Incorporating Negative Information to Process Discovery of Complex Systems

Hernán Ponce de León[1 a], Lucio Nardelli[b], Josep Carmona[c], Seppe K.L.M vanden Broucke[d]

[a] *fortiss GmbH, Germany*
[b] *Universidad Nacional de Rosario, Rosario, Argentina*
[c] *Universitat Politècnica de Catalunya, Barcelona, Spain*
[d] *Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Leuven, Belgium*

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 one-class supervised learning problem where only a set of process instances is available. Moreover, the majority of techniques cannot generate complex models, a crucial feature in some areas like manufacturing. In this paper we present a fresh look at process discovery where undesired process behaviors can also be taken into account. This feature may be crucial for deriving process models which are less complex, fitting and precise, but also good on generalizing the right behavior underlying an event log. The technique is based on the theory of convex polyhedra and satisfiability modulo theory (SMT) and can be combined with other process discovery approach as a post processing step to further simplify complex models. We show in detail how to apply the proposed technique in combination with a recent method that uses numerical abstract domains. Experiments performed in a new prototype implementation show the effectiveness of the technique and the ability to be combined with other discovery techniques.

Email addresses: `ponce@fortiss.org` (Hernán Ponce de León[1]), `lucio@fceia.unr.edu.ar` (Lucio Nardelli), `jcarmona@cs.upc.edu` (Josep Carmona), `seppe.vandenbroucke@kuleuven.be` (Seppe K.L.M vanden Broucke)

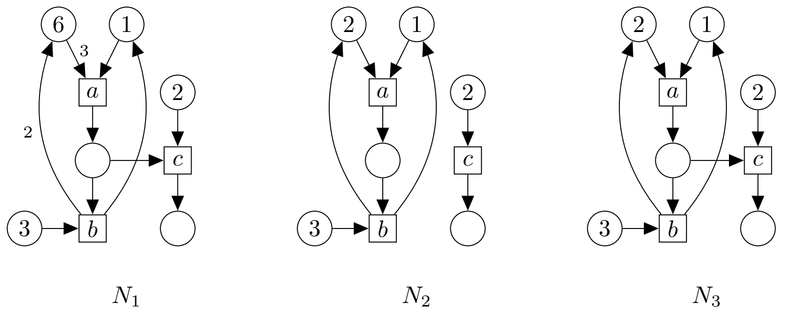
[1] This work was carried out when the author was at Aalto University.

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. Process Mining is considered to be a viable solution to this problem: by using the event log 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 [23]. Process discovery, the main focus of this paper, is a family of techniques for deriving process models expected to be good in four quality dimensions: *fitness* (ability of the model to reproduce the traces in the event log), *precision* (ability of the model to avoid reproducing undesired behavior), *generalization* (ability of the model to reproduce desired behavior not found in the event log) and *simplicity* (the well-known *Occam's Razor* principle). Process discovery is a *learning* technique: given a set of training examples (traces denoting process executions) the goal is to derive a process model which encloses the behavior underlying the training set. Most techniques that have been proposed for process discovery so far assume a positive label for each given trace, i.e. the example is an instance of behavior that must be in the process model to be derived. In that sense, most discovery algorithms can be regarded as one-class supervised learning. Very few techniques have been presented that consider the discovery problem as a binary, two-class supervised learning task, i.e. using the real process executions as positive examples, but also traces representing behavior that is forbidden in the underlying system and should hence not be accepted by the process model to be derived. 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 forbidden behavior. Another benefit is the ability to reduce the complexity of a model on those parts that do not contribute to differentiate between positive and negative examples. We ground our binary-class supervised approach on the duality between the marking equation of a Petri net [21] and the domain of convex polyhedra, which has been already exploited for process discovery in [5]. Remarkably, this approach is among the few that can discover the full class of pure P/T-nets, i.e. those with arbitrary arc weights and tokens. This aspect makes the approach well suited for domains where systems are complex (e.g. manufacturing). Even if the theory of polyhedra allows to capture non-unitary relations, it might make the model unnecessarily complex. In order to avoid this issue, it is necessary to remove or simplify the parts of the net that may not be essential for the underlying process. The technique based on the theory of polyhedra suffers from three main limitations, namely *(i)* it may discover large arc weights and tokens, *(ii)* it may allow for unwanted behavior, and *(iii)* it only uses heuristics to simplify the model. Hence, our previous work [19] extended the technique from [5] by an extra step to reduce the complexity of the polyhedron. This step focuses on half-spaces representing complex restrictions that can be relaxed while preserving the initial solutions, i.e. the positive traces. Additionally, forbidden traces can be encoded as negative points which must not be enclosed by the polyhedron, thus preventing some of the previously mentioned problems. Remarkably, in contrast with the work of [5], this step is automated with the help of satisfiability modulo theory (SMT): constraints expressed as formulas in first-order-logic that enable to derive an optimal rotation and shift of the polyhedron half-spaces.

Example 1. *Consider the three models (Petri nets) of Figure 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*[3]. *N_1 can replay all the traces in $\mathcal{L}^+$, but not those in $\mathcal{L}_-$; we can conclude that it is fitting and*

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



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

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

precise. N_2 is also fitting, but it is too imprecise since it accepts some of the undesired behavior in $\mathcal{L}_-$, e.g. action c can be fired independently of the firing of a. Using the approach by [5] the first net is discovered while the second one can be discovered by the algorithms from [19] and this article using only positive information. It can be considered that the structure of the latter is less complex since it has less arcs and smaller weights. The problem with the transformation from N_1 into N_2 is that it introduces undesired behavior. Then net N_3 can be discovered using also negative information; it does not accept any undesired behavior and it is still less complex than N_1.

This paper extends our previous work [19]; the main contributions are: *(i)* an improved SMT-encoding that reduces the complexity of the net globally rather than locally, i.e. it simplifies the whole incidence matrix rather than one half-space at a time, *(ii)* an automatic post processing step to remove complex half-spaces when possible, *(iii)* a new prototype tool that unifies the results from [5, 19] and the contributions of this article, and *(iv)* an experimental set up using k-fold cross validation.

2. Preliminaries and Problem Statement

2.1. Event Logs and their Parikh Representation

The behavior of a process is observed as sequences of events from a given alphabet T of activities. A trace is a word $\sigma \in T^*$ that describes a finite sequence of events, i.e. occurrences of an activity[4]. A *log* $\mathcal{L}$ is a set of traces from a given alphabet[5]. By abuse of notation we say that $\sigma \in \mathcal{L}$ if σ is the prefix of some trace of $\mathcal{L}$. In other words, a log $\mathcal{L}$ is a finite set of traces over a certain alphabet representing the footprints of the real process executions of a system that is only (partially) visible through these runs.

We use $|\sigma|_a$ to represent the number of occurrences of a in a trace σ. Given an alphabet of events $T = \{t_1, \ldots, t_n\}$, the *Parikh vector* of a sequence of events is a function $\widehat{\ }: T^* \to \mathbb{N}^n$ 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 $\Pi(\mathcal{L}) = \{\widehat{\sigma} \mid \sigma \in \mathcal{L}\}$. Given the trace $\sigma = abababac$, its Parikh representation is $\widehat{\sigma} = (4, 3, 1)$. Notice that a trace is in a log if it is a prefix of a log trace. Then, $\Pi(\{\sigma\}) = \{(0,0,0), (1,0,0), (1,1,0), (2,1,0), \ldots, (4,3,1)\}$. This implies that $\Pi(\{ab\}) \neq \Pi(\{ba\})$.

2.2. Petri Nets

A *Petri net* [18] is a tuple (P, T, F, M_0) where P and T represent respectively finite and disjoint sets of places and transitions, the weighted flow relation is given by $F : (P \times T) \cup (T \times P) \to \mathbb{N}$. A marking M is a function $M : P \to \mathbb{N}$. The initial marking M_0 defines the initial state of the Petri net.

Notice that the same symbol T will be used to denote the transitions of the Petri net and the alphabet of events for the traces in the log, i.e. each transition in the net corresponds exactly to one activity in the log. For the same reason

[4]The language operators $()^*$ and $()^+$ represent sequences of *any* length and length *at least* one respectively.

[5]Logs can be defined more generally as multi-sets where some traces can be observed multiple times. Since we do not consider frequency of traces, we abstract from this and only consider the presence or absence of certain behavior by using sets.

silent transitions are not allowed in the net and two different transitions cannot represent the same action.

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 called *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. This is a restriction on *(i)* the output (Petri net) produced by the discovery technique, but not on the input (log) and *(ii)* a restriction on the Petri net input in case the technique is used for simplification. In the latter one would however be able to apply a pre-processing step on the Petri net to resolve this.

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 $\sigma = t_1 t_2 \dots t_n$ is fireable if there is a sequence of markings $M_1, M_2, \dots, M_n$ 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, $\mathscr{L}(N)$ 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 $\mathsf{RS}(N)$.

Consider a place p with ${}^\bullet p = \{x_1, \dots, x_k\}$, $p^\bullet = \{y_1, \dots, y_l\}$ and a flow relation F. Assume that the place contains $M_0(p)$ tokens in its initial marking. Then, the following equality holds for any sequence of events σ

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

If we formulate the previous equation for all places in a Petri net, we can compress it using a matrix notation: $M = M_0 + A \times \widehat{\sigma}$, where M and M_0 are vectors and A is the *incidence matrix* with $|P|$ rows and $|T|$ columns that represents the flow relation of the net. The previous equation is called the *Marking Equation* of the Petri net [18]. The set of solutions for which the following inequality holds

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

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. $\mathsf{RS}(N) \subseteq \mathsf{PRS}(N)$.

The set of inequalities (in its matrix notation) representing the PRS of the left Petri net in Fig. 2 is the following

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

2.3. Numerical Abstract Domains and Process Discovery

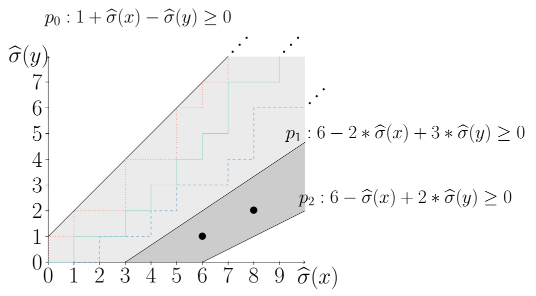
An n-dimensional convex polyhedron is a convex set of points in $\mathbb{Z}^n$. A *half-space* is that portion of an n-dimensional space obtained by removing that part lying on one side of an $(n-1)$-dimensional hyperplane. It can be specified by a linear inequality $a_1 \times x_1 + a_2 \times x_2 + \cdots + a_n \times x_n \geq b$. The H-representation of a convex polyhedron $\mathcal{P}$ denotes it as the intersection of a finite set of half-spaces

$$\mathcal{P} = \{x \in \mathbb{Z}^n \mid A \times x + b \geq \mathbf{0}\} \tag{3}$$

where $A \in \mathbb{Z}^{k \times n}$ and $b \in \mathbb{Z}^k$ are the matrix and vector that represent k half-spaces. For the sake of brevity, all polyhedra mentioned in this work will be assumed to be convex.

Several techniques for the discovery of Petri nets from Parikh vectors were introduced in [5]. 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. Given a Petri net N, by comparing the expressions (1) and (3) we can observe that $\mathsf{PRS}(N)$ is the Z-polyhedron of a convex polyhedron defined by two matrices: $A \in \mathbb{Z}^{|P| \times |T|}$ and $M_0 \in \mathbb{N}^{|P|}$. These guarantee that the initial marking is not negative and only markings with positive token values are reachable.

The link between logs and Petri nets is illustrated in Figure 2. The light grey area represents a polyhedron covering the points visited by the walks. The



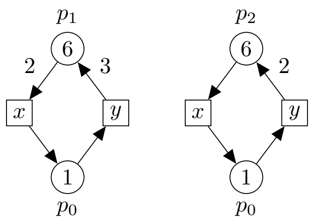


Figure 2: Convex polyhedra and Petri net.

polyhedron can be represented by the intersection of two half-spaces in $\mathbb{Z}^2$:

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

The polyhedron can also be represented by equation (2) obtained from the interpretation of the marking equation for the net on the left in Figure 2. Each half-space of the polyhedron is represented by a place. In summary, given a set of Parikh vectors from a log, the techniques by [5] find the polyhedron which can finally be translated to a Petri net as shown in the example.

2.4. Inducing Negative Information from a Log or Model

Different interpretations exist for the term *negative information.* In machine learning, the term refers to a behavior that was observed, but with a negative label, e.g. from a group of student taking a course, we want to classify what makes a student pass the course (positive instance) or fail it (negative one). The notion that we give here to negative traces in the setting of process mining is somewhat different; a negative trace represents a behavior that is forbidden by the system and thus it should not be allowed by the model.

Due to the fact that real-life event logs seldom contains forbidden behavior, scholars have proposed alternative ways to induce negative information to guide the learning task. A technique to induce so called *artificial negative events* based on the positive information contained in the log was proposed in [12]. The obtention of forbidden traces from event logs can be done efficiently in a manner which is robust to differing levels of event log completeness [27]. Also, when a prescriptive, ground-truth process model is known, forbidden traces can be obtained by replaying the positive traces over the model and querying the latter to investigate which activities in the activity alphabet are not enabled.

The kind of forbidden traces that we consider in this article are of the form $\sigma = \sigma'\sigma''$ where σ' is executable in the system (i.e. it is a positive trace), but after it the system cannot perform σ''.

Definition 1. *Given a log $\mathcal{L}$ over the alphabet T, we say that $\sigma = \sigma'\sigma''$ is a forbidden (or prohibited, or negative) trace iff $\sigma'' \in T^+$ and $\sigma' \in \mathcal{L}$, but $\sigma \notin \mathcal{L}$.*

Notice that our definition is more general than the one from [12, 27] where negative traces are traces of the log followed by a unique negative event, i.e. σ'' contains only one event. Even if such traces fulfill Definition 1, we use negative information that is within a certain distance from positive information. In practice we use randomly generated postfixes of the prohibited traces obtained by the algorithm proposed by [27]. The behavior of this trace should still be considered as forbidden since every postfix of a forbidden trace is also a forbidden trace according to Definition 1.

2.5. Problem Statement

Given a log $\mathcal{L}^+$ and a collection of forbidden traces $\mathcal{L}^-$ (artificially created or not), the goal of the techniques of this paper is to derive a Petri net N with the following characteristics: *(i)* N is a weighted P/T net; *(ii)* N fits $\mathcal{L}^+$; *(iii)* for every forbidden trace $\sigma \in \mathcal{L}^-$, we have $\sigma \notin \mathscr{L}(N)$; and *(iv)* N has the least complexity with respect to elements, weights and tokens.

The first three items are guaranteed by construction, while for complexity we will evaluate derived models with respect to a tailored fine-grain complexity metric which takes into account not only the number of elements but also its weight. In the evaluation section, we will use current metrics for precision and generalization to estimate whereas the derived models are in good balance between underfitting and overfitting the log.

3. Binary-class Supervised Process Discovery

In this section we show how the approach by [5] can be adapted to use two classes of traces: positive and negative ones. We additionally show how to reduce the complexity of models obtained by other discovery techniques.

The idea is to transform the traces in the log into a set of points of an n-dimensional space and then to find a convex envelope of these points. The final step is to convert the convex polyhedron into a Petri net using the duality between the convex polyhedra and the marking equation of nets. One of the limitations of the algorithm used for computing a polyhedron covering the set of points is that it actually computes the *minimal* polyhedron (called minimal convex hull). This minimality generally introduces unnecessary restrictions that generates complicated models. A post processing step by [5] consists of a simplification done by manually selecting a subset of the constraints within the H-representation of the polyhedron computed. However the derived model may be generalizing too much, i.e. may be imprecise. Since we are only interested in the information that allows to differentiate between positive and negative instances, the selection of constraints can be done using negative information to avoid too much generalization. A half-space is removed only if its removal does not introduce any negative point. The use of negative information allows to automatically detect and remove such constraints.

3.1. Stages of the Approach

The proposed approach is illustrated in Figure 3. The upper part of the figure (enclosed in a black box) represents the approach from [5] from which this work is grounded. The bottom part of the figure shows how the complexity of a model can be reduced with the use also of forbidden traces, at the price of accepting more points (more traces are fireable in the net).

Algorithm 1 formalizes our approach step by step using pseudocode. A set of allowed and forbidden traces is given as input (event log and negative information in Figure 3). Lines 2-7 compute the positive points pp for every prefix of a positive trace in $\mathcal{L}^+$ and the negative points np for the traces in $\mathcal{L}_-$ (but

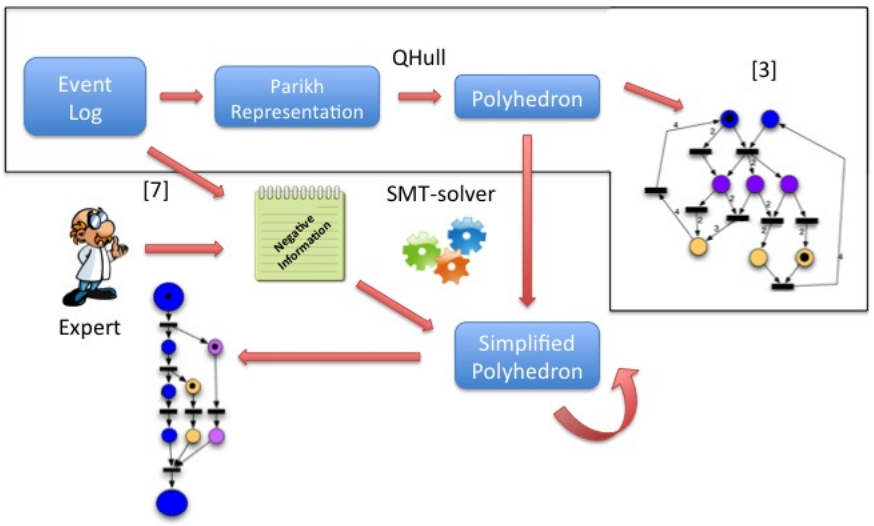


Figure 3: Flow for supervised process discovery (below) compared with the approach in [5] (enclosed in a black box).

not for its prefixes since some of them correspond to positive traces following Definition 1). A polyhedron covering the set of points *pp* is computed by ConvexHull using any algorithm that computes the minimal convex hull for a set of points (e.g., Qhull by [3]). The complexity of the obtained polyhedron is then reduced by Shift&Rotate using the SMT-encoding presented in Sections 3.2 and 3.3. Since the transformation is encoded as an SMT-instance, there might be several solutions. The SMT-solver does not necessarily returns the optimal one (simplest polyhedron) and therefore we follow an iterative approach to find this optimal solution. The Removal procedure iterates over the sets of half-spaces and removes all of them for which this does not introduce any negative point as a solution; this step replaces the manual selection done by [5]. Finally the set of half-spaces is transformed into a Petri net by Hull2Net using the duality between polyhedra and the marking equation of a Petri net.

Since ConvexHull may depend on sampling and projection techniques and Shift&Rotate is encoded as an instance of SMT which may have several solutions and the obtained one depends on the solver, our algorithm is not

Algorithm 1 Binary-class Supervised Process Discovery

Require: allowed (positive) traces $\mathcal{L}^+$ and forbidden (negative) traces $\mathcal{L}_-$
Ensure: a Petri net N with $\forall \sigma \in \mathcal{L}^+ : \sigma \in L(N)$ and $\forall \sigma \in \mathcal{L}_- : \sigma \notin L(N)$

```
1: procedure Discover($\mathcal{L}^+, \mathcal{L}_-$)
2:     $pp, np \leftarrow \emptyset$
3:     for $\sigma_p \in \mathcal{L}^+$ do
4:         for $\sigma$ prefix of $\sigma_p$ do
5:             add $\widehat{\sigma}$ to $pp$
6:     for $\sigma_n \in \mathcal{L}_-$ do
7:         add $\widehat{\sigma}_n$ to $np$
8:     $H$ = ConvexHull($pp$)
9:     $H_{smt}$ = Shift&Rotate($H, np$)
10:    $H'$ = Removal($H_{smt}, np$)
11:    $N$ = Hull2Net($H'$)
12:    return N
```

deterministic, i.e. given the same set of allowed and forbidden traces, it can produce different nets.

3.2. Generalization on the Positive Perspective

Section 2.3 explains how 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; those corresponds to lines 2-8 in Algorithm 1. However, the structure of the obtained model might be too complicated due to the fact that we are not computing any polyhedron covering all the points, but the *minimal convex hull.* The minimality constraint generates a complex model and in order to make sense of real-life process and obtain valuable information, one thus has to abstract from the particular details, hence simplify.

We propose to first modify the polyhedron to obtain less complex half-spaces preserving as much as possible the behavior of the obtained polyhedron (Shift&Rotate procedure detailed in the remainder of this section and Section 3.3). If after this step a half-space still needs to be removed, the new polyhedron is less restrictive and therefore more points satisfy the set of remaining constraints; in the obtained Petri net, more traces are possible, thus generalizing the underlying behavior. We make this removal automatically by checking that

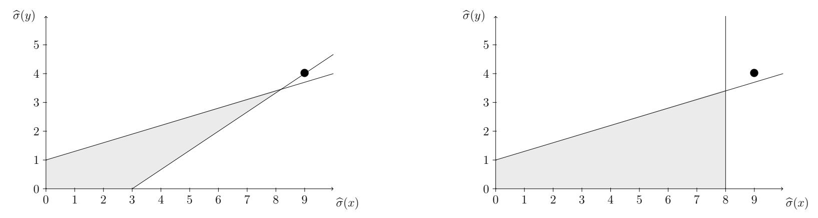


Figure 4: Individiual half-space reduction vs. matrix reduction.

this step does not introduce forbidden behavior (Remove procedure detailed in Section 3.4).

Example 2. *Figure 2 (left) shows a polyhedron (light grey area) defined by the H-representation $\{p_0, p_1\}$. A more general polyhedron, i.e. one with larger Z-polyhedron 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 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* $\{p_0, p_1\}$.

In order to simplify a net, one could try to reduce the complexity of the H-representation of its corresponding polyhedron by modifying its half-spaces; this can be achieved by reducing the coefficients of each half-space in isolation. Each new half-space should accept at least the same solutions as the original one to avoid loosing fitness. However since the behavior of the model is not defined by a single half-space, but by the conjunction of them, loosing some solutions allowed by a single half-space should be tolerated as far as the set of solutions of the whole system is not reduced. Figure 4 illustrates this idea; the point $(9, 4)$ is a solution on the left for the half-space (call it p) bounding the grey polyhedron from below. If we try to simplify p in isolation with the approach by [19], the half-space $x = 8$ on the right could *not* be obtained since the solution $(9, 4)$ of p is lost. However point $(9, 4)$ is not part of the whole system (the one bounded by the conjunction of both half-spaces) and therefore this rotation should be allowed since the solution set on the right contains the solution set on the left.

We propose then to get a net with lower complexity by transforming the whole incidence matrix and initial token distribution from the marking equation rather than its half-spaces individually. Given a system of the form

$$\begin{array}{ccccccccc}
\alpha_{1,0} & + & \alpha_{1,1} \times x_1 & + & \ldots & + & \alpha_{1,n} \times x_n & \geq & 0 \\
\alpha_{2,0} & + & \alpha_{2,1} \times x_1 & + & \ldots & + & \alpha_{2,n} \times x_n & \geq & 0 \\
\vdots & & & & & & \vdots & & \\
\alpha_{m,0} & + & \alpha_{m,1} \times x_1 & + & \ldots & + & \alpha_{m,n} \times x_n & \geq & 0
\end{array}$$

we need to find new coefficients $\beta_{1,0}, \beta_{1,1}, \ldots, \beta_{m,n}$ such that

$$\sum_{i,j=1}^{m,n} \beta_{i,j} > 0 \text{ and } \sum_{i=1}^{m} \beta_{i,0} > 0 \tag{NZ}$$

for each $0 \leq i \leq m$ and $1 \leq j \leq n$

$$|\beta_{i,j}| \leq |\alpha_{i,j}| \tag{MIN}$$

and for all $x_j \geq 0$ with $1 \leq j \leq n$

$$\bigwedge_{i=1}^{m} (\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} \times x_j) \geq 0 \;\Rightarrow\; \bigwedge_{i=1}^{m} (\beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} \times x_j) \geq 0 \tag{PC}$$

Constraint (NZ) specifies that *(i)* at least one of the coefficients should be different than zero to eliminate trivial solutions and *(ii)* some place should be initially marked. The meaning of constraint (MIN) is that the new matrix should be less complex than the original one, i.e. each transition should consume or produce less tokens. Finally, every solution of the original system should also be a solution of the discovered one to preserve fitness (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 theory. For the incidence matrix of the left net in Figure 2 the proposed encoding results in

$$(\beta_{1,1} + \beta_{1,2} + \beta_{2,1} + \beta_{2,2} > 0) \wedge (\beta_{1,0} \geq 0) \wedge (\beta_{2,0} \geq 0) \wedge$$
$$(|\beta_{1,1}| \leq 2) \wedge (|\beta_{1,2}| \leq 3) \wedge (|\beta_{2,1}| \leq 1) \wedge (|\beta_{2,2}| \leq 1) \wedge$$
$$\forall \widehat{\sigma}(x), \widehat{\sigma}(y) : (6 - 2 \times \widehat{\sigma}(x) + 3 \times \widehat{\sigma}(y) \geq 0 \wedge 1 + \widehat{\sigma}(x) - \widehat{\sigma}(y) \geq 0)$$
$$\Rightarrow (\beta_{1,0} + \beta_{1,1} \times \widehat{\sigma}(x) + \beta_{1,2} \times \widehat{\sigma}(y) \geq 0 \wedge \beta_{2,0} + \beta_{2,1} \times \widehat{\sigma}(x) + \beta_{2,2} \times \widehat{\sigma}(y) \geq 0)$$

which has as a solution for example $\beta_{1,0} = 6, \beta_{1,1} = -1, \beta_{1,2} = 2, \beta_{2,0} = 1, \beta_{2,1} = 1, \beta_{2,2} = -1$ corresponding to the marking equation of the right net in Figure 2.

The method that we propose does not sacrifice fitness of the model since the 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' is fitting for $\mathcal{L}$.*

Proof: Let $\mathcal{P} = \bigwedge_{i=1}^{m} (\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} \times x_j) \geq 0$ and $\mathcal{P}' = \bigwedge_{i=1}^{m} (\beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} \times x_j) \geq 0$ be respectively the polyhedra obtained by the interpretation of the marking equation of nets N and N'. Since N is fitting w.r.t $\mathcal{L}$, all the points in the Parikh representation of $\mathcal{L}$ are solutions of $\mathcal{P}$. By the constraint (PC) all the points are also solutions of $\mathcal{P}'$ and thus all the traces in $\mathcal{L}$ are fireable in N'. Finally N' is fitting for $\mathcal{L}$. $\square$

3.3. Improving Generalization via Negative Information

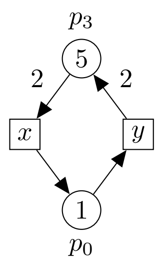
The generalization method proposed in Section 3.2 may introduce extra behavior in the discovered model since the new polyhedron covers more points. If we take into account negative information (forbidden behavior), the proposed encoding needs to be refined to rule out certain solutions. In order to avoid forbidden traces to be executable in the final model, each of them (but not its prefixes since they contain some positive behavior, see Definition 1) is converted into its Parikh representation. If one intends to reduce the complexity of each half-space in isolation, the SMT-encoding should be such that each negative point should not be a solution of the new half-space. However this might be too restrictive since a solution accepted by a single half-space may be ruled out by the rest of the system. If we consider Figure 4 (right) as the original model with $(9, 4)$ as a negative point, it can be seen that one of the half-spaces of the figure on the left accepts this solution and thus an approach that considers half-spaces in isolation would not allow this transformation. However this point is still ruled out by the other half-space and therefore the new encoding presented in this article may generate this polyhedron.

As in the case of the positive perspective, the transformation can be improved by considering the complete set of half-spaces (the whole matrix) rather than one half-space at a time. To avoid forbidden traces, the following encoding is used; for each negative point $(k_1, \ldots, k_n)$

$$\bigvee_{i=1}^{m} (\beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} \times k_j) < 0 \tag{NP}$$

The encoding above forces for each negative point at least one half-space to forbid it as a solution. By using negative information we can avoid constraint (NZ); when using just positive information this constraint is needed to avoid the trivial solution that removes every half-space. In the case of negative information being present, we can remove some places as far as they do not introduce negative behavior (the trivial solution will never be a solution since it accepts negative points).

Going back to Example 2, if we want to reduce the complexity of the system while ruling out the point $(6, 1)$ corresponding to the behavior $xxxyxxx$, the new encoding should add the constraint $(\beta_{1,0} + \beta_{1,1} \times 6 + \beta_{1,2} \times 1 < 0) \vee (\beta_{2,0} + \beta_{2,1} \times 6 + \beta_{2,2} \times 1 < 0)$ which rules out $\beta_{1,0} = 6, \beta_{1,1} = -1, \beta_{1,2} = 2, \beta_{2,0} = 1, \beta_{2,1} = 1, \beta_{2,2} = -1$ as a solution. The new method using negative information proposes to replace the first half-space by $5 - 2 \times \widehat{\sigma}(x) + 2 \times \widehat{\sigma}(y) \geq 0$ while keeping the second one unchanged. The simplified net (see above) and does not accept $xxxyxxx$ as a trace.



Limitations of the Negative Information Abstraction. The abstraction of the negative information based on the Parikh representation of the traces imposes two limitations in our technique. *(i)* 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, geometrically this is not true in

Algorithm 2 Automatic Half-space Removal

```
1: procedure Removal($H, np$)
2:     for $h \in half\text{-}spaces(H)$ do
3:         if $\neg$SomeInside($H \setminus h, np$) then
4:             remove $h$ from $H$
5:     return $H$
```

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, negative points must not be inside the polyhedron. *(ii)* Since we only translate the whole negative trace into a point, but not its prefixes as in the case of positive traces, the abstraction "looses precision". Consider the positive trace ab and the negative trace ba (where b is allowed, but ba is not). This situation leads to point $(1, 1)$ (one instance of a and one instance of b) being both a positive and negative point and therefore any set containing all the positive points using the Parikh abstraction will also contain a negative point.

3.4. Automatic Half-space Removal

The last step of our approach is an automatic removal of half-spaces done by the Removal procedure. This procedure takes as an input a polyhedron and a set of negatives points. It then iterates over all half-spaces and checks if the new polyhedron obtained by removing such half-space accepts any negative point. If the answer is no, then the half-space can be removed without introducing any prohibited behavior to the final net. The pseudo-code of the procedure can be found in Algorithm 2. SomeInside($H \setminus h, np$) returns true if some point in np is a solution of H after removing h.

3.5. Complexity Reduction

We formalize now the notion of complexity reduction used in this article. The idea is to minimize the coefficients of the incidence matrix of the net. The complexity of a given net is therefore the sum of the initial tokens and the tokens consumed or produced by each transition. Given a net with initial marking $(\alpha_{1,0}, \ldots, \alpha_{m,0})$ and incidence matrix

$$\begin{pmatrix} \alpha_{1,1} & \dots & \alpha_{1,n} \\ \vdots & & \vdots \\ \alpha_{m,1} & \dots & \alpha_{m,n} \end{pmatrix}$$

its *structural complexity* is given by $\sum\limits_{i=1}^{m}(|\alpha_{i,0}| + \sum\limits_{j=1}^{n}|\alpha_{i,j}|)$. With this definition the complexity of polyhedra $\{p_0, p_1\}$ and $\{p_0, p_2\}$ in Figure 2 are 14 and 12 respectively. Therefore we consider the second polyhedron and the corresponding net simpler since its complexity is smaller, i.e. simplicity is computed as the inverse of complexity.

3.6. Reducing the Complexity of Arbitrary Models

An important observation can be made at this point: the techniques presented in the previous section can be applied on top of any Petri net satisfying our assumption (pure and with neither silent transitions nor two transitions representing the same action) and hence are not dependent on the discovery technique from [5]. In Section 2.3, the correspondence between a polyhedron and a Petri net is shown by observing that the H-representation of $\mathcal{P}$ represents the marking equation of the corresponding Petri net N. In the previous sections we were using this correspondence in the forward direction, i.e. for computing N from $\mathcal{P}$. To enable the application of the SMT-based reduction 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. Hence, a Petri net can be obtained by some discovery technique and the SMT-encoding can be used to reduce the complexity of the model as a post processing technique.

4. Evaluation

We run our approach as described in Section 3 on several artificial and real-life logs. To illustrate the general applicability of the approach as described in Section 3.6, we also apply our technique on models obtained by ILP miner by [25] which can also discover fitting nets. We evaluate the quality of discovered models using the state-of-the-art techniques by [1] to measure precision and the approach by [27] to measure generalization; complexity is measured using the metric described in Section 3.5. We remark that, since the discovery method based on the theory of polyhedra and the ILP miner generate fitting models, Theorem 1 guarantees that all the generated nets are fitting.

4.1. The PacH Tool

Algorithm 1 has been implemented in a new command-line tool called PacH which is written in Python. The ConvexHull procedure is implemented using the pyhull package, a Python wrapper for Qhull [3] and uses the sampling and projection techniques introduced by [5]. All the encodings (Shift&Rotate procedure) are implemented using the SMT-solver Z3 [8]. The tool supports not only the SMT-encoding presented in this article, but also the encoding to reduce the complexity of half-spaces in isolation. The tool is available from

`http://github.com/lucionardelli/PacH`

It reads logs (containing allowed and forbidden traces) in XES format and generates Petri nets in PNML format. A detailed list of commands for the tool can be found on its web site.

4.2. Supervised Process Discovery

Here we report on the results on the discovery time together with the complexity, precision and generalization of the obtained nets. The set of benchmarks (allowed and forbidden traces) is introduced in Table 1 where we report the number of traces in the log $|\mathcal{L}|$, its number of events $|E|$ (i.e. the sum of the lengths of the traces) and the total number of activities $|T|$. Forbidden traces were generated using the algorithm by [27] with an additional random postfix added to the end of each generated forbidden trace, with the length of the postfix equal to the length of the forbidden trace (effectively doubling the length of each resulting forbidden trace).

	Positive Traces			Negative Traces		
Benchmark	$\lvert\mathcal{L}\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert E\rvert$	$\lvert T\rvert$
A(32)	100	2483	32	100	3134	32
A(42)	100	3308	42	100	3484	42
Choice	300	2400	12	300	3144	12
ConfDimB	500	3725	11	500	5476	11
Cycles(5)	100	4000	20	100	3728	20
DbMut(2)	500	8204	32	500	11904	32
DocumentFlow	1000	5328	59	1000	7570	59
FHMexample	1000	13837	13	1000	19188	13
Incident	1000	4931	18	1000	9168	18
Receipt	1434	8577	27	1434	13968	27
Svn	765	7959	13	765	24612	13
T(32)	100	3766	33	100	3716	33
Telecom	17812	83286	42	17812	159090	42

Table 1: The set of logs included in the benchmark and their corresponding sizes.

In Tables 2-5, for each benchmark we report results for the following experimental set up (the results highlighted with a grey background corresponds to the new methods introduced by this paper)

- **Polyhedra:** the net was mined using the theory of polyhedra with no complexity reduction.
- **Half-space:** the net was mined using the theory of polyhedra and the SMT-encoding that reduces the complexity of half-spaces in isolation.
- **Matrix:** the net was mined using the theory of polyhedra and the SMT-encoding that reduces the complexity of the whole incidence matrix.
- **Removal:** the Removal procedure was run on top of the net obtained in the 1st column (Baseline) or on top of the net obtained in the 2nd column (SMT).

To calculate the results, we apply a ten-fold cross-validation strategy: for each input event log, we split the log in ten equally sized new logs. Models are discovered and simplified over nine out of ten sub-logs, leaving out the remaining 10 percent of traces, repeated ten times (e.g. in the first fold, a net is discovered whilst leaving out log number 1, in the second fold, a net is discovered whilst leaving out log number 2, and so on). To obtain conformance checking results, each originally left-out sub-log in each of the ten folds is replayed over the discovered net, hence obtaining out-of-fold results. At the end of the run, final

		Positive		Positive/Negative		Removal	
Benchmark	Polyhedra	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	40.07	273.65	9.11	15.73	9.57	317.28	366.59
A(42)	3.82	182.78	7.18	11.63	9.97	363.10	357.57
Choice	0.10	0.41	0.73	0.29	0.75	0.64	0.62
ConfDimB	0.05	0.32	0.46	0.26	0.73	0.56	0.58
Cycles(5)	0.16	1.37	0.88	0.59	0.98	2.09	1.87
DbMut(2)	0.26	3.35	1.13	0.97	1.39	10.57	10.52
DocumentFlow	0.19	0.62	1.01	0.53	1.20	1.27	1.23
FHMexample	0.21	5.95	0.93	0.69	0.84	17.97	17.60
Incident	0.12	15.03	1.13	1.07	1.19	7.94	8.36
Receipt	0.23	10.48	1.35	1.50	1.70	42.45	55.61
Svn	0.22	63.96	2.10	3.26	1.53	341.09	372.00
T(32)	871.47	610.30	22.08	38.95	30.06	3451.96	2101.92
Telecom	1.66	21.48	1.47	1.46	2.00	77.10	61.31

Table 2: Computational times (in secs) of PacH.

values are obtained by averaging the results over all folds.

Results on the computation time of PacH are reported in Table 2. The first column reports the time to compute the convex hull; the rest of the columns reports on the complexity reduction time. In general the Removal procedure is the most time consuming method since it needs to iterate over all the half-spaces and all the negative points. Since for the SMT encodings we follow an iterative approach to obtain the optimal solution, several iterations are usually needed for the results in the 2^{nd} and 4^{th} columns. This produces a high increase in times of the 2^{nd} column for nets having several places such as A(32), A(42), Svn or T(32); the same is not true for the 4^{th} column since the SMT solver normally returns an *unsat* solution and then the iterative mode to find the optimal solution is not used in all the cases. On the other hand, the encoding to reduce the complexity of the whole matrix obtains an optimal solution after just a couple of iterations or not solution at all and thus the times are usually low. From the first column, it can be observed that A(32) and T(32) consume much more time than the rest; this is due to the fact that the projection technique needs to be used and this consumes much of the computational time.

In Table 3 we report the results on structural complexity achieved by our algorithms. Trivially the first column reports the highest values since no reduction is done. For all the cases the best results (shown in bold font) are obtained

		Positive		**Positive/Negative**		Removal	
Benchmark	Polyhedra	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	13129	10324	✗	✗	6489	132	**130**
A(42)	7813	6209	✗	✗	5009	117	**114**
Choice	33	29	✗	✗	✗	25	**21**
ConfDimB	33	31	29	✗	30	**25**	27
Cycles(5)	105	104	✗	✗	102	**45**	**45**
DbMut(2)	134	120	✗	✗	126	74	**72**
DocumentFlow	56	52	50	✗	52	35	**29**
FHMexample	297	215	295	✗	235	72	**43**
Incident	406	316	✗	✗	292	104	**75**
Receipt	588	412	✗	✗	462	129	**101**
Svn	27831	21051	✗	✗	7399	8778	**1562**
T(32)	50117	40741	✗	✗	34389	188	**114**
Telecom	840	592	✗	✗	688	197	**111**

Table 3: Experimental results on the complexity of the models mined by PacH.

after applying the Removal procedure on top of the net obtained after SMT-half-space reduction with positive information (i.e. the net obtained in the 2^{nd} column). It can be also observed that no reduction was achieved (entries with a cross) when using either the matrix reduction with just positive information or the half-space reduction with negative traces. The former is due to the fact that the search space for the SMT-solver is big and in most of the cases a timeout (600 milliseconds) is reached. However by using negative points, new constraints are used to encode negative points and this reduces the search space as can be seen by the solutions obtained in the 5^{th} column. The crosses on the 4^{th} column correspond to the fact that for the half-space reduction, each half-space must forbid *every* negative point and the generated encoding is very restrictive (this is due to the phenomenon explained in Figure 4). In this case the SMT solver returned either *unsat* or reached a timeout.

Tables 4 and 5 report on the precision and generalization of the nets mined by PacH. For the entries marked with N/A, the conformance checking metric was unable to retrieve a result due to out-of-memory errors. From Table 4, it can be observed that the SMT-based reduction with only positive information drastically decreases precision of the nets in some cases (see Choice or ConfDimB for example). However, the drop is very small when adding negative information; in most of the cases the matrix simplification did not even

		Positive		**Positive/Negative**		Removal	
Benchmark	Polyhedra	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	**0.16**	0.13	✗	✗	0.15	0.12	0.11
A(42)	**0.11**	0.10	✗	✗	**0.11**	0.08	0.08
Choice	**0.95**	0.21	✗	✗	**0.95**	**0.95**	0.20
ConfDimB	**0.87**	0.26	0.41	✗	**0.87**	**0.87**	0.25
Cycles(5)	**0.23**	**0.23**	✗	✗	**0.23**	0.22	0.22
DbMut(2)	0.26	0.24	✗	✗	**0.27**	0.25	0.23
DocumentFlow	**0.04**	**0.04**	**0.04**	✗	**0.04**	**0.04**	**0.04**
FHMexample	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Incident	**0.13**	0.12	✗	✗	0.11	0.12	0.12
Receipt	**0.15**	0.13	✗	✗	**0.15**	0.14	0.12
Svn	**0.15**	0.14	✗	✗	**0.15**	0.14	0.14
T(32)	0.13	0.12	✗	✗	**0.14**	0.11	0.11
Telecom	**0.08**	**0.08**	✗	✗	**0.08**	**0.08**	**0.08**

Table 4: Experimental results on the precision of the models mined by PacH.

		Positive		**Positive/Negative**		Removal	
Benchmark	Polyhedra	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	0.65	0.66	✗	✗	0.66	**0.70**	**0.70**
A(42)	0.63	0.64	✗	✗	0.63	0.71	**0.72**
Choice	**1.00**	**1.00**	✗	✗	**1.00**	**1.00**	**1.00**
ConfDimB	**0.83**	**0.83**	**0.83**	✗	0.81	**0.83**	**0.83**
Cycles(5)	**0.90**	**0.90**	✗	✗	**0.90**	**0.90**	**0.90**
DbMut(2)	**1.00**	**1.00**	✗	✗	**1.00**	**1.00**	**1.00**
DocumentFlow	0.97	0.97	0.97	✗	0.97	**1.00**	**1.00**
FHMexample	N/A	N/A	N/A	✗	N/A	N/A	N/A
Incident	**0.98**	**0.98**	✗	✗	**0.98**	**0.98**	**0.98**
Receipt	**0.87**	**0.87**	✗	✗	**0.87**	0.86	0.86
Svn	0.99	0.99	✗	✗	0.99	**1.00**	**1.00**
T(32)	0.53	0.54	✗	✗	0.51	**0.56**	**0.56**
Telecom	**1.00**	**1.00**	✗	✗	**1.00**	**1.00**	**1.00**

Table 5: Experimental results on the generalization of the models mined by PacH.

reduce the precision at all. In terms of generalization (see Table 5) minimal and maximal values are very close. For half of the benchmarks, PacH with no reduction obtains already the best results; for the remaining cases, the best generalization is obtained after performing Removal.

To summarize, the 5th and 6th columns from Tables 4 and 5 show that precision and generalization remain almost the same when using the matrix complexity reduction with negative information or the Removal procedure; this combined with the results of Table 3 that show that those methods perform very well on complexity reduction suggest that these combinations result on the best trade-off between the quality metrics.

		Positive		Positive/Negative		Removal	
Benchmark	ILP	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	15.19	7.16	1.66	2.43	1.62	8.50	18.96
A(42)	67.79	17.95	4.10	6.11	4.12	32.85	98.49
Choice	2.41	0.33	0.16	0.17	0.16	0.06	0.15
ConfDimB	2.20	0.30	0.31	0.20	0.16	0.18	0.26
Cycles(5)	10.33	0.37	0.19	0.31	0.19	0.20	0.79
DbMut(2)	19.92	1.52	0.55	0.72	0.55	N/A	N/A
DocumentFlow	549.82	25.61	7.02	9.07	6.93	61.79	76.45
FHMexample	31.29	1.29	0.44	0.57	0.44	1.00	3.40
Incident	8.83	3.10	0.70	0.88	0.70	2.84	3.17
Receipt	8.69	3.36	0.79	1.03	0.76	8.67	10.36
Svn	23.21	1.12	0.37	0.46	0.36	2.16	6.81
T(32)	64.60	10.97	2.44	3.66	2.41	15.01	44.54
Telecom	1347.14	14.43	2.73	3.81	2.73	409.43	528.16

Table 6: Computational times (in secs) of the ILP miner and the reductions made by PacH.

4.3. Improving the Complexity of Arbitrary Discovered Nets

To show the broad applicability of our method, we run the ILP miner by [25] on the same set of benchmarks and fed PacH with the resulting nets to apply the Removal procedure and SMT-based reductions. For Tables 6-9 we use the same experimental set up as above, but the baseline discovery method is the ILP miner instead of the theory of polyhedra.

Results on the computation time of the ILP miner and PacH are reported in Table 6. The first column reports the time to mine the net; the rest of the columns reports on the reductions time obtained by PacH. For all the cases, but those requiring projection, PacH performs much faster than the ILP miner. For the DbMut(2) benchmark, our tool crashed when trying to apply the Removal procedure and thus for this combination of methods no answer is reported in any of the tables.

In terms of complexity (see Table 7), for most of the results the same conclusions can be made as those using the theory of polyhedra. The only difference is that no reduction was done by matrix reduction with negative information due either to an *unsat* answer by the SMT solver or a reached timeout. This coincides with our intuition that the theory of polyhedra generates complex models due to the fact that it computes not any hull covering the points, but the *minimal* one. Due to this extra complexity, there is a lot of space for improvement

		Positive		**Positive/Negative**		REMOVAL	
Benchmark	ILP	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	396	289	✗	✗	✗	100	**97**
A(42)	933	678	✗	✗	✗	**124**	175
CHOICE	24	**21**	✗	✗	✗	✗	**21**
CONFDIMB	24	**23**	**23**	✗	✗	✗	**23**
CYCLES(5)	30	✗	✗	✗	✗	✗	✗
DBMUT(2)	92	**77**	✗	✗	✗	N/A	N/A
DOCUMENTFLOW	2426	1463	✗	✗	✗	937	**695**
FHMEXAMPLE	102	85	✗	✗	✗	✗	**72**
INCIDENT	194	107	✗	✗	✗	150	**77**
RECEIPT	183	121	✗	✗	✗	120	**89**
SVN	82	68	✗	✗	✗	65	**47**
T(32)	606	441	✗	✗	✗	**108**	149
TELECOM	777	486	✗	✗	✗	341	**194**

Table 7: Experimental results on the complexity of the models mined by the ILP miner and simplified by PACH.

in terms of complexity and SMT performs very well.

		Positive		**Positive/Negative**		REMOVAL	
Benchmark	ILP	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	**0.46**	0.08	✗	✗	✗	0.21	0.07
A(42)	**0.32**	0.09	✗	✗	✗	0.13	0.08
CHOICE	**0.98**	0.21	✗	✗	✗	✗	0.21
CONFDIMB	**0.99**	0.28	0.62	✗	✗	✗	0.28
CYCLES(5)	0.20	**0.22**	✗	✗	✗	✗	✗
DBMUT(2)	**0.22**	0.12	✗	✗	✗	N/A	N/A
DOCUMENTFLOW	**0.15**	0.12	✗	✗	✗	**0.15**	0.11
FHMEXAMPLE	**0.27**	**0.27**	✗	✗	✗	✗	0.26
INCIDENT	**0.24**	0.13	✗	✗	✗	0.23	0.12
RECEIPT	**0.22**	0.08	✗	✗	✗	0.21	0.08
SVN	**0.11**	0.10	✗	✗	✗	0.10	0.10
T(32)	**0.39**	0.09	✗	✗	✗	0.22	0.08
TELECOM	N/A	0.03	✗	✗	✗	0.22	0.20

Table 8: Experimental results on the precision of the models mined by the ILP miner and simplified by PACH.

Precision results for the ILP miner are reported in Table 8. Here the results differ from the ones of the theory of polyhedra. The best values are obtained from the ILP miner (as expected since a reduction in complexity usually decreases precision) and the closest ones are still obtained by applying REMOVAL on top of baseline; however the drop in precision is higher than in the case of polyhedra. For the first two benchmarks, precision drops to half and one third

		Positive		Positive/Negative		REMOVAL	
Benchmark	ILP	Halfspace	Matrix	Halfspace	Matrix	Baseline	SMT
A(32)	0.63	0.77	✗	✗	✗	0.68	**0.80**
A(42)	0.60	0.68	✗	✗	✗	0.68	**0.75**
CHOICE	**1.00**	**1.00**	✗	✗	✗	✗	**1.00**
CONFDIMB	**0.83**	**0.83**	**0.83**	✗	✗	✗	**0.83**
CYCLES(5)	**0.90**	**0.90**	✗	✗	✗	✗	✗
DBMUT(2)	0.42	**0.56**	✗	✗	✗	N/A	N/A
DOCUMENTFLOW	0.88	0.91	✗	✗	✗	0.91	**0.93**
FHMEXAMPLE	0.88	**0.96**	✗	✗	✗	✗	**0.96**
INCIDENT	0.97	**0.99**	✗	✗	✗	0.97	**0.99**
RECEIPT	0.67	**0.84**	✗	✗	✗	0.66	**0.84**
SVN	**1.00**	**1.00**	✗	✗	✗	**1.00**	**1.00**
T(32)	0.48	0.63	✗	✗	✗	0.53	**0.68**
TELECOM	**1.00**	**1.00**	✗	✗	✗	**1.00**	**1.00**

Table 9: Experimental results on the generalization of the models mined by the ILP miner and simplified by PACH.

respectively where in the case of polyhedra the difference between these two columns were always below 5%. In terms of generalization (Table 9), the best results are always obtained after applying REMOVAL on top of the half-space reduction. However for most of the cases the same results are already obtained in the 2^{nd} column.

4.4. Comparison with an Unfolding-based Simplification Techniques

Our complexity reduction can be seen as a simplification technique; we now compare the results of PACH with respect to a recently introduced simplification method based on unfoldings by [10] which is implemented in the UMA package of the PROM framework. Since the latter cannot be applied to the models mined using the theory of polyhedra (due to the restriction that unfolding techniques must be applied to safe nets with no weight arcs[6]) we only apply both algorithms on the models mined by the ILP miner and use the same ten-fold cross-validation strategy as in the previous experiments. Since this simplification technique is complementary to our method, we also evaluate the results of applying them both, i.e. we mine the net with the ILP miner, we make a first simplification

[6]The safeness restriction can be relaxed following [13] to apply unfoldings to general Petri nets, however we are not aware of any tool implementing such general approach.

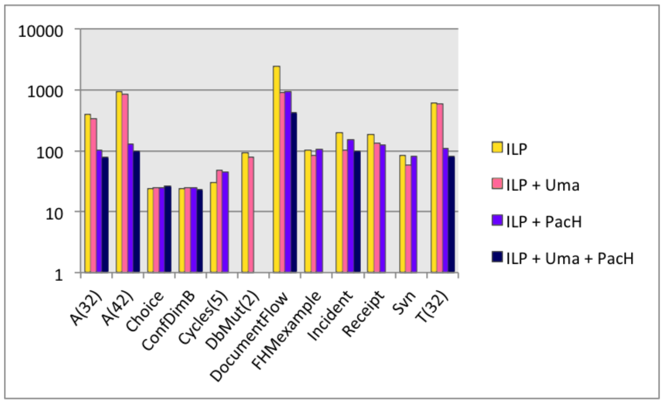


Figure 5: Results on the complexity of the nets generated by our method and an unfolding-based simplification one (optimal values are the lowest ones).

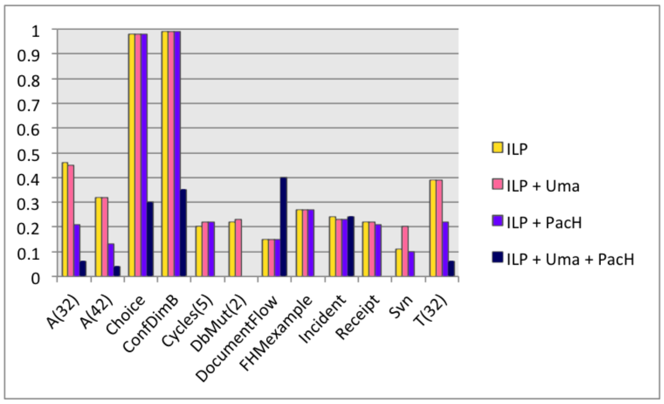


Figure 6: Results on the precision of the nets generated by our method and an unfolding-based simplification one (optimal values are the highest ones).

using Uma and then we apply PacH. Figures 5-7 display the results of both methods in terms of complexity, precision and generalization. We report on the original values (no simplification), the values after applying Uma, the ones after applying the Removal procedure from PacH and the results of applying both methods together. Surprisingly, even if Uma does not consider negative traces, for this set of benchmarks and the automatically generated negative information, the resulting models do not accept any negative trace.

From Figure 5 it can be observed that the Removal procedure from PacH is the one obtaining the best results in terms of reducing the complexity of the

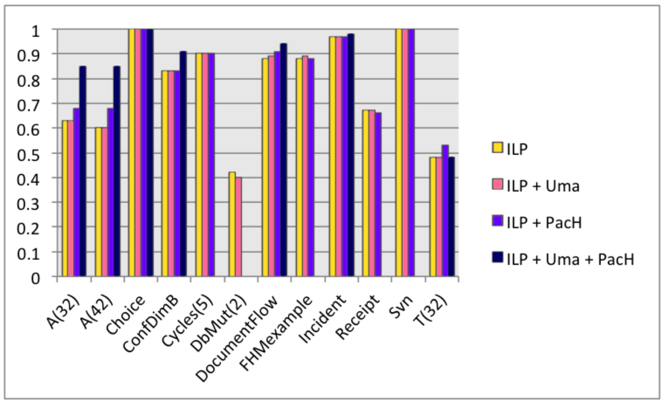


Figure 7: Results on the generalization of the nets generated by our method and an unfolding-based simplification one (optimal values are the highest ones).

net (lowest values correspond to the best results). However this procedure can be applied on top of the results obtained by Uma resulting in the optimal results. Figure 6 shows what can be considered as a drawback of our technique: the degradation of precision (specifically when using the Removal procedure which is the most aggressive one in terms of simplification). This happens in both set ups where the Removal procedure is applied ("ILP + PacH" and "ILP + Uma + PacH"). This is not a surprise considering that reducing complexity and improving generalization normally imply a reduction in precision. In terms of generalization (Figure 7) most of the scores are similar, however for some benchmarks the combination "ILP + Uma + PacH" reports the optimal results. From the three figures it can be concluded that both techniques are complementary and the use of both of them together guarantees a good compromise in terms of the three quality metrics.

If we do not combine both approaches, it is clear that for this set of benchmarks Uma outperforms PacH. However for complex systems like found in manufacturing, the safeness assumption made by Uma would generate models with low precision values and which miss important relations between activities. Consider the following set of traces (taken from [5]) which shows a non-unitary synchronic distance between the firing of activities a and b.

a b b a a b b a a b a a b a b b a a a a

```
a a a b b b a a b a b b b a a b b a b a
b a a b a a a a b a b b b b b a a b a b
b a b a a b b a a b a b a a a a b a b a
a a a b a b b a a b a b b a b b a a a a
b a a a a b a a b b a a a b a b a a b b
b a b a b a a b a a a a b b a b a a b a
b a b a b a b a b a a a b b a b b a b a
a b b a a a b b b a b a a a a a a a b b
a b b a b a a a b a a a b a a b a b b a
```

In this case, it is possible to have three a's without any b, so the relation between them is non-unitary. This represents a typical situation in manufacturing systems where manufacturing a product requires several parts, each being repeated several times. Uma generates a model that cannot capture these relations between a and b and instead overuses duplicate labels. On the contrary, the theory of polyhedra is general enough to produce the whole class of P/T nets which can capture the non unitary relation between activities a and b.

4.5. Discussion

In summary, the evaluation performed in this section shows important tendencies that we would like to explicit now. First, it is clear that some of the methods presented in this paper tend to alleviate the complexity of the derived net. This capability can be boosted when the techniques are combined with state-of-the art simplification techniques like [10]. Second, precision is often negatively correlated with respect to the complexity alleviation, a drawback of the techniques proposed in this paper. In contrast, generalization is positively correlated with the produced complexity alleviation, showing a clear benefit of using the techniques of this paper. Finally, the computation overhead is acceptable given the quality impact in the derived nets.

5. Related Work

Several techniques exist for the discovery of workflow nets [23], a very restricted class of Petri nets. Other techniques which focus on similar formalisms (*heuristics/causal nets* [28] or *fuzzy nets* [24]) are again restricted in terms of expressivity, making them unsuitable for capturing the type of general behavior considered in this paper. In terms of expressive power and capability to generate fitting models, the only approaches in the literature closer to ours are grounded on the *theory of regions*: the works by [4, 26, 6, 22] can discover unrestricted models but do not incorporate negative information as we report in this paper.

Very few approaches exist towards binary, two-class supervised process discovery, compared to the multitude of process discovery techniques that work in a one-class setting where only one class of traces is given. Among the first to investigate the use of binary-class supervised techniques to predict dependency relationships between activities is [16], in which a binary classification algorithm is trained on a table of metrics for each activity. Inductive logic programming and partial-order planning techniques are applied to derive a process model in [11]. Here, 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. An extension of logic programming (SCIFF) is applied towards supervised declarative process discovery by [14, 15] and [2], 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, [12] 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. To guide the learning process, an input event log is supplemented with induced artificial negative events, similar as in this work.

A recent discovery technique that also uses negative information is presented by [20]. The intuitive idea is to obtain an *unfolding* from an event log using some independence information between activities which is given as input. Then a folding step is performed to derive a process model that further generalizes the behavior in a controlled manner, e.g., including cycles, but without introducing any forbidden behavior. Unfortunately, an artifact of using the unfolding as intermediate representation is the restriction to *safe* Petri nets, i.e., nets with at most one token per place. This contrasts with the general models considered in this paper.

In terms of simplicity, several metrics have been proposed in the literature (see for example the work of [17]), most of them related to the visualization of the final model, i.e. its graphical representation. However those metrics were usually defined for more restricted classes of Petri nets where the main objective is the representation of the workflow of the process. The kind of nets we consider (P/T nets) are more general and allow to represents concepts such as resources and costs.

Murata mentioned in his survey about Petri nets [18] several simplification rules which preserve liveness and reachability of the system; however they do not necessarily preserve the behavior of the net (i.e. fitness). Redundant places (those that do not restrict transition firing) can be detected and removed following [7]. Some more recent techniques are also worth mentioning. First, the simplification technique presented by [10] also describes an automatic method to simplify a Petri net that relies on the computation of the unfolding of the net in order to preserve only the paths that lead to a sound generalization. Since this technique is based on unfoldings, it has the same limitation as the technique introduced by [20], i.e. it can only be applied to safe nets which restricts its applicability. Additionally, this technique does not consider negative information at all and cannot thus guarantee that certain unwanted behavior is not introduced in the simplification step.

Another recent simplification technique is presented by [9]. This technique makes a trade-off between the graphical representation and the quality metrics of the net and decides what arcs to remove by encoding the problem as an optimization problem. They approach can be used in combination with the methods of this paper to further simplify a model.

6. Conclusions and Future Work

We have presented a process discovery approach based on numerical abstract domains and SMT which is able to reduce the complexity 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 binary-class discovery techniques and argue that this feature may be crucial for deriving process models which are less complex, fitting and precise, but also good on generalizing the right behavior underlying an event log. Experiments performed in our implementation show the effectiveness of the techniques both as a discovery algorithm or a post processing technique.

Regarding 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, as mentioned above, 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 polyhedra covering only the positive points, for which merging methods would need to be investigated. Related to this issue is the aspect of noise, which also forms an interesting avenue for future work. The following strategies can be identified as possible ways to tackle noise. First, using existing frequency-based pre-processing filters on the event log to remove "outlier"-activities (i.e. activities occurring in a very rarely seen context in the event log). Second, the removal of halfspaces even if they lead to the inclusion of negative information also forms an interesting possibility. If the number of negative points that would be included are few, one might still consider the removal as the negative points might be due to noise being present in the negative information. These are potentially useful improvements worth exploring in follow-up work. Finally, we plan to set up a thorough experiment in to investigate the effects of our approach on models mined by various miners and plan to continue developing PacH.

Acknowledgments

We thank Jordi Cortadella for interesting discussions on the topic of this paper, Sebastiaan J. van Zelst for providing a tailored version of the *hybrid ILP* technique, and Javier De San Pedro for his help with the PNsimpl tool. 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 project 277522.

References

[1] Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Measuring precision of modeled behavior. *Inf. Syst. E-Business Management*, 13(1):37–67, 2015.

[2] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Verifiable agent interaction in abductive logic programming: The SCIFF framework. *ACM Trans. Comput. Log.*, 9(4), 2008.

[3] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, December 1996.

[4] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In *BPM*, pages 375–383, 2007.

[5] Josep Carmona and Jordi Cortadella. Process discovery algorithms using numerical abstract domains. *IEEE Trans. Knowl. Data Eng.*, 26(12):3064–3076, 2014.

[6] Josep Carmona, Jordi Cortadella, and Michael Kishinevsky. New region-based algorithms for deriving bounded petri nets. *IEEE Trans. Computers*, 59(3):371–384, 2010.

[7] José Manuel Colom and Manuel Silva. Improving the linearly based characterization of P/T nets. In *Petri Nets*, pages 113–145, 1989.

[8] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, pages 337–340, 2008.

[9] Javier de San Pedro, Josep Carmona, and Jordi Cortadella. Log-based simplification of process models. In *BPM*, pages 457–474, 2015.

[10] Dirk Fahland and Wil M. P. van der Aalst. Simplifying discovered process models in a controlled manner. *Inf. Syst.*, 38(4):585–605, 2013.

[11] Hugo M. Ferreira and Diogo R. Ferreira. An integrated life cycle for workflow management based on learning and planning. *Int. J. Cooperative Inf. Syst.*, 15(4):485–505, 2006.

[12] Stijn Goedertier, David Martens, Jan Vanthienen, and Bart Baesens. Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10:1305–1340, 2009.

[13] Jonathan Hayman and Glynn Winskel. The unfolding of general petri nets. In *FSTTCS*, volume 2 of *LIPIcs*, pages 223–234. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.

[14] Evelina Lamma, Paola Mello, Marco Montali, Fabrizio Riguzzi, and Sergio Storari. Inducing declarative logic-based models from labeled traces. In *BPM*, pages 344–359, 2007.

[15] Evelina Lamma, Paola Mello, Fabrizio Riguzzi, and Sergio Storari. Applying inductive logic programming to process mining. In *ILP*, pages 132–146, 2007.

[16] Laura Maruster, A. J. M. M. Weijters, Wil M. P. van der Aalst, and Antal van den Bosch. A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. *Data Min. Knowl. Discov.*, 13(1): 67–87, 2006.

[17] Jan Mendling, Gustaf Neumann, and Wil M. P. van der Aalst. Understanding the occurrence of errors in process models based on metrics. In *DOA, ODBASE, GADA, and IS*, pages 113–130, 2007.

[18] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[19] Hernán Ponce de León, Josep Carmona, and Seppe K. L. M. vanden Broucke. Incorporating negative information in process discovery. In *BPM*, pages 126–143, 2015.

[20] Hernán Ponce de León, César Rodríguez, Josep Carmona, Keijo Heljanko, and Stefan Haar. Unfolding-based process discovery. In *ATVA*, pages 31–47, 2015.

[21] Manuel Silva, Enrique Teruel, and José Manuel Colom. Linear algebraic and linear programming techniques for the analysis of place or transition net systems. In *Lectures on Petri Nets*, pages 309–373, 1996.

[22] Marc Solé and Josep Carmona. Light region-based techniques for process discovery. *Fundam. Inform.*, 113(3-4):343–376, 2011.

[23] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.

[24] Wil M. P. van der Aalst and Christian W. Günther. Finding structure in unstructured processes: The case for process mining. In *ACSD*, pages 3–12, 2007.

[25] Jan Martijn E. M. van der Werf, Boudewijn F. van Dongen, Cor A. J. Hurkens, and Alexander Serebrenik. Process discovery using integer linear programming. *Fundam. Inform.*, 94(3-4):387–412, 2009.

[26] Jan Martijn E. M. van der Werf, Boudewijn F. van Dongen, Cor A. J. Hurkens, and Alexander Serebrenik. Process discovery using integer linear programming. *Fundam. Inform.*, 94(3-4):387–412, 2009.

[27] Seppe K. L. M. vanden Broucke, Jochen De Weerdt, Jan Vanthienen, and Bart Baesens. Determining process model precision and generalization with weighted artificial negative events. *IEEE Trans. Knowl. Data Eng.*, 26(8): 1877–1889, 2014.

[28] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible heuristics miner (FHM). In *CIDM*, pages 310–317, 2011.