Incorporating Negative Information in Process Discovery

Hernan Ponce-de-León[1], Josep Carmona[2], and Seppe K.L.M vanden Broucke[3]

[1] Helsinki Institute for Information Technology HIIT and Department of Computer Science and Engineering, School of Science, Aalto University, Finland
hernan.poncedeleon@aalto.fi

[2] Universitat Politecnica de Catalunya, Barcelona, Spain
jcarmona@cs.upc.edu

[3] Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Leuven, Belgium
seppe.vandenbroucke@kuleuven.be

Abstract. The discovery of a formal process model from event logs describing real process executions is a challenging problem that has been studied from several angles. Most of the contributions consider the extraction of a model as a *semi-supervised* problem where only positive information is available. In this paper we present a fresh look at process discovery where also negative information can be taken into account. This feature may be crucial for deriving process models which are not only simple, fitting and precise, but also good on generalizing the right behavior underlying an event log. The technique is based on numerical abstract domains and Satisfiability Modulo Theories (SMT), and can be combined with any process discovery technique. As an example, we show in detail how to supervise a recent technique that uses numerical abstract domains. Experiments performed in our prototype implementation show the effectiveness of the techniques and the ability to improve the results produced by selected discovery techniques.

1 Introduction

The digital revolution that is taking place in the last decade is abruptly changing the way organizations, industry and people access, store and analyze the vast amount of digital information currently available. The challenge is to be able to extract value from this information in an effective way. In the context of information systems and business process management, where processes are responsible for the correct undertaking of system functionalities, end-users desire to extract process-oriented aspects that can contribute to a better understanding of the process perspective of the reality observed.

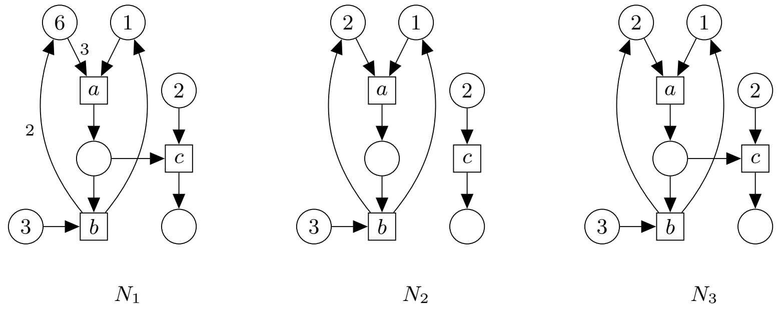
Process Mining is considered to be a viable solution to this problem: by using the *event logs* containing the footprints of real process-executions, process mining techniques aim at discovering, analyzing and extending formal process models revealing the real processes in a system [1]. From its arising around a decade ago, the process mining field has evolved into several directions, with process discovery perhaps being the most difficult challenge demonstrated by the large amount of techniques available nowadays. What makes process discovery hard is the fact that derived process models are expected to be good in four quality dimensions which often are opposed: *fitness* (ability of the model to reproduce the traces in the event log), *precision* (how precise is the model in representing the behavior in the log), *generalization* (is the model able to generalize for behavior not in the log) and *simplicity* (the well-known *Occam's Razor* principle).

Process discovery is a *learning* technique: a set of training examples (traces denoting process executions) are used to derive a process model which encloses the behavior underlying in the training set. Most techniques that have been proposed for process discovery so far assume a positive label in each trace, i.e. the example is an instance of behavior that must be in the process model to be derived. A slight extension of this assumption may be obtained if extra information is considered that enables to weight different traces: for instance, if the frequency of a trace is also considered, there exist some techniques that are able to extract only the most frequent patterns into a process model [2,3].

In literature, very few techniques have been presented that consider the discovery problem as a *supervised learning* task, i.e. using both the real process executions as positive examples, but also incorporating negative examples, that is, traces representing behavior that cannot be executed in the underlying system and should hence not be present in the process model to be derived. Such information might be crucial to derive the right model [4,5,6,7,8]. Clearly, the use of negative information can bring significant benefits, e.g. enable a controlled generalization of a process model: the patterns to generalize should never include the negative behavior. Another benefit is the ability to simplify a model on those parts that do not contribute to differentiate between positive and negative examples. The existence of few techniques for supervised process discovery is due to the fact that most real-life event logs do not provide easy ready-to-use negative examples.

This paper proposes a *novel methodology for supervised process discovery*, and shows how this technique can be adapted to be used in combination with arbitrary process discovery methods. The two main techniques combined to this end are numerical abstract domains [9] and Satisfiability Modulo Theories [10].

We ground the supervisory approach on the duality between the marking equation of a Petri net and the domain of convex polyhedra which has been already exploited for process discovery [11] and which we summarize now informally. The idea of [11] is to transform the traces in the log into points of an n-dimensional space (where n is the number of different activities in the log) and then to find a convex envelope of these points representing the concept to learn. The domain of *convex polyhedra* is used as it is a good compromise between expressivity and complexity of the operations [12]. The final step is to convert the convex polyhedron into a process model (a Petri net [13]) by extracting half-spaces of the polyhedron and transforming them into Petri net places that restrict behavior in the derived Petri net. Remarkably, this approach is among the few ones that can discover the full class of pure P/T-nets, i.e. Petri nets with arbitrary arc weights and tokens. This aspect makes the approach well suited for domains like manufacturing, where the flow relation between activities may be non-unitary. Most of the techniques in the literature do not aim for such a general class of process models.



$$\mathcal{L}^+ = \left\{ \begin{array}{c} a \cdot c \\ a \cdot b \cdot a \cdot c \\ a \cdot b \cdot a \cdot b \cdot a \cdot c \\ a \cdot b \cdot a \cdot b \cdot a \cdot b \cdot a \cdot c \end{array} \right\} \qquad \mathcal{L}_- = \left\{ \begin{array}{c} c \\ a \cdot c \cdot c \end{array} \right\}$$

Fig. 1: Three process models to illustrate supervised process discovery.

The technique presented in [11] suffers from two main limitations. First, since it is tailored to P/T nets, the number of half-spaces describing the concept learned are often large and complex which significantly hampers the practical use of the corresponding model to understand the underlying process. Among the problems, we highlight deriving overfitting and/or *spaghetti* models as the most stringent ones. Second, the technique follows a semi-supervised paradigm, i.e. only positive instances of a process are considered. This implies that the model obtained may be accepting behavior that is against the expected functioning of the process represented; in other words: the model is imprecise.

We extend the technique from above by an extra simplification step on the polyhedron before transforming it into a net. The restrictions on the polyhedron can be relaxed as far as they preserve the initial solutions, i.e. the positive traces. Additionally, negative information can be encoded as negative points which must be not enclosed by the polyhedron and thus preventing some of the problems from [11]. This step is automated with the help of SMT instances that enable the rotation and shifting of the polyhedron.

Example 1. Consider the three models of Fig. 1 and the logs $\mathcal{L}^+$ and $\mathcal{L}_-$ representing respectively the observed and the undesired behavior of the system. The model on the left (N_1) represents a system where an action c can only be fired once and when it is preceded by action a[4]. N_1 can replay all the traces in $\mathcal{L}^+$, but not those in $\mathcal{L}_-$; we can conclude that it is fitting, precise and generalizes well the intended behavior. N_2 is also fitting, but it is too general since it accepts some of the undesired behaviors in $\mathcal{L}_-$, e.g. action c can be fired independently of the firing of a. Using the approach from [11] both nets could be discovered, but the structure of the latter is simpler (it has less arcs and smaller weights). The problem with the simplification from N_1 into N_2 is that it introduces undesired behaviors as commented previously. With the contributions of this paper

[4] Notice that there is a safe Petri net which includes $\mathcal{L}^+$ and excludes $\mathcal{L}_-$: we are using the unsafe models in Fig. 1 just as an illustrative example.

net N_3 can be discovered, which is fitting, precise, does not accept any undesired behavior and it is still simpler than N_1.

The remainder of this paper is organized as follows: Section 2 introduces all the necessary background to understand the contribution of this paper. Then in Section 3 the approach for supervised process discovery is presented. A small discussion in Section 4 is devoted to decouple the methods of this paper from the particular discovery technique used. The approach is evaluated in Section 5, and compared with related work in Section 6. Section 7 concludes.

2 Preliminaries

In this section we introduce some basic definitions and ideas used in the subsequent sections.

2.1 Parikh Representation of an Event Log

The behavior of a process is observed as sequences of events from a given alphabet. For convenience, we use T to denote the set of symbols that represent the alphabet of events. A trace is a word $\sigma \in T^*$ that represents a finite sequence of events; $|\sigma|_a$ represents the number of occurrences of a in σ.

A *log* $\mathcal{L}$ is a set of traces from a given alphabet. We say that $\sigma \in \mathcal{L}$ if σ is the prefix of some trace of $\mathcal{L}$. Given an alphabet of events $T = \{t_1, \ldots, t_n\}$, the *Parikh vector* of a sequence of events is a function $\widehat{\ }\colon T^* \to \mathbb{N}^n$ defined as $\widehat{\sigma} = (|\sigma|_{t_1}, \ldots, |\sigma|_{t_n})$. For simplicity, we will also represent $|\sigma|_{t_i}$ as $\widehat{\sigma}(t_i)$. Given a log $\mathcal{L}$, the set of Parikh vectors of $\mathcal{L}$ is defined as $\Pi(\mathcal{L}) = \{\widehat{\sigma} \mid \sigma \in \mathcal{L}\}$.

2.2 Petri Nets and Process Discovery

A *Petri net* [13] is a tuple (P, T, F, M_0) where P and T represent respectively finite and disjoint sets of places and transitions, $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the weighted flow relation. A marking M is a function $M : P \to \mathbb{N}$. M_0 is the initial marking that defines the initial state of the Petri net.

The preset and postset of a place p are respectively denoted as ${}^\bullet p$ and $p^\bullet$ and defined by ${}^\bullet p = \{t \in T \mid F(t,p) > 0\}$, $p^\bullet = \{t \in T \mid F(p,t) > 0\}$. A Petri net is said to be *pure* if it does not have any self-loop, i.e. $\forall p \in P : {}^\bullet p \cap p^\bullet = \emptyset$. Henceforth, we will assume that all Petri nets referred to in the paper are pure.

The dynamic behavior of a Petri net is defined by its firing rules. A transition $t \in T$ is *enabled* in a marking M if $M(p) \geq F(p,t)$ for any $p \in P$. Firing an enabled transition t in a marking M leads to the marking M' defined by $M'(p) = M(p) - F(p,t) + F(t,p)$, for any $p \in P$, and is denoted by $M \xrightarrow{t} M'$. A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is fireable if there is a sequence of markings $M_1, M_2, \ldots, M_n$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \cdots \xrightarrow{t_n} M_n$. Given a Petri net N, $L(N)$ denotes the language of N, i.e. the set of fireable sequences of transitions. The set of markings reachable from the initial marking M_0 is called the *Reachability Set* of N and denoted as $\mathsf{RS}(N)$.

The Marking Equation: Let us consider a place p with ${}^\bullet p = \{x_1, \dots, x_k\}$, $p^\bullet = \{y_1, \dots, y_l\}$ and all flow relations having weight 1. Let us assume that the place contains $M_0(p)$ tokens in its initial marking. Then, the following equality holds for any sequence of events σ:

$$M(p) = M_0(p) + \widehat{\sigma}(x_1) + \cdots + \widehat{\sigma}(x_k) - \widehat{\sigma}(y_1) - \cdots - \widehat{\sigma}(y_l).$$

The previous equation can be generalized for weighted flows:

$$M(p) = M_0(p) + \sum_{x_i \in {}^\bullet p} F(x_i, p) \cdot \widehat{\sigma}(x_i) - \sum_{y_i \in p^\bullet} F(p, y_i) \cdot \widehat{\sigma}(y_i).$$

If we formulate the previous equation for all places in a Petri net, we can compress it using a matrix notation: $M = M_0 + A \cdot \widehat{\sigma}$, where M and M_0 are place vectors and A is the *incidence matrix* with $|P|$ rows and $|T|$ transitions that represents the flow relation of the net. The previous equation is called the *Marking Equation* of the Petri net [13].

The set of solutions for which the following inequality holds

$$M = M_0 + A \cdot \widehat{\sigma} \geq 0 \tag{1}$$

is called the *Potentially Reachable Set* ($\mathsf{PRS}(N)$). All reachable markings of a Petri net fulfill (1). However the opposite is not always true. In general there can be unreachable markings for which (1) also holds, i.e. $\mathsf{RS}(N) \subseteq \mathsf{PRS}(N)$.

Process Discovery: The problem of process discovery requires the computation of a model M that adequately represents a log $\mathcal{L}$. A model M is *overfitting* with respect to log $\mathcal{L}$ if it is too specific and too much driven by the information in $\mathcal{L}$. On the other hand, M is an *underfitting* model for $\mathcal{L}$ if the behavior of M is too general and allows for things "not supported by evidence" in $\mathcal{L}$. Whereas overfitting denotes lack of generalization, underfitting represents too much generalization. A good balance between overfitting and underfitting is a desired feature in any process discovery algorithm [1].

2.3 Convex Polyhedra and Integer Lattices

An n-dimensional convex polyhedron is a convex set of points in $\mathbb{R}^n$. Convex polyhedra admit two equivalent representations: the H-representation and the V-representation [9]. The former denotes a convex polyhedron $\mathcal{P}$ as the intersection of a finite set of half-spaces, i.e.

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid A \cdot x + b \geq 0\} \tag{2}$$

where $A \in \mathbb{R}^{k \times n}$ and $b \in \mathbb{R}^k$ are the matrix and vector that represent k half-spaces. Given a polyhedron $\mathcal{P}$, the set of integer points inside $\mathcal{P}$ are called the Z-polyhedron of $\mathcal{P}$. For the sake of brevity, all polyhedra mentioned in this work will be assumed to be convex.

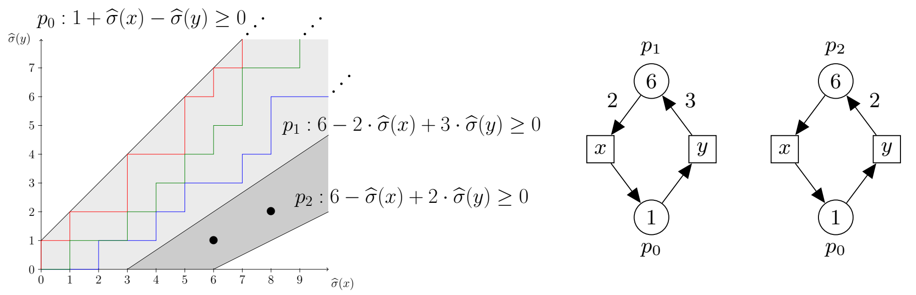

$p_0 : 1 + \widehat{\sigma}(x) - \widehat{\sigma}(y) \geq 0$
$p_1 : 6 - 2 \cdot \widehat{\sigma}(x) + 3 \cdot \widehat{\sigma}(y) \geq 0$
$p_2 : 6 - \widehat{\sigma}(x) + 2 \cdot \widehat{\sigma}(y) \geq 0$
$\widehat{\sigma}(y)$
$\widehat{\sigma}(x)$
p_1
6
2
3
x
y
1
p_0
p_2
6
2
x
y
1
p_0

Fig. 2: Walks in the integer lattice and Petri net.

2.4 Numerical Abstract Domains and Process Discovery

In [11] several techniques are presented for the discovery of Petri nets from Parikh vectors. In particular, given a log $\mathcal{L}$, the set $\Pi(\mathcal{L})$ is used to find A and M_0 in (1) such that the associated Petri net is a good approximation of the process behavior. We now summarize the approach.

Given a Petri net N, by comparing the expressions (1) and (2) we can observe that $\mathsf{PRS}(N)$ is the Z-polyhedron of a convex polyhedron that has two properties: $A \in \mathbb{Z}^{|P| \times n}$ and $M_0 \in \mathbb{N}^{|P|}$. These properties guarantee that the initial marking is not negative and only markings with integral token values are reachable.

The n-dimensional integer lattice $\mathbb{Z}^n$ is the lattice of n-tuples of integers. For describing a log, each lattice point represents a Parikh vector from an alphabet with n symbols and hence the points belong to $\mathbb{N}^n$. A log can be represented as a set of walks in $\mathbb{N}^n$. Every step in a walk moves from one lattice point to another by only increasing one of the components of the n-tuple by one unit.

The link between logs and Petri nets is illustrated in Fig. 2. The figure at the left represents three different walks in a 2-dimensional space. The light grey area represents a polyhedron that *covers* the points visited by the walks. The polyhedron can be represented by the intersection of two half-spaces in $\mathbb{R}^2$:

$$\begin{aligned} 1 + \widehat{\sigma}(x) - \widehat{\sigma}(y) &\geq 0 \\ 6 - 2 \cdot \widehat{\sigma}(x) + 3 \cdot \widehat{\sigma}(y) &\geq 0 \end{aligned}$$

The polyhedron can also be represented in matrix notation with a direct correspondence with the marking equation (1) of a Petri net:

$$\begin{bmatrix} 1 \\ 6 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} \cdot \begin{bmatrix} \widehat{\sigma}(x) \\ \widehat{\sigma}(y) \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The Petri net on the left represents the one obtained from the interpretation of the marking equation. Each face of the polyhedron is represented by a place (row in the matrix). The set of Parikh vectors generated by the Petri net corresponds to the Z-polyhedron of the polyhedron depicted at the left.

In summary, given a set of Parikh vectors from a log, the techniques in [11] find the polyhedron which can finally be translated to a Petri net as shown in the example.

2.5 Inducing Negative Information from a Log or Model

Due to the fact that real-life event logs seldom contain negative information, i.e. behavior that the system should not allow, scholars have proposed alternative ways to induce negative information to guide the learning task. In [8], a technique is proposed to induce so called "artificial negative events" based on the positive information contained in the log. Recent contributions have shown that the obtention of negative information from event logs can be done efficiently in a manner which is robust to differing levels of event log completeness [14]. Finally, when a prescriptive, ground-truth process model is known, negative information can also be appended to the known, positive traces contained in a given event log by replaying the traces over the model and querying the latter to investigate which activities in the activity alphabet are not enabled in a given position in the trace at hand, from which a set of negative traces can be derived.

3 Supervised Process Discovery

In this section we show in detail how to make the approach from [11] supervised. Next section shows how to make an arbitrary discovery technique supervised.

3.1 Stages of the Approach

The proposed approach for supervised process discovery and simplification is illustrated in Fig. 3. The upper part of the figure (enclosed in a round box) represents the approach from [11] from which this work is grounded; the detailed explanation of this is found in Section 2.4. This approach suffers from two shortcomings. First, it is exponential on the number of different activities in the log: in [11] a divide-and-conquer strategy is presented that uses sampling and projection to overcome this limitation for large event logs. Sampling and projection techniques alleviate the complexity of the monolithic approach considerably, but on the other hand the quality metrics regarding precision and simplicity may become considerably degraded, thus deriving an underfitting and complex process model. The reason for this is due to the fact that sampling tends to extract an overfitting representation of the samples used, which may be simplified if the whole set of Parikh vectors (instead of using samples) was used to construct the polyhedron. Additionally, the representations for the samples obtained may miss important relations, a problem that causes a precision degradation.

The second limitation of the approach in [11] is the manual selection of the constraints within the H-representation of the polyhedron computed. Only constraints with simple coefficients (those with gray background in Fig. 3) are used, leaving the rest of constraints (half-spaces that mean to separate the observed behavior from the rest of behavior) out of the model, and hence the model derived may be generalizing too much, i.e. may be imprecise. The selection of simple constraints is guided by the assumption that in reality (and specially, in the scope of business process management), process models tend to be defined by simple constructs.

Having only event logs with positive information at our disposal, we use the technique from [14] to extract accurate negative execution traces. The proposed

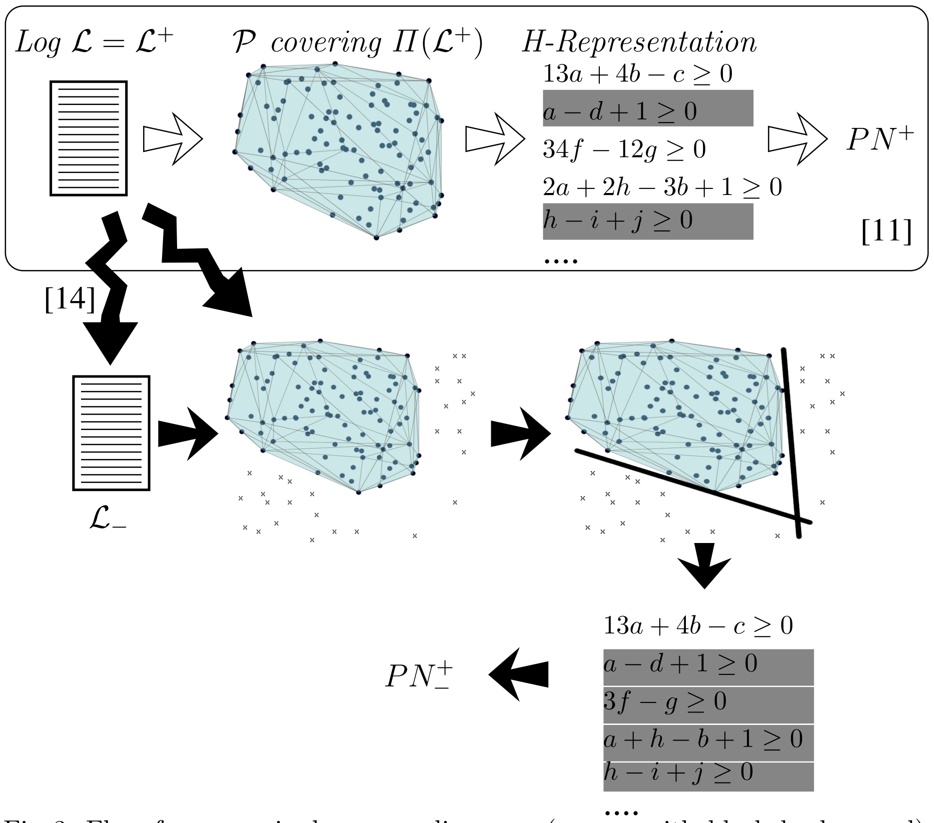


Fig. 3: Flow for supervised process discovery (arrows with black background) compared with the approach in [11] (arrows with white background).

approach inserts the negative Parikh vectors into the n-dimensional space and uses a tailored SMT-based optimization technique to *shift* and *rotate* the half-spaces covering the positive information to simplify and generalize them while keeping the negative information away from the transformed polyhedron (see Sections 3.2 and 3.3). In the figure, it can be seen that $34f - 12g \geq 0$ is a constraint obtained from the polyhedron only covering the positive information. This constraint is then simplified and generalized to $3f - g \geq 0$ which is then considered as a simple constraint, and therefore will be also translated to the final process model. This enables relating in the model activities f and g.

Remarkably, the approach of Fig. 3 can also be applied without any negative information, leading to a simplification (and generalization) of the models derived by the approach presented in [11]. As the shifting and rotation techniques presented in the following sections are implemented as instances of an SMT problem, leaving out the parts related to negative information will enable end users focusing only on simplification of the model in the positive perspective. Next section presents this idea.

3.2 Generalization and Simplification on the Positive Perspective

Section 2.4 explains how to to compute a Petri net containing a set of traces using the minimal convex hull of its Parikh vectors and then extracting its H-representation that can in turn be translated to a Petri net. However, the structure of the obtained model might be too complicated; e.g. actions might consume and produce big amounts of resources. Real-life business process usually have a simpler structure and therefore the discovered models need to be simplified. This task is usually done manually using expert knowledge to detect situations where the net can be simplified. When discovery algorithms based on numerical abstract domains are used, the simplification consists on removing manually inequalities (half-spaces defining the polyhedron) from the H-representation. Since each inequality defines a place in the net, removing them reduces the number of places in the net and simplifies it.

We shift and rotate the polyhedron to obtain simpler inequalities and thus preserve as much as possible the behavior of the system. In Fig. 3 only inequalities in dark background are transformed into places in the final net. Whenever an inequality is removed, the new polyhedron is less restrictive and therefore more points satisfy the set of remaining constraints; in the mined Petri net, more traces are possible, thus generalizing the underlying behavior.

Example 2. Fig. 2 (left) shows a polyhedron (light grey area) defined by the H-representation $\{p_0, p_1\}$, and some of its walks. A more general polyhedron, i.e. one with larger Z-polyhedron (see Section 2.3) is defined by $\{p_0, p_2\}$ (light and dark grey area). Points marked as $\bullet$ are solutions of $\{p_0, p_2\}$, but not of $\{p_0, p_1\}$. The right of the figure shows the Petri nets representing both polyhedra; the sequence $xxxyxxx$ is a trace of the second net, however it cannot be fireable in the first net. This is represented in the left part of the figure by the point $(6, 1)$ which is a solution of $\{p_0, p_2\}$, but not of $\{p_0, p_1\}$.

The approach that we propose in this section simplifies the H-representation of a given polyhedron by modifying its inequalities; this is achieved by trying to reduce the coefficients of each inequality. Each new inequality should accept at least the same solutions as the original one to avoid loosing the fitness of the model. Given an inequality of the form $\alpha_0 + \alpha_1 \cdot x_1 + \cdots + \alpha_n \cdot x_n \geq 0$ we need to find new coefficients $\beta_0, \beta_1, \ldots, \beta_n$ such that:

$$\sum_{i=1}^{n} \beta_i > 0 \text{ and } \beta_0 \geq 0 \tag{NZ}$$

for each $0 \leq i \leq n$:

$$|\beta_i| \leq |\alpha_i| \tag{MIN}$$

and for all $x_i \geq 0$ with $i \leq n$:

$$(\alpha_0 + \sum_{i=1}^{n} \alpha_i \cdot x_i) \geq 0 \;\Rightarrow\; (\beta_0 + \sum_{i=1}^{n} \beta_i \cdot x_i) \geq 0 \tag{PC}$$

Constraint (NZ) specifies that at least one of the variable's coefficients should be different than zero to eliminate trivial solutions and that the independent coefficient should not be negative since it represents the initial marking. The meaning of constraint (MIN) is that the new inequality should be simpler than the original one, i.e. each transition should consume or produce less tokens. Finally, every solution of the original inequality should also be a solution of the discovered one (PC).

To obtain the H-representation of a polyhedron representing a simpler and more general net, constrains (NZ), (MIN) and (PC) can be encoded using Satisfiability Modulo Theories; we have implemented the proposed encoding using the Z3 SMT solver [15]. For the inequality $6-2\cdot\widehat{\sigma}(x)+3\cdot\widehat{\sigma}(y)\geq 0$ the proposed encoding results in:

$$\begin{array}{l}(\beta_1+\beta_2>0)\wedge(\beta_0\geq 0)\wedge(|\beta_1|\leq 2)\wedge(|\beta_2|\leq 3)\wedge\\ \forall\widehat{\sigma}(x),\widehat{\sigma}(y):(6-2\cdot\widehat{\sigma}(x)+3\cdot\widehat{\sigma}(y)\geq 0)\Rightarrow(\beta_0+\beta_1\cdot\widehat{\sigma}(x)+\beta_2\cdot\widehat{\sigma}(y)\geq 0)\end{array}$$

which has as a solution for example $\beta_0=6,\beta_1=-1,\beta_2=2$. The original inequality can be thus replaced in the H-representation of the polyhedron by $6-\widehat{\sigma}(x)+2\cdot\widehat{\sigma}(y)\geq 0$. The new polyhedron generates a simpler Petri net (less tokens are consumed by x and produced by y), but more traces are accepted as it is shown in Example 2.

Since our approach only simplifies inequations, it might be still necessary to remove some of them manually; in Fig. 3 two inequalities are simplified and remain in the final model, but $13a+4b-c\geq 0$ cannot be simplified and thus is still removed.

The method that we propose does not sacrifices fitness of the model since the Z-polyhedron obtained by the transformations is a superset of the original one:

Theorem 1. *Let $\mathcal{L}$ be a log, N a fitting model of $\mathcal{L}$ and N' the model obtained by our method, then N' if fitting for $\mathcal{L}$.*

Proof. The proof is immediate by the constraint (PC) in the encoding of the new polyhedron.

Structural Simplification Ratio: Given an inequality $p_i=\alpha_0+\sum\limits_{i=1}^{n}\alpha_i\cdot x_i\geq 0$ its *structural complexity* is given by $C_{p_i}=\sum\limits_{i=0}^{n}|\alpha_i|$; the complexity of the H-representation of a polyhedron is the sum of the complexity of its inequalities. With this definition the complexity of polyhedra $\{p_0,p_1\}$ and $\{p_0,p_2\}$ are 14 and 12 respectively. Hence we consider the second polyhedron and the corresponding net simpler since its complexity is smaller. The effectiveness of our method is defined as the reduction in the complexity of the new polyhedron.

Example 3. Fig. 1 shows the result of applying our method; the net N_1 has complexity $c_1=C_{p_0}+C_{p_1}+C_{p_2}+C_{p_3}+C_{p_4}+C_{p_5}=(6+2+3)+(1+1+1)+(2+1)+(1+1+1)+(3+1)+1=25$ while the net N_2 obtained by our method has complexity $c_2=C_{p_0}+C_{p_1}+C_{p_2}+C_{p_3}+C_{p_4}+C_{p_5}=(2+1+1)+(1+1+1)+(2+1)+(1+1)+(3+1)+1=17$. In this example the efficiency of our method is $100-100\times(c_2/c_1)=32\%$.

3.3 Improving Generalization and Simplicity via Negative Information

The generalization and simplification method proposed in Section 3.2 may introduce extra behaviors in the discovered model since the new polyhedron covers more points. If we take into account negative information (forbidden traces), the proposed encoding needs to be refined to rule out certain solutions.

We use the method proposed in [14] to generate negative information for our supervised process discovery. This method generates negative traces which are in the frontier of a polyhedron, but since any postfix of a negative trace is also a negative trace, we use extrapolation to generate traces that are not close to the positive behaviors, i.e. the half-spaces defining the polyhedron. If this step is avoided, in most of the cases the method presented in this section does not reduce the complexity of the model discovered simply by using [11] since rotation is very restricted.

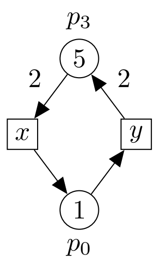
In order to avoid the negative traces derived, each of them is converted into its Parikh representation. Each negative point should not be a solution of the new inequality; this can be encoded as follows; for each negative point $(k_1, \dots, k_n)$:

$$\beta_0 + \sum_{i=1}^{n} \beta_i \cdot k_i < 0 \tag{NP}$$

Going back to Example 2, if we want to simplify inequality p_1 while ruling out the point $(6, 1)$, the new encoding should add the constraint

$$(\beta_0 + \beta_1 \cdot 6 + \beta_2 \cdot 1 < 0)$$

which rules out $\beta_0 = 6, \beta_1 = -1, \beta_2 = 2$ as a solution. The method using negative information proposes $5 - 2 \cdot \widehat{\sigma}(x) + 2 \cdot \widehat{\sigma}(y) \geq 0$ as the simplified inequality resulting in the net on the right which does not accept $xxxyxxx$ as a trace.



3.4 Discussion

The approach presented in this section comes with a hidden assumption that we would like to acknowledge here. It is assumed that negative information can be separated from positive information linearly, i.e. by a set of half-spaces representing a convex polyhedron. However, it is clear that geometrically this is not true in general, i.e. there may be negative points inside the polyhedron constructed. Due to the prefix-closed nature of the positive points in the convex polyhedron (see Fig. 2), negative points must be near the polyhedron half-spaces, and not in the center since a negative point cannot be the prefix of a positive point. In case of dealing with negative points inside a polyhedron, the learning can be oriented to not one but a set of convex polyhedron covering only the positive points, and the merge of several models can then be applied.

Another interesting source of negative information apart from the techniques used in this paper is the use of expert knowledge. This has not been considered in this paper, but extracting negative points from such expert knowledge can be done easily and may contribute to improve the method considerably.

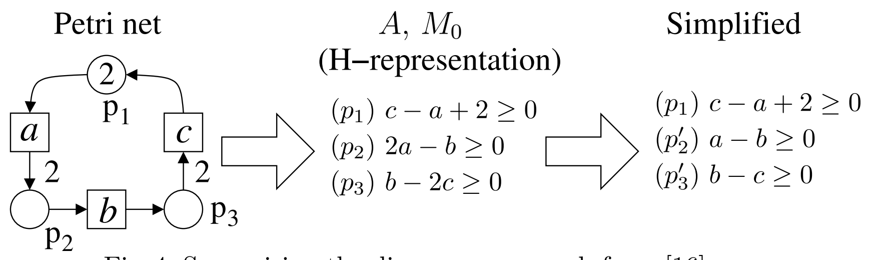


Fig. 4: Supervising the discovery approach from [16].

4 Supervising Arbitrary Process Discovery Techniques

An important observation can be made at this point of the paper: the techniques presented in the previous section can be applied on top of any Petri net and hence are not dependent on the discovery technique from [11]. In Section 2.4 it has been shown the correspondence between a polyhedron and a Petri net by observing that the H-representation of $\mathcal{P}$ represents the marking equation of the corresponding Petri net N. This correspondence is used in [11] in the forward direction, i.e. for computing N from $\mathcal{P}$. To enable the application of the techniques in the previous section to an arbitrary Petri net N, one can simply use the aforementioned correspondence in the backward direction (to compute $\mathcal{P}$ from N) by taking the adjacency matrix of N and the initial marking and use them as the H-representation of a polyhedron corresponding to N.

Example 4. Fig. 4 (left) shows a Petri net that has been derived with the approach from [16] using the state-based theory of regions. The Petri net accepts traces where after firing a, there might be twice the number of bs, e.g., $ababbbc$. Now imagine that these traces are now forbidden, i.e. they are negative traces and only traces with the same amount of as, bs and cs should be included in the model. The technique presented will derive the inequalities in the right, denoting a Petri net similar to the original but where the weights in the arcs are now unitary and the net is conformant with the negative information provided.

5 Experiments

We run our approach as described in Section 3 on several real-life logs. To illustrate the general applicability of the approach as described in Section 4, we also apply our technique on models obtained by ILP Miner [17]. Results on the effectiveness (reduction in the complexity of the simplified polyhedron with regards to the original one) are reported. We also evaluate the precision of discovered models using the state-of-the-art technique from [18], as well as the generalization using the approach from [14]. Finally we compare our method with AGNEsMiner [8], a supervised technique for process discovery.

5.1 Supervising Process Discovery Techniques

The results of the simplification/generalization approach are shown in Tables 1 and 2. For all the examples the simplification step took less than a minute, showing that the overall performance of the discovery method is not degraded. The

Table 1: Experimental results on models mined by the approach of Section 3.

log	total	prec	gen	$\mathcal{L}^+$				$\mathcal{L}^+$ and $\mathcal{L}_-$			
				iter	effec	prec	gen	iter	effec	prec	gen
caise2014	425	0,08	0.56	6	7.41	**0,09**	**0.59**	4	0.31	0,08	0.56
complex	28	**0,32**	0.54	2	2.81	**0,32**	**0.60**	1	0.00	**0,32**	0.54
confdimblocking	15	**1,00**	**0.84**	2	6.25	0,25	**0.84**	1	0.00	**1,00**	**0.84**
documentflow	26	**0,16**	**0.99**	2	11.94	**0,16**	**0.99**	2	7.46	0,15	**0.99**
fhmexamplen5	11	0,32	0.90	3	16.21	0,35	**0.93**	2	2.70	**0,36**	0.90
incident	15	**0,26**	**1.00**	2	17.46	0,22	**1.00**	2	4.76	0,24	**1.00**
purchasetopay	15	0,17	**1.00**	2	4.16	**0,20**	**1.00**	2	2.08	**0,20**	**1.00**
receipt	49	**0,26**	0.62	2	19.31	0,23	**0.69**	3	3.44	0,24	0.62

following information is given: total number of inequations[5] ("total"), precision ("prec") and generalization ("gen") in the original model[6]; the efficiency ("effec") precision and generalization of the enhanced models when only positive information is used ("$\mathcal{L}^+$") and when negative information is also added ("$\mathcal{L}^+$ and $\mathcal{L}_-$"). Since the SMT encoding gives one solution (not necessarily the minimal one), we applied our method iteratively until the effectiveness between two iterations is not improved; the number of iterations is also reported in the table ("iter").

The results show that the complexity can be reduced up to 20% when only positive information is considered; in all the cases the penalty of this reduction is rather small, since the precision of the new model is similar to the original one (except for *confdimblocking* where a drop in precision occurs). When negative information is added, the effectiveness is reduced (below 8%) but precision values are almost coinciding with the original models. The same remark can be made for generalization: when only positive information is considered, all models exhibit a slight increase. After adding negative information, generalization scores are comparable to the original models. We can conclude that applying the ideas of Sections 3 and 4 results in models which are much simpler without a big impact on precision and which retain original generalization capabilities.

5.2 Empirical Comparison

Few approaches have been proposed in literature towards supervised process discovery (see next Section 6 for an overview on related work). We have chosen to compare our approach with [8], a supervised technique which is also able to utilize artificially generated negative events. Table 3 provides a comparative overview of fitness, precision and generalization scores for our approach applied on the numerical abstract domains based miner, ILP Miner, and AGNEsMiner[7].

[5] Although in Section 3 we comment on the fact that inequations can be chosen manually to retain only simple constructs, in the experiments we have avoided such manual selection for the sake of a fair comparison.

[6] As both the numerical abstract domains based miner and ILP Miner discover perfectly fitting models, fitness is not reported in the result tables.

[7] Attentive readers will observe that it is in fact possible to apply our supervised simplification approach to models mined by supervised process discovery techniques, e.g.

Table 2: Experimental results on models mined by ILP Miner [17].

log	total	prec	gen	$\mathcal{L}^+$				$\mathcal{L}^+$ and $\mathcal{L}_-$			
				iter	effec	prec	gen	iter	effec	prec	gen
caise2014	112	0,17	0.57	7	24.04	0,05	**0.65**	3	9.21	**0,24**	0.58
complex	15	**0,57**	0.56	2	14.89	0,35	**0.75**	2	2.12	0,40	0.58
confdimblocking	10	**1,00**	**0.84**	2	8.33	0,26	**0.84**	2	4.16	0,28	**0.84**
documentflow	128	0,37	0.95	5	10.23	**0,39**	**0.96**	3	2.13	0,36	**0.96**
fhmexamplen5	22	0,38	0.92	2	9.70	**0,42**	**0.94**	2	0.97	0,38	0.92
incident	27	0,25	**0.99**	9	25.25	0,28	**0.99**	4	10.60	**0,29**	**0.99**
purchasetopay	12	**0,35**	**0.99**	3	13.13	0,09	**0.99**	2	12.12	0,20	**0.99**
receipt	35	**0,35**	0.62	6	25.26	0,29	**0.83**	2	3.15	0,32	0.62

Table 3: Fitness, precision, and generalization for supervised techniques.

log	Supervised Polyhedra			Supervised ILP Miner			AGNEsMiner		
	fit	prec	gen	fit	prec	gen	fit	prec	gen
caise2014	**1.00**	0,08	0.56	**1.00**	**0,24**	**0.58**	–	–	–
complex	**1.00**	0,32	0.54	**1.00**	0,40	**0.58**	0.82	**0.71**	0.43
confdimblocking	**1.00**	**1,00**	**0.84**	**1.00**	0,28	**0.84**	**1.00**	**1.00**	**0.84**
documentflow	**1.00**	0,15	**0.99**	**1.00**	**0,36**	0.96	–	–	–
fhmexamplen5	**1.00**	0,36	0.90	**1.00**	0,38	**0.92**	0.94	**0,53**	0.32
incident	**1.00**	0,24	**1.00**	**1.00**	0,29	0.99	0.84	**0,65**	0.63
purchasetopay	**1.00**	0,20	**1.00**	**1.00**	0,20	0.99	0.86	**0,82**	0.26
receipt	**1.00**	0,24	**0.62**	**1.00**	0,32	**0.62**	0.92	**0.81**	0.23

The following conclusions are derived from the results: first, we note that AGNEsMiner generally performs well on the dimension of precision, although at a cost of deriving models which are not perfectly fitting. In addition, the miner did not succeed to find a model within the allotted time period (one day of calculation, the dash mark "–" represents a time out). In terms of generalization, our proposed approach outperforms AGNEsMiner since the best results are obtained either by the supervised polyhedra or the supervised ILP Miner.

6 Related Work

Very few approaches exist towards supervised process discovery, especially when compared to the multitude of process discovery techniques which work in an unsupervised fashion. Maruster et al. [19] were among the first to investigate the use of supervised techniques (in this case: rule-induction learners) to predict dependency relationships between activities. Instead of relying on negative information, the authors apply the learner on a table of metrics for each activity derived from the positive information.

AGNEsMiner. We have not done so in this section, however, to keep the comparison between various supervised discovery strategies pure.

Ferreira and Ferreira [4] apply inductive logic programming and partial-order planning techniques to derive a process model. Negative information is collected from users and domain experts who indicate whether a proposed execution plan is feasible or not, iteratively combining planning and learning to discover a process model.

Lamma et al. [5,6,7] apply an extension of logic programming, SCIFF, towards supervised declarative process discovery, i.e. the process model is represented as a set of logic constraints and not as a visual process model as done in this work. The authors assume the presence of negative information.

Similarly, Goedertier et al. [8] represent the process discovery task as a multi-relational first-order classification problem and apply inductive logic programming in their AGNEsMiner algorithm to learn the discriminating preconditions that determine whether an event can take place or not, given a history of events of other activities. These preconditions are then converted to a graphical model after applying a pruning and post-processing step. To guide the learning process, an input event log is supplemented with induced artificial negative events, similar as in this work.

7 Conclusions and Future Work

We have presented a supervised approach based on numerical abstract domains and SMT which is able to simplify and generalize discovered process models based on negative information found in event logs, derived artificially or supplied by domain experts. We believe this contribution opens the door for supervising (either manually or automatically) discovery techniques, a crucial feature for improving the quality of derived process models.

With regard to future work, we plan to pursue to following avenues. First, we have made use of an artificial negative event induction technique in order to derive negative information for a given event log. We plan to investigate the possibilities towards incorporating domain knowledge to simplify and generalize models using our technique. Second, we have assumed that negative information can be separated from positive information in a linear fashion, i.e. by a set of half-spaces representing a convex polyhedron. However, there may be negative points inside the polyhedron constructed. As such, the learning task can be oriented to not one but a set of convex polyhedron covering only the positive points, for which merging methods would need to be investigated. Third we may combine the techniques of this paper with other simplification techniques developed by some of the authors that enrich the model with log-based simulation scores. Finally, we plan to set up a thorough experiment in which we investigate the effects of our approach on models mined by various miners. As we have argued, our approach can be applied on top of any Petri net in order to generalize and simplify it without loss of fitness.

Acknowledgments. This work as been partially supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R) and the Academy of Finland projects 139402 and 277522.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: BPM. (2007) 328–343
3. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: CIDM. (2011) 310–317
4. Ferreira, H., Ferreira, D.: An integrated life cycle for workflow management based on learning and planning. International Journal of Cooperative Information Systems **15**(4) (2006) 485–505
5. Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Inducing declarative logic-based models from labeled traces. In: BPM, Springer (2007) 344–359
6. Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. Inductive Logic Programming (2008) 132–146
7. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the sciff framework. ACM Transactions on Computational Logic (TOCL) **9**(4) (2008) 29
8. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research **10** (2009) 1305–1340
9. Rockafellar, R.T.: Convex Analysis. Princeton University Press (1970)
10. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(t). J. ACM **53**(6) (2006) 937–977
11. Carmona, J., Cortadella, J.: Process discovery algorithms using numerical abstract domains. IEEE Trans. Knowl. Data Eng. **26**(12) (2014) 3064–3076
12. Fukuda, K., Picozzi, S., Avis, D.: On canonical representations of convex polyhedra. In: Proc. of the First International Congress of Mathematical Software. (2002) 350–360
13. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (April 1989) 541–580
14. vanden Broucke, S.K.L.M., Weerdt, J.D., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Trans. Knowl. Data Eng. **26**(8) (2014) 1877–1889
15. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008. Volume 4963 of Lecture Notes in Computer Science., Springer (2008) 337–340
16. Carmona, J., Cortadella, J., Kishinevsky, M.: New region-based algorithms for deriving bounded Petri nets. IEEE Trans. Computers **59**(3) (2010) 371–384
17. Van der Werf, J.M.E., van Dongen, B.F., Hurkens, C.A., Serebrenik, A.: Process discovery using integer linear programming. In: Applications and Theory of Petri Nets. Springer (2008) 368–387
18. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Measuring precision of modeled behavior. Inf. Syst. E-Business Management **13**(1) (2015) 37–67
19. Maruster, L., Weijters, A., van der Aalst, W., van den Bosch, A.: A Rule-Based Approach for Process Discovery: Dealing with Noise and Imbalance in Process Logs. Data Mining and Knowledge Discovery **13**(1) (2006) 67–87