniquely labeled transitions, the eST-Miner models such behavior using a place with a loop. On the one hand this correctly allows to skip the looping behavior (maintains fitness), on the other hand precision is decreased drastically by allowing the behavior to be repeated arbitrarily often. Thus, the models returned by the eST-Miner often contain imprecise substructures, most prominently, "flower-like" places (places with one-loops). In this paper, we aim to replace such places by more meaningful and precise submodels that include silent transitions, while preserving desirable structures existing in the input uniquely labeled Petri net. We describe an approach to distill the behavior related to a specific, imprecise place from the event log such that the resulting log projection can be used to discover and insert a more precise Petri net. Furthermore, extensions of the approach to more complex imprecise structures are discussed.

Keywords

Process Mining, Process Discovery, Process Enhancement, Petri Nets, eST-Miner, Place Projection

1. Introduction and Related Work

Today's organizations collect increasing amounts of data corresponding to processes they handle, so-called *event data*. Every event corresponds to the execution of an activity within the run of an instance of the process. From such event data one can extract an *event log*, which groups the events into cases, i.e., the sequences of activities sequentially recorded for one process instance. While additional data attributes may be stored for events, they are not considered in this work.

ATAED'23: Algorithms & Theories for the Analysis of Event Data 2023, June 26–27, 2023, Lisbon, Portugal

*Corresponding author.


✉ rennert@pads.rwth-aachen.de (C. Rennert); mannel@pads.rwth-aachen.de (L. L. Mannel);

wvdaalst@pads.rwth-aachen.de (W. M. P. v. d. Aalst)

ORCID [0000-0003-4614-6171](https://orcid.org/0000-0003-4614-6171) (C. Rennert); [0000-0001-6158-356X](https://orcid.org/0000-0001-6158-356X) (L. L. Mannel); [0000-0002-0955-6940](https://orcid.org/0000-0002-0955-6940) (W. M. P. v. d. Aalst)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

An area that provides methods for gaining insights and extract information and value from such data is the field of *process mining*, which connects data mining and process science.

The sub-field of *process discovery* applies discovery algorithms to an event log to find a *process model* that describes the relations of the occurrences of activities. Such relations include conditionality, concurrency or choice between activities in a process and are reflected in the *behavior* of the event log. A good process model can help to better understand and subsequently improve the process, e.g., by reducing inefficiencies or detecting quality issues. An ideal discovery algorithm returns a process model for an input event log that represents all process executions in the event log (*fitness*) while not allowing for behavior not contained in the event log (*precision*), is easy to understand (*simplicity*) and, finally, is likely to also include future process executions similar to the examples given in the event log, i.e., is not overfitting the log (*generalization*). For real-life event logs, these goals are usually in conflict with each other and it is impossible to perfectly satisfy all of them simultaneously.

The discovery algorithm *eST-Miner* (efficient Statespace Traversal-Miner) [1] aims to balance the different quality dimensions. It can provide fitness guarantees, includes options to filter infrequent behavior and is able to discover Petri nets that include complex control-flow structures, like long-term dependencies (non-free choice constructs) between activities in a process, enabling increased precision compared to algorithms that are limited to the discovery of basic structures. This approach can achieve high fitness and precision for event logs that can be represented by the class of *uniquely labeled Petri nets*. However, many processes require silent or duplicate transition labels to be fully described. In such cases, the *eST-Miner* discovers less precise process models in comparison to algorithms that return *non-uniquely labeled Petri nets*.

In this work, we aim to remedy this issue by introducing a framework that combines the *eST-Miner*'s ability to discover complex control-flows with the added expressiveness of silent transitions. Based on a uniquely labeled Petri net discovered by the *eST-Miner*, the goal of the framework is to introduce silent transitions to improve precision while preserving fitness. The approach first identifies imprecise structures in the given Petri net and computes the corresponding parts of the event log. Based on each sublog, it discovers a new, non-uniquely labeled Petri net and replaces the imprecise structure with this model without impacting the behavior expressed by the surrounding parts of the model.

A technique for the computation of the sublog and for the replacement of the imprecise structure are both presented in this paper. Considering the identification of imprecise structures, in this work, we focus on easily detectable candidate imprecise structures such as loops.

The proposed framework is related to ideas presented in the field of incremental process discovery techniques. In particular, the work in [2] is the most similar to our approach. The authors propose an architecture that incrementally discovers process models on individual logs and merges them into an existing process model. However, while their approach incrementally adds new traces to the seen behavior, our framework considers the complete event log from the beginning and refines on sublogs. Another approach related to our work is presented in [3]. They introduce a repair technique to insert process models into existing process models to increase fitness. To this end, they first identify the location of deviations in a given process model for a given log. Based on this information, a log is constructed to discover a process model that can be inserted at the identified location. A major difference to the approach proposed in this work is that we do not intend to extend the behavior of the given accepting Petri net

but rather to reduce it. The repair approach presented in [4] also aims to increase precision of a given net by constraining behavior that is not in the input log. However, their approach addresses a different reason for impreciseness: they use region-theory to add missing non-local constraints, while we focus on repairing imprecise structures caused by the inability to model optional behavior by adding silent transitions.

In this work we focus on Petri nets discovered by the eST-Miner. However, the ideas and concepts can be extended to Petri nets produced by other approaches. In particular, approaches rooted in region theory [5, 6, 7, 8] often exclude silent transition labels and therefore may synergize well with the presented approach.

The framework described in this paper was inspired by a master thesis [9]. Here, we refine the approach while focusing on the eST-Miner and include a detailed evaluation.

The remainder of this work is structured as follows. In Section 2, we present the mathematical notations and concepts used in this work. Section 3 introduces the eST-Miner. Section 4 presents the approach to obtain a more precise process model by replacing imprecise structures. Section 5 applies the approach presented on real-life logs. In Section 6, we conclude this paper and give an outlook for future work.

2. Basic Notations, Event Logs and Process Models

We use of the following definitions and notations: A set, e.g. $\{a, b, c, d\}$, contains every element once, while a multiset can contain an element more than once, e.g. $[a, b, a, a, b, c] = [a^3, b^2, c]$. Set union, intersection and difference are defined as usual. By $\mathbb{P}(X)$ we refer to the power set of the set X , and $\mathbb{M}(X)$ is the set of all multisets over the set X . In contrast to sets and multisets, a sequence is ordered. It can contain an element more than once, e.g., $\langle a, a, b, a, b, b \rangle$. We denote the number of elements of a set, multiset or sequence X as $|X|$. A projection of a sequence $\sigma \in X^*$ on a subset of activities $Y \subseteq X$ is denoted by $\sigma \upharpoonright_Y$. For example, $\langle a, b, a, c, b \rangle \upharpoonright_{\{a,c\}} = \langle a, a, c \rangle$. Further, we uplift all functions that are applicable to single elements of a sequence to be applicable to the full sequence, i.e., for a function $f : X \rightarrow Y$ and a sequence $\langle x_1, x_2, \dots, x_n \rangle$ we write $f(\langle x_1, x_2, \dots, x_n \rangle) = \langle f(x_1), f(x_2), \dots, f(x_n) \rangle$. By \mathcal{A} we denote the universe of all possible *activities* (e.g., actions or operations). A *trace* is a finite sequence where each element is an activity from the universe of activities \mathcal{A} . The universe of such traces is denoted by \mathcal{T} . A *log* $L \in \mathbb{M}(\mathcal{T})$ is a multiset of traces. We aim to represent the behavior described by an event log using a Petri net with transitions labeled according to the activities in the event log. A sequence of transitions fired describes a trace corresponding to the sequence of the transition labels. A so-called silent transition is labeled with the silent activity label τ and adds no activity to a trace.

Definition 2.1 (Labeled Petri Net) *We define a labeled Petri net as a quintuple $N = (P, T, F, A, l)$ where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, $A \subseteq \mathcal{A}$ with $\tau \notin A$ is a set of non-silent activity labels, and $l : T \rightarrow A \cup \{\tau\}$ is a labeling function assigning to each transition a (possibly silent) activity label.*

We define the *preset* and *postset* of places and transitions as usual, i.e., given a Petri net $N = (P, T, F, A, l)$ and a place or transition $x \in P \cup T$, the preset of x is $\bullet x = \{x_{in} \in P \cup T \mid (x_{in}, x) \in F\}$ and the postset of x is $x \bullet = \{x_{out} \in P \cup T \mid (x, x_{out}) \in F\}$. The current state of a labeled Petri net is

described by its marking. A marking is a mapping function $M : P \rightarrow \mathbb{N}_0$ for a set of places P and describes the number of tokens that each place holds. An accepting Petri net is a labeled Petri net having an initial marking and a final marking, which we use to define its behavior.

Definition 2.2 (Accepting Petri Net) *An accepting Petri net is defined as a triplet $SN = (N, M_i, M_f)$, where $N = (P, T, F, A, l)$ is a labeled Petri net and $M_i : P \rightarrow \mathbb{N}_0$ an initial marking and $M_f : P \rightarrow \mathbb{N}_0$ a final marking.*

The behavior of an accepting Petri net is based on the firing rule for transitions. A transition t is *enabled*, if all ingoing places in $\bullet t$ hold at least one token in the current marking M , i.e., if $\forall p \in \bullet t : M(p) \geq 1$ holds. An enabled transition can *fire*, consuming a token from each place in its preset and producing a token in each place in its postset. Thus, firing a transition t changes the current marking M to marking M' , denoted as $M[t]M'$, where for M' it holds that $M'(p) = M(p) - 1$, if $p \in (\bullet t \setminus t \bullet)$ or $M(p) + 1$, if $p \in (t \bullet \setminus \bullet t)$ or $M(p)$ otherwise. The behavior of an accepting Petri net is given by the set of its valid firing sequences, which are sequences of transitions that can be fired conclusively to lead from the initial marking to the final marking.

Definition 2.3 (Valid Firing Sequences) *Let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$ be an accepting Petri net and let $\omega = \langle t_1, t_2, \dots, t_n \rangle$ with $\omega \in T^*$ be a sequence of transitions. We call ω a firing sequence in SN from marking M_0 if we have $M_0[t_1]M_1[t_2]M_2 \dots M_{n-1}[t_n]M_n$. This is denoted as $M_0[\omega]M_n$. We call ω a valid firing sequence in SN if $M_0 = M_i$ and $M_n = M_f$.*

The non-silent transition labels of a valid firing sequence describe a trace that can be replayed using this firing sequence.

Definition 2.4 (Replayable Traces) *Consider an accepting Petri net $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$ and a trace $\sigma \in A^*$. We say that σ is replayable on SN if and only if there exists a valid firing sequence $\omega = \langle t_1, t_2, \dots, t_n \rangle$ such that the projection of ω on its non-silent transition labels equals σ , i.e., $\sigma_\omega = \langle l(t_1), l(t_2), \dots, l(t_n) \rangle \setminus_A = \sigma$.*

The approach presented in this paper aims to improve a given accepting Petri net SN based on an event log L , such that fitness is preserved while precision is ideally increased. Various metrics have been introduced for measuring or approximating fitness and precision with different advantages and disadvantages. In this work, we abstract from concrete metrics as we are interested only in the relation between these quality aspects when comparing two accepting Petri nets SN and SN' . To this end, we compare the traces replayable by both nets.

Definition 2.5 (Comparing Fitness and Precision) *Consider an event log L and two accepting Petri nets SN and SN' . SN' is at least as fitting as SN , if every trace $\sigma \in L$ replayable on SN is also replayable on SN' . SN' is at least as precise as SN , if every trace $\sigma \in \mathcal{T}$ replayable on SN' is also replayable on SN . SN' is more precise than SN , if additionally a trace $\sigma' \in \mathcal{T} \setminus L$ exists that is replayable on SN but not replayable on SN' .*

3. Introducing the eST-Miner

Several variants and extensions of the eST-Miner have been proposed in the past years [1, 10]. In the following, we briefly introduce the variant used as the basis of this work. For further details, we refer the interested reader to the respective papers.

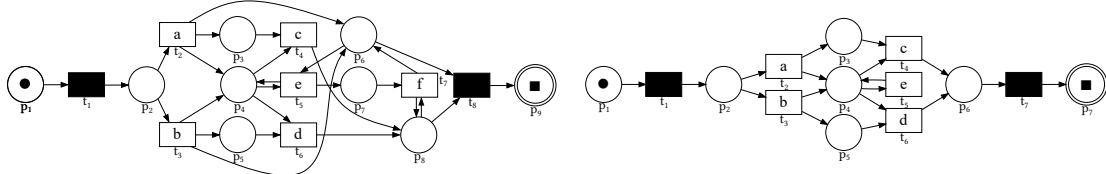
As input, the algorithm takes a log L and returns an accepting Petri net as output. Inspired by language-based regions, the basic strategy of the approach is to begin with a Petri net whose transition labels correspond exactly to the activities used in the given log. From the finite set of unmarked, intermediate candidate places, the subset of all *fitting* places is computed and inserted by connecting them to their uniquely labeled ingoing and outgoing transitions. Such places are uniquely identifiable by their sets of ingoing and outgoing transitions, thus there are $|\mathbb{P}(A) \times \mathbb{P}(A)|$ candidate places. The eST-Miner is able to filter infrequent behavior by deeming a place to be fitting even if it does not allow for replaying the complete event log. However, in this paper, we require it to accept places as fitting only if all traces in the event log are replayable (feasible places). Places are evaluated using token-based replay. To avoid replaying the log on all candidate places (exponential in the number of activities), it organizes the potential places as a set of trees. When traversing these trees, their special structure allows to cut off subtrees, and thus candidates, based on the replay result of their parent [1]. This greatly increases efficiency, while still guaranteeing that all fitting places are found.

To facilitate further computations and human readability, *implicit* places are identified and removed [11, 12, 13]. A place is implicit if its removal does not increase the behavior of the accepting Petri net. Implicit places can be detected based on the structure of the accepting Petri net as proposed for the first eST-Miner variant [1], or by using the faster replay-based implicit place removal strategy introduced in [14].

We emphasize the following details since they become relevant in the context of this work. Note that the eST-Miner adds designated start and designated end activities to the input event log, which are reflected by correspondingly labeled start and end transitions. A source place, marked in the initial marking, is added as the preset of the start transition, and a sink place, marked in the final marking, forms the postset of the final transition. In the context of this work, we relabel the start and end transitions of each model discovered using the eST-Miner as silent transitions. Furthermore, the eST-Miner variant used here is restricted to only discover accepting Petri nets that have perfect fitness for the input log, i.e., all log traces are replayable. Thus, with the start and end transitions relabeled to be silent, the input log without added artificial start and end activities has perfect fitness on the discovered net.

As an example, consider the accepting Petri net SN_a shown in Figure 1a discovered on the log $L_a = [\langle a, c \rangle, \langle b, d \rangle, \langle a, e, c, f \rangle, \langle b, e, d, f \rangle]$ using the eST-Miner. SN_a illustrates the discovery of complex control-flow structures and is perfectly fitting and precise with respect to L as it does not replay any trace not in L . Nevertheless, undesired places which are decreasing readability are inserted due to the limitation of the eST-Miner to not use silent transitions for the representation of skippable behavior such as the activities e and f being skippable.

Consider $L_b = [\langle a, c \rangle, \langle b, d \rangle, \langle a, e, c \rangle, \langle b, e, d \rangle]$ as a similar log, for which the eST-Miner discovers the accepting Petri net SN_b shown in Figure 1b. SN_b allows for arbitrarily many occurrences of e between a and c or between b and d respectively. Such a looping structure, which allows for a transition to be fired arbitrarily often (or not at all) and in arbitrary order, is commonly referred to as a flower construct [15]. Such a place is usually undesired as inserting it marginally restricts the behavior of the corresponding Petri net and does not reflect the input event log precisely. Unfortunately, due to its inability to model optional behavior with the help of silent transitions, the eST-Miner is often forced to resort to such structures. In the following section, we target such imprecise accepting Petri nets and replace such loops with more precise structures.



(a) Accepting Petri net SN_a discovered on L_a using the eST-Miner. (b) Accepting Petri net SN_b discovered on L_b using the eST-Miner.

Figure 1: Introductory examples of accepting Petri nets discovered by the eST-Miner.

4. Identification and Replacement of Imprecise Structures

The eST-Miner frequently returns Petri nets that over-approximate optional behavior using rather imprecise looping structures. Specifically, we refer to k -loops as *candidate imprecise structures*.

Definition 4.1 (k -Loops) Let $N = (P, T, F, A, I)$ be a Petri net that contains a sequence of unique nodes $\langle n_1, n_2, \dots, n_l \rangle \in (P \cup T)^*$ of length l , which are forming a directed cycle from and to a place $p \in P$, i.e., $p \in (\bullet n_1 \cup n_l \bullet) \wedge \forall_{i \in [1, l-1]} : n_i \in \bullet n_{i+1}$ and $\forall_{i, j \in [1, l]} : n_i = n_j \Rightarrow i = j$ holds. We refer to such a sequence as a k -loop of the place p , where k denotes the number of transitions in the node sequence.

Given an accepting Petri net, the so-called *Projection Discovery with the eST-Miner* presented in the following replaces a place that has *one-looping* transitions with a more precise accepting Petri net while preserving fitness, i.e., all traces from the input log remain replayable. Replayable behavior outside of the event log is never increased, with the goal being to reduce it.

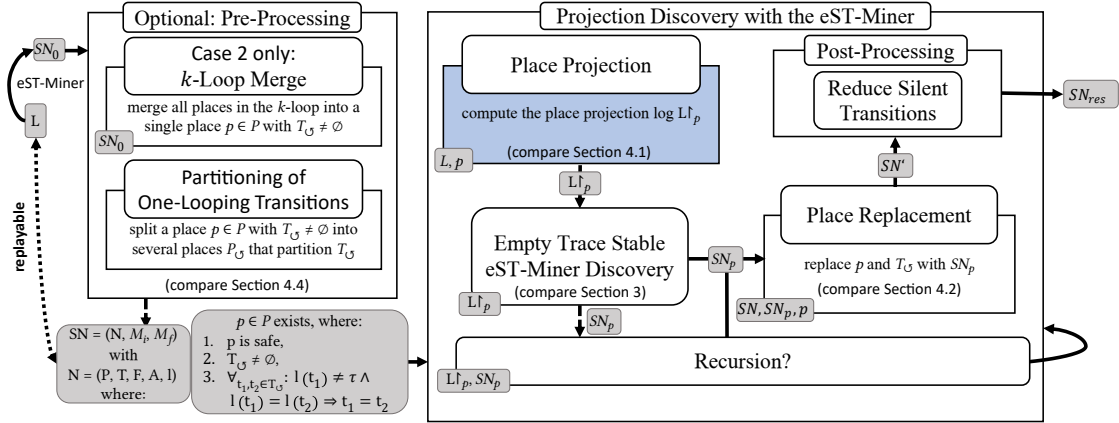


Figure 2: High-level overview of the proposed algorithmic framework. The newly introduced place projection (represented as a blue box) is proposed in Section 4.1.

The overall algorithmic framework is outlined in Figure 2. Initially, a log L and an accepting Petri net SN with perfect fitness on this log are given. The projection discovery starts with the procedure detailed in Section 4.1, for each safe place p with uniquely labeled and non-silent

one-loops we compute a *place projection log* representing the behavior of the place with respect to the input log and one-looping transitions. Notably, this projected log will contain the empty trace in case the looping transitions reflect optional behavior. Based on the place projection log, we use the eST-Miner to discover a new subnet SN_p and insert it into SN replacing p and its one-looping transitions. The replacement technique is detailed in Section 4.2. Note, that the projection discovery can be applied recursively to the discovered subnets until no improvement can be found. Further, we enable the eST-Miner to discover a more precise model by removing all occurrences of the empty trace from the projected event log before starting the discovery. If such traces indeed exist, then we add a silent transition to the discovered SN_p , making the complete net *skippable*.

Our input of the projection discovery can be refined by pre-processing SN such that k -loops are considered or such that better results are obtained by using a place partitioning technique. For a k -loop contained in SN , we merge all places in the k -loop into a single place as detailed in Section 4.3. This reduces the k -loops to one-loops. Place partitioning splits p into a set of places to be handled individually and to find more meaningful logs, as detailed in Section 4.3.

Note that in our implementation and presented examples, we reduce silent transitions in the final accepting Petri net while preserving its behavior [16] to improve readability.

The approach guarantees to preserve fitness and precision, as detailed in Section 4.4.

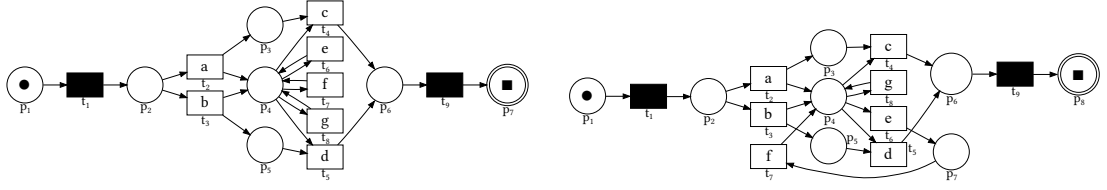
4.1. Projection of a Log on a Safe Place

As input, the place projection expects an accepting Petri net $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$, a firing sequence ω on the accepting Petri net SN and a safe place $p \in P$ that is unmarked in M_i and M_f and that has a non-empty set of uniquely labeled, non-silent one-looping transitions T_\circ . As a result, the place projection of ω on p returns a place projection log L_p satisfying the following properties:

Property 1 L_p holds a trace for each non-extendable subsequence of ω for which p holds a token during firing of ω on SN and

Property 2 for all $t \in T_\circ$ fired while a token is contained in p , their label (if non-silent) is appended to the same trace of L_p , and for no $t' \notin T_\circ$ their label is added.

In the following, we give further explanation for a place projection log to satisfy the two criteria. As we replace the place p and its one-looping transitions T_\circ with an accepting Petri net SN_p discovered on the log L_p , we do not want to model behavior that is modeled by other transitions (*Property 2*). Further, for our replacement technique, we expect the replacing accepting Petri net SN_p to be able to reach its final marking from its initial marking when sequentially firing the activities corresponding to the one-looping transitions contained in a subsequence of ω , for which p is non-empty when firing ω on SN (*Property 1*, *Property 2*). As our used discovery algorithm returns a perfectly fitting Petri net on its input log, we conclude *Property 1* and *Property 2* to be strongly influential towards the behavior of an accepting Petri net discovered on a place projection log. If both properties hold for a place projection function, we expect an accepting Petri net discovered on the place projection log L_p to cover the same behavior from ω as the place p to be replaced.



(a) Accepting Petri net SN_1 discovered on the input $\log L_1 = [\langle a, c \rangle, \langle b, d \rangle, \langle a, e, c \rangle, \langle b, e, d \rangle, \langle a, e, f, c \rangle, \langle b, e, f, d \rangle, \langle a, e, g, c \rangle, \langle b, e, g, d \rangle]$ using the eST-Miner. (b) Accepting Petri net SN_c discovered on the input $\log L_c = [\langle a, c \rangle, \langle a, e, f, c \rangle, \langle a, e, f, g, c \rangle, \langle b, d \rangle, \langle b, e, f, d \rangle, \langle b, e, f, g, d \rangle]$ using the eST-Miner.

Figure 3: Example input Petri nets with a one- and two-looping place respectively.

To detect increments and decrements in the number of tokens of a place between two subsequent markings, we introduce the token sequence of a place and a valid firing sequence as a new concept.

Definition 4.2 (Token Sequence of a Place and a Valid Firing Sequence) Let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$ be an accepting Petri net, $p \in P$ be a place and $\omega = \langle t_1, t_2, \dots, t_n \rangle$ be a valid firing sequence in SN . We define the sequence of token states of p and ω as the sequence $\varphi(p, \omega) = \langle M_i(p), M_1(p), M_2(p), \dots, M_{n-1}(p), M_f(p) \rangle$, where $M_i [t_1] M_1 [t_2] M_2 [t_3] \dots [t_{n-1}] M_{n-1} [t_n] M_f$ holds.

We use the concept of the token sequences to construct a projection function satisfying *Property 1* and *Property 2* described above. Definition 4.3 gives the projection of a valid firing sequence on a safe place as the multiset of all non-extendable subsequences for which p is non-empty during firing of ω .

Definition 4.3 (Projection of a Valid Firing Sequence on a Safe Place) Let $SN = (N, M_i, M_f)$ be an accepting Petri net with $N = (P, T, F, A, l)$, $p \in P$ be a safe place in the Petri net where $M_i(p) = 0 \wedge M_f(p) = 0$ holds, $T_\circ = \bullet p \cap p \bullet \neq \emptyset$ the set of non-silent and uniquely labeled one-looping transitions of the place p , i.e., $\forall_{t_1, t_2 \in T_\circ} : l(t_1) \neq l(t_2) \Rightarrow t_1 = t_2$, σ_ω a trace, $\omega = \langle t_1, t_2, \dots, t_n \rangle$ a valid firing sequence of SN replaying σ_ω and $\varphi(p, \omega) = \langle c_0, c_1, c_2, \dots, c_n \rangle$ a token sequence with $n \in \mathbb{N}$. The projection $\omega \upharpoonright_p$ of the firing sequence ω on the safe place p is the multiset of sequences

$$\omega \upharpoonright_p = [l(\langle t_{i+1}, t_{i+2}, \dots, t_j \rangle) \upharpoonright_{T_\circ} \mid i, j \in [2, n-1] \wedge i < j \wedge c_{i-} = 0 \wedge c_{j+1} = 0 \wedge \forall k \in [i, j] : c_k > 0].$$

In the context of this work, we expect the firing sequence of a trace to be *non-ambiguous*. This holds trivially for the eST-Miner as it discovers uniquely labeled accepting Petri nets only.

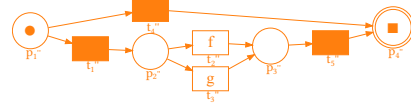
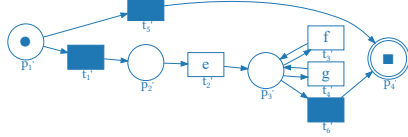
We uplift the concept of the place projection to logs to obtain a place projection $\log L \upharpoonright_p$ by considering the union of all projections of firing sequences on p corresponding to each trace in the log, i.e., $L \upharpoonright_p = \bigcup_{\sigma_\omega \in L} \omega \upharpoonright_p$. In Table 1, examples of token sequences and place projections are shown for p_4 of the accepting Petri net SN_c shown in Figure 3b.

The next step in our framework is to discover an accepting Petri net $SN_p = (N_p, M_{i,p}, M_{f,p})$ with $N_p = (P_p, T_p, F_p, A_p, l_p)$ on the projection $\log L \upharpoonright_p$. Note that if the log contains the empty trace, i.e., $\langle \rangle \in L \upharpoonright_p$, that an empty trace stable eST-Miner used in this work then discovers

Table 1

Example for token sequences and place projections for the place p_4 in SN_c shown in Figure 3b.

Input trace	Firing sequence	Token sequence for p_4	Projected on p_4
$\langle a, c \rangle$	$\langle t_1, t_2, t_4, t_9 \rangle$	$\langle 0, 0, 1, 0, 0 \rangle$	$[\langle \rangle]$
$\langle a, e, f, e, f, g, c \rangle$	$\langle t_1, t_2, t_6, t_7, t_6, t_7, t_8, t_4, t_9 \rangle$	$\langle 0, 0, 1, 0, 1, 0, 1, 1, 0, 0 \rangle$	$[\langle \rangle^2, \langle g \rangle]$
$\langle b, g, e, f, g, g, d \rangle$	$\langle t_1, t_3, t_8, t_6, t_7, t_8, t_8, t_5, t_9 \rangle$	$\langle 0, 0, 1, 1, 0, 1, 1, 1, 0, 0 \rangle$	$[\langle g \rangle, \langle g, g \rangle]$



(a) Accepting Petri net $SN_2 = (N_2, M_{i,2}, M_{f,2})$ with $N_2 = (P_2, T_2, F_2, A_2, L_2)$ discovered on L_2 . (b) Accepting Petri net $SN_3 = (N_3, M_{i,3}, M_{f,3})$ with $N_3 = (P_3, T_3, F_3, A_3, L_3)$ discovered on L_3 .

Figure 4: Finding accepting Petri nets based on place projection logs.

an accepting Petri net on $L \upharpoonright_p \setminus \{\langle \rangle\}$ and inserts a skipping silent transition t_τ , i.e., $l(t_\tau) = \tau \wedge \bullet t_\tau = \{p_i \in P_p \mid M_{i,p}(p_i) > 0\} \wedge t_\tau \bullet = \{p_f \in P_p \mid M_{f,p}(p_f) > 0\}$.

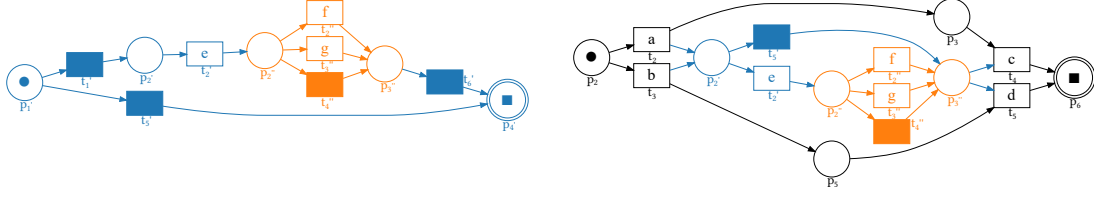
Considering our running example, we obtain the place projection log $L_2 = L_1 \upharpoonright_{p_3} = [\langle \rangle^2, \langle e \rangle^2, \langle e, f \rangle^2, \langle e, g \rangle^2]$. We discover the accepting Petri net SN_2 shown in Figure 4a with the empty trace stable eST-Miner on L_2 . Here, we apply the projection discovery recursively with the place p_2 in SN_2 and L_2 as input. We obtain the place projection log $L_3 = L_2 \upharpoonright_{p'_2} = [\langle \rangle^2, \langle f \rangle^2, \langle g \rangle^2]$ and discover the accepting Petri net SN_3 shown in Figure 4b. Here, no further recursion is applied. At this point, place replacement is applicable. Intuition on the replacement of a place with an accepting Petri net and corresponding desired properties are given in the next section.

4.2. Replacing a Place with an Accepting Petri Net

In this section, we describe the replacement of a place p with an accepting Petri net $SN_p = (N_p, M_{i,p}, M_{f,p})$ discovered on the place projection log $L \upharpoonright_p$. First, the place p and all its one-looping transitions $T_\odot = \bullet p \cap p \bullet$ are removed (Step 1). Further, we connect all other ingoing (respectively outgoing) transitions of p as ingoing (respectively outgoing) to each place that holds a token in the initial marking $M_{i,p}$ (Step 2) (respectively, final marking $M_{f,p}$) (Step 3). Lastly, we convey all restrictions that a transition $t \in T_\odot$ has to all inserted transitions that share the label with the transition t (Step 4).

Steps (1)-(3) aim to replace the place p and its one-looping transitions T_\odot with the accepting Petri net SN_p such that, with respect to the replay of all traces in the log, a token is produced in all places marked by $M_{i,p}$ whenever a token was produced in p and to consume a token from all places marked by $M_{f,p}$ whenever a token was consumed from p . Step (4) aims to convey all restrictions on the former one-looping transitions that are not imposed by p itself. We formalize these replacement steps in Definition 4.4.

Definition 4.4 (Replacing a Place with an Accepting Petri Net) Let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$ be an accepting Petri net, $p \in P$ a place with its one-looping transitions



(a) Accepting Petri net $SN_{2,3}$, which is the result of replacing the place p'_2 in SN_2 in Figure 4a with the accepting Petri net SN_3 in Figure 4b. (b) Accepting Petri net $SN_{1,2,3}$, which is the result of replacing the place p_3 in SN_1 in Figure 3a with the accepting Petri net $SN_{2,3}$ in Figure 5a.

Figure 5: Replacement of place projection places in Figure 3a and 4a with the accepting Petri nets in Figures 4b and 5a respectively.

$T_{\odot} = \bullet p \cap p \bullet \neq \emptyset$ to be replaced with the accepting Petri net $SN_p = (N_p, M_{i,p}, M_{f,p})$ with $N_p = (P_p, T_p, F_p, A_p, l_p)$. This results in an accepting Petri net $SN' = (N', M'_i, M'_f)$ with $N' = (P', T', F', A', l')$, where the following holds:

$$\begin{aligned}
 P' &= (P \cup P_p) \setminus \{p\} & F' &= (F \cup F_p) \setminus (\bullet p \times \{p\} \cup \{p\} \times p \bullet) \cup \\
 T' &= (T \cup T_p) \setminus T_{\odot} & & \{(t_i, p_i) \in (\bullet p \setminus T_{\odot}) \times P_p \mid M_{i,p}(p_i) > 0\} \cup \\
 A' &= A & & \{(p_f, t_o) \in P_p \times (p \bullet \setminus T_{\odot}) \mid M_{f,p}(p_f) > 0\} \cup \\
 M'_i &= M_i, M'_f = M_f & & \bigcup_{t_i \in T_{\odot}} (\{(p_i, t_r) \in (\bullet t_i \setminus \{p\}) \times T_p \mid l(t_i) = l_p(t_r)\} \cup \\
 & & & \{(t_r, p_o) \in T_p \times (t_i \bullet \setminus \{p\}) \mid l(t_i) = l_p(t_r)\}).
 \end{aligned}$$

In the following we continue our example from Figure 3a and Figure 4. In Figure 5a, we show the accepting Petri net $SN_{2,3}$ resulting from replacing p'_2 in SN_2 with SN_3 . Here, L_2 can still be replayed on $SN_{2,3}$ while precision is improved compared to SN_3 . The accepting Petri net $SN_{1,2,3}$ shown in Figure 5b results from replacing p_3 in SN_1 with $SN_{2,3}$. All traces occurring in L_1 are replayable on $SN_{1,2,3}$ and no other. Thus, fitness is preserved and precision is improved compared to SN_1 .

In the following section we introduce pre-processing steps that can be applied optionally in our framework to also consider k -loops as candidate imprecise structures and to improve the quality of the results.

4.3. Pre-Processing of Accepting Petri Nets

In this section, we are going to discuss how to make place projection feasible for k -loops and an approach on how to handle the case where we discover an accepting Petri net that does not improve precision when being replaced. First, we have a look on the place projection on k -loops.

Place Projection on k-Loops So far, we limited our method to project a log on a single place with one-looping transitions. Nevertheless, the eST-Miner and other discovery algorithms

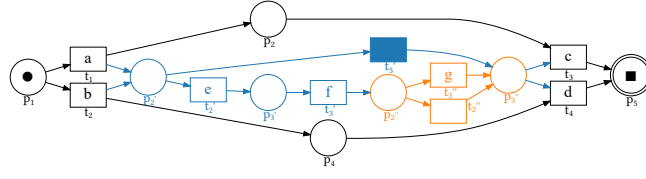


Figure 6: Resulting accepting Petri net of applying the framework to L_c and SN_c shown in Figure 3b.

frequently return Petri nets that include k -loops. Therefore, we extend our approach to also allow for the projection of a log on multiple places.

Consider a k -loop in an accepting Petri net $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, I)$ that contains a set of places $P_\odot \subseteq P$. Combining all places into a single place p , where $\bullet p = \bigcup_{p' \in P_\odot} \bullet p'$ and $p \bullet = \bigcup_{p' \in P_\odot} p' \bullet$ holds, does not restrict the behavior of the accepting Petri net SN . Further, the framework is applicable as the combined place has a non-empty set of one-looping transitions. An example of this approach is given for SN_c discovered on L_c from our example shown in Figure 3b. Here, we merge p_4 and p_7 into a single place resulting in an accepting Petri net equivalent to SN_1 shown in Figure 3a. The result of the projection discovery on SN_1 with L_c is shown in Figure 6.

In the following, we present that splitting a place into several places and partitioning the one-looping transitions can be applied to improve precision when the discovered accepting Petri net SN_p does not improve precision.

Partitioning of One-Looping Transitions Let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, I)$ be an accepting Petri net and let $p \in P$ be a place with one-looping transitions $T_\odot = \bullet p \cap p \bullet \neq \emptyset$. Splitting the place p into a set of parallel places P_\odot , where $\bigcup_{p' \in P_\odot} \bullet p' \cap p' \bullet = T_\odot \wedge \forall p' \in P_\odot : \bullet p' = \bullet p \wedge p' \bullet = p \bullet$ holds, does not decrease fitness. The place projection log for each place $p' \in P_\odot$ is different from the place projection log for the place p . Thus, we conclude different accepting Petri nets to be discovered which we can insert. Therefore, we can apply this technique when our framework does not further improve precision. Heuristic strategies for partitioning the transitions are explored in [9] but out of scope of this work.

In the following section we formalize the guarantees on fitness and precision preservation and sketch the corresponding proofs.

4.4. Guarantees of the Framework

In this section, we give a more detailed view on the guarantees provided by our framework. The framework guarantees to preserve fitness and precision as formalized in Theorem 4.1 and Theorem 4.2.

Theorem 4.1 (Preservation of Fitness) *Let L be an event log and let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, I)$ be an accepting Petri net on which each trace $\sigma \in L$ has a unique valid firing sequence ω . Further, let $p \in P$ be a safe place that has a set of uniquely labeled non-silent one-looping transitions $T_\odot = \bullet p \cap p \bullet \neq \emptyset$ with $\forall t_1, t_2 \in T_\odot : l(t_1) \neq \tau \wedge l(t_1) = l(t_2) \Rightarrow t_1 = t_2$ that does not a*

hold a token in the initial or final marking, i.e., $M_i(p) = M_f(p) = 0$. Let $SN' = (N', M'_i, M'_f)$ with $N' = (P', T', F', A', l')$ be the accepting Petri net obtained by applying the projection discovery. For such an accepting Petri net SN' , it holds that all traces $\sigma \in L$ have a valid firing sequence ω' and therefore fitness to be preserved.

Proof. For any $\sigma \in L$ that has a valid firing sequence $\omega = \langle t_1, t_2, \dots, t_n \rangle$ on SN , we conclude a valid firing sequence $\omega' = \langle t'_1, t'_2, \dots, t'_m \rangle$ to exist on SN' where $n, m \in \mathbb{N}$ and which replays σ on SN' . Recall, that for the place projection to be unambiguous and for our replacement technique to be applicable, we expect each trace to be replayed by the same firing sequence throughout the whole framework. We consider the token sequence $\varphi(p, \omega) = \langle c_0, c_1, \dots, c_n \rangle$. If every token state is equal to zero, we conclude the firing sequence ω' to be equal to ω as the replacement technique does not influence the token movement and the transitions fired.

Next, we consider a token sequence where a value is non-zero. We are interested in the first non-extendable subsequence $\langle c_k, c_{k+1}, \dots, c_j \rangle$ where $\forall_{i \in [k, j]} : c_i = 1$ with $j \in [2, n]$ and $k \in [1, j-1]$.

Consider the prefix $\omega_a \omega_b$ of ω , with $\omega_a = \langle t_1, t_2, \dots, t_k \rangle$ and $\omega_b = \langle t_{k+1}, t_{k+2}, \dots, t_{j-1} \rangle$. Since ω_a does not consume a token from p , we conclude it to be exceptionable on SN' and ω' to start with ω_a . Considering the sequence $\omega_b = \langle t_{k+1}, t_{k+2}, \dots, t_{j-1} \rangle$, we distinguish for each transition whether its in the set of shared transitions $T \setminus T_\circ$ (1) or in the set T_\circ (2).

In Case (1), we know that such a transition $t_1 \in T \setminus T_\circ$ is neither ingoing nor outgoing to p as t_1 reproduces a non-zero token state and as it is not a one-looping transition of p . Therefore, all ingoing places of t are in P . Thus, we conclude such a transition to be fired as well in ω' when fired in ω . We conclude the transition to be enabled in ω' considering Property (4) of Definition 4.4 if each preceding transition in ω_b has a transition with the same labeling being added to ω' .

The same arguing holds in Case (2) for all ingoing places of a transition $t' \in T_\circ$. Nevertheless, we cannot add t' directly to ω' as it is replaced in SN' and as a transition with the same label has other ingoing places than p . If we consider ω_b to only contain transitions in T_\circ , i.e., $\omega_c = \omega_b \upharpoonright_{T_\circ}$, we identify this firing sequence to be considered for the place projection according to Definition 4.3. Therefore, we conclude that for each t' a transition with the same label exists and is enabled in SN' . Further, according to Definition 4.4, we preserve exiting transitions of p being correspondingly enabled in SN' as we connect outgoing non-looping transitions with the places containing a token in the final marking of the replacing subnet. Similarly, according to Definition 4.4, we preserve the transitions corresponding to the one-looping transitions to be enabled accordingly as we connect the places containing a token in the initial marking of the replacing subnet as outgoing places to all ingoing transitions of p .

We can continue our argument on the alternating subsequences of all-zero and all-one token sequences for p following. Considering those arguments, we conclude for each ω replaying σ on SN that a corresponding ω' on SN' also replaying σ exists. Thus, we conclude fitness being preserved. \square

Theorem 4.2 (Preservation of Precision) *Let L be an event log and let $SN = (N, M_i, M_f)$ with $N = (P, T, F, A, l)$ be an accepting Petri net on which each trace $\sigma \in L$ has a unique valid firing sequence ω . Further, let $p \in P$ be a safe place that has a set of uniquely labeled non-silent one-looping transitions $T_\circ = \bullet p \cap p \bullet \neq \emptyset$ with $\forall_{t_1, t_2 \in T_\circ} : l(t_1) \neq \tau \wedge l(t_1) = l(t_2) \Rightarrow t_1 = t_2$ that does not hold a token in the initial or final marking, i.e., $M_i(p) = M_f(p) = 0$. Let $SN' = (N', M'_i, M'_f)$*

Table 2

Overview of the filtered real-life logs used for evaluation.

Log	Activities	Trace Variants	Source
Road Traffic Fines Management (with appeal only) [RTFM+]	11	189	[17]
Road Traffic Fines Management (without appeal only) [RTFM-]	6	42	[17]
Sepsis (Filtered) [Sepsis]	9	803	[18]
BPI Challenge Log 2017 - Offer Log [BPI17]	8	16	[19]

with $N' = (P', T', F', A', l')$ be the accepting Petri net obtained by projection discovery. For such an accepting Petri net SN' , it holds that any traces σ that is replayable in SN' is also replayable by SN' . This implies, that the proposed method does not increase the behavior of the model, i.e., preserves precision.

Proof. We consider the set of places P_a that only occur in SN' , i.e., $P_a = P' \setminus P$. According to Definition 4.3, we can conclude that it is possible to merge all places P_a into a single place p_m resulting in an accepting Petri net SN_m , i.e., $\bullet p_m = \bigcup_{p' \in P_a} \bullet p' \wedge p_m \bullet = \bigcup_{p' \in P_a} p' \bullet$ holds. Such a modification does not reduce behavior. Considering Definition 4.3, Definition 4.4 and the qualities of the eST-Miner that we apply to the place projection logs, we can directly conclude such an accepting Petri net SN_m to be equal to SN , i.e., $SN_m = SN$ holds. Thus, every trace replayable on SN' is replayable on SN and therefore precision preserved. \square

In the following, we apply our framework to real-life event logs to evaluate its capacities.

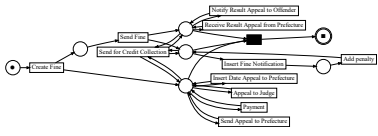
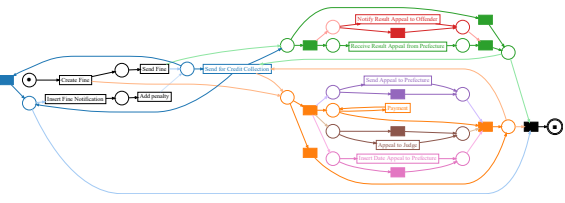
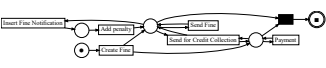
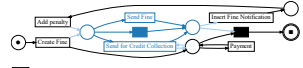
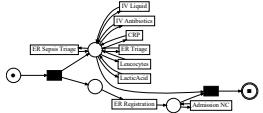
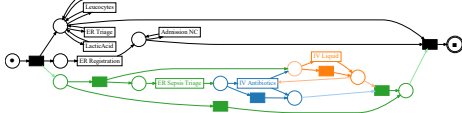
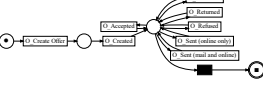

5. Application to Real-Life Event Logs

In this section, we apply the framework presented in Figure 2 to one artificial event log and three real-life event logs. First, there is the 'Road Traffic Fine Management' log [17] which contains the log of an information system managing road traffic fines. In order to find more meaningful Petri nets, we split the log into two logs RTFM+ and RTFM-, based on the presence of the appeal attribute of a trace. The second log used is the 'Sepsis' log [18] describing the pathway of a patient with a sepsis through a hospital. We filtered this log for the nine most frequent activities. The third log is a subset of the BPI Challenge log of 2017 [19] which describes a loan application process of a Dutch financial institution that is filtered for all accepted application offers. We provide an overview of the main properties of the logs in Table 2.

For each of the four logs, we give accepting Petri nets discovered by the eST-Miner and discovered by our framework each. Comparing both the input and the result of our framework, we want to show that precision improves while fitness is preserved. However, existing automated precision metrics fail to show the improvement of precision numerically as they are negatively impacted in our framework when silent transitions are added. Therefore, we show that the application of our framework is reducing the possible behavior. The resulting accepting Petri

Table 3

Accepting Petri nets discovered by the eST-Miner and our framework on the evaluation logs. The models can be accessed individually online. (Click here: [Link to GitLab instance](#))

Log	eST-Model	Projection Model
RTFM+		
RTFM-		
Sepsis		
BPI17		

nets are shown in Table 3. The different colors in the accepting Petri nets discovered by our framework each represent a projection place replacement.

Our framework is implemented in ProM as the so-called ProjectionMiner plugin. For the application of the eST-Miner within our framework, we limited the maximal search depth of the candidate tree to a depth of six as experience has shown that most interesting places are covered already even though we do not traverse the full candidate tree. This limits each place discovered to contain in sum at most six arcs ingoing and outgoing. In our evaluation, we applied the full framework except for the k -loop merge. As there is currently no heuristic implemented for the partitioning of one-looping transitions, it is applied manually. Nevertheless, the framework is able to discover the models on the RTFM-, Sepsis and BPI17 event log without user guidance.

As a result, for all accepting Petri nets discovered by our framework that are shown in Table 3, we can conclude their one-looping transitions to be meaningful as they model activities that appear repeatedly in traces of the original log. Further, our evaluation shows that our framework extends the eST-Miner to find more meaningful constructs than flower models. In the investigated examples, precision is improved for all inputs while fitness is preserved. In conclusion, there are real-life event logs and artificial logs that our framework can be applied to and where our framework extends the eST-Miner to achieve higher precision by adequately modeling optional behavior without hampering its ability to discover complex control-flow structures or reducing fitness.

6. Conclusion and Future Work

In this paper, we introduced a framework which uses place projection techniques to replace imprecise loop-structures in eST-Miner models, stemming from the algorithm's inability to discover silent transitions, with more precise subnets suitably modeling optional behavior. In addition to the projection and replacement methods we have extended the approach to loops of arbitrary length and considered a partitioning technique for one-looping transitions. As proven, the framework guarantees to preserve fitness and precision. As expected, for our investigated example logs we identify precision to be improved. In particular, our application to real-life event logs has shown that our framework is capable of finding accepting Petri nets containing silent transitions with the eST-Miner that are more precise than those discovered using the standard eST-Miner.

We conclude our framework to be a promising step towards improving discovery algorithms unable to return silent transitions, e.g. many region-based approaches, and towards the enhancement of process models containing flower structures. Nevertheless, further research has to be done. In [9], we introduced a first approach for the non-trivial problem of projecting on unsafe places. Unfortunately, such projections are non-unique and heuristics to choose the best are yet to be developed. Further research on models with duplicate transition labels is of interest, as they may have more than one valid firing sequence for a trace and thus also may have a non-unique place projection log. As indicated before, tailored approaches for the partitioning of on one-looping transitions are missing and may further improve results. Finally, our current approach is limited to perfectly fitting logs and an extension to also cover non-perfectly fitting logs is worthwhile.

Any of these ideas can be taken as starting point for further research. Nevertheless, the concept of place projection is promising as it is not limited to eST-Miner models only and as it thus extends our view on process discovery and process enhancement. Moreover, our introduced techniques extend the general toolset of incremental process discovery. Notably, it already significantly extends the capabilities of the eST-Miner to make it more applicable to real-life logs.

Acknowledgments

SPONSORED BY THE



Federal Ministry
of Education
and Research

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research (grant no. 1191945). The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project Bridging AI (grant no. 16DHBKI023).

References

- [1] L. L. Mannel, W. M. P. van der Aalst, Finding complex process-structures by exploiting the token-game, in: S. Donatelli, S. Haar (Eds.), *PETRI NETS 2019, Proceedings*, volume 11522 of *LNCS*, Springer, 2019, pp. 258–278.
- [2] E. Kindler, V. A. Rubin, W. Schäfer, Incremental workflow mining based on document versioning information, in: M. Li, B. W. Boehm, L. J. Osterweil (Eds.), *Unifying the Software Process Spectrum, SPW 2005, Revised Selected Papers*, volume 3840 of *LNCS*, Springer, 2005, pp. 287–301.
- [3] D. Fahland, W. M. P. van der Aalst, Model repair - aligning process models to reality, *Inf. Syst.* 47 (2015) 220–243.
- [4] A. A. Kalenkova, J. Carmona, A. Polyvyanyy, M. L. Rosa, Automated repair of process models with non-local constraints using state-based region theory, *Fundam. Informaticae* 183 (2021) 293–317.
- [5] É. Badouel, L. Bernardinello, P. Darondeau, *Petri Net Synthesis*, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2015.
- [6] R. Bergenthum, J. Desel, R. Lorenz, S. Mauser, Process mining based on regions of languages, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), *BPM 2007, Proceedings*, volume 4714 of *LNCS*, Springer, 2007, pp. 375–383.
- [7] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, A. Serebrenik, Process discovery using integer linear programming, *Fundam. Informaticae* 94 (2009) 387–412.
- [8] J. Carmona, J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, A symbolic algorithm for the synthesis of bounded petri nets, in: K. M. van Hee, R. Valk (Eds.), *PETRI NETS 2008, Proceedings*, volume 5062 of *LNCS*, Springer, 2008, pp. 92–111.
- [9] C. Rennert, *Improving the eST-Miner Models using Non-Unique Transition Labels*, Master's thesis, RWTH Aachen University, Chair of Process and Data Science, Templergraben 55, 52074 Aachen, Germany, 2022.
- [10] L. L. Mannel, W. M. P. van der Aalst, Discovering process models with long-term dependencies while providing guarantees and handling infrequent behavior, in: L. Bernardinello, L. Petrucci (Eds.), *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*, volume 13288 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 303–324.
- [11] F. García-Vallés, J. M. Colom, Implicit places in net systems, in: *PNPM 1999, Proceedings*, IEEE Computer Society, 1999, pp. 104–113.
- [12] B. Berthomieu, D. L. Botlan, S. Dal-Zilio, Petri net reductions for counting markings, *CoRR* abs/1807.02973 (2018).
- [13] J. M. Colom, M. S. Suárez, Improving the linearly based characterization of P/T nets, in: G. Rozenberg (Ed.), *PETRI NETS 1989, Proceedings*, volume 483 of *LNCS*, Springer, 1989, pp. 113–145.
- [14] L. L. Mannel, R. Bergenthum, W. M. P. van der Aalst, Removing implicit places using regions for process discovery, in: W. M. P. van der Aalst, R. Bergenthum, J. Carmona (Eds.), *ATAED 2020, Satellite event of PETRI NETS 2020*, volume 2625 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 20–32.
- [15] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process

models from event logs - A constructive approach, in: J. M. Colom, J. Desel (Eds.), PETRI NETS 2013, Proceedings, volume 7927 of LNCS, Springer, 2013, pp. 311–329.

- [16] T. Murata, Petri nets: Properties, analysis and applications, Proc. IEEE 77 (1989) 541–580.
- [17] M. de Leoni, F. Mannhardt, Road Traffic Fine Management Process (2015).
- [18] F. Mannhardt, Sepsis Cases - Event Log (2016).
- [19] B. van Dongen, BPI Challenge 2017 - Offer log (2021).