



**HAL**  
open science

# Anti-Alignments – Measuring The Precision of Process Models and Event Logs

Thomas Chatain, Mathilde Boltenhagen, Josep Carmona

► **To cite this version:**

Thomas Chatain, Mathilde Boltenhagen, Josep Carmona. Anti-Alignments – Measuring The Precision of Process Models and Event Logs. Information Systems, 2021, 10.1016/j.is.2020.101708 . hal-02383546v3

**HAL Id: hal-02383546**

**<https://inria.hal.science/hal-02383546v3>**

Submitted on 4 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Anti-Alignments – Measuring The Precision of Process Models and Event Logs

Thomas Chatain<sup>a,\*</sup>, Mathilde Boltenhagen<sup>a</sup>, Josep Carmona<sup>b</sup>

<sup>a</sup>*Université Paris-Saclay, ENS Paris-Saclay, CNRS, LSV, Gif-sur-Yvette, France*

<sup>b</sup>*Universitat Politècnica de Catalunya, Barcelona, Spain*

---

## Abstract

Processes are a crucial artefact in organizations, since they coordinate the execution of activities so that products and services are provided. The use of models to analyse the underlying processes is a well-known practice. However, due to the complexity and continuous evolution of their processes, organizations need an effective way of analysing the relation between processes and models. Conformance checking techniques assess the suitability of a process model in representing an underlying process, observed through a collection of real executions. One important metric in conformance checking is to assess the precision of the model with respect to the observed executions, i.e., characterize the ability of the model to produce behavior unrelated to the one observed. In this paper we present the notion of *anti-alignment* as a concept to help unveiling runs in the model that may deviate significantly from the observed behavior. Using anti-alignments, a new metric for precision is proposed. The proposed anti-alignment based precision metric satisfies most of the required axioms highlighted in a recent publication. Moreover, a complexity analysis of the problem of computing anti-alignments is provided, which sheds light into the practicality of using anti-alignment to estimate precision. Experiments are provided that witness the validity of the concepts introduced in this paper.

*Keywords:*

---

\*Corresponding author. *Phone number:* XXX  
*Email addresses:* [chatain@lsv.fr](mailto:chatain@lsv.fr) (Thomas Chatain), [boltenhagen@lsv.fr](mailto:boltenhagen@lsv.fr) (Mathilde Boltenhagen), [jcarmona@cs.upc.edu](mailto:jcarmona@cs.upc.edu) (Josep Carmona)

## 1. Introduction

Relating observed and modelled process behavior is the lion's share of conformance checking [9]. Observed behavior is often recorded in form of *event logs*, that store the footprints of process executions. Symmetrically, process models are representations of the underlying process, which can be automatically discovered or manually designed. With the aim of quantifying this relation, conformance checking techniques consider four quality dimensions: fitness, precision, generalization and simplicity [24]. For the first three dimensions, the *alignment* between a process model and an event log is of paramount importance, since it allows relating modeled and observed behavior [1].

Given a process model and a trace in the event log, an alignment provides the run in the model which mostly resembles the observed trace. When alignments are computed, the quality dimensions can be defined on top [1, 20]. In a way, alignments are optimistic: although observed behavior may deviate significantly from modeled behavior, it is always assumed that the least deviations are the best explanation (from the model's perspective) for the observed behavior.

In this paper we present a somewhat symmetric notion to alignments, denoted as *anti-alignments*. Given a process model and a log, an anti-alignment is a run of the model that mostly deviates from any of the traces observed in the log. The motivation for anti-alignments is precisely to compensate the optimistic view provided by alignments, so that the model is queried to return a highly deviating behavior that has not been seen in the log. In contexts where the process model should adhere to a certain behavior and not leave much room for exotic possibilities (e.g., banking, healthcare), the absence of highly deviating anti-alignments may be a desired property for a process model. Using anti-alignments one cannot only catch deviating behavior, but also use it to improve some of the current quality metrics considered in conformance checking. In this paper we highlight the strong relation of anti-alignments and the

precision metric: a highly-deviating anti-alignment may be considered as a witness for a loss in precision. Some of the reference metrics for precision lack this ability of exploring the model behavior beyond what is observed in the log, thus being considered as short-sighted (e.g., [2]).

We cast the problem of computing anti-alignments as the satisfiability of a Boolean formula, and provide high-level techniques which can for instance compute the most deviating anti-alignment for a certain run length, or the shortest anti-alignment for a given number of deviations.

Anti-alignments are related to the *completeness of the log*; a log is complete if it contains all the behavior of the underlying process [28]. For incomplete logs, the alternatives for computing anti-alignments grow, making it difficult to tell the difference between behavior not observed but meant to be part of the process, and behavior not observed which is not meant to be part of the process. Since there exists already some metrics to evaluate the completeness of an event log (e.g., [36]), we assume event logs have a high level of completeness before they are used for computing anti-alignments. Notice that in presence of an incomplete event log, anti-alignments can be used to interactively complete it: an anti-alignment that is certified by the stake-holder as valid process behavior can be appended to the event log to make it more complete.

This work is an extension of recent publications related to anti-alignments: in [10] we established for the first time the notion of anti-alignments based on the Hamming distance, and proposed a simple metric to estimate precision. Then, the work in [30] elaborated the notion of anti-alignments, heuristically computing them for the Levenshtein distance by adapting the  $A^*$  search technique, and proposed two new notions for trace-based and log-based precision, that can be combined to estimate precision of process models. However, as it was claimed recently in a survey paper advocating for properties precision metrics should have [26], it was not known the satisfiability of the properties for the aforementioned metrics.

The contributions of the paper with respect to our previous work are now enumerated.

- We show how anti-alignments can be computed in an optimal way for the Levenshtein distance, without increasing the complexity class of the problem. Moreover we relate the two available distance encodings (Hamming and Levenshtein), and show the implications of using each one for anti-alignment based precision.
- We adapt the precision metrics from [30] to not depend on a particular length defined apriori.
- We prove the adherence of one of the new metrics proposed in this paper to most of the properties in [26].
- A novel implementation is provided, with several improvements and heuristics, which makes it able to deal with larger instances.
- A new evaluation section is reported, that shows empirically the capabilities of the proposed technique for large and real-life instances.

The remainder of the paper is organized as follows: in the next section, a simple example is used to emphasize the importance of anti-alignments and their application to estimate precision is shown. Then in Section 3 the basic theory needed for the understanding of the paper is introduced. Section 4 provides the formal definition of anti-alignments, whilst Section 5 formalizes the encoding into SAT of the problem of computing anti-alignments. In Section 6, we define a new metric, based on anti-alignments, for estimating precision of process models. Experiments are reported in Section 7, and related work in Section 8. Section 9 concludes the paper and gives some hints for future research directions.

## 2. A Motivating Example

Let us use the example shown in Figure 1 for illustrating the notion of anti-alignment. The example was originally presented in [31], and in this paper we present a very abstract version of it in Figure 1(a): The modeled process describes a realistic transaction process within a banking context. The process

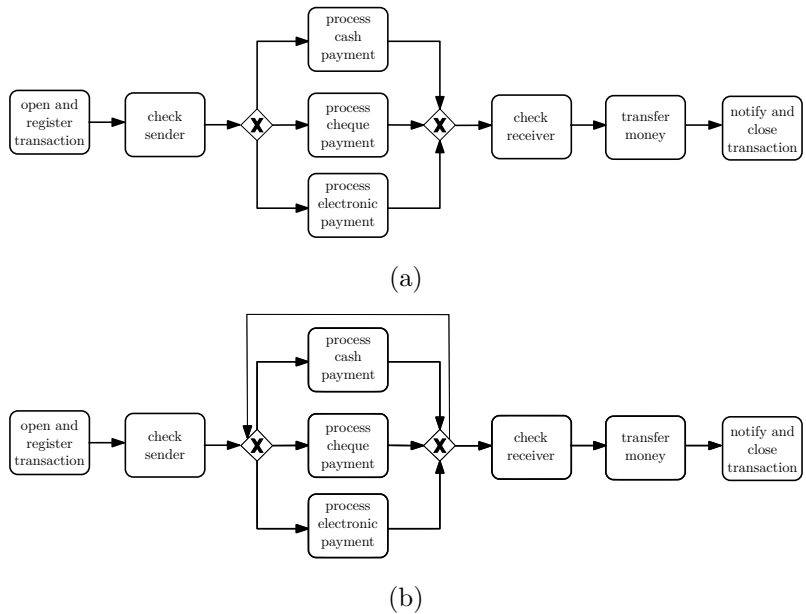


Figure 1: Example (adapted from [31]). (a) Initial process model, (b) Modified process model.

contains all sort of monetary checks, authority notifications, and logging mechanisms. The process is initiated when a new transaction is requested, opening a new instance in the system and registering all the components involved. The second step is to run a check on the person (or entity) origin of the monetary transaction. Then, the actual payment is processed differently, depending of the payment modality chosen by the sender (cash, cheque and payment). Later, the receiver is checked and the money is transferred. Finally, the process ends registering the information, notifying it to the required actors and authorities, and emitting the corresponding receipt.

Assume that a log covering all the three possible variants (corresponding to the three possible payment methods) with respect of the model in Figure 1(a) is given. The three different variants for this log will be:

```

ort, cs, pcap, cr, tm, nct
ort, cs, pchp, cr, tm, nct
ort, cs, pep, cr, tm, nct

```

where we use the acronym for each one of the actions performed, e.g., `ort` stands for *open and register transaction*.

For this pair of model and log, most of the current metrics for precision (e.g., [2]) will rightly assess a very high precision. In fact, since no deviating anti-alignment can be obtained because every model run is in the log, the anti-alignment based precision metric from this paper will also assess a high (in our case, perfect) precision.

Now assume that we modify a bit the model, adding a loop around the alternative stages for the payment. Intuitively, this (malicious) modification in the process model may allow to pay several times although only one transfer will be done. The modified high-level overview is shown in Figure 1(b). The aforementioned metric for precision will not consider this modification as a severe one: the precision of the model with respect to the log will be very similar to the one for the model in Figure 1(a).

Remarkably, this modification in the process model comes with a new highly deviating anti-alignment denoting a run of the model that contains more than one iteration of the payment:

`ort, cs, pcap, pchp, pchp, pep, pcap, cr, tm, nct`

Clearly, this model execution where five payments have been recorded is possible in the process of Figure 1(b). Correspondingly, the precision of this model in describing the log of only three variants will be significantly lowered in the metric proposed in this paper, since the anti-alignment produced is very different from any of the three variants recorded in the event log.

### 3. Preliminaries

**Definition 1 ((Labeled) Petri net).** A (labeled) Petri Net [21] is a tuple  $N = \langle P, T, \mathcal{F}, M_0, M_f, \Sigma, \lambda \rangle$ , where  $P$  is the set of places,  $T$  is the set of transitions (with  $P \cap T = \emptyset$ ),  $\mathcal{F} : (P \times T) \cup (T \times P) \rightarrow \{0, 1\}$  is the flow relation,  $M_0$  is the initial marking,  $M_f$  is the final marking,  $\Sigma$  is an alphabet of actions and  $\lambda : T \rightarrow \Sigma$  labels every transition by an action.

A marking is an assignment of a non-negative integer to each place. If  $k$  is assigned to place  $p$  by marking  $M$  (denoted  $M(p) = k$ ), we say that  $p$  is marked with  $k$  tokens. Given a node  $x \in P \cup T$ , its pre-set and post-set are denoted by  $\bullet x$  and  $x^\bullet$  respectively.

A transition  $t$  is *enabled* in a marking  $M$  when all places in  $\bullet t$  are marked. When a transition  $t$  is enabled, it can *fire* by removing a token from each place in  $\bullet t$  and putting a token to each place in  $t^\bullet$ . A marking  $M'$  is *reachable* from  $M$  if there is a sequence of firings  $t_1 \dots t_n$  that transforms  $M$  into  $M'$ , denoted by  $M[t_1 \dots t_n]M'$ . We define the *language* of  $N$  as the set of *full runs* defined by  $\mathcal{L}(N) := \{\lambda(t_1) \dots \lambda(t_n) \mid M_0[t_1 \dots t_n]M_f\}$ . A Petri net is *k-bounded* if no reachable marking assigns more than  $k$  tokens to any place. A Petri net is bounded if there exist a  $k$  for which it is  $k$ -bounded. A Petri net is *safe* if it is 1-bounded. A bounded Petri net has an *executable loop* if it has a reachable marking  $M$  and sequences of transitions  $u_1 \in T^*$ ,  $u_2 \in T^+$ ,  $u_3 \in T^*$  such that  $M_0[u_1]M[u_2]M[u_3]M_f$ .

An event log is a collection of traces, where a trace may appear more than once. Formally:

**Definition 2 (Event Log).** An event log  $L$  (over an alphabet of actions  $\Sigma$ ) is a multiset of traces  $\sigma \in \Sigma^*$ .

Process mining techniques aim at extracting from a log  $L$  a process model  $N$  (e.g., a Petri net) with the goal to elicit the process underlying a system  $\mathcal{S}$ .  $\mathcal{S}$  is considered a language for the sake of comparison. By relating the behaviors of  $L$ ,  $\mathcal{L}(N)$  and  $\mathcal{S}$ , particular concepts can be defined [8]. A log is *incomplete* if  $\mathcal{S} \setminus L \neq \emptyset$ . A model  $N$  *fits* log  $L$  if  $L \subseteq \mathcal{L}(N)$ . A model is *precise* in describing a log  $L$  if  $\mathcal{L}(N) \setminus L$  is small. A model  $N$  represents a *generalization* of log  $L$  with respect to system  $\mathcal{S}$  if some behavior in  $\mathcal{S} \setminus L$  exists in  $\mathcal{L}(N)$ . Finally, a model  $N$  is *simple* when it has the minimal complexity in representing  $\mathcal{L}(N)$ , i.e., the well-known *Occam's razor principle*.



#### 4. Anti-Alignments

The idea of *anti-alignments* is to seek in the language of a model  $N$  what are the runs which differ considerably with all the observed traces. Hence, this is the opposite of the notion of *alignments* [1] which is central in process mining: for many tasks in conformance checking like *process model repair* or *decision point analysis*, one needs indeed to find the run which is the most similar to a given log trace [28]. In this paper, we are focusing on precision and for this, traces which are not similar to any observed trace in the log serve as witnesses for bad precision. All these notions anyway depend on a definition of distance between two traces (typically a model trace, i.e. a run of the model, and an observed log trace). We assume a given distance function  $dist : \Sigma^* \times \Sigma^* \rightarrow [0, 1]$  computable in polynomial time and such that <sup>1</sup>

- for every  $\gamma \in \Sigma^*$ ,  $dist(\gamma, \gamma) = 0$ ,
- for every  $\sigma \in \Sigma^*$ ,  $dist(\gamma, \sigma)$  converges to 1 when  $|\gamma|$  diverges to  $\infty$ .

**Definition 3 (Anti-alignment).** For a distance threshold  $m \in [0, 1]$ , an  $m$ -anti-alignment of a model  $N$  w.r.t. a log  $L$  is a full run  $\gamma \in \mathcal{L}(N)$  such that  $dist(\gamma, L) \geq m$ , where  $dist(\gamma, L)$  is defined as the  $\min_{\sigma \in L} dist(\gamma, \sigma)$ .<sup>2</sup>

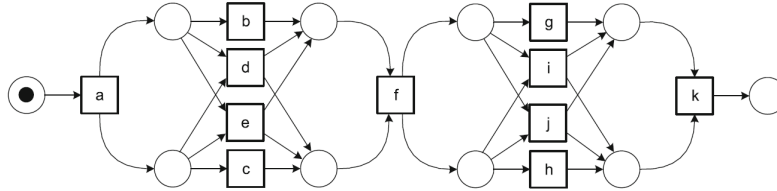
For the following examples, we show anti-alignments w.r.t. two possible choices of distance  $dist$ : Levenshtein's distance and Hamming distance.

**Definition 4 (Levenshtein's edit distance  $dist^L$ ).** Levenshtein's edit distance  $dist^L(\gamma, \sigma)$  between two traces  $\gamma, \sigma \in \Sigma^*$  is based on the minimum number  $\#edits(\gamma, \sigma)$  of deletions and insertions needed to transform  $\gamma$  to  $\sigma$ . In order to get a normalized distance between 0 and 1, we define Levenshtein's edit distance  $dist^L(\gamma, \sigma) = \frac{\#edits(\gamma, \sigma)}{\max(1, |\gamma| + |\sigma|)}$ .

---

<sup>1</sup>Actually, we do not require that  $dist$  satisfies the usual properties of distance functions like symmetry or triangle inequality.

<sup>2</sup>Since the function  $dist$  takes its values in  $[0, 1]$ , we define by convention  $dist(\gamma, \emptyset) := 1$ .



$$\begin{aligned}
L = \{ & \langle a, b, c, f, g, h, k \rangle, \\
& \langle a, c, b, f, g, h, k \rangle, \\
& \langle a, c, b, f, h, g, k \rangle, \\
& \langle a, b, c, f, h, g, k \rangle, \\
& \langle a, e, f, i, k \rangle, \\
& \langle a, d, f, g, h, k \rangle, \\
& \langle a, e, f, h, g, k \rangle \} ,
\end{aligned}$$

Figure 2: A process model (taken from [29]) and an event log. The full run  $\langle a, b, c, f, i, k \rangle$  is a  $\frac{3}{13}$ -anti-alignment for Levenshtein’s distance and a  $\frac{3}{7}$ -anti-alignment for Hamming distance.

**Example 1.** Consider the Petri net and log shown in Figure 2. With Levenshtein’s distance, the full run  $\langle a, b, c, f, i, k \rangle$  is at distance  $\frac{3}{13}$  from the log trace  $\langle a, b, c, f, g, h, k \rangle$  (two deletions and one insertion). It is at larger distance from the other log traces. Therefore, it is a  $\frac{3}{13}$ -anti-alignment.

Another interesting choice is Hamming distance. It is in general less informative than Levenshtein’s distance for relating observed and modelled behavior, but it has the interest of being very simple to compute. Variants of Hamming distance can also provide good compromises. In Sections 5 and 6, we will show how to efficiently compute anti-alignments for Hamming distance using SAT solvers.

**Definition 5 (Hamming distance  $dist^H$ ).** For two traces  $\gamma = \gamma_1 \dots \gamma_n$  and  $\sigma = \sigma_1 \dots \sigma_n$ , of same length  $n > 0$ , define  $dist^H(\gamma, \sigma) := \frac{1}{n} \cdot |\{i \in \{1 \dots n\} \mid \gamma_i \neq \sigma_i\}|$ . For  $\gamma$  longer than  $\sigma$ , we define  $dist^H(\gamma, \sigma) := dist^H(\gamma, \sigma \cdot w^{|\gamma| - |\sigma|})$ , where  $w \notin \Sigma$  is a special padding symbol ( $w$  for ‘wait’); we proceed symmetrically

when  $\gamma$  is shorter than  $\sigma$ .

**Lemma 1.** *Observe that, for every  $\gamma$  and  $\sigma$  (assuming one of them at least is nonempty),  $dist^L(\gamma, \sigma) \leq dist^H(\gamma, \sigma)$ .*

PROOF. Assume w.l.o.g.  $0 < |\gamma| < |\sigma|$ . Let  $n := |\{i \in \{1 \dots |\gamma|\} \mid \gamma_i \neq \sigma_i\}|$ . We have,  $dist_H(\gamma, \sigma) = \frac{n+|\sigma|-|\gamma|}{|\sigma|}$ . We have also  $dist_L(\gamma, \sigma) \leq \frac{2n+|\sigma|-|\gamma|}{|\gamma|+|\sigma|}$  because one way to transform  $\gamma$  to  $\sigma$  is to replace  $\gamma_i$  by  $\sigma_i$  (one deletion and one insertion) at each position  $i \leq |\gamma|$  where they differ ( $2n$  edits), and then to insert the letters  $\sigma_{|\gamma|+1} \dots \sigma_{|\sigma|}$  ( $|\sigma| - |\gamma|$  edits). It remains to see that  $n \leq |\gamma|$ , which implies

$$\begin{aligned} n(|\sigma| - |\gamma|) &\leq |\gamma|(|\sigma| - |\gamma|) \\ n|\sigma| &\leq |\gamma|(n + |\sigma| - |\gamma|) \\ n|\sigma| + (n + |\sigma| - |\gamma|)|\sigma| &\leq |\gamma|(n + |\sigma| - |\gamma|) + (n + |\sigma| - |\gamma|)|\sigma| \\ (2n + |\sigma| - |\gamma|)|\sigma| &\leq (|\gamma| + |\sigma|)(n + |\sigma| - |\gamma|) \\ \frac{2n + |\sigma| - |\gamma|}{|\gamma| + |\sigma|} &\leq \frac{(n + |\sigma| - |\gamma|)}{|\sigma|} \\ dist^L(\gamma, \sigma) &\leq dist^H(\gamma, \sigma). \end{aligned}$$

□

**Example 2.** *Consider the Petri net and log shown in Figure 2. With Hamming distance, the full run  $\gamma = \langle a, b, c, f, i, k \rangle$  is at distance  $\frac{3}{7}$  from the log trace  $\sigma = \langle a, b, c, f, g, h, k \rangle$  ( $i$  and  $k$  do not match with  $g$  and  $h$ , and  $\gamma$  is shorter than  $\sigma$ , which counts for the third mismatch). It is at larger distance from the other log traces. Therefore, it is a  $\frac{3}{7}$ -anti-alignment.*

**Lemma 2.** *For every log  $L$  and finite model  $N$ , we have:*

1. *if the model has finitely many full runs, then there exists (at least) one maximal anti-alignment  $\gamma$  of  $N$  w.r.t.  $L$ , i.e.  $\gamma$  that maximizes the distance  $dist(\gamma, L)$ ;*
2. *if the model has infinitely many full runs, then there exist anti-alignments  $\gamma$  with  $dist(\gamma, L)$  arbitrarily close to 1. Yet, there may not exist any 1-anti-alignment, i.e. there is no guarantee that the limit  $dist(\gamma, L) = 1$  is reached for any  $\gamma$ .*

PROOF. If the model has finitely many full runs, then one of them must be a maximal anti-alignment.

Conversely, if  $N$  is finite and has infinitely many runs, then there must exist arbitrary long full runs; more formally, there exists an infinite sequence of full runs  $(\gamma_i)_{i \in \mathbb{N}}$  of strictly increasing length. For every  $\sigma \in L$ , the sequence of  $(\text{dist}(\gamma_i, \sigma))_{i \in \mathbb{N}}$  converges to 1. Since there are finitely many  $\sigma \in L$ ,  $(\min_{\sigma \in L} \text{dist}(\gamma_i, \sigma))_{i \in \mathbb{N}}$  converges to 1 as well.  $\square$

Maximal anti-alignments will be used in Section 6 to define our precision metric. The case of models with executable loops will be discussed in Subsection 6.1.1.

**Lemma 3.** *The problem of deciding, given a finite model  $N$  and a log  $L$ , whether there exists a 0-anti-alignment of  $N$  w.r.t.  $L$ , has the same complexity as reachability for Petri nets.*

PROOF. This is equivalent to checking whether  $\mathcal{L}(N) \neq \emptyset$ , i.e. whether  $M_f$  is reachable from  $M_0$ .  $\square$

By definition, a 0-anti-alignment of  $N$  w.r.t.  $L$  is a full run  $\gamma \in \mathcal{L}(N)$  satisfying the trivial inequality  $\text{dist}(\gamma, L) \geq 0$ . The same problem with a strict inequality is also of interest. We will need it in Section 6.1.1.

**Lemma 4.** *The problem of deciding, given a finite model  $N$  and a log  $L$ , whether there exists a full run  $\gamma \in \mathcal{L}(N)$  satisfying  $\text{dist}(\gamma, L) > 0$  (or equivalently deciding if  $\mathcal{L}(N) \not\subseteq L$ ), has the same complexity as reachability for Petri nets.*

PROOF. The reachability problem reduces to the existence of a full run  $\gamma \in \mathcal{L}(N)$  satisfying  $\text{dist}(\gamma, \emptyset) > 0$ : indeed  $\text{dist}(\gamma, \emptyset) = 1$  for every  $\gamma$ .

Conversely, deciding if  $\mathcal{L}(N) \not\subseteq L$  reduces to deciding reachability of  $M_f \cup \{p_{esc}\}$  in the synchronous product of  $N$  with a deterministic Petri net which represents as a tree the log traces sharing their common prefixes, and, from the leaves, marks a sink place  $p_{esc}$ , as illustrated in Figure 3. Hence, every full run  $\gamma$  of  $N$ , when synchronized with the Petri net representation of the log  $L$ , leads to a marking of the form  $M_f \cup \{p\}$ , and  $p = p_{esc}$  iff  $\gamma \notin L$ .  $\square$

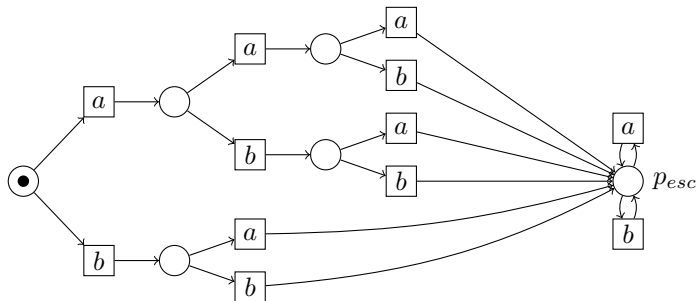


Figure 3: A deterministic Petri net representing the log  $L = \{\langle a \rangle, \langle a, b \rangle, \langle a, a \rangle, \langle b \rangle\}$ , for the alphabet of actions  $\Sigma = \{a, b\}$ . Place  $p_{esc}$  is reached only by the runs which do not appear in the log.

The problem of reachability in Petri nets is known to be decidable, but non-elementary [12]. Yet, the complexity drops to NP if a bound is given on the length of the anti-alignment.

**Lemma 5.** *The problem of deciding, for a Petri net  $N$ , a log  $L$ , a rational distance threshold  $m \in \mathbb{Q}$  and a bound  $n \in \mathbb{N}$ , if there exists an  $m$ -anti-alignment  $\gamma$  such that  $|\gamma| \leq n$ , is NP-complete. We assume that  $n$  is encoded in unary.<sup>3</sup>*

PROOF. The problem is clearly in NP: checking that a run  $\gamma$  is a  $m$ -anti-alignment of  $N$  w.r.t.  $L$  takes polynomial time (remember that we consider distance functions computable in polynomial time).

For NP-hardness, we propose a reduction from the problem of reachability of a marking  $M_f$  in a 1-safe acyclic<sup>4</sup> Petri net  $N$ , known to be NP-complete [25, 11]. Notice that, since  $N$  is acyclic, each transition can fire only once; hence, the length of the firing sequences of  $N$  is bounded by the number of transitions  $|T|$ . Finally,  $M_f$  is reachable in  $N$  iff there exists a  $\gamma$  of length less or equal to  $|T|$  which is a 0-anti-alignment of  $N$  (with  $M_f$  as final marking) w.r.t. the empty log.  $\square$

<sup>3</sup>Since  $n$  has typically the same order of magnitude as the length of the longest traces in the log, encoding  $n$  in unary does not significantly affect the size of the problem instances.

<sup>4</sup>a Petri net is acyclic if the transitive closure  $\mathcal{F}^+$  of its flow relation is irreflexive.

## 5. SAT-encoding of Anti-Alignments

In this section, we give hints on how SAT solvers can help to find anti-alignments. We detail the construction of a SAT formula  $\Phi_m^n(N, L)$ , where  $N$  is a Petri net,  $L$  a log,  $n$  and  $m$  two integers. This formula will be used in the search of anti-alignments of  $N$  w.r.t.  $L$  for Hamming distance (see Section 5.3 for the encoding using the Levenshtein distance). The formula  $\Phi_m^n(N, L)$  characterizes precisely the full runs  $\gamma = \lambda(t_1) \dots \lambda(t_n) \in \mathcal{L}(N)$  of  $N$  of length  $n$  which differ in at least  $m$  positions with every log trace in  $L$ .

### 5.1. Coding $\Phi_m^n(N, L)$ Using Boolean Variables

The formula  $\Phi_m^n(N, L)$  is coded using the following Boolean variables:

- $\tau_{i,t}$  for  $i = 1 \dots n, t \in T$  (remind that  $w$  is the special symbol used to pad the log traces, see Definition 5) means that transition  $t_i = t$ .
- $m_{i,p}$  for  $i = 0 \dots n, p \in P$  means that place  $p$  is marked in marking  $M_i$  (remind that we consider only safe nets, therefore the  $m_{i,p}$  are Boolean variables).
- $\delta_{i,\sigma,k}$  for  $i = 1 \dots n, \sigma \in L, k = 1, \dots, m$  means that the  $k^{\text{th}}$  mismatch with the observed trace  $\sigma$  is at position  $i$ .

The total number of variables is  $O(n \times (|T| + |P| + |L| \times m))$ .

Let us decompose the formula  $\Phi_m^n(N, L)$ .

- The fact that  $\gamma = \lambda(t_1) \dots \lambda(t_n) \in \mathcal{L}(N)$  is coded by the conjunction of the following formulas:

– Initial marking:

$$\left( \bigwedge_{p \in M_0} m_{0,p} \right) \wedge \left( \bigwedge_{p \in P \setminus M_0} \neg m_{0,p} \right)$$

– Final marking:

$$\left( \bigwedge_{p \in M_f} m_{n,p} \right) \wedge \left( \bigwedge_{p \in P \setminus M_f} \neg m_{n,p} \right)$$

– One and only one  $t_i$  for each  $i$ :

$$\bigwedge_{i=1}^n \bigvee_{t \in T} (\tau_{i,t} \wedge \bigwedge_{t' \in T} \neg \tau_{i,t'})$$

– The transitions are enabled when they fire:

$$\bigwedge_{i=1}^n \bigwedge_{t \in T} (\tau_{i,t} \implies \bigwedge_{p \in \bullet t} m_{i-1,p})$$

– Token game (for safe Petri nets):

$$\begin{aligned} & \bigwedge_{i=1}^n \bigwedge_{t \in T} \bigwedge_{p \in t^\bullet} (\tau_{i,t} \implies m_{i,p}) \\ & \bigwedge_{i=1}^n \bigwedge_{t \in T} \bigwedge_{p \in \bullet t \setminus t^\bullet} (\tau_{i,t} \implies \neg m_{i,p}) \\ & \bigwedge_{i=1}^n \bigwedge_{t \in T} \bigwedge_{p \in P, p \notin \bullet t, p \notin t^\bullet} (\tau_{i,t} \implies (m_{i,p} \iff m_{i-1,p})) \end{aligned}$$

- Now, the constraint that  $\gamma$  deviates from the observed traces (for every  $\sigma \in L$ ,  $dist(\gamma, \sigma) \geq m$ ) is coded as:

$$\bigwedge_{\sigma \in L} \bigwedge_{k=1}^m \bigvee_{i=1}^n \delta_{i,\sigma,k}$$

with the  $\delta_{i,\sigma,k}$  correctly affected w.r.t.  $\lambda(t_i)$  and  $\sigma_i$ :

$$\bigwedge_{\sigma \in L} \bigwedge_{k=1}^m \bigwedge_{i=1}^n \left( \delta_{i,\sigma,k} \iff \bigvee_{t \in T, \lambda(t) \neq \sigma_i} \tau_{i,t} \right)$$

and that for  $k \neq k'$ , the  $k^{\text{th}}$  and  $k'^{\text{th}}$  mismatch correspond to different  $i$ 's (i.e. a given mismatch cannot serve twice):

$$\bigwedge_{\sigma \in L} \bigwedge_{i=1}^n \bigwedge_{k=1}^{m-1} \bigwedge_{k'=k+1}^m \neg (\delta_{i,\sigma,k} \wedge \delta_{i,\sigma,k'})$$

## 5.2. Size of the Formula

In the end, the first part of the formula ( $\gamma = \lambda(t_1) \dots \lambda(t_n) \in \mathcal{L}(N)$ ) is coded by a Boolean formula of size  $O(n \times |T| \times |N|)$ , with  $|N| := |T| + |P|$ .

The second part of the formula (for every  $\sigma \in L$ ,  $dist(\gamma, \sigma) \geq m$ ) is coded by a Boolean formula of size  $O(n \times m^2 \times |L| \times |T|)$ .

The total size for the coding of the formula  $\Phi_m^n(N, L)$  is

$$O(n \times |T| \times (|N| + m^2 \times |L|)) .$$

### 5.3. SAT-encoding of Anti-Alignments for Levenshtein's Edit Distance

Our SAT-encoding of anti-alignments for Levenshtein's edit distance uses the same boolean variables as the SAT-encoding of anti-alignments for Hamming distance of the previous section, completed with  $\delta$  variables used to encode the edit distance.

Our encoding is based on the same relations that are used by the classical dynamic programming recursive algorithm for computing the edit distance between two words  $u = \langle u_1, \dots, u_n \rangle$  and  $v = \langle v_1, \dots, v_m \rangle$ :

$$\left\{ \begin{array}{l} dist(\langle u_1, \dots, u_i \rangle, \epsilon) = i \\ dist(\epsilon, \langle v_1, \dots, v_j \rangle) = j \\ dist(\langle u_1, \dots, u_{i+1} \rangle, \langle v_1, \dots, v_{j+1} \rangle) = \\ \left\{ \begin{array}{ll} dist(\langle u_1, \dots, u_i \rangle, \langle v_1, \dots, v_j \rangle) & \text{if } u_{i+1} = v_{j+1} \\ 1 + \min(dist(\langle u_1, \dots, u_{i+1} \rangle, \langle v_1, \dots, v_j \rangle), \\ \quad dist(\langle u_1, \dots, u_i \rangle, \langle v_1, \dots, v_{j+1} \rangle)) & \text{if } u_{i+1} \neq v_{j+1} \end{array} \right. \end{array} \right.$$

We encode this computation in a SAT formula  $\phi$  over variables  $\delta_{i,j,d}$ , for  $i = 0, \dots, n$ ,  $j = 0, \dots, m$  and  $d = 0, \dots, n + m$ . Formula  $\phi$  will have exactly one solution, in which each variable  $\delta_{i,j,d}$  is **true** iff  $\langle u_1 \dots u_i \rangle$  and  $\langle v_1 \dots v_j \rangle$  differ by at least  $d$  edits.

In order to test equality between the  $u_i$  and  $v_j$ , we use variables  $\lambda_{i,a}$  and  $\lambda'_{j,a}$ , for  $i = 0, \dots, n$ ,  $j = 0, \dots, m$  and  $a \in \Sigma$ , and we set their value such that  $\lambda_{i,a}$  is **true** iff  $u_i = a$ , and  $\lambda'_{j,a}$  is **true** iff  $v_j = a$ . Hence, the test  $u_{i+1} = v_{j+1}$  becomes in our formulas:  $\bigvee_{a \in \Sigma} (\lambda_{i+1,a} \wedge \lambda'_{j+1,a})$ . For readability of the formulas, we refer to this coding by  $[u_{i+1} = v_{j+1}]$ . We also write similarly  $[u_{i+1} \neq v_{j+1}]$ .

In the following, we describe the different clauses of the formula  $\phi$  of our



SAT encoding of the edit distance.

$$\delta_{0,0,0} \wedge \bigwedge_{d>0} \neg \delta_{0,0,d} \quad (1)$$

$$\bigwedge_d \bigwedge_{i=0}^n (\delta_{i+1,0,d+1} \Leftrightarrow \delta_{i,0,d}) \quad (2)$$

$$\bigwedge_d \bigwedge_{j=0}^m (\delta_{0,j+1,d+1} \Leftrightarrow \delta_{0,j,d}) \quad (3)$$

$$\bigwedge_d \bigwedge_{i=0}^n \bigwedge_{j=0}^m [u_{i+1} = v_{j+1}] \Rightarrow (\delta_{i+1,j+1,d} \Leftrightarrow \delta_{i,j,d}) \quad (4)$$

$$\bigwedge_d \bigwedge_{i=0}^n \bigwedge_{j=0}^m [u_{i+1} \neq v_{j+1}] \Rightarrow (\delta_{i+1,j+1,d+1} \Leftrightarrow (\delta_{i+1,j,d} \wedge \delta_{i,j+1,d})) \quad (5)$$

**Example 3.** At instants  $i = 1$  and  $j = 1$  of words  $u = \langle s, g, c \rangle$  and  $v = \langle s, b, c, a \rangle$ , the letters are the same, then, by (4), the distance is only higher or equal to 0 :  $(u_1 = v_1) \Rightarrow (\delta_{1,1,0} \Leftrightarrow \delta_{0,0,0})$ .

However at instants  $i = 2$  and  $j = 2$ , the letters  $u_2$  and  $v_2$  are different. A step before,  $\delta_{1,2,1}$  and  $\delta_{2,1,1}$  are **true** because of the length of the subwords. Then, by (5), the distance at instants  $i = 2$  and  $j = 2$  is higher or equal to 2 :  $\delta_{2,2,2}$ . The result is understandable because the edit distance costs the deletion of  $g$  and the addition of  $b$  to transform  $u$  to  $v$ .

In order to insert this encoding of Leventshein's edit distance into our formulas for anti-alignments, we need to compute the edit distance between the expected anti-alignment and every trace of the log, which requires to use variables  $\delta_{i,\sigma,j,k}$  for  $i = 0 \dots n$ ,  $\sigma \in L$ ,  $j = 0 \dots |\sigma|$ ,  $k = 0, \dots, \max(n, |\sigma|)$  to represent the fact that  $u_1 \dots u_i$  and  $v_1 \dots v_j$  differ by at least  $k$  edits.

#### 5.4. Solving the Formula in Practice

In practice, the coding of the formula  $\Phi_m^n(N, L)$  can be done using the Boolean variables  $\tau_{i,t}$ ,  $m_{i,p}$  and  $\delta_{i,\sigma,k}$ .

Then we need to transform the formula in conjunctive normal form (CNF) in order to pass it to the SAT solver. We use Tseytin's transformation [27] to get a formula in conjunctive normal form (CNF) whose size is linear in the size of the original formula. The idea of this transformation is to replace recursively the disjunctions  $\phi_1 \vee \dots \vee \phi_n$  (where the  $\phi_i$  are not atoms) by the following

equivalent formula:

$$\exists x_1, \dots, x_n \left\{ \begin{array}{l} x_1 \vee \dots \vee x_n \\ \wedge x_1 \implies \phi_1 \\ \wedge \dots \\ \wedge x_n \implies \phi_n \end{array} \right.$$

where  $x_1, \dots, x_n$  are fresh variables.

In the end, the SAT solver tells us if there exists a run  $\gamma = \lambda(t_1) \dots \lambda(t_n) \in \mathcal{L}(N)$  which differs by at least  $m$  edits with every observed trace  $\sigma \in L$ . If a solution is found, we extract the run  $\gamma$  using the values assigned by the SAT solver to the Boolean variables  $\tau_{i,t}$ .

## 6. Using Anti-Alignments to Estimate Precision

In this section we show how to use anti-alignments to estimate precision of process models. Remarkably, we show how to modify the definitions of [10, 30] so that the new metric does not depend on a predefined length. In Section 6.2 we dive into the adherence of the metric with respect to a recent proposal for properties of precision metrics [26].

### 6.1. Precision

Our precision metric is an adaptation of our previous versions presented in [10, 30]. It relies on anti-alignments to find the model run that is as distant as possible to the log traces. Like anti-alignments, the definition of precision is parameterized by a distance *dist*. In the examples, we will specify each time if we use Levenshtein's edit distance (Definition 4) or Hamming distance (Definition 5).

**Definition 6 (Precision).** *Let  $L$  be an event log and  $N$  a model. We define precision as follows:*

$$P_{aa}(N, L) := 1 - \sup_{\gamma \in \mathcal{L}(N)} \text{dist}(\gamma, L).$$

For instance, consider the model and log shown in Figure 2. With Levenshtein’s distance, the full run  $\langle a, b, c, f, i, k \rangle$  is a maximal anti-alignment. It is at distance  $\frac{3}{13}$  to any of the log traces, and hence  $P_{aa}(N, L) = 1 - \frac{3}{13} = 0.77$ .

### 6.1.1. Handling Process Models with Loops

Notice that a model with arbitrary long runs (i.e., a process model that contains loops) may cause the formula in Definition 6 to converge to 0. This is a natural artifact of comparing a finite language (the event log), with a possibly infinite language (the process model). Since process models in reality contain loops, an adaptation of the metric is done in this section, so that it can also handle this type of models without penalizing severely the loops.

**Definition 7 (Precision for Models with Loops).** *Let  $L$  be an event log and  $N$  a model. We define  $\epsilon$ -precision as follows:*

$$P_{aa}^\epsilon(N, L) := 1 - \sup_{\gamma \in \mathcal{L}(N)} \frac{\text{dist}(\gamma, L)}{(1 + \epsilon)^{|\gamma|}}$$

with some  $\epsilon \geq 0$  which is a parameter of this definition.

Informally, the formula computes the anti-alignment that provides maximal distance with any trace in the log, and at the same time tries to minimize its length. The penalization for the length is parametrized over the  $\epsilon^5$ . Observe that  $P_{aa}^0(N, L)$  is precisely the precision  $P_{aa}(N, L)$  of Definition 6. By making Definition 7 not dependant on a predefined length, it deviates from the log-based precision metrics defined in previous work [10, 30].

Let us now consider the model of Figure 4, and the log  $L = [\langle a \rangle, \langle a, b, c, d \rangle, \langle a, f, g, h \rangle, \langle a, b, i, b, c, d \rangle]$ . Assume that  $\epsilon = 0.05$ . A possible anti-alignment is  $\gamma_1 = \langle a, c, b, e \rangle$  which is at least at Levenshtein’s distance  $\frac{1}{2}$  to any of the log traces. For  $\gamma_1$  the value of the formula is  $1 - \frac{\frac{1}{2}}{(1.05)^4} = 0.589$ .

---

<sup>5</sup>Although, admittedly,  $\epsilon$  is a parameter that should be decided apriori, in practice one can use a particular value to this parameter through several instances, without impacting significantly the insights obtained through this metric.

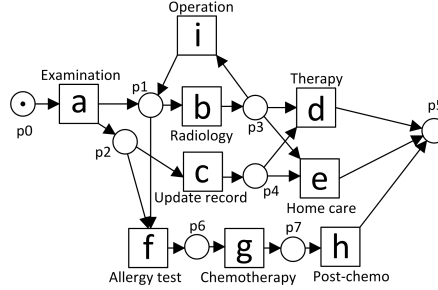


Figure 4: Example from [2]

Another possible anti-alignment is  $\gamma_2 = \langle a, c, b, i, b, i, b, i, b, i, b, e \rangle$  which is at least at distance  $\frac{10}{18}$  to any of the log traces. For  $\gamma_2$  the value of the formula is  $1 - \frac{\frac{10}{18}}{(1.05)^{12}} = 0.691$ . Hence, since the anti-alignment that maximizes the second term of the formula is  $\gamma_1$ , the precision computed is  $P_{aa}^{0.05}(N, L) = 0.589$ . If instead,  $\epsilon$  is set to a lower value, e.g.,  $\epsilon = 0.02$ , the corresponding value of the formula for the anti-alignment  $\langle a, c, b, i, b, i, b, i, b, i, b, i, b, i, b, e \rangle$  will be the mainimal, and therefore it will be selected as the anti-alignment resulting in  $P_{aa}^{0.02}(N, L) = 0.533$ .

### 6.1.2. Computing $P_{aa}^\epsilon$

By incorporating the  $\epsilon$  parameter in the definition of precision, now the metric can deal with models containing loops without predefining the length of the anti-alignment. In this section we show that the proposed extension is well-defined and can be computed, and provide some complexity results of the algorithms involved.

**Lemma 6.** *For every finite model  $N$ , log  $L$  and  $\epsilon > 0$ , the supremum in the definition of  $P_{aa}^\epsilon$  is reached, i.e. there exists a full run  $\gamma \in \mathcal{L}(N)$  such that  $P_{aa}^\epsilon(N, L) = 1 - \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}}$ .*

PROOF. Two cases have to be distinguished: if  $\mathcal{L}(N) \subseteq L$ , then the supremum equals 0, is obviously reached by any  $\gamma \in \mathcal{L}(N)$ , and  $P_{aa}^\epsilon(N, L) = 1$ ; otherwise, let  $\gamma_0 \in \mathcal{L}(N) \setminus L$  and let  $m := \frac{\text{dist}(\gamma_0, L)}{(1+\epsilon)^{|\gamma_0|}}$ ; we show that the supremum in the definition of  $P_{aa}^\epsilon$  becomes now a maximum over a finite set of runs, bounded by

a given length  $n$  that depends on  $m$  and  $\epsilon$ :

$$\sup_{\gamma \in \mathcal{L}(N)} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}} = \max_{\gamma \in \mathcal{L}(N), |\gamma| \leq n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}},$$

with  $n := \left\lfloor \frac{-\log m}{\log(1+\epsilon)} \right\rfloor$ . Indeed, for every  $\gamma$  strictly longer than  $n$ , we have  $\frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}} \leq \frac{1}{(1+\epsilon)^{n+1}} = \exp(-(n+1) \cdot \log(1+\epsilon)) \leq \exp(\log m) = m$ , which also shows that  $|\gamma_0| \leq n$ . Hence  $\gamma_0$  is considered in our max, and then  $\max_{\gamma \in \mathcal{L}(N), |\gamma| \leq n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}} \geq m > \sup_{\gamma \in \mathcal{L}(N), |\gamma| > n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}}$ .  $\square$

Lemma 6 gives us the key for an algorithm to compute  $P_{aa}^\epsilon$ .

**Algorithm 1.** *Algorithm for computing  $P_{aa}^\epsilon(N, L)$ :*

- if  $\mathcal{L}(N) \not\subseteq L$ , then
  - select  $\gamma_0 \in \mathcal{L}(N) \setminus L$
  - let  $m := \frac{\text{dist}(\gamma_0, L)}{(1+\epsilon)^{|\gamma_0|}}$
  - explore the reachability graph of  $N$  until depth  $n := \left\lfloor \frac{-\log m}{\log(1+\epsilon)} \right\rfloor$  and return  $1 - \max_{\gamma \in \mathcal{L}(N), |\gamma| \leq n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}}$ ;
- else return 1 (the model has perfect precision).

The correctness of this algorithm follows directly from Lemma 6. Its complexity resides essentially in the initial test, which corresponds to simply deciding if  $P_{aa}^\epsilon(N, L) < 1$ , whose complexity is given by the following lemma:

**Lemma 7.** *The problem of deciding, for a finite model  $N$  and a log  $L$ , if  $P_{aa}^\epsilon(N, L) < 1$ , is equivalent to deciding reachability in Petri nets.*

PROOF. We simply observe that  $P_{aa}^\epsilon(N, L) < 1$  iff  $\mathcal{L}(N) \not\subseteq L$ . Deciding this is equivalent to deciding reachability in Petri nets, as showed in Lemma 4.  $\square$

However, in practice, one would generally skip the first check and jump directly to the exploration until some depth  $n$ , possibly computed for a given threshold  $m$ , like the one given by the  $\gamma_0$  in Algorithm 1. Notice that the

algorithm explores less deep (i.e.  $n$  is smaller) when  $m$  is large (close to 1), i.e.  $\gamma_0$  is close to the optimal anti-alignment. We can summarize this with the following variation of Algorithm 1:

**Algorithm 2.** *Algorithm for estimating  $P_{aa}^\epsilon(N, L)$  using a threshold  $0 < m \leq 1$  as input:*

- explore the reachability graph of  $N$  until depth  $n := \left\lfloor \frac{-\log m}{\log(1+\epsilon)} \right\rfloor$
- if the exploration finds a full run  $\gamma \in \mathcal{L}(N) \setminus L$
- then output “ $P_{aa}^\epsilon(N, L) = 1 - \max_{\gamma \in \mathcal{L}(N), |\gamma| \leq n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}}$ ”
- else output “ $P_{aa}^\epsilon(N, L) \geq 1 - m$ ”.

**Lemma 8.** *For any fixed  $\epsilon > 0$ , the problem of deciding, for a finite model  $N$ , a log  $L$  and a rational constant  $m > 0$ , if  $P_{aa}^\epsilon(N, L) < 1 - m$ , is NP-complete.*

PROOF. The proof is similar to the one of Lemma 6; here, the bound  $m$  is given directly, and we have the same equality

$$\sup_{\gamma \in \mathcal{L}(N)} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}} = \max_{\gamma \in \mathcal{L}(N), |\gamma| \leq n} \frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}},$$

with  $n := \left\lfloor \frac{-\log m}{\log(1+\epsilon)} \right\rfloor$ . This means, in order to check that  $P_{aa}^\epsilon(N, L) < 1 - m$ , it suffices to guess a full run  $\gamma$  of length  $|\gamma| \leq n$ , where  $n$  depends linearly on the size of the representation of  $m$  (number of bits in the numerator and denominator). Then one can check in polynomial time that  $\frac{\text{dist}(\gamma, L)}{(1+\epsilon)^{|\gamma|}} > m$ .

For completeness, we proceed like in Lemma 5: we reduce reachability of  $M_f$  in a 1-safe acyclic Petri net  $N$  to  $P_{aa}^\epsilon(N, L) < 1 - m$  with  $L = \emptyset$  and  $m = \frac{1}{(1+\epsilon)^{|T|+1}}$ . □

## 6.2. Discussion about Reference Properties for Precision

Recently, an effort to consolidate a set of desired properties for precision metrics has been proposed [26]. Five axioms are described that establish different features of a precision metric  $\text{prec}(N, L)$ . Summarizing, the axioms proposed in [26] are:

- A1: A precision metric should be a function, i.e. it should be deterministic.
- A2: If a process model  $N_2$  allows for more behavior not seen in a log  $L$  than another model  $N_1$  does, then  $N_2$  should have a lower precision than  $N_1$  regarding  $L$ :

$$L \subseteq \mathcal{L}(N_1) \subseteq \mathcal{L}(N_2) \implies \text{prec}(N_1, L) \geq \text{prec}(N_2, L)$$

- A3: Let  $N_1$  be a model that allows for the behavior seen in a log  $L$ , and at the same time its behavior is properly included in a model  $N_2$  whose language is  $\Sigma^{*6}$  (called a flower model). Then the precision of  $N_1$  on  $L$  should be strictly greater than the one for  $N_2$ .
- A4: The precision of a log on two language equivalent models should be equal:

$$\mathcal{L}(N_1) = \mathcal{L}(N_2) \implies \text{prec}(N_1, L) = \text{prec}(N_2, L)$$

- A5: Adding fitting traces to a fitting log can only increase the precision of a given model with respect to the log:

$$L_1 \subseteq L_2 \subseteq \mathcal{L}(N) \implies \text{prec}(N, L_1) \leq \text{prec}(N, L_2)$$

In the aforementioned paper, it is shown that the previous version of our antialignment-based precision metric (from [30]) does not satisfy axiom A5 (the satisfaction of the rest of axioms are declared as *unknowns* in the paper). With the new version of the metric presented in this paper, we here provide proofs for these axioms, except for A3. But at the same time, we show that any precision metric can be adapted in order to satisfy A3.

**Lemma 9.** *The metric  $P_{aa}^\epsilon$  (for any fixed  $\epsilon$ ) satisfies A1.*

PROOF. Everything in our definitions is functional. □

---

<sup>6</sup>Actually, [26] writes “ $\mathcal{L}(N_1) \subset \mathcal{P}(\Sigma^*)$ ”, with  $\mathcal{P}$  for powerset, but we believe this is a mistake.

**Lemma 10.** *The metric  $P_{aa}^\epsilon$  satisfies A2.*

PROOF. Let  $L \subseteq \mathcal{L}(N_1) \subseteq \mathcal{L}(N_2)$ . The definitions take the  $\sup_{\gamma \in \mathcal{L}(N)}$  of an expression which does not depend on the model. Since  $\mathcal{L}(N_1) \subseteq \mathcal{L}(N_2)$ , the  $\sup_{\gamma \in \mathcal{L}(N_2)}$  ranges over a  $\subseteq$ -larger set than the  $\sup_{\gamma \in \mathcal{L}(N_1)}$ . Therefore the result cannot be smaller, and we get  $prec(N_1, L) \geq prec(N_2, L)$ .  $\square$

Our metrics may not satisfy the strict inequality required by A3: they satisfy only a weaker version of A3 with non-strict inequality, but, as observed in [26], this is then simply subsumed by A2. The authors of [26] precisely introduced A3 after arguing that, in case of a flower model, a strict inequality should be required.

Anyway, we show in Lemma 11 that any precision metric  $prec$  can be modified so that it satisfies this requirement of strict inequality for the flower models.

**Lemma 11.** *Let  $prec$  be any precision metric. It is possible to define a metric  $prec'$  from  $prec$  such that  $prec'$  satisfies A3: it suffices to set the precision of the flower models to a value smaller than all the other precision values (after possibly extending the target set of the function). This guarantees A3 and preserves all the other axioms.*

PROOF. The new metric  $prec'$  satisfies A3 by construction. Moreover, if  $prec$  is deterministic (A1), then  $prec'$  also is. For preservation of A2, A4 and A5, it suffices to study the different cases (separate flower model and others) to show that the equality and non-strict inequalities are preserved.  $\square$

We consider that satisfying A3 is a very artificial issue. However, if really the transformation defined in Lemma 11 had to be implemented, it would imply that, in order to compute the precision, one would have to decide if the model  $N$  is a flower model, i.e. if  $\mathcal{L}(N) = \Sigma^*$ . This is known as the universality problem. This problem is, in theory, highly intractable<sup>7</sup>. But again, this is very

---

<sup>7</sup>This universality problem is PSPACE-complete for non-deterministic finite state automata



artificial: in practice it suffices to explore at a very short finite horizon to detect many many non-flower models.

**Lemma 12.** *The metric  $P_{aa}^\epsilon$  satisfies A4.*

PROOF. This trivially holds since both metrics are behaviorally defined. Also, we copied this axiom A4 from [26], but observe that it is a simple corollary of A2 as soon as  $L \subseteq \mathcal{L}(N_1) = \mathcal{L}(N_2)$ .  $\square$

**Lemma 13.** *Metrics  $P_{aa}^\epsilon$  (for any  $\epsilon$ ) satisfies A5.*

PROOF. With  $L_1 \subseteq L_2$ , for every  $\gamma \in \mathcal{L}(N)$ , we have  $\text{dist}(\gamma, L_1) \geq \text{dist}(\gamma, L_2)$ , so the  $\sup_{\gamma \in \mathcal{L}(N)}$  cannot be smaller for  $L_1$  than for  $L_2$ . The rest does not depend on the log.  $\square$

### 6.3. Heuristics

Due to the complexity of the problem, we present two heuristics to approximate anti-alignment based precision.

#### 6.3.1. Prefix Limitation

Anti-alignment based precision requires the computation of a run of the process model that maximizes the minimal distance to the log traces. A first approximation is then to compute only a prefix of it.

**Definition 8 (Prefixes of Petri net).** *A Prefix of size  $n$  for a Petri net  $N = \langle P, T, \mathcal{F}, M_0, M_f, \Sigma, \lambda \rangle$  is a firing sequence  $M_0[u_1] \dots M_{n-1}[u_n]M_n$ , starting from the initial marking  $M_0$  and of maximal length  $n$ .*

For example, the prefixes of size 3 for the model of Fig. 4 are the firing sequences:  $\{p_0\}\{a\}\{p_1, p_2\}\{b\}\{p_2, p_3\}\{c\}\{p_3, p_4\}$  and  $\{p_0\}\{a\}\{p_1, p_2\}\{f\}\{p_1, p_6\}\{g\}\{p_1, p_7\}$  and give the runs  $\langle a, b, c \rangle$  and  $\langle a, f, g \rangle$ .

---

(NFSAs) [17], and here the NFSAs to consider would be the reachability graph of  $N$  (for  $N$   $k$ -bounded), which is exponential in the size of  $N$ . Hence, deciding universality for  $k$ -bounded labeled Petri nets is in EXPSpace.

Similarly, the log traces are also truncated.

For instance, for  $n = 3$  and  $\log L = [\langle a \rangle, \langle a, b, c, d \rangle, \langle a, f, g, h \rangle, \langle a, b, i, b, c, d \rangle]$ , the prefixes of size 3 of the model will be compared with the truncated log traces  $[\langle a \rangle, \langle a, b, c \rangle, \langle a, f, g \rangle, \langle a, b, i \rangle]$ .

Solving the problem of anti-alignment on prefixes dramatically improves efficiency, and already gives very relevant results in practice (see experimental Section 7.4).

Furthermore, the optimal prefix based anti-alignment brings a lower bound for model precision under the assumption that the model is a sound workflow net, i.e., every prefix can be completed to reach the final marking. The intuition is that the most misaligned full run that we can expect is the optimal anti-alignment found on the prefixes, completed with a sequence of perfectly misaligned actions. Following this intuition, one can deduce a lower bound on the precision for the full model and full log. This lower bound is quite tedious to compute in the general case but we illustrate it in a restricted case. We explain it in a simplified case of a model  $N$  and a log  $L$  where all log traces and full runs have the same length  $2n$ .

PROOF. Let  $\gamma'$  be an anti-alignment found for the prefixes of size  $n$ , i.e.  $\gamma'$  is a prefix of size  $n$  for  $N$ , maximizing the  $\text{dist}(\gamma', L')$  where  $L'$  is the truncated log. Let  $\sigma' \in L'$  achieving  $\text{dist}(\gamma', L')$ , such that  $\text{dist}(\gamma', L') = \text{dist}(\gamma', \sigma')$ . Consider now any full run  $\gamma$  of  $N$ . Let  $\gamma_1$  and  $\gamma_2$  be the prefix and suffix respectively of size  $n$  of  $\gamma$ , such that  $\gamma = \gamma_1 \cdot \gamma_2$  and  $|\gamma_1| = |\gamma_2| = n$ . By assumption, the distance between  $\gamma_1$  and the truncated log  $L'$  is less or equal to  $\text{dist}(\gamma', L')$ . This means that there exists a log trace  $\sigma \in L$  whose prefix  $\sigma_1$  of size  $n$  satisfies  $\#edits(\gamma_1, \sigma_1) \leq \#edits(\gamma', \sigma')$ . From this we deduce that

$$\#edits(\gamma, \sigma) \leq \#edits(\gamma', \sigma') + 2n$$

Indeed, one way to edit  $\gamma$  to  $\sigma$  is to delete all the suffix  $\gamma_2$  ( $n$  deletions), then edit  $\gamma_1$  to  $\sigma_1$  ( $\leq \#edits(\gamma', \sigma')$  actions), and finally complete  $\sigma_1$  to  $\sigma$  ( $n$  insertions).

It follows that

$$\text{dist}(\gamma, L) \leq \frac{\#\text{edits}(\gamma', \sigma') + 2n}{4n} = \frac{1}{2} \left( \frac{\#\text{edits}(\gamma', \sigma')}{2n} + 1 \right) = \frac{1}{2}(\text{dist}(\gamma', \sigma') + 1)$$

and

$$P_{aa}^\epsilon(N, L) \geq 1 - \frac{\text{dist}(\gamma', \sigma') + 1}{2(1 + \epsilon)^{2n}}.$$

□

Notice that we did not compute this lower bound in the experiment part because the general case (where traces and runs have various lengths) is tedious to obtain. In section 7.4, we give the prefix-based anti-alignments and the precision of those prefixes only, i.e., we reduce the problem to the set of prefixes. Definition 7 is then given by

$$P_{aa}^n(N, L, n) = 1 - \sup_{\gamma \in \text{Prefixes}(N, n)} \text{dist}(\gamma, \text{Truncate}(L, n))$$

where  $\text{Prefixes}(N, n)$  denotes the set of prefixes of length  $n$  for the process model  $N$ , and  $\text{Truncate}(L, n)$  denotes the log  $L$  with all traces truncated to length  $n$ . This precision on prefixes can already help one to analyze the *beginning* of a process model.

### 6.3.2. Maximal Number of Edits

The SAT encoding of the Levenshtein distance between two words  $u$  and  $v$ , presented in Section 5.1, uses boolean variables  $\delta_{i,j,d}$  for  $i = 0, \dots, |u|$ ,  $j = 0, \dots, |v|$  and  $d = 0, \dots, |u| + |v|$ . The maximal number of edits  $\text{max\_d}$  heuristic is an approximation of the SAT formula that aims at reducing the size of the formula by removing some clauses. Optimal formula of Levenshtein distance between words  $u$  and  $v$  requires boolean variables for  $|u| + |v|$  edits (parameter  $d$  of  $\delta_{i,j,d}$ ). The maximal number of edits heuristic uses variables  $\delta_{i,j,d}$  only with  $d$  smaller than some bound  $\text{max\_d} < |u| + |v|$ . This way, it reduces also the size of the formula and the computation time to solve it.

The heuristic precisely encodes the computation of the following value  $\text{dist}_{\text{max\_d}}(u, v)$  defined by slightly modifying the definition of Levenshtein's

distance (compare with the definition of  $dist(u, v)$  in Section 5.3):

$$\left\{ \begin{array}{l} dist_{max\_d}(\langle u_1, \dots, u_i \rangle, \epsilon) = \min(d, i) \\ dist_{max\_d}(\epsilon, \langle v_1, \dots, v_j \rangle) = \min(d, j) \\ dist_{max\_d}(\langle u_1, \dots, u_{i+1} \rangle, \langle v_1, \dots, v_{j+1} \rangle) = \\ \left\{ \begin{array}{ll} dist_{max\_d}(\langle u_1, \dots, u_i \rangle, \langle v_1, \dots, v_j \rangle) & \text{if } u_{i+1} = v_{j+1} \\ \min(d, 1 + \min(dist_{max\_d}(\langle u_1, \dots, u_{i+1} \rangle, \langle v_1, \dots, v_j \rangle), \\ dist_{max\_d}(\langle u_1, \dots, u_i \rangle, \langle v_1, \dots, v_{j+1} \rangle))) & \text{if } u_{i+1} \neq v_{j+1} \end{array} \right. \end{array} \right.$$

**Lemma 14.** *For every  $u$  and  $v$ ,  $dist_{max\_d}(u, v) \leq dist(u, v)$ . Moreover, if  $dist(u, v) \leq d$  then  $dist_{max\_d}(u, v) = dist(u, v)$ .*

PROOF. Both parts come naturally by strong induction on  $i + j$ . □

Hence, the maximal number of edits heuristic gives a lower bound of the Levenshtein distance. From Definition 7 of the precision metric  $P_{aa}^\epsilon$  defined by  $1 - \sup_{\gamma \in \mathcal{L}(N)} \frac{dist(\gamma, L)}{(1+\epsilon)^\gamma}$ , we conclude that the heuristic returns an upper-bound of the precision value based on  $L$ .

## 7. Tool Support and Experiments

In this section we present the tools implementing the results of this paper, and both a qualitative and quantitative evaluation on state-of-the-art benchmarks from the literature. To compare the different distances based results, we denoted Levenshtein distance based anti-alignment precision by  $P_{aa}^{L, \epsilon}$  and Hamming distance based anti-alignment precision by  $P_{aa}^{H, \epsilon}$ .

### 7.1. da4py: A Python Library Supporting Anti-Alignments

Several tools implement anti-alignments. **Darksider**, an Ocaml command line software, has already been presented in [30]. It creates the SAT formulas and calls the solver **Minisat+** [14] to get the result. **ProM** software [33] also has an anti-alignment plugin that computes anti-alignments in a brute force way. Recently, we have created a Python library in order to make our technique

Trace
$\langle A, B, D, E, I \rangle$
$\langle A, C, D, G, H, F, I \rangle$
$\langle A, C, G, D, H, F, I \rangle$
$\langle A, C, H, D, F, I \rangle$
$\langle A, C, D, H, F, I \rangle$

Table 1: An example event log.

more accessible: `da4py`<sup>8</sup>. Thanks to the use of the SAT library `PySAT` [16], `da4py` allows one to run different state-of-the-art SAT solvers. Moreover, this SAT library uses an implementation of the RC2 algorithm [15] in order to get MaxSAT solutions, a variant that improves a lot the efficiency of computing anti-alignments. Finally, `da4py` is compatible with the library `pm4py` [7], and uses the same data objects.

Remarkably, in order to deal with large logs (as the ones shown in the quantitative evaluation part), `da4py` has two variants of the precision metric. The first one, used in the first experiments, computes the exact precision as it is introduced in Definition 7 with a search of best anti-alignments (Algorithm 1). For the quantitative evaluations, Definitions 8 and  $P_{aa}^n$  precision are used, thus alleviating the complexity by not requiring a full run but only a prefix. Accordingly, the corresponding precision measure is then a variant, that is normalized by the size of the prefix.

Furthermore, for anti-alignments based on Levenshtein’s distance, the simplification to add a threshold on the number of edits (presented in Section 6.3.2) is also experimented below. We then discuss the limitations of this heuristic.

## 7.2. Qualitative Comparison of Precision Metrics

A set of examples are taken from page 64 of [23], and consist of the simple event log shown in Table 1 aligned with 10 different process models. The log consists of only five different traces, with various frequencies. The models in

<sup>8</sup><https://github.com/BoltMaud/da4py>, a Python version of `Darksider`

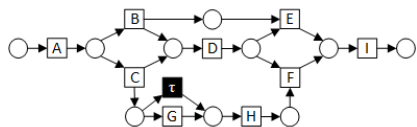


Figure 5: The ideal model. Fitting, fairly precise and properly generalizing.

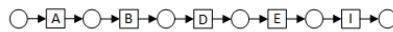


Figure 6: Most frequent trace. Precise, but not fitting or generalizing.

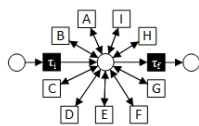


Figure 7: The flower model. Fitting and generalizing, but very imprecise.

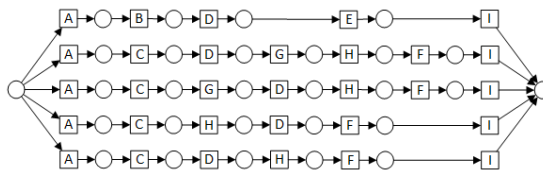


Figure 8: All traces separate. Fitting, precise, but not generalizing.

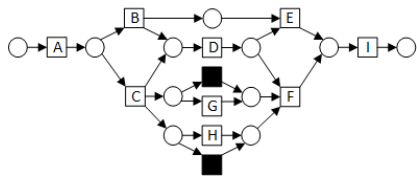


Figure 9: A model with G and H in parallel.

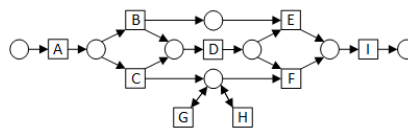


Figure 10: A model with G and H in self-loops

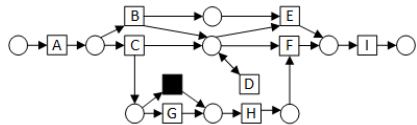


Figure 11: A model with D in a self-loop

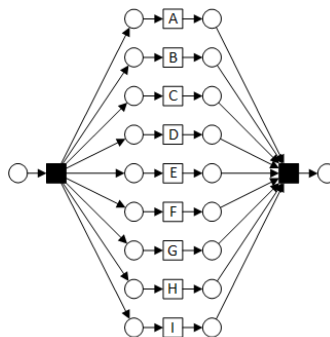


Figure 12: A model with all transitions in parallel.

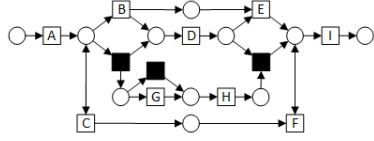


Figure 13: A model where C and F are in a loop, but need to be executed equally often to reach the final marking.

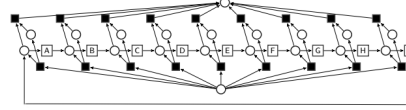


Figure 14: Modified Round-robin model (introduced in [30] and modified in [4]). The outer loop can be started at any point and then either exited one transition before completing the loop either finished the run.

Model	$P_{ET}$	$P_{ETC}$	$P_a$	$P_{n\epsilon}$	$MAP^3$	$MAP^7$	$P_{aa}^{H,.05}$	$P_{aa}^{L,.05}$	$P_{aa}^{L,.01}$
Fig. 5 Generating model	0.992	0.994	0.982	0.995	0.880	0.852	0.797	0.945	0.928
Fig. 6 Single trace	1.000	1.000	1.000	0.893	1.000	1.000	1.000	1.000	1.000
Fig. 7 Flower model	0.136	0.119	0.142	0.117	0.003	0.000	0.408	0.352	0.295
Fig. 8 Separate traces	1.000	0.359	1.000	0.985	1.000	1.000	1.000	1.000	1.000
Fig. 9 G,H in parallel	0.894	0.936	0.947	0.950	0.564	0.535	0.797	0.945	0.928
Fig. 10 G,H as self-loops	0.884	0.889	0.947	0.874	0.185	0.006	0.570	0.780	0.496
Fig. 11 D as self-loop	0.763	0.760	0.797	0.720	0.349	0.069	0.506	0.759	0.469
Fig. 12 All parallel	0.273	0.170	0.336	0.158	0.006	0.000	0.468	0.468	0.420
Fig. 13 C,F equal loop	0.820	0.589	0.839	0.600	–	–	0.751	0.881	0.845
Fig. 14 Round-robin	0.579	0.185	0.889	0.194	0.496	0.274	0.456	0.352	0.300

Table 2: Precision measures of artificial models

Figures 5 to 8 are four examples of models often used to show the differences between fitness, precision and generalization. The model in Figure 5 shows the “ideal” process discovery result, i.e. the model that is fitting, fairly precise and properly generalizing. The models in Figures 9 to 12 present the same set of activities with varying loop and/or parallel constructs. Two new process models that describe particularly different routing logic from the previous models are depicted in Figures 13 and 14.

Table 2 compares some precision metrics for the models in Figures 5 to 14 with the two possible metrics proposed in this paper:  $P_{aa}^{H,\epsilon}$  and  $P_{aa}^{L,\epsilon}$ , representing the precision for the Hamming and Levenshtein distance. The values of  $P_a$  are defined as in [1]. The precision values in  $P_{ET}$  and  $P_{ETC}$  are defined in

[2]. The value  $P_{ne}$  denotes the precision metric from [32]. Finally, the values  $MAP^3$  and  $MAP^7$  are defined in [4]<sup>9</sup>.

Clearly, the existing precision metrics do not agree on all models and do not always agree with the intuition for precision. While all methods rank the All-parallel model just after the Flower model, our AA-based precisions position the Round-Robin model in second place instead. This is due to the fact that our method found the anti-alignment  $\langle \tau, \tau \rangle$  which is far from the log traces. In fact, this run does not exist in the first version of this model in [30] for which the situation would not occur.

Figure 8, which consists of all the traces in separate paths, is considered to have a low precision by the  $P_{ETC}$  metric while all the other methods describe this model as perfectly precise.

We observe that our methods give higher precisions than the rest for the flower model. This is due to the requirement of the  $\tau$  transitions in order to get a full run of the model, thus alleviating the distance to the traces. A similar situation, with the same antidote, happens with the parallel model.

More generally, we note that our method, due to the parameter  $\epsilon$ , gives a good balance for models with loops. The precision of those models are very mixed. We see on one side  $P_{ET}$ ,  $P_{ETC}$ ,  $P_a$  and  $P_{ne}$  are from 0.947 to 0.720 and on the other side,  $MAP$  gives a precision close to zero. However, all the methods agree on the ranking of the loops compared to the parallel model.

Now, notice that model 5 and 9 are considered to be equally precise with the anti-alignment precision, which is not the case for the other metrics. Model 9 contains the behaviors of model 5 plus some permutation and skip activities between  $G$  and  $H$ , thus adding runs in the models. The fact that our method finds the same precision is a consequence of the input log. Model precision is defined in accordance to the log. In this case, there is not anti-alignment of Fig. 9 that is further to the log traces than the anti-alignment of Fig. 5, despite

---

<sup>9</sup>Notice that the metrics  $MAP^3$  and  $MAP^7$  are not applicable for one of the benchmarks of this paper, due to the existence of unbounded constructs.



Model	Size of Prefix	Hamming distance Anti-alignments	Edit distance Anti-alignments	$P_{aa}^{H./n}$	$P_{aa}^{L./n}$	
Fig. 5	Generating model	5	$\langle A, C, G, H, D \rangle$	$\langle A, C, G, H, D \rangle$	0.600	0.800
		10	$\langle A, C, G, H, D, F, I \rangle$	$\langle A, C, G, H, D, F, I \rangle$	0.714	0.923
		15	$\langle A, C, G, H, D, F, I \rangle$	$\langle A, C, G, H, D, F, I \rangle$	0.714	0.923
Fig. 6	Single	5	$\langle A, B, D, E, I \rangle$	$\langle A, B, D, E, I \rangle$	1.000	1.000
		10	$\langle A, B, D, E, I \rangle$	$\langle A, B, D, E, I \rangle$	1.000	1.000
		15	$\langle A, B, D, E, I \rangle$	$\langle A, B, D, E, I \rangle$	1.000	1.000
Fig. 7	Flower model	5	$\langle \tau, G, F, B, C \rangle$	$\langle \tau, G, G, E, B \rangle$	0.200	0.300
		10	$\langle \tau, C, I, C, C, C, C, F, E, I \rangle$	$\langle \tau, G, G, G, G, G, G, G, G, G \rangle$	0.100	0.176
		15	$\langle \tau, D, E, B, B, B, B, E, E, B, H, B, B, C \rangle$	$\langle \tau, G, G, G, G, G, G, G, G, G, G, G, G, G, G \rangle$	0.067	0.136
Fig. 8	Separate traces	5	$\langle A, B, D, E, I \rangle$	$\langle A, C, D, G, H, F, I \rangle$	1.000	1.000
		10	$\langle A, B, D, E, I \rangle$	$\langle A, C, D, G, H, F, I \rangle$	1.000	1.000
		15	$\langle A, B, D, E, I \rangle$	$\langle A, C, D, G, H, F, I \rangle$	1.000	1.000
Fig. 9	G,H in parallel	5	$\langle A, C, H, G, D \rangle$	$\langle A, C, \tau, D, \tau \rangle$	0.600	0.800
		10	$\langle A, C, \tau, H, D, F, I \rangle$	$\langle A, C, D, \tau, \tau, F, I \rangle$	0.714	0.923
		15	$\langle A, C, G, H, D, F, I \rangle$	$\langle A, C, \tau, D, G, F, I \rangle$	0.714	0.929
Fig. 10	G,H as self-loops	5	$\langle A, C, D, F, I \rangle$	$\langle A, C, G, G, G \rangle$	0.600	0.600
		10	$\langle A, C, H, H, D, H, H, G, H, H \rangle$	$\langle A, C, G, G, G, G, G, G, G, G, G, G \rangle$	0.300	0.353
		15	$\langle A, C, H, H, G, D, G, H, G, H, G, G, G, G \rangle$	$\langle A, C, G, G, G, G, G, G, G, G, G, G, G, G, G, G \rangle$	0.200	0.272
Fig. 11	D as self-loop	5	$\langle A, B, E, I \rangle$	$\langle A, B, D, D, D \rangle$	0.400	0.600
		10	$\langle A, B, D, D, D, D, D, D, D, E, I \rangle$	$\langle A, C, D, D, D, D, D, D, D, D, D, D \rangle$	0.300	0.375
		15	$\langle A, B, D, D, D, D, D, D, D, D, D, D, D, D, E, I \rangle$	$\langle A, C, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D \rangle$	0.200	0.286
Fig. 12	All parallel	5	$\langle \tau, G, E, I, C \rangle$	$\langle \tau, G, C, E, B \rangle$	0.200	0.300
		10	$\langle \tau, I, F, B, G, A, E, H, C, D \rangle$	$\langle \tau, D, I, F, B, H, A, G, C, E \rangle$	0.100	0.313
		15	$\langle \tau, G, B, A, D, E, F, H, I, C, \tau \rangle$	$\langle \tau, F, H, G, C, I, E, D, B, A, \tau \rangle$	0.091	0.353
Fig. 13	C,F equal loop	5	$\langle A, \tau, \tau, H, D \rangle$	$\langle A, C, \tau, D, \tau \rangle$	0.400	0.600
		10	$\langle A, C, B, D, E, F, I \rangle$	$\langle A, C, B, D, E, F, I \rangle$	0.714	0.833
		15	$\langle A, C, B, D, E, F, I \rangle$	$\langle A, C, B, D, E, F, I \rangle$	0.667	0.833
Fig. 14	Round-robin	5	$\langle \tau, I, A, B, C \rangle$	$\langle \tau, D, E, F, G \rangle$	0.200	0.500
		10	$\langle \tau, I, A, B, C, D, E, F, G, H \rangle$	$\langle \tau, E, F, G, H, I, A, B, C, D \rangle$	0.100	0.529
		15	$\langle \tau, A, B, C, D, E, F, G, H, I, A, B, C, D, E \rangle$	$\langle \tau, B, C, D, E, F, G, H, I, A, B, C, D, E, F \rangle$	0.067	0.500

Table 3: Anti-alignments prefix of size 5, 10 and 15 for all artificial models

their are more anti-alignments.

Finally, we would like to point out the practical aspect of the parameter  $\epsilon$ . See precisions of Figure 10 for  $\epsilon = 0.05$  and  $\epsilon = 0.01$ . The score considerably drops from  $P_{aa}^{L,0.05} = 0.780$  to  $P_{aa}^{L,0.01} = 0.496$ . For large  $\epsilon$  values, traces with a thousand G's is possible in the model and our anti-alignment can penalizes it a lot by allowing long runs.

### 7.3. Qualitative Experiment on Prefix Heuristic

To deal with large logs, our library has a variant of precision that works on a fixed size of prefix. In this sub-section, we show and discuss the consequences of using only a prefix of anti-alignment.

Table 3 figures, for the artificial models, prefix anti-alignment of a size of prefix to 5, 10 and 15 and the corresponding adapted precision  $P_{aa}^{/n}$  given in

section 6.3.1. With the use of prefix only, the method does not require full runs. This can be notified for models with loops for which anti-alignments do not reach the final marking. As the size of the run is fixed, parameter  $\epsilon$ , used to limit the size of the run, is not required and corresponds to  $\epsilon = 0$ .

In most cases, results for distance to 10 and 15 are similar. However, we remark two types of differences. When using a small size of prefix, a model with a loop is qualified as more precise than when the size of prefix is large, thus alleviating the use of the loop. For well precise model, reducing the size of the prefix can induce a decrease of the metric. For instance, for Fig. 5 the transitions  $F$  and  $I$  are well presented in the traces and do not raise the distances, thus raising the denominator of the normalization of the precision.

The comparison between Hamming and Levenshtein distances interestingly shows some cases where anti-alignments are equal or very similar, which validates the use of the former (whose SAT encoding is simpler) as alternative for the latter. Also notice that, for a same size of run,  $P_{aa}^{H./n} \leq P_{aa}^{L./n}$  as a consequence of Lemma 1.

Finally, we invite readers to compare anti-alignment prefix based precision (Table 3) to exact anti-alignment precision (Table 2). We see that for most models, approximated precision tends to be close to optimal one.

As a conclusion of this section, we motivate users to use the prefix based method, which is more time efficient. A preliminary study of the length of full runs of the models can help one to set the size of the prefix.

#### 7.4. Qualitative Experiment for Maximal Number of Edits Heuristic

A second heuristic, used for large logs, is to reduce the maximal number of edits  $max\_d$  of the Levenshtein distance in order to considerably reduce the size of the SAT formula (Section 6.3.1).

We present in Table 4 a concrete example of the consequences of this parameter. For log of Table 1, model of Fig. 10 and a prefix of run to 10, the maximal number of edits is 17 (the maximal length in log is 7). The first row of Table 4 shows the exact anti-alignment which is at distance 11 to any log

Parameter $max\_d$ , Threshold on the Number of Edits	Returned Anti-alignment	Returned Threshold Lower Bound Distance to Log	Correct Minimal Distance to Log	Precision
17	$\langle A, C, G, G, G, G, G, G, G, G \rangle$	11	11	0.353
5	$\langle A, C, G, G, G, G, G, G, G, G \rangle$	5	11	0.353
5	$\langle A, C, D, G, G, G, G, H, H \rangle$	5	7	0.588
5	$\langle A, C, D, H, G, H, G, G, G, F \rangle$	5	5	0.706

Table 4: Consequences of the Threshold on Number of Edits of Levenshtein Distance for log of Table 1, model of Fig. 10 and a prefix of run to 10. The last line is forced to reach the final marking.

trace. When we reduce the number of edits, the tool might still find the same anti-alignment. Indeed for next row of the table,  $max\_d$  to 5, the tool finds anti-alignment  $\langle A, C, G, G, G, G, G, G, G, G \rangle$  which is indeed *at least* at distance 5 to any log trace. In fact, this anti-alignment is optimal and it is at distance 11, but has been returned by chance. In fact, for  $max\_d$  to 5, the tool also returned the two last lines of the table where the optimal anti-alignment has not been found. For them, the correct distance to the log is not 11 which implies a difference in precision.

### 7.5. Quantitative Comparison

In this section, we evaluate the techniques of this paper for 12 available real-life logs of varying sizes, and which have been recently used for benchmarking process discovery methods [6] presented in Table 5. They cover different fields, ranging from finance through to healthcare and government. The logs are publicly available at the 4DTU Data Center<sup>10</sup>. The demonstrations have been executed on a virtual machine with Debian 3.16.43-2+deb8u2 system, 12 CPU Intel Xeon 2.67GHz and 50GB RAM.

#### 7.5.1. Benchmark Description

For each of the twelve logs, we compute precision for the models automatically discovered with two state-of-the-art discovery algorithms: Inductive Miner

<sup>10</sup>[https://data.4tu.nl/repository/collection:event\\_logs\\_real](https://data.4tu.nl/repository/collection:event_logs_real)

Log	Total Traces	Trace Variants	Events	Activites
<i>BPIC</i> <sub>2012</sub>	13 087	4366	262 200	36
<i>BPIC</i> <sub>2013<sub>cp</sub></sub>	1 487	183	6 660	7
<i>BPIC</i> <sub>2013<sub>inc</sub></sub>	7 554	1511	65 533	13
<i>BPIC</i> <sub>2014<sub>f</sub></sub>	41 353	14 948	369485	9
<i>BPIC</i> <sub>2015<sub>1f</sub></sub>	902	295	21 656	70
<i>BPIC</i> <sub>2015<sub>2f</sub></sub>	681	420	24 678	82
<i>BPIC</i> <sub>2015<sub>3f</sub></sub>	1 369	826	43 786	62
<i>BPIC</i> <sub>2015<sub>4f</sub></sub>	860	451	29 403	65
<i>BPIC</i> <sub>2015<sub>5f</sub></sub>	975	446	30 030	74
<i>BPIC</i> <sub>2017<sub>f</sub></sub>	21 861	8 767	714 198	41
<i>RTFMP</i>	150 370	231	561 470	11
<i>SEPSIS</i>	1 050	846	15 214	16

Table 5: Quantitative Benchmark Descriptions of the 12 real-life Logs

[19], and Split Miner [5]. We compare the performance of computing our metric against the one required for the *MAP* metric [4], which is known to be the most recent precision contribution, and which as in our case, satisfies the axioms described in [26].

Given that our methods are known to incur into a high complexity (specially in terms of memory footprint), we use samples of size 10 and 100 of the initial log. We present charts of runtimes in Fig. 15 and Fig. 16. The notation  $P_{aa}^{/n=10}$  indicates that we use the simpler prefix variant of anti-alignments (see Section 7.4). As explained in Section 6.3.1, a second parameter, *max\_d*, helps to reduce complexity of the SAT formula by bounding the number of maximal differences: in all but the precision using the Levenshtein distance and 100 traces, we use *max\_d* = 20 which is optimal for a prefix of size 10. For 100 traces and the Levenshtein distance, we set *max\_d* to 10, due to the computational requirements of that last set of benchmarks.

### 7.5.2. Discussion

By relying on approximations of anti-alignments (with prefixes instead of full runs, but also by limiting the maximal number of differences), our approach tends to run in reasonable time. We compare our method against *MAP*<sup>3</sup> which is also behavioral, satisfies the axioms and is also biased towards a pessimistic

positioning of precision<sup>11</sup>. Also, in [4] the method has shown good performances. From Fig. 15, we see that neither the size of the model in term of nodes, neither the log size is correlated to runtimes of *MAP*<sup>3</sup>. In opposite, and as expected, our methods fully rely on the size of the models. The complexity of computing the Levenshtein based anti-alignment is very well highlighted by the two Figures 15 and 16. Moreover the charts show that Hamming based precision, which gives an approximation of Levenshtein based precision, is often the quicker approach. Despite that method of [4] is quite insensitive to the log size, which admittedly is not the case of our approach, we remark a difference between runtime of *MAP*<sup>3</sup> for when process models are learned with the Inductive miner and the Split mine. This measure is sensitive to the state-space and the Inductive miner creates many choice structure. The peak that appears in the charts of Fig. 15 corresponds to a model that contains a choice between 30 transitions. In our metric, this structure is encoded in a SAT clause and does not influence the runtime compare to other aspect like run size. Finally, it should be stressed that several engineering efforts can be incorporated into our current tool, to drastically reduce part of its computational demands. But we see that for the first time, our tool can deal with larger instances, when it is compared to the previous implementations.

## 8. Related Work

The seminal work in [24] was the first one in relating observed behavior (in form of a set of traces), and a process model. In order to asses how far can the model deviate from the log, the *follows* and *precedes* relations for both model and log are computed, storing for each relation whereas it *always* holds or only *sometimes*. In case of the former, it means that there is more variability. Then, log and model follows/precedes matrices are compared, and in those matrix cells where the model has a *sometimes* relation whilst the log has an *always* relation

---

<sup>11</sup>The selection of  $k = 3$  was advised by the authors of [4] in a personal communication.

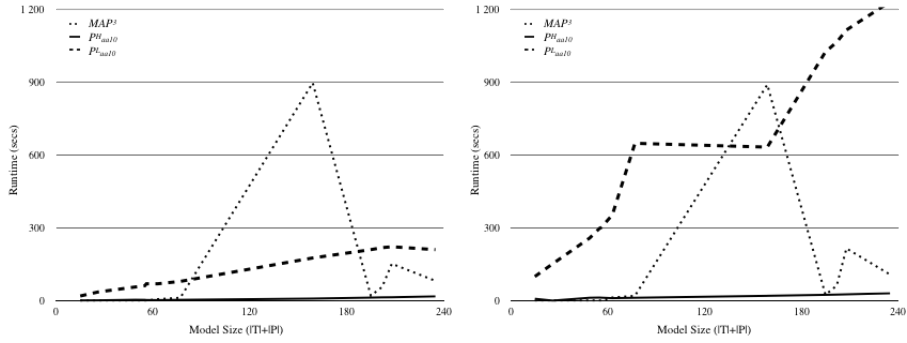


Figure 15: Runtimes Comparison of  $MAP^3$ ,  $P_{aa}^{H, /n=10}$  and  $P_{aa}^{L, /n=10}$  for 10 traces (left) and 100 traces (right) of Models discovered using Inductive Miner for the 12 Real-life Logs. For 100 traces,  $P_{aa}^{L, /n=10}$  is given for  $max\_d = 10$ .

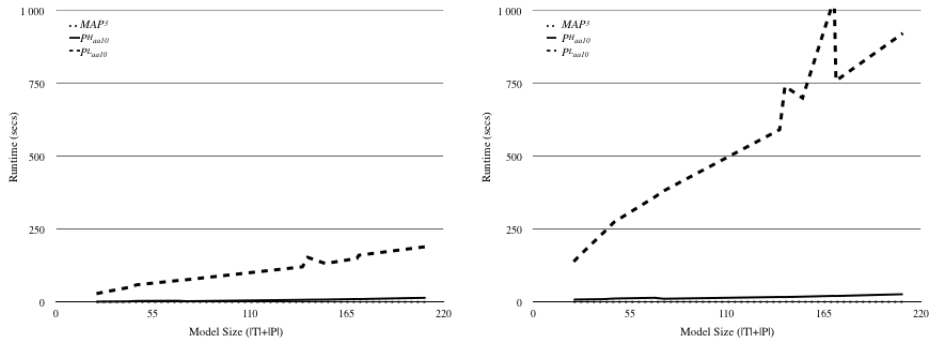


Figure 16: Runtimes Comparison of  $MAP^3$ ,  $P_{aa}^{H, /n=10}$  and  $P_{aa}^{L, /n=10}$  for 10 traces (left) and 100 traces (right) of Models discovered using Split Miner for the 12 Real-life Logs. For 100 traces,  $P_{aa}^{L, /n=10}$  is given  $max\_d = 10$ .

indicate that the model allows for more behavior, i.e., a lack of precision. This technique has important drawbacks: first, it is not general since in the presence of loops in the model the characterization of the relations is not accurate [24]. Second, the method requires a full state-space exploration of the model in order to compute the relations, a stringent limitation for models with large or even infinite state spaces.

In order to overcome the limitations of the aforementioned technique, a different approach was proposed in [20]. The idea is to find *escaping arcs*, denoting those situations where the model starts to deviate from the log behavior, i.e., events allowed by the model not observed in the corresponding trace in the log. The exploration of escaping arcs is restricted by the log behavior, and hence the complexity of the method is always bounded. By counting how many escaping arcs a pair (model, log) has, one can estimate the precision of a model. Although being a sound estimation for the precision metric, it may hide the problems we are considering in this paper, i.e., models containing escaping arcs that lead to a large behavior.

Less related is the work in [32], where the introduction of *weighted artificial negative events* from a log is proposed. Given a log  $L$ , an artificial negative event is a trace  $\sigma' = \sigma \cdot a$  where  $\sigma \in L$ , but  $\sigma' \notin L$ . Algorithms are proposed to weight the confidence of an artificial negative event, and they can be used to estimate the precision and generalization of a process model [32]. Like in [20], by only considering one step ahead of log/model's behavior, this technique may not catch serious precision/generalization problems.

Finally, an automata-based technique is presented in [18]. The technique projects event log and process model onto all subsets of activities of size  $k$ , and generates minimal deterministic finite automata that conjunctively represents both perspectives, so that precision can be estimated by confronting the model and the conjunction automata.

As we mentioned in Section 6.2, a recent study elaborated on the guarantees provided by the aforementioned precision metrics [26]. The study shows that none of the precision metrics up to that moment (including an earlier version

of the metrics described in this paper [30]) was able to satisfy all the properties together. Recently, however, a metric was proposed in [4] which satisfies the 5 axioms. Likewise to the metric in [4], the new precision metric proposed in this paper is shown to satisfy either these properties, or a weaker version of them. The metric in [4] relies on behavioral abstractions which are based on refinements of the directly-follows graph [19]. Due to this, they can dramatically overestimate the behavior of the model, thus have a tendency to be pessimistic. For instance, for well chosen values of  $k$ ,  $MAP^k$  would use the same abstraction for two models  $N_1$  and  $N_2$  having languages  $\mathcal{L}(N_1) = a^*b^3c^*|d^*b^3e^*$  and  $\mathcal{L}(N_2) = (a^*|d^*)b^3(c^*|e^*)$  and assign them the same precision. Intuitively, this is not satisfactory: assume that the log contains only traces in  $\mathcal{L}(N_1)$ , then model  $N_1$  should score much better than  $N_2$ .

#### *The Hidden Bias of Anti-Alignment Based Precision*

To conclude this section on related work, we would like to emphasize that, as it happens with some of the aforementioned metrics of this section, anti-alignment based precision checking presented in this paper may provide unintuitive values for certain, extreme case, situations. Let us use a very simple example to illustrate this.

Consider a model that has two main parts, that are in conflict: the first part is the single activity  $a$ , whilst the second part is a set of  $n$  concurrent activities, all different from  $a$ . Moreover, consider a log that consists of all the traces corresponding to the interleaving of the  $n$  concurrent activities of the second part of the model. Anti-alignment  $\langle a \rangle$  is then at normalized distance 1 to any log trace (the normalized distance counts edits in the run, and edits in the trace and normalizes it by the length of both). This induces a precision of the model to be 0. However, when removing the transition labeled by  $a$  in the process model, the precision of the new model raises from 0 to 1. Hence, on this regard, the precision metric arising from anti-alignments can be sometimes too pessimistic (if one thinks on the initial model, a slight edit has made it to change completely its precision). We will consider in the future ways for



compensating this pessimistic bias, perhaps by incorporating the number of available anti-alignments in the metric.

## 9. Conclusions and Future Work

Conformance checking is becoming an important tool to certify the correct execution of processes in organizations. Quality metrics for conformance checking are crucial to evaluate quantitatively process models, and among the four dimensions to attain this task, precision is an important one. This paper proposes anti-alignments as a crucial artefact to compute an accurate metric for precision. In contrast to existing metrics, anti-alignment based precision metrics satisfy most of the properties for precision described in [26].

As future work, we plan to explore new algorithms to compute anti-alignments so that the complexity of the problem can be alleviated in practice. Also, we will consider new areas for the application of anti-alignments, like process model comparison or novel ways of computing the generalization of process models (we already suggested one such approach in a recent paper [30]).

We see potential for other notions of anti-alignments, for instance anti-alignments can be generalized between two nets: to find behavior of the former that cannot be found in the latter. The application of this generalization goes beyond the field of process mining: it can be used, for instance, to allow generating behavior that is not possible in a model, by simply aligning the net that allows for all the possible behaviors with the original net. This way, one can compute the most deviating trace of a net with respect to its own behavior, up to a given length. A second application of anti-alignments between two nets is for assessing behavioral process model similarity: this way, example behavior that is in one model and not in the other can be provided as a hint on the dissimilarity between two models. Previous techniques focus on the differences on either the causal relations, or those parts of the models that are different [3, 13, 22, 34, 35].

*Acknowledgments.*

This work has been partially supported by funds from ENS Paris-Saclay (project ProMAut of the Farman institute), the Spanish Ministry for Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant GRAMM (TIN2017-86727-C2-1-R).

- [1] Adriansyah, A. (2014). *Aligning observed and modeled behavior*. PhD thesis, Technische Universiteit Eindhoven.
- [2] Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B. F., and van der Aalst, W. M. P. (2015). Measuring precision of modeled behavior. *Inf. Syst. E-Business Management*, 13(1):37–67.
- [3] Armas-Cervantes, A., Baldan, P., Marlon, D., and García-Bañuelos, L. (2014). Behavioral comparison of process models based on canonically reduced event structures. *Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, pages 267–282.
- [4] Augusto, A., Armas-Cervantes, A., Conforti, R., Dumas, M., Rosa, M. L., and Reißner, D. (2018). Abstract-and-compare: A family of scalable precision measures for automated process discovery. In *Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, pages 158–175.
- [5] Augusto, A., Conforti, R., Dumas, M., and La Rosa, M. (2017). Split miner: Discovering accurate and simple business process models from event logs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1–10. IEEE.
- [6] Augusto, A., Conforti, R., Dumas, M., Rosa, M. L., Maggi, F. M., Marrella, A., Mecella, M., and Soo, A. (2019). Automated discovery of process models from event logs: Review and benchmark. *IEEE Trans. Knowl. Data Eng.*, 31(4):686–705.

- [7] Berti, A., van Zelst, S. J., and van der Aalst, W. (2019). Process Mining for Python (PM4Py): Bridging the Gap Between Process-and Data Science. page 13–16.
- [8] Buijs, J. C. A. M., van Dongen, B. F., and van der Aalst, W. M. P. (2014). Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1).
- [9] Carmona, J., van Dongen, B. F., Solti, A., and Weidlich, M. (2018). *Conformance Checking - Relating Processes and Models*. Springer.
- [10] Chatain, T. and Carmona, J. (2016). Anti-alignments in conformance checking - the dark side of process models. In *Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings*, pages 240–258.
- [11] Cheng, A., Esparza, J., and Palsberg, J. (1995). Complexity results for 1-safe nets. *Theor. Comput. Sci.*, 147(1&2):117–136.
- [12] Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., and Mazowiecki, F. (2019). The reachability problem for petri nets is not elementary. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 24–33. ACM.
- [13] Dijkman, R. M. (2008). Diagnosing differences between business process models. In *BPM 2008, Milan, Italy, September 2-4*, pages 261–277.
- [14] Eén, N. and Sörensson, N. (2003). An extensible sat-solver. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, pages 502–518.
- [15] Ignatiev, A., Morgado, A., and Marques-Silva, J. Rc2: a python-based maxsat solver. *MaxSAT Evaluation 2018*, page 22.

- [16] Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018). PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437.
- [17] Krötzsch, M., Masopust, T., and Thomazo, M. (2017). Complexity of universality and related problems for partially ordered NFAs. *Information and Computation*, 255:177 – 192.
- [18] Leemans, S., Fahland, D., and van der Aalst, W. (2016). Scalable process discovery and conformance checking. *Software and Systems Modeling*, pages 1–33.
- [19] Leemans, S. J., Fahland, D., and van der Aalst, W. M. (2013). Discovering block-structured process models from event logs—a constructive approach. In *International conference on applications and theory of Petri nets and concurrency*, pages 311–329. Springer.
- [20] Munoz-Gama, J. (2014). *Conformance Checking and Diagnosis in Process Mining*. PhD thesis, Universitat Politècnica de Catalunya.
- [21] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–574.
- [22] Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M. L., and ter Hofstede, A. H. M. (2014). The 4c spectrum of fundamental behavioral relations for concurrent systems. In *PETRI NETS 2014, Tunis, Tunisia, June 23-27*, pages 210–232.
- [23] Rozinat, A. (2010). *Process Mining: Conformance and Extension*. PhD thesis, Technische Universiteit Eindhoven.
- [24] Rozinat, A. and van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95.
- [25] Stewart, I. A. (1995). Reachability in some classes of acyclic Petri nets. *Fundam. Inform.*, 23(1):91–100.

- [26] Tax, N., Lu, X., Sidorova, N., Fahland, D., and van der Aalst, W. M. (2018). The imprecisions of precision measures in process mining. *Information Processing Letters*, 135:1 – 8.
- [27] Tseytin, G. S. (1970). On the complexity of derivation in propositional calculus. In Slisenko, A., editor, *Studies in Constructive Mathematics and Mathematical Logic, Part II*, Seminars in Mathematics, page 115–125. Steklov Mathematical Institute. Translated from Russian: Zapiski Nauchnykh Seminarov LOMI 8 (1968), pp. 234–259.
- [28] van der Aalst, W. M. P. (2016). *Process Mining - Data Science in Action, Second Edition*. Springer.
- [29] van der Aalst, W. M. P., Kalenkova, A., Rubin, V., and Verbeek, E. (2015). Process discovery using localized events. In *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*, pages 287–308.
- [30] van Dongen, B. F., Carmona, J., and Chatain, T. (2016). A unified approach for measuring precision and generalization based on anti-alignments. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 39–56.
- [31] vanden Broucke, S. K. L. M., Munoz-Gama, J., Carmona, J., Baesens, B., and Vanthienen, J. (2014a). Event-based real-time decomposed conformance analysis. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings*, pages 345–363.
- [32] vanden Broucke, S. K. L. M., Weerdt, J. D., Vanthienen, J., and Baesens, B. (2014b). Determining process model precision and generalization with weighted artificial negative events. *IEEE Trans. Knowl. Data Eng.*, 26(8):1877–1889.

- [33] Verbeek, H., Buijs, J., Van Dongen, B., and van der Aalst, W. M. (2010). Prom 6: The process mining toolkit. *Proc. of BPM Demonstration Track*, 615:34–39.
- [34] Weidlich, M., Mendling, J., and Weske, M. (2011a). Efficient consistency measurement based on behavioral profiles of process models. *IEEE Tr. Soft. Eng.*, 37(3):410–429.
- [35] Weidlich, M., Polyvyanyy, A., Mendling, J., and Weske, M. (2011b). Causal behavioural profiles - efficient computation, applications, and evaluation. *Fundam. Inform.*, 113(3-4):399–435.
- [36] Yang, H., van Dongen, B. F., her Hofstede, A. H. M., Wynn, M. T., and Wang, J. (2012). Estimating completeness of event logs. Technical Report BPM-12-04, BPM Center.