




# Enhancing the Applicability of the eST-Miner: Efficient Precision-Guided Implicit Place Avoidance

1<sup>st</sup> Groß, Felix C.   
Process and Data Science (PADS),  
RWTH Aachen University  
Aachen, Germany  
felix.carl.gross@rwth-aachen.de

2<sup>nd</sup> Mannel, Lisa L.   
Process and Data Science (PADS),  
RWTH Aachen University  
Aachen, Germany  
mannel@pads.rwth-aachen.de

3<sup>rd</sup> van der Aalst, Wil M. P.   
Process and Data Science (PADS),  
RWTH Aachen University  
Aachen, Germany  
wvdaalst@pads.rwth-aachen.de

**Abstract**—In process discovery, we aim to find a model that describes the underlying process best, given an event log. The eST-Miner is a discovery technique inspired by language-based regions. It discovers precise Petri nets containing exactly one transition for each activity in the event log by efficiently traversing the space of all possible places. It then expands the model iteratively with places. The final model consists of the maximal set of places considered fitting with respect to a user-definable fraction of the behavior in the log evaluated by token-based replay. Therefore, the eST-Miner can derive complex control-flow structures that other approaches fail to discover and handle noise and infrequent behavior. Although the eST-Miner discovers high-quality models, its feasibility has previously been limited by its runtime: When naively inserting places, the discovered models contain many implicit places, i.e., places whose removal does not change its language and is time-consuming. Therefore, we propose an efficient strategy that avoids adding implicit places by exploiting log information while including non-fitting log traces. We extend the discovery with information on the precision of the (expanding) model based on escaping edges. With our extensions, the eST-Miner becomes a competitive choice among other process discovery techniques. We demonstrate the effectiveness of our heuristics using the eST-Miner as an exemplary use case and run various experiments on real-life and artificial event logs.

**Index Terms**—Process Discovery, Petri nets, eST-Miner, Implicit Places, ETC-Precision

## I. INTRODUCTION AND RELATED WORK

More and more organizations support the execution of their processes using information systems that capture behavior in so-called *event logs*. For each event in the log, amongst other attributes, we store a name identifying the executed step (activity name), a case identifier that is necessary to distinguish between different executions of the process (case ID), and the time the event was observed (timestamp). Various algorithms can utilize such data. Here, we focus on *process discovery*. We aim to construct a Petri net that reflects the behavior of the process that generated the log.

For several reasons, finding a process model for a given event log is non-trivial. Too few events may have been recorded to discover some of the structure of the process correctly, or the log may contain infrequent behavior that is different from the process's nature. An optimal process

discovery technique should discard such noise while still capturing the critical behavior of the process. It should discover process models that can replay the behavior in the event log (*fitness*) while remaining uncomplicated enough to be understood by a human (*simplicity*). The model should generalize the example behavior (*generalization*) without allowing for unrelated behavior (*precision*). It may be impossible to satisfy all quality dimensions. Different process discovery techniques favor certain quality aspects while neglecting others. Hence, the resulting models may vary greatly depending on the method used.

Many discovery algorithms abstract from the complete information in the log or generate places heuristically to decrease the computation time and complexity of the discovered models. While this is desirable in many applied settings, the resulting models often are underfitting and fail to discover complex control-flow structures, such as long-term dependencies. Examples are the Alpha Miner [1], Inductive Miner [2], genetic algorithms [3], and Heuristic Miner [4]. The Split Miner [5] combines several heuristics on directly-follows relations of the event log to discover more precise process models. In contrast to these approaches, algorithms based on region theory [6, 7] discover models whose behavior is a minimal superset of the behavior captured in the input event log. On the downside, these approaches are time-consuming, cannot handle noise, and tend to produce overly complex and overfitting models.

In this work, we extend the *eST-Miner* [8], a process discovery technique inspired by language-based regions. It can derive complex structures (e.g., long-term dependencies) many other algorithms fail to discover. Additionally, it can handle infrequent behavior and noise. It efficiently traverses the finite search space of all possible places resulting from combining all uniquely labeled transitions. By naively adding all (fractionally) fitting places, the final Petri net contains many *implicit places*, i.e., places whose removal does not change its language and, therefore making the model overly complex. Different approaches exist to remove implicit places (see [9, 10]). Several extensions to the eST-Miner have been proposed to improve its runtime. However, although it discovers *high-quality models*, i.e., models that score well with respect to selected fitness and precision metrics, its feasibility on real-life data has so far been limited by the time-consuming Implicit Place Removal (IPR) step.

The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project BridgingAI. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.



This work addresses this issue by suggesting a precision-based heuristic to avoid adding implicit places, the so-called *ETC-based composer*. In contrast to other techniques, we exploit log information and can handle noise and infrequent behavior. We show how to incrementally update the precision of the expanding model efficiently during the discovery based on escaping edges [11]. This enables us to only add places that increase its precision, i.e., non-implicit places. Furthermore, we can identify places whose removal does not decrease precision, i.e., implicit places. We illustrate the effects of the proposed heuristics through experiments on real-life and artificial event logs. We show that we can *significantly decrease the computation time* of the eST-Miner and its Delta variant [12] while keeping the discovered models' fitness and precision high. Our experiments show that when using our heuristics, we can now avoid implicit places up to 92% faster compared to all other available IPR techniques. Thus, for the first time, the eST-Miner becomes a competitive choice for discovering process models on mid-sized real-life event logs (for a size reference, see Fig. 4).

In Section II, we introduce basic definitions and notation before briefly presenting the eST-Miner in Section III and our extensions to it in Section IV. In Section V, we perform several experiments whose results we discuss in Section VI. Finally, we conclude the paper in Section VII.

## II. PRELIMINARIES

A set, e.g.,  $\{a, b\}$ , contains every element at most once, while multisets allow containing elements more than once. We write multisets as  $[a, a] = [a^2]$ . For a set  $X$ , we denote its powerset by  $\mathbb{P}(X)$  and the set of all multisets over  $X$  by  $\mathbb{M}(X)$ . Contrary to sets and multisets, the order of the elements matters in sequences. We denote sequences over some set  $X$  as  $\langle a, b, c \rangle$ .  $X^*$  denotes the set of all possible sequences over  $X$ , including the empty sequence  $\langle \rangle$ . We denote the size of a set, multiset, or sequence  $X$ , i.e., the number of elements in  $X$ , by  $|X|$ . For a sequence  $\sigma$ , we denote  $\#_\sigma(a)$  as the frequency of an element  $a$  in  $\sigma$ . We denote the prefix of length  $k$  of  $\sigma$ , with  $0 \leq k \leq |\sigma|$ , by  $pref^k(\sigma)$ . We denote the set of all (unique) prefixes of a (multi)set of sequences,  $X$ , by  $Pref(X)$ .

When defining event logs, we focus on discovering control-flow and abstract from concrete timestamps and optional attributes. We require each trace to begin with the designated start activity  $\blacktriangleright$  and end with a designated end activity  $\blacksquare$ . Note that we can transform any event log accordingly.

**Definition 1** (Activity, Trace, Log). *Let  $\mathcal{A}$  be the universe of all activities, let  $\blacktriangleright \in \mathcal{A}$  be the designated start activity and  $\blacksquare \in \mathcal{A}$  the designated end activity. A trace  $\sigma$  is a sequence  $\langle \blacktriangleright, a_1, \dots, a_n, \blacksquare \rangle \in \mathcal{A}^*$ , such that  $a_i \in \mathcal{A} \setminus \{\blacktriangleright, \blacksquare\}$  for all  $1 \leq i \leq n$ . Let  $\mathcal{T}$  be the universe of such traces. A log  $L \in \mathbb{M}(\mathcal{T})$  is a multiset of traces.*

Given a set of activities, its characteristic function evaluates to 1 for every element in the set and 0 otherwise.

**Definition 2** (Characteristic Function). *For a set of activities  $A \subseteq \mathcal{A}$ , we define the characteristic function*

$$\mathbb{I} : \mathbb{P}(\mathcal{A}) \times \mathcal{A}, \mathbb{I}(A, a) = \begin{cases} 1 & a \in A \\ 0 & \text{otherwise} \end{cases}.$$

There are many different representations of models. This work focuses on Petri nets. The eST-Miner discovers Petri nets without duplicate labels and silent transitions. We assume that firing a transition corresponds to the execution of a (distinct) activity and vice versa. Therefore, the set of transitions of the discovered model and the set of all activities contained in the given event log coincide, and we will be using these terms interchangeably. We denote Petri nets that we discover given an event log  $L$  over the set of activities  $A \subseteq \mathcal{A}$  by  $N = (A, P, p_s, p_t)$ . We sometimes refer to  $N$  by its set of intermediate places  $P$  only. We denote a place  $p = (I|O)$  by its set of incoming transitions  $\bullet p = I$  and its set of outgoing transitions  $p \bullet = O$ . Additionally, given an activity  $a \in A$ , we refer to its set of incoming places by  $\bullet_P a$  and its set of outgoing places by  $a \bullet_P$ . If it is clear which Petri net we refer to, we only write  $\bullet a$  and  $a \bullet$ , respectively. We require Petri nets always to have a dedicated source place  $p_s = (\emptyset | \blacktriangleright)$  and target place  $p_t = (\blacksquare | \emptyset)$ .

**Definition 3** (Petri Net). *Let  $A \subseteq \mathcal{A}$ , with  $\blacktriangleright, \blacksquare \in A$  be a finite set of activities. We denote a Petri net by  $N = (A, P, p_s, p_t)$ , where  $P \subseteq \{(I|O) \mid I \subseteq A \setminus \{\blacksquare\} \wedge I \neq \emptyset \wedge O \subseteq A \setminus \{\blacktriangleright\} \wedge O \neq \emptyset\}$  is the set of intermediate places,  $p_s = (\emptyset | \blacktriangleright)$  the source, and  $p_t = (\blacksquare | \emptyset)$  the target place.*

We determine the state of a Petri net by the distribution of tokens over places, a *marking*. We consider  $m_i = [p_s]$  to be the initial marking of a Petri net corresponding to the state before the execution of the process. During its execution, a transition  $a \in A$  is enabled if at least one token is in each place  $\bullet a$ . Then, it can fire, consuming a token from each place in  $\bullet a$  and producing one in each place in  $a \bullet$ . After executing a valid firing sequence, we obtain the final marking  $m_f = [p_t]$ . We define the behavior of a Petri net as the execution of traces such that all (intermediate) places are empty at the end of the trace and never have a negative number of tokens.

There are different metrics to measure the quality of a process model given an event log. The eST-Miner considers place-wise fitness by playing the token game: We call an intermediate place  $p \in P$  *fitting*, if a user-definable fraction of traces in the log can be perfectly replayed on  $p$ , i.e., without causing missing or remaining tokens during or after replay. Our approach to extending the eST-Miner is based on ETC-precision [11], a commonly used precision metric. It compares the state spaces of the log and model. An exhaustive exploration of the model's state space may be infeasible due to the state space explosion problem. Thus, we restrict our consideration of the model's behavior to the one captured in the log. Therefore, we look at the *prefix automaton* of a log  $L$ , a Deterministic Finite Automaton (DFA) with one state for every (unique) prefix of all of its traces derived using the sequence, past, and infinite log abstraction as presented in [13].

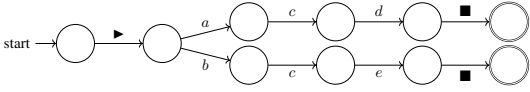


Fig. 1: Prefix automaton for the log  $L = [(a, c, d)^{13}, (b, c, e)^{42}]$  annotated with frequencies.

**Definition 4** (Prefix Automaton). Let  $L$  be an event log over the set of activities  $A \subseteq \mathcal{A}$ . The prefix automaton of  $L$  is the DFA  $PA_L = (S, A, t, s_i, F)$ , with  $S = Pref(L)$  the (finite) set of states,  $A$  the (finite) alphabet,  $t : S \times A \rightarrow S$  a transition function,  $s_i = \langle \rangle$  the initial state, and  $F = \{\sigma \in L\}$  the set of final states. For a prefix  $\sigma' \in S \setminus L$  of a trace  $\sigma \in L$ , such that  $\sigma' \circ \langle a \rangle \in S$  for an  $a \in A$ , we have  $t(\sigma', a) = \sigma' \circ \langle a \rangle$ . Thus,  $PA_L$ 's language equals  $L$ .

Figure 1 shows an example of a prefix automaton. To compute precision, the log is replayed concurrently on the intermediate model and the prefix automaton, comparing and searching for discrepancies between the behavior allowed in  $N$  and the observed behavior in  $L$ . Therefore, for each state  $s$  of  $PA_L$  corresponding to a prefix  $\sigma'$  of a trace  $\sigma \in L$ , we look at the set of enabled transitions in the model  $N$  after executing  $\sigma'$ . These activities are *allowed* in  $s$ . Then, we look at the activities executed after the prefix  $\sigma'$  of some (possibly different) trace in  $L$ , the *reflected* activities in  $s$ . Activities are *escaping* in  $s$  if their execution is allowed but not reflected in  $s$ .

### III. EST-MINER

Over the past years, several variants of the eST-Miner have been proposed. This section gives an overview of the core idea of the eST-Miner [8] and other concepts relevant to the context of this paper. For further details, we refer the reader to the respective papers.

a) *Efficient State Space Traversal*: For an input event log  $L$  over the set of activities  $A \subseteq \mathcal{A}$  and a noise threshold parameter  $\tau \in [0, 1]$ , the eST-Miner returns a Petri net  $N = (A, P, p_s, p_t)$ . Inspired by language-based regions, initially, the Petri net contains exactly one transition for each activity in  $A$  and a source and target place, i.e.,  $P = \emptyset$ . In the *searching phase* of the algorithm, we traverse the finite space of all possible candidate places, only storing those that are fitting with respect to at least a fraction of  $\tau$  traces.

Since evaluating all possible places in a brute-force approach becomes infeasible with an increasing number of activities in the log, the eST-Miner uses the results of the place-wise fitness evaluation to skip large parts of the search space containing non-fitting places. In [8], it is proposed to organize the candidate places in a set of trees. This data structure allows only keeping a limited number of places in memory at any time and traversing the search-space efficiently by pruning the so-called *candidate tree* (CT). We traverse the CT using breadth-first search (BFS). Sometimes, we limit the CT's depth to  $d \in \mathbb{N}$ . Then, all places in consideration have at most  $d$  connecting, i.e., incoming and outgoing activities. Excluding complex places located at deeper levels of the

candidate tree improves runtime significantly and achieves a trade-off across all four quality dimensions. While some complex process behavior may need such places to be expressed precisely, inserting them into the Petri net is usually devastating to readability. In practical applications, simpler places can usually approximate their constraints sufficiently.

The eST-Miner applies noise filtering on a place-wise level: When evaluating the fitness of a place, certain traces are temporarily ignored. Furthermore, varying  $\tau$  gives us a tool to discover models with different quality aspects.

Often, when combining the places evaluated individually to be fitting with respect to some noise threshold during the searching phase, the final Petri net may contain places that contradict each other, leading to deadlocks. Thus, the discovered models sometimes have low fitness. To tackle this issue, the *Delta variant* [12] of the eST-Miner guarantees that at least a fraction of  $\tau$  traces of the input event log  $L$  is replayable by the discovered Petri net. It only adds places whose inclusion does not decrease the fraction of traces in  $L$  perfectly fitting to the intermediate model by more than  $\delta \in [0, 1]$ .

b) *Implicit Place Removal*: When naively adding all possible fitting places to the model,  $P$  may consist of thousands of places, most of which are implicit. To improve the model's simplicity, depending on the choice of  $\tau$ , the eST-Miner has different approaches at its disposal: We can *remove implicit places using regions* (region-IPR) if  $\tau = 1$ , or resort to *solving an ILP* (LP-IPR) regardless of the value of  $\tau$ .

If  $\tau = 1$ , we only include places that are perfectly fitting with respect to  $L$ . Hence, we guarantee that  $L$  perfectly fits  $N$ . Then, as proposed in [9], the so-called *marking histories* of two places can be compared to determine whether one of them is (going to become) implicit in the final model. For a candidate place  $p$ , its marking history on some sequence  $\sigma$  represents the number of tokens in  $p$  when firing its connected transitions only during the replay of  $\sigma$ .

**Definition 5** (Marking History).

For a place  $p = (I|O)$ , with  $I \subseteq \mathcal{A}$  and  $O \subseteq \mathcal{A}$ , we define its marking history on a trace  $\sigma = \langle a_1, \dots, a_n \rangle \in L$  as  $H_\sigma(p) = \langle h_{pref^1(\sigma)}(p), \dots, h_{pref^n(\sigma)}(p) \rangle \in \mathbb{Z}^*$ , with

$$h_{pref^k(\sigma)}(p) = \sum_{i \in I} \#_{pref^k(\sigma)}(i) - \sum_{o \in O} \#_{pref^k(\sigma)}(o)$$

for all  $1 \leq k \leq n$ .

Let  $\sigma_1, \dots, \sigma_m$  be the trace variants of the event log  $L$  in an arbitrary but fixed order. Then, we define the marking history of  $p$  on  $L$  to be the concatenation of the marking histories of all its trace variants:  $H_L(p) = H_{\sigma_1}(p) \circ \dots \circ H_{\sigma_m}(p)$ .

A place that always contains more tokens during replay than others can be identified as implicit since removing it does not reduce the constraints imposed on its connected transitions. Implicit places can be avoided during the *searching phase* of the algorithm (region-CIPR): Before adding a fitting place, its marking history is compared to those of all other places in  $P$ . Alternatively, the marking histories of all fitting places are compared in post-processing (region-PPIPR).

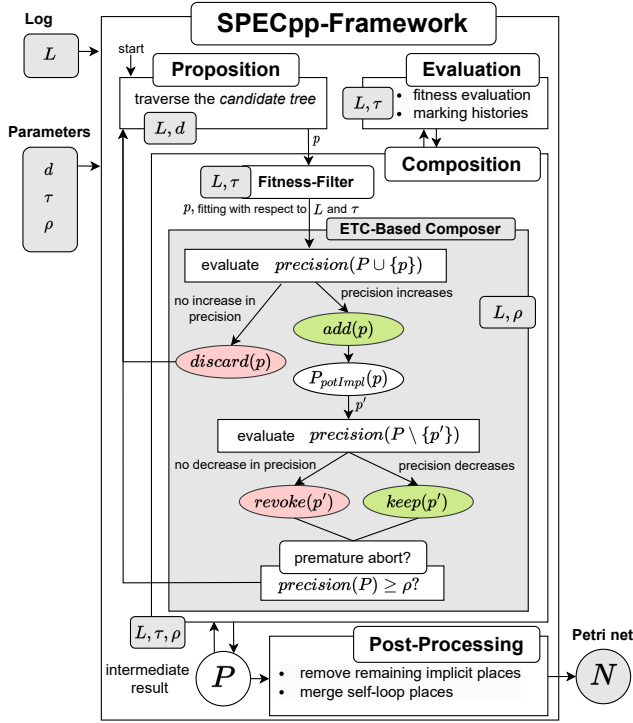


Fig. 2: Overview of the SPECpp-framework [14] including input, output, and parameter use.

Region-IPR, as defined in [9], requires all log traces to fit the model perfectly and can, therefore, not be used for  $\tau < 1$ . We then solve an ILP as suggested originally in [10] instead, identifying and removing all implicit places of the model based on its structure only (LP-IPR). We do not incorporate any log information in the process. In [8], solving the ILP once as an additional post-processing step (ILP-PPIPR) is proposed. Alternatively, similar to the replay-based approach, we can use the LP-based approach to remove implicit places “concurrently” (LP-CIPR): Each time we add a place to  $P$ , we solve the ILP and remove any places (newly) made implicit.

c) *Implementation*: We have integrated our approach into the *SPECpp-framework* [14]. Figure 2 gives an overview of the algorithmic framework. It aims to give developers easy access to working on the eST-Miner. Although intended to run independently, the SPECpp-framework also provides an interactive ProM-plugin [15]. Apart from being able to supervise the discovery of process models using the eST-Miner and all its extensions, SPECpp iterates over the search space in so-called *PEC-cycles*: First, we traverse the next candidate place  $p$  of the CT (*proposition*). Next, we evaluate the fitness of  $p$ . In case  $p$  is fitting, we forward it to our new subroutine, the ETC-based composer, which avoids implicit places using marking histories (*evaluation/composition*). In post-processing, we remove any remaining implicit places and merge places with the same non-looping connecting activities.

#### IV. ETC-BASED COMPOSER

This section introduces a heuristic extension to the eST-Miner to avoid adding implicit places. In contrast to existing

IPR methods, it heavily relies on log information and can handle Petri nets that cannot perfectly replay the input event log. In each iteration of the discovery, we traverse the next place of the CT,  $p$ , and evaluate its fitness. In case  $p$  is fitting with respect to the input event log  $L$  over the set of activities  $A \subseteq \mathcal{A}$  and  $\tau$ , we do not simply add it to the intermediate model  $N = (A, P, p_s, p_t)$  but instead propose it to a new subroutine called the *ETC-based composer*. It evaluates the impact adding  $p$  has on the precision of  $N$ . We add  $p$  if it increases precision. Otherwise, we consider it implicit and discard it. If we add  $p$ , we revoke places we recognize as implicit after inserting  $p$ , i.e., places  $p' \in P$  whose removal does not decrease precision. Figure 2 gives an overview of this approach, indicating inputs, outputs, and parameter use.

Note that recalculating ETC-precision naively in each iteration is not feasible. Additionally, for  $\tau < 1$ , we cannot guarantee that all traces in  $L$  are perfectly replayable on  $N$  at any time. We could discard all non-fitting log traces or compute alignments as described in [11, 16]. However, we want to be able to handle noise and infrequent behavior while keeping the computation time reasonable. Thus, we show how to efficiently approximate the precision of the expanding model  $N$  by incorporating all log information.

a) *Initialization*: First, we construct the prefix automaton of  $L$ ,  $PA_L$ . When calculating precision, we reuse the concept of marking histories introduced in [9] to define the sets of allowed, reflected, and escaping activities for each state  $s$  of  $PA_L$ . If all incoming places of an activity  $a$  contain at least one token after replaying the prefix corresponding to  $s$ ,  $a$  is allowed in  $s$ . Then, we can check whether  $a$  is reflected by looking at the (outgoing) transitions of  $s$  in  $PA_L$ . If  $a$  is not reflected in the log,  $a$  is escaping in  $s$ .

**Definition 6** (Indicator Sets). *Let  $L$  be an event log over the set of activities  $A \subseteq \mathcal{A}$ . Let  $PA_L = (S, A, t, s_i, F)$  be the prefix automaton of  $L$ . Let  $N = (A, P, p_s, p_t)$  be a Petri net. For a state  $s \in S$  corresponding to a prefix  $\sigma'$  of a trace  $\sigma \in L$ , we define  $R(s) = \{a \in A \mid s \circ \langle a \rangle \in S\}$  as the set of activities that are reflected in  $L$  in  $s$ ,  $A_P(s) = \{a \in A \mid \forall p \in \bullet_{Pa} : h_s(p) \geq 1\}$  as the set of activities that are allowed in  $N$  in  $s$ , i.e., after executing  $\sigma'$ , and  $E_P(s) = A_P(s) \setminus R(s)$  as the set of activities that are escaping in  $s$ .*

To (re)compute precision, we use so-called *activity mappings*.

**Definition 7** (Activity Mappings). *Let  $L$  be the input event log over the set of activities  $A \subseteq \mathcal{A}$ . The activity mapping  $M_A : A \mapsto \mathbb{N}_0$  defines the (current) number of times an activity  $a \in A$  is allowed and  $M_E : A \mapsto \mathbb{N}_0$  the (current) number of times  $a$  is escaping when replaying  $L$ .*

Initially, at the start of the discovery, we have  $P = \emptyset$ . At the beginning of each trace,  $PA_L$  is in its initial state, and  $N$  is in marking  $m_i$ . All activities are allowed in  $N$ , but only the starting activity  $\blacktriangleright$  is reflected in this state. In all other states of  $L$ , all activities except for  $\blacktriangleright$  are allowed. We can check whether they are reflected by looking at  $PA_L$ .

**Definition 8** (Initial Activity Mappings).

Let  $L = [\sigma_1^{f_1}, \dots, \sigma_n^{f_n}]$  be the input event log over the set of activities  $A \subseteq \mathcal{A}$ . To obtain the initial activity mappings, we set  $M_A(\blacktriangleright) = \sum_{i=1}^n f_i$  and  $M_E(\blacktriangleright) = 0$  for the starting activity, and for all other activities  $a \in A \setminus \{\blacktriangleright\}$ , we set

$$M_A(a) = \sum_{i=1}^n f_i \cdot |\sigma_i| \text{ and}$$

$$M_E(a) = \sum_{i=1}^n f_i \cdot \sum_{j=0}^{|\sigma_i|-1} 1 - \mathbb{I}(R(\text{pref}^j(\sigma_i)), a).$$

**Definition 9** (Precision). Let  $M_A$  and  $M_E$  be the current activity mappings. We define the precision of the current intermediate model  $N = (A, P, p_s, p_t)$  as

$$\text{precision}(P) = 1 - \frac{\sum_{a \in A} M_E(a)}{\sum_{a \in A} M_A(a)}.$$

*b) Recalculating Precision Concurrently:* After the initialization, in each call of the ETC-based composer, we receive a fitting candidate place  $p = (I|O)$  as input. We have also stored  $M_A$  and  $M_E$  resulting from the previous iterations.

Since the place  $p$  models conditions which constrain executing its outgoing activities, by adding  $p$  to  $P$ , only those activities may be allowed less often when replaying  $L$ . Analogously, by removing  $p$  from  $P$ , only activities  $o \in O$  may be allowed more often during replay. Therefore, when adding or removing  $p$ , we only need to reevaluate the entries  $M_A(o)$  and  $M_E(o)$  for every activity  $o \in O$  to recalculate precision.

For an activity  $a \in A$ , we recalculate  $M_A(a)$  and  $M_E(a)$  as follows:

**Definition 10** (Recalculating Precision).

Let  $L = [\sigma_1^{f_1}, \dots, \sigma_n^{f_n}]$  be the input event log over the set of activities  $A \subseteq \mathcal{A}$ ,  $N = (A, P, p_s, p_t)$  the current intermediate model, and  $M_A$  and  $M_E$  the current activity mappings. Let  $p = (I|O)$ , with  $I \subseteq A$  and  $O \subseteq A$ , be a place. Let  $P'$  be the set of places obtained by either adding  $p$ , ( $P' = P \cup \{p\}$ ) or removing  $p$  ( $P' = P \setminus \{p\}$ ). In both cases, we update the entries  $M_A(o)$  and  $M_E(o)$  for all activities  $o \in O$  as follows:

$$M_A(o) = \sum_{i=1}^n f_i \cdot \sum_{j=0}^{|\sigma_i|-1} \mathbb{I}(A_{P'}(\text{pref}^j(\sigma_i)), o) \text{ and}$$

$$M_E(o) = \sum_{i=1}^n f_i \cdot \sum_{j=0}^{|\sigma_i|-1} \mathbb{I}(E_{P'}(\text{pref}^j(\sigma_i)), o).$$

Note that for all other activities  $a \in A \setminus O$ ,  $M_A(a)$  and  $M_E(a)$  remain unchanged.

Thus, we recalculate the precision efficiently regarding runtime and memory: When adding or removing a place  $p = (I|O)$ , we only need to update  $|O|$  entries of each activity mapping. Therefore, for an activity  $o \in O$ , we traverse (parts of) the marking histories of  $|\bullet o|$  places on  $L$ . Note that a marking history of a place  $p$  only depends on its set of

incoming and outgoing activities and the log. Therefore, we can compute a marking history of each candidate place only once when we propose it and keep the ones of all places in  $P$  in memory.

For  $\tau = 1$ , we thereby (re)calculate the same value as returned by ETC-precision [11]. For  $\tau < 1$ , some traces of  $L$  may be unfitting. Then, we provide a reasonable approximation when updating the entries of the activity mappings for an activity  $a \in A$ : By only looking at the marking histories of its incoming places, e.g.,  $H_L(p)$ , with  $p \in \bullet a$ , at any time during the replay of a trace  $\sigma \in L$ , we neglect any (earlier) missing tokens in other places: We assume  $p$ 's input transitions fire when their corresponding activities are executed. Furthermore, we count missing tokens in  $p$  itself negatively.

*c) Place Classification:* Being able to efficiently recalculate the precision of the intermediate model  $P$  when adding or removing a place allows us to avoid implicit places by classifying the place  $p$  proposed to the ETC-based composer as follows:

**Definition 11** (Classification of Proposed Places). Let  $L$  be the input event log over the set of activities  $A \subseteq \mathcal{A}$ , let  $N = (A, P, p_s, p_t)$  be the current intermediate model, and  $M_A$  and  $M_E$  the current activity mappings. Let  $p = (I|O)$ , with  $I \subseteq A$  and  $O \subseteq A$ , be a candidate place proposed to the ETC-based composer. Reevaluating  $\text{precision}(P \cup \{p\})$  according to Definition 10, we get the updated mappings  $M'_A : A \rightarrow \mathbb{N}_0$ , and  $M'_E : A \rightarrow \mathbb{N}_0$ . Then, we have:  $\text{add}(p) = \exists a \in A : M'_E(a) < M_E(a)$  and  $\text{discard}(p)$ , otherwise.

We consider  $p$  implicit if its addition to the intermediate model does not constrain any escaping activities. Otherwise, we add  $p$  by setting  $P := P \cup \{p\}$  and update the activity mappings accordingly. Then, we check all places  $p' \in P \setminus \{p\}$  whose set of outgoing activities  $p' \bullet$  intersects with the one of the newly added place,  $p \bullet$ , for implicitness.

**Definition 12** (Classification of Potentially Implicit Places).

Let  $L$  be the input event log over the set of activities  $A \subseteq \mathcal{A}$ ,  $N = (A, P, p_s, p_t)$  the current intermediate model, and  $M_A$  and  $M_E$  the current activity mappings. Let  $p = (I|O)$ , with  $I \subseteq A$  and  $O \subseteq A$ , be a candidate place proposed to the ETC-based composer with  $\text{add}(p)$ . Then, we look at the set  $P_{\text{potImpl}}(p) = \{(I'|O') \in P \setminus \{p\} \mid O \cap O' \neq \emptyset\}$ . For every  $p' \in P_{\text{potImpl}}(p)$ , reevaluating  $\text{precision}(P \setminus \{p'\})$ , we get  $M'_A$  and  $M'_E$ . Then, we have  $\text{revoke}(p') = \forall a \in A : M'_A(a) = M_A(a) \wedge M'_E(a) = M_E(a)$  and  $\text{keep}(p')$  otherwise.

We recognize  $p'$  as implicit if its removal from the intermediate model does not change its behavior regarding allowed and escaping activities.

Additionally, we can abort the search prematurely once the precision of our current model meets a user-definable threshold  $\rho \in [0, 1]$ . By default, we set  $\rho$  to one. Then, aborting the search prematurely has no impact on the discovered model.

*d) Example:* This section provides a complete example of using the eST-Miner in combination with the ETC-based

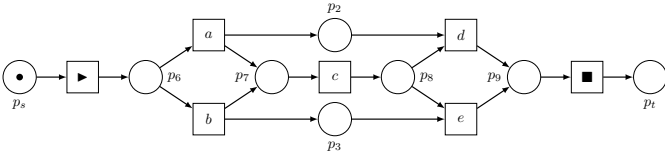


Fig. 3: Final result of example run when given  $L = [\langle \blacktriangleright, a, c, d, \blacksquare \rangle^{13}, \langle \blacktriangleright, b, c, e, \blacksquare \rangle^{42}]$  and  $\tau = 1$  as input.

name	#activities	#traces	#variants	ref.
HD2017	16	4580	226	[18]
RTFM	13	150370	231	[19]
Sepsis	18	1050	846	[20]
Repair	10	1104	77	[21]
Reviewing	16	100	96	[21]
Teleclaims	13	3512	12	[21]

Fig. 4: Event logs used in experiments.

composer to discover a model when given the event log  $L = [\langle \blacktriangleright, a, c, d, \blacksquare \rangle^{13}, \langle \blacktriangleright, b, c, e, \blacksquare \rangle^{42}]$  and  $\tau = \rho = 1$ . In the following, we refer to the place proposed in and the intermediate model after the  $i$ -th call of the ETC-based composer by  $p_i$  and  $P^i$ , respectively.

At first, we have  $P^0 = \emptyset$ . The initial model consists of the source and target places and one transition for each activity in the event log. We build the initial prefix automaton (see Fig. 1) and calculate the initial activity mappings according to Definition 8. Thus, we have  $\text{precision}(P^0) \approx 0.194$ . After the fourth call, we have  $P^4 = \{\langle \blacktriangleright | c \rangle, (a|d), (b|e), (c|\blacksquare)\}$ . Adding  $p_5 = \langle \blacktriangleright | \blacksquare \rangle$  to  $P^4$  does not constrain any escaping activities. Hence, we have  $\text{discard}(p_5)$ . Next, we add  $p_6 = \langle \blacktriangleright | a, b \rangle$  and  $p_7 = (a, b|c)$ . Then, we have to check  $P_{\text{potImpl}}(p_7) = \{\langle \blacktriangleright | c \rangle\}$ . Removing  $p_1 = \langle \blacktriangleright | c \rangle$  does not change the activity mappings. Hence, we have  $\text{revoke}(p_1)$ . When adding  $p_8 = (c|d, e)$ , we have to check the places indicating the long-term dependencies for implicitness. However, removing either of those places results in more allowed/escaping activities. Hence, we have  $\text{keep}(a|d)$  and  $\text{keep}(b|e)$ . Finally, we add  $p_9 = (d, e|\blacksquare)$  and revoke the addition of  $p_4 = (c|\blacksquare)$ . After the ninth call, we reach optimal precision and prematurely abort the search. Figure 3 shows the final model.

## V. EVALUATION

In this section, we evaluate the process models discovered by the eST-Miner using the ETC-based composer and compare its performance to other IPR techniques.

We use the event logs listed in Fig. 4 for our experiments. As real-life event data, we use the following three logs: HD2017 contains the behavior of a ticketing management process of the help desk of a software company. Event data about managing road traffic fines is captured in RTFM. The event log Sepsis contains event data of a hospital. Moreover, we look at the artificial event logs Repair, Reviewing, and Teleclaims used in [17]. We evaluate the discovered models' alignment-based fitness [22] and ETC-precision [11] and quantify their simplicity by their number of arcs. Note that unlike the approximation in precision calculated by the ETC-based composer, the measured ETC-precision only considers

$\tau$	1	0.9	0.8	0.7	0.6	0.5
HD2017	0/20	14/4390	32/5282	40/8125	43/8622	47/8642
RTFM	0/32	3/567	11/846	20/989	10/1352	5/2535
Sepsis	0/29	15/543	15/978	18/1521	33/2987	42/3614
Repair	0/21	5/59	4/63	5/104	7/127	8/252
Reviewing	0/88	0/88	0/88	0/88	0/88	2/681
Teleclaims	0/82	3/103	3/238	3/372	3/414	2/1035

Fig. 5: Number of remaining implicit places (RIP) in the models discovered with the eST-Miner using the ETC-based composer and total numbers of (fractionally) fitting places (TFP). Notation:  $RIP_{TFP}$ .

the traces of the event log perfectly fitting the process model. We compare models by their F1-score, the harmonic mean of their fitness and precision.

a) *Suitability for Implicit Place Removal*: In this subsection, we want to check the ETC-based composer's suitability to avoid implicit places. For this experiment, we vary  $\tau \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ , set  $\rho = 1$ , and limit the depth of the CT to  $d = 4$ . When allowing for greater depth, most test runs using the standard eST-Miner (with LP-PPIPR) do not finish within two hours.

Figure 5 shows the number of remaining implicit places in the models discovered using the ETC-based composer. For  $\tau = 1$ , no implicit places remain. For  $\tau < 1$ , only a few implicit places are left, especially compared to the total number of fitting places: Although there are only comparably few perfectly fitting places for all logs, when lowering  $\tau$  the amount of (fractionally) fitting places increases, especially when increasing  $d$ . For  $d = 4$ , there are up to multiple thousand for the real-life event logs, and for the artificial event logs, up to multiple hundred fractionally fitting places for  $\tau = 0.5$ . The few remaining implicit places can be removed quickly in post-processing by solving an ILP.

Looking at the quality of the models, Fig. 6a and Fig. 6b show that they score comparably regarding fitness and precision to those discovered using another IPR approach (ILP-CIPR). This implies that we do not discard (significant) non-implicit places either. Section VIa focuses on the simplicity of models.

b) *Comparison to Other Implicit Place Removal Techniques*: In this subsection, we compare our approach's performance to other techniques that remove/avoid implicit places of the models discovered by the eST-Miner (LP-CIPR, LP-PPIPR, replay-CIPR, and replay-PPIPR). We use the same experimental setup as in the previous experiment.

For  $\tau = 1$ , the set of fitting places is comparatively small. Thus, we only have to spend little runtime to avoid/remove them and observe that the runtime of all approaches, including ours, is comparable. For  $\tau < 1$ , recall that we cannot reliably use the region-IPR approaches. The standard approach (LP-PPIPR) is much slower than the other approaches: For example, for  $\tau \in \{0.5, 0.6\}$ , none of the test runs on the real-life logs finishes within two hours. LP-CIPR is faster than LP-PPIPR since it keeps the set of places of the intermediate model small. However, using the ETC-based composer, on average, Fig. 6d shows that we can discover models even ten seconds (30%) faster on the real-life logs than when

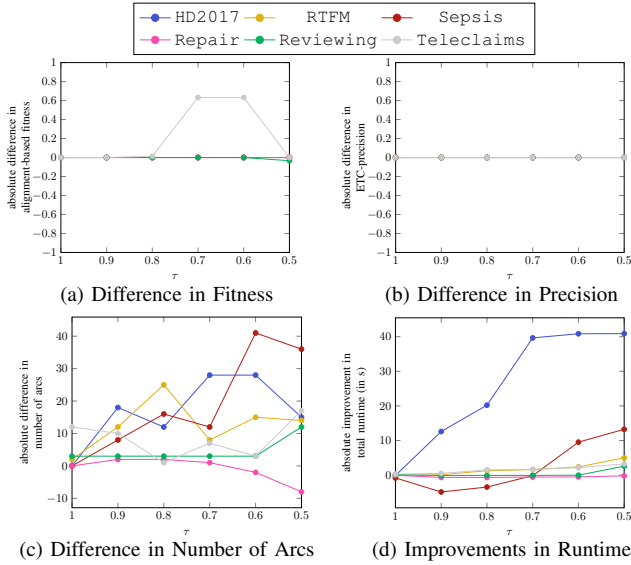


Fig. 6: Difference in Quality of resulting process models and improvements in performance when using the ETC-based composer compared to using LP-CIPR.

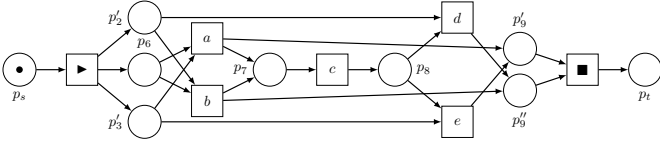


Fig. 7: Overly complex model discovered alternatively to the model in Fig. 3.

using LP-CIPR. Such improvements are highly relevant for a discovery technique’s practical application. The speedup for artificial event logs is only minor, but the runtime is already low there. Crucially, we can abort the search prematurely in some instances, e.g., for the *Teleclams* log and  $\tau \in \{0.5, 0.6, 0.7, 0.8\}$ , resulting in an improvement of two seconds (81%) on average. There are even more significant improvements compared to LP-CIPR, not only in cases where we abort prematurely, when increasing the maximum depth of the CT.

## VI. DISCUSSION

This section discusses the problems of our approach, compares the eST-Miner (with our extensions) to other discovery techniques, and explores its limitations.

*a) Challenges:* Experiments show that, although free of any implicit places (after applying post-processing), models discovered using the ETC-based composer sometimes are overly complex (see Fig. 6c). The reason for that is that the choice of places kept in the final model heavily depends on the proposal order of candidate places: Looking back at the example in Section IV d, if we propose the fitting places in a different order, we may discover the model shown in Fig. 7 which is overly complex compared to the one in Fig. 3 and thus not desirable.

*b) eST-Miner in Process Discovery:* In this subsection, we use the Delta variant of the eST-Miner [12] in combination with the ETC-based composer. It not only proposes fewer

log	E	A	H	I	S
HD2017	<b>0.983</b>	0.354	0.952	0.916	0.962
RTFM	0.966	-	0.974	0.895	<b>0.979</b>
Sepsis	0.731	-	<b>0.829</b>	0.724	-
Repair	0.826	0.310	0.757	0.820	<b>0.848</b>
Reviewing	0.809	0.877	<b>0.999</b>	0.877	0.959
Teleclams	0.962	-	0.976	0.959	<b>1</b>
NPFS	0	0.5	0.444	0	0.391

Fig. 8: Comparison of the best F1-scores of the models discovered by different process discovery techniques: eST-Miner (E), Alpha (A), Heuristic (H), Inductive (I), and Split Miner (S). The bottom row (NPFS) shows the fraction of models we discovered (per technique) that do not have any possible firing sequence. Thus, not all discovered models satisfy the requirements to compute alignment-based fitness (within 2 hours).

places and therefore improves the simplicity of the discovered models, but it also guarantees that they can perfectly replay at least a fraction of  $\tau$  traces. We perform a grid search across the parameters  $\tau$  and  $\delta$  and limit  $d = 4$ . We select the *best model* by choosing the simplest one among those that have the highest F1-scores. We then compare its quality to the best models discovered by varying the parameters of the Alpha, Heuristic, Inductive, and Split Miner. We exclude algorithms based on region theory from this comparison because they are too time-consuming. Note that a detailed comparison of the models’ quality is beyond the scope of this work, we focus on improving performance.

When using the ETC-based composer, the discovery takes at most 13 seconds for the *Sepsis* event log. For the artificial logs, the discovery takes less than two seconds. Thus, we can discover high-quality models within a competitive amount of time: The Alpha, Heuristic, Inductive, and Split Miner only take several seconds to discover process models on all tested event logs. As stated above, algorithms based on region theory are much more time-consuming.

Figure 8 shows the quality of the best models discovered by each technique. In five out of six cases, the eST-Miner can discover better models than the Inductive and Alpha Miners. Although the quality of the best models discovered by the eST-Miner is high, the Heuristic and Split Miner can discover models that score slightly better concerning the F1-score in most instances. However, unlike the eST-Miner, they fail to provide guarantees: Many of the models we discovered do not have any possible firing sequence (see Fig. 8). Furthermore, they are more time-consuming to analyze algorithmically and by humans because they contain many silent transitions. The quality of the models discovered by the eST-Miner, particularly precision, is possibly impacted by their representational bias: In the absence of silent transitions and duplicate labels, repetitions and skips are modeled by self-loops.

Figure 9 shows the model discovered by the eST-Miner on the well-known *RTFM* event log. The eST-Miner balances well between fitness and precision while being able to handle noise and infrequent behavior. In [12], we show that for the *Sepsis* event log, we can discover a model that contains complex control-flow structures the inductive miner cannot discover.

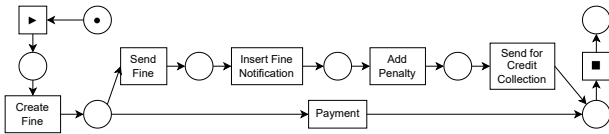


Fig. 9: Best Model discovered by the Delta variant of the eST-Miner on the RTFM event log. For reasons of simplicity, we have removed all activities that are not part of any possible firing sequence.

However, due to the challenges pointed out in Sec. VI a, when using the ETC-based composer, the model is of comparable quality but more complex.

c) *Limitations:* The runtime of the eST-Miner heavily depends on the size of the CT and, therefore, on the choice of  $d$  and the number of activities contained in the event log. Previously, when given  $\tau = 0.5$  and the Sepsis event log as input, and when limiting the depth of the CT to a greater depth than before ( $d = 7$ ), we have been unable to discover a process model within two hours using the eST-Miner and the fastest available IPR technique (LP-CIPR). Now, using the ETC-based composer, the discovery takes approximately 28 minutes. However, on a much bigger event log, e.g., the one used in the BPI-challenge in 2019 [23], which contains over 1.5 million events over 44 (unique) activities, we are still unable to discover a process model efficiently when limiting the depth of the CT to  $d = 4$  for any value for  $\tau < 1$ .

## VII. CONCLUSION AND FUTURE WORK

The ETC-based composer recalculates the precision of the expanding model during the discovery efficiently regarding runtime and memory. This enables a place classification strategy to avoid adding implicit places to the expanding model. Additionally, we can abort the search once the precision of the model reaches a user-definable threshold. In instances where the eST-Miner has previously been unable to discover a model within two hours, we can now discover models notably more efficiently. Thus, the eST-Miner becomes a competitive choice for process discovery on mid-sized real-life event logs. We can discover high-quality models in a reasonable amount of time while guaranteeing user-definable minimal fitness.

Apart from its ability to avoid implicit places, the ETC-based composer extends the discovery with information on precision which we can use in future work to extend the eST-Miner further, e.g., by pruning the search space. Some problems of our approach are yet to be solved: We need to investigate why some implicit places remain undetected when using the proposed heuristics if we consider non-fitting log traces. Furthermore, we would like to explore different (precision-guided) heuristics to change the search space's traversal order to improve the simplicity of the discovered models and other quality aspects. Lastly, it could be worth exploring whether efficiently (re)calculating precision while incorporating non-fitting log traces, as presented in this work, can be used for conformance checking or to improve other process discovery techniques.

## REFERENCES

- [1] W. M. P. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, 2004.
- [2] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs - A constructive approach," in *Petri NETS 2013, Proceedings*, ser. LNCS, J. M. Colom and J. Desel, Eds., vol. 7927. Springer, pp. 311–329.
- [3] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic process mining: an experimental evaluation," *Data Min. Knowl. Discov.*, vol. 14, no. 2, pp. 245–304, 2007.
- [4] A. J. M. M. Weijters and J. T. S. Ribeiro, "Flexible heuristics miner (FHM)," in *CIDM 2011, Proceedings*. IEEE, pp. 310–317.
- [5] A. Augusto, R. Conforti, M. Dumas, and M. L. Rosa, "Split miner: Discovering accurate and simple business process models from event logs," in *ICDM 2017, Proceedings*, V. Raghavan, S. Aluru, G. Karypis, L. Miele, and X. Wu, Eds. IEEE Computer Society, pp. 1–10.
- [6] J. Carmona, J. Cortadella, and M. Kishinevsky, "A region-based algorithm for discovering Petri nets from event logs," in *BPM 2008, Proceedings*, ser. LNCS, M. Dumas, M. Reichert, and M. Shan, Eds., vol. 5240. Springer, pp. 358–373.
- [7] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *Petri NETS 2008, Proceedings*, ser. LNCS, K. M. van Hee and R. Valk, Eds., vol. 5062. Springer, pp. 368–387.
- [8] L. L. Mannel and W. M. P. van der Aalst, "Finding complex process-structures by exploiting the token-game," in *PETRI NETS 2019, Proceedings*, ser. LNCS, S. Donatelli and S. Haar, Eds., vol. 11522. Springer, pp. 258–278.
- [9] L. L. Mannel, R. Bergenthum, and W. M. P. van der Aalst, "Removing implicit places using regions for process discovery," in *Petri NETS 2020, ATAE Workshop Proceedings*, ser. CEUR Workshop Proceedings, W. M. P. van der Aalst, R. Bergenthum, and J. Carmona, Eds., vol. 2625. CEUR-WS.org, pp. 20–32.
- [10] F. García-Vallés and J. M. Colom, "Implicit places in net systems," in *PNPM 1999*. IEEE Computer Society, 1999, pp. 104–113.
- [11] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *BPM 2010, Proceedings*, ser. LNCS, R. Hull, J. Mendling, and S. Tai, Eds., vol. 6336. Springer, 2010, pp. 211–226.
- [12] L. L. Mannel and W. M. P. van der Aalst, "Discovering process models with long-term dependencies while providing guarantees and handling infrequent behavior," in *Petri NETS 2022, Proceedings*, ser. LNCS, L. Bernardinello and L. Petrucci, Eds., vol. 13288. Springer, pp. 303–324.
- [13] W. M. P. van der Aalst, V. A. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: a two-step approach to balance between underfitting and overfitting," *Softw. Syst. Model.*, vol. 9, no. 1, pp. 87–111, 2010.
- [14] L. Tacke genannt Unterberg, "SPECpp-Framework," URL: <https://github.com/leahuh/specpp>, (Accessed: 2023-09-03).
- [15] H. Verbeek, J. Buijs, B. Dongen, and W. Aalst, "ProM 6: The Process Mining Toolkit," in *Proc. of BPM Demonstration Track 2010*, ser. CEUR Workshop Proceedings, M. L. Rosa, Ed., vol. 615, 2010, pp. 34–39.
- [16] A. Adriansyah, J. Muñoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," in *BPM 2012 Workshops, Revised Papers*, ser. LNBIP, M. L. Rosa and P. Soffer, Eds., vol. 132. Springer, pp. 137–149.
- [17] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.
- [18] M. Polato, "Dataset belonging to the help desk log of an Italian Company," 7 2017.
- [19] M. de Leoni and F. Mannhardt, "Road traffic fine management process," 02 2015.
- [20] F. Mannhardt, "Sepsis cases - event log," 01 2016.
- [21] W. van der Aalst, "Event logs and models used in process mining: Data science in action (2016)," URL: <https://processmining.org/old-version/event-book.html>, (Accessed: 2023-09-03).
- [22] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining Knowl. Discov.*, vol. 2, no. 2, pp. 182–192, 2012.
- [23] B. van Dongen, "Bpi challenge 2019," 2019.