# Unfolding-Based Process Discovery

Hernán Ponce-de-León[1], César Rodríguez[2], Josep Carmona[3], Keijo Heljanko[1], and Stefan Haar[4]

[1] Helsinki Institute for Information Technology HIIT and Department of Computer Science and Engineering, School of Science, Aalto University, Finland
[2] Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, France
[3] Universitat Politècnica de Catalunya, Barcelona, Spain
[4] INRIA and LSV, École Normale Supérieure de Cachan and CNRS, France

**Abstract.** This paper presents a novel technique for process discovery. In contrast to the current trend, which only considers an event log for discovering a process model, we assume two additional inputs: an independence relation on the set of logged activities, and a collection of negative traces. After deriving an intermediate net unfolding from them, we perform a controlled folding giving rise to a Petri net which contains both the input log and all independence-equivalent traces arising from it. Remarkably, the derived Petri net cannot execute any trace from the negative collection. The entire chain of transformations is fully automated. A tool has been developed and experimental results are provided that witness the significance of the contribution of this paper.

## 1 Introduction

The derivation of process models from partial observations has received significant attention in the last years, as it enables eliciting evidence-based formal representations of the real processes running in a system [1]. This discipline, known as *process discovery*, has similar premises as in *regression analysis*, i.e., only when moderate assumptions are made on the input data one can derive faithful models that represent the underlying system.

Formally, a technique for process discovery receives as input an *event log*, containing the footprints of a process' executions, and produces a model (e.g., a Petri net) describing the real process. Many process discovery algorithms in the literature make strong implicit assumptions. A widely used one is *log completeness*, requiring every possible trace of the underlying system to be contained in the event log. This is hard to satisfy by systems with cyclic or infinite behavior, but also for systems that evolve continuously over time. Another implicit assumption is the lack of *noise* in the log, i.e., traces denoting exceptional behavior that should not be contained in the derived process model. Finally, every discovery technique has a *representational bias*. For instance, the $\alpha$-algorithm [2] can only discover Petri nets of a specific class (*structured workflow nets*).

Few attempts have been made to remove the aforementioned assumptions. One promising direction is to relieve the discovery problem by assuming that more knowledge about the underlying system is available as input. On this line, the works in [3,4,5] are among the few that use domain knowledge in terms of

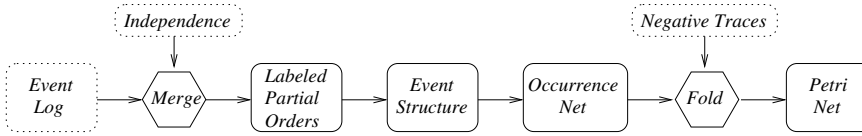


**Fig. 1.** Unfolding-based process discovery.

*negative information*, expressed by traces which do not represent process behavior. In this paper we follow this direction, but additionally incorporate a crucial information to be used for the task of process discovery: when a pair of activities are *independent* of each other. One example could be the different tests that a patient should undergo in order to have a diagnosis: blood test, allergy test, and radiology test, which are independent each other. We believe that obtaining this coarse-grain independence information from a domain expert is an easy and natural step; however, if they are not available, one can estimate them from analysing the log with some of the techniques in the literature, e.g., the relations computed by the $\alpha$-algorithm [1].

The approach of this paper is summarized in Fig. 1. Starting from an event log and an independence relation on its set of activities, we conceptually construct a collection of *labeled partial orders* whose linearizations include both the sequences in the log as well those in the same Mazurkiewicz trace [6], i.e., those obtained via successive permutations of independent activities. We then merge (the common prefixes of) this collection into an *event structure* which we next transform into an occurrence net representing the same behavior. Finally, we perform a controlled generalization by selectively folding the occurrence net into a Petri net. This step yields a net that (a) can execute all traces contained in the event log, and (b) generalizes the behavior of the log in a controlled manner, introducing no execution given in the collection of negative traces. The folding process is driven by a *folding equivalence relation*, which we synthesize using SMT. Different folding equivalences guarantee different properties about the final net. The paper proposes three different classes of equivalences and studies their properties. In particular we define a class of *independence-preserving* folding equivalences, guaranteeing that the natural independence relation in the final net will equal the one given by the expert.

In summary, the main contributions of the paper are:

- A general and efficient translation from prime event structures to occurrence nets (Section 3).
- Three classes of folding equivalences of interest not only in process discovery but also in formal verification of concurrent systems (Section 4).
- A method to synthesize folding equivalences using SMT (Section 5).
- An implementation of our approach and experimental results witnessing its capacity to even rediscover the original model (Section 6).

Remarkably, the discovery technique of this paper solves for the first time one of the foreseen operations in [7], which advocates for the unified use of event structures to support process mining operations.

Proofs for all formal results stated in the paper can be found in [8].

## 2 Preliminaries

*Events:* given an alphabet of actions $A$, several occurrences of a given action can happen on a run or execution. In this paper we consider a set $E$ of events representing the occurrence of actions in executions. Each event $e \in E$ has the form $e := \langle a, H \rangle$, where $a \in A$ and $H \subseteq E$ is a subset of events causing $e$ (its history). The label of an event is given by a function $\lambda : E \to A$ defined as $\lambda(\langle a, H \rangle) := a$.

*Labeled partial orders (lpos):* we represent a labelled partial order by a pair $(E, \leq)$, where $\leq \subseteq E \times E$ is a reflexive, antisymmetric and transitive relation on the set $E$ of events. Two distinct events $e, e' \in E$ can be either ordered ($e \leq e'$ or $e' \leq e$) or concurrent ($e \not\leq e'$ and $e' \not\leq e$). Observe that all events are implicitly labelled by $\lambda$.

*Petri nets:* a net consists of two disjoint sets $P$ and $T$ representing respectively places and transitions together with a set $F$ of flow arcs. The notion of state of the system in a net is captured by its markings. A marking is a multiset $M$ of places, i.e., a map $M : P \to \mathbb{N}$. We focus on the so-called safe nets, where markings are sets, i.e., $M(p) \in \{0, 1\}$ for all $p \in P$. A Petri net (PN) is a net together with an initial marking and a total function that labels its transitions over an alphabet $A$ of observable actions. Formally a PN is a tuple $\mathcal{N} := (P, T, F, \lambda, M_0)$ where *(i)* $P \neq \varnothing$ is a set of places; *(ii)* $T \neq \varnothing$ is a set of transitions such that $P \cap T = \varnothing$; *(iii)* $F \subseteq (P \times T) \cup (T \times P)$ is a set of flow arcs; *(iv)* $\lambda : T \to A$ is a labeling mapping; and *(v)* $M_0 \subseteq P$ is an initial marking. Elements of $P \cup T$ are called the nodes of $\mathcal{N}$. For a transition $t \in T$, we call ${}^\bullet t := \{p \mid (p, t) \in F\}$ the preset of $t$, and $t^\bullet := \{p \mid (t, p) \in F\}$ the postset of $t$. In figures, we represent as usual places by empty circles, transitions by squares, $F$ by arrows, and the marking of a place $p$ by black tokens in $p$. A transition $t$ is enabled in marking $M$, written $M \xrightarrow{t}$, iff ${}^\bullet t \subseteq M$. This enabled transition can fire, resulting in a new marking $M' := (M \setminus {}^\bullet t) \cup t^\bullet$. This firing relation is denoted by $M \xrightarrow{t} M'$. A marking $M$ is reachable from $M_0$ if there exists a firing sequence, i.e. transitions, $t_1, \ldots, t_n$ such that $M_0 \xrightarrow{t_1} \ldots \xrightarrow{t_n} M$. The set of reachable markings from $M_0$ is denoted by $\mathit{reach}(\mathcal{N})$. The set of co-enabled transitions is $\mathit{coe}(\mathcal{N}) := \{(t, t') \mid \exists M \in \mathit{reach}(\mathcal{N}) : {}^\bullet t \subseteq M \wedge {}^\bullet t' \subseteq M\}$. The set of observations of a net is the image over $\lambda$ of its fireable sequences, i.e., $\sigma \in \mathit{obs}(\mathcal{N})$ iff $M_0 \xrightarrow{t_1} \ldots \xrightarrow{t_n} M$ and $\lambda(t_1) \ldots \lambda(t_n) = \sigma$.

*Occurrence nets:* occurrence nets can be seen as infinite Petri nets with a special acyclic structure that highlights conflict between transitions that compete for resources. Places and transitions of an occurrence net are usually called conditions and events. Formally, let $N := (P, T, F)$ be a net, $<$ the transitive closure of $F$, and $\leq$ the reflexive closure of $<$. We say that transitions $t_1$ and $t_2$ are in structural conflict, written $t_1 \#_s t_2$, if and only if $t_1 \neq t_2$ and ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \varnothing$. Conflict is inherited along $<$, that is, the conflict relation $\#$ is given by $a \# b \Leftrightarrow \exists t_a, t_b \in T : t_a \#_s t_b \wedge t_a \leq a \wedge t_b \leq b$. Finally, the concurrency relation **co** holds between nodes $a, b \in P \cup T$ that are neither ordered nor in conflict, i.e. $a \ \mathbf{co} \ b \Leftrightarrow \neg(a \leq b) \wedge \neg(a \# b) \wedge \neg(b \leq a)$.

A net $\beta := (B, E, F)$ is an occurrence net iff *(i)* $\leq$ is a partial order; *(ii)* for all $b \in B$, $|{}^\bullet b| \in \{0, 1\}$; *(iii)* for all $x \in B \cup E$, the set $[x] := \{y \in E \mid y \leq x\}$ is finite; *(iv)* there is no self-conflict, i.e. there is no $x \in B \cup E$ such that $x \# x$. The initial marking $M_0$ of an occurrence net is the set of conditions with an empty preset, i.e. $\forall b \in B : b \in M_0 \Leftrightarrow {}^\bullet b = \varnothing$. Every $\leq$-closed and conflict-free set of events $C$ is called a configuration and generates a reachable marking defined as $Mark(C) := (M_0 \cup C^\bullet) \setminus {}^\bullet C$. We also assume a labeling function $\lambda : E \to A$ from events in $\beta$ to alphabet $A$. Conditions are of the form $\langle e, X \rangle$ where $e \in E$ is the event generating the condition and $X \subseteq E$ are the events consuming it. Occurrence nets are the mathematical form of the partial order unfolding semantics of a Petri net [9]; we use indifferently the terms occurrence net and unfolding.

Conditions in an occurrence net can be removed by keeping the causal dependencies and introducing a conflict relation; the obtained object is an event structure [10].

*Event structures:* an event structure is a tuple $\mathcal{E} := (E, \leq, \#)$ where $E$ is a set of events; $\leq \subseteq E \times E$ is a partial order (called causality) satisfying the property of finite causes, i.e. $\forall e \in E : |[e]| < \infty$ where $[e] := \{e' \in E \mid e' \leq e\}$; $\# \subseteq E \times E$ is an irreflexive symmetric relation (called conflict) satisfying the property of conflict heredity, i.e. $\forall e, e', e'' \in E : e \# e' \wedge e' \leq e'' \Rightarrow e \# e''$. Note that in most cases one only needs to consider reduced versions of relations $\leq$ and $\#$, which we will denote $\lessdot$ and $\#_d$, respectively. Formally, $\lessdot$ (which we call direct causality) is the transitive reduction of $\leq$, and $\#_d$ (direct conflict) is the smallest relation inducing $\#$ through the property of conflict heredity. A configuration is a computation state represented by a set of events that have occurred; if an event is present in a configuration, then so must all the events on which it causally depends. Moreover, a configuration does not contain conflicting events. Formally, a configuration of $(E, \leq, \#)$ is a set $C \subseteq E$ such that $e \in C \Rightarrow (\forall e' \leq e : e' \in C)$, and $(e \in C \wedge e \# e') \Rightarrow e' \notin C$. The set of configurations of $\mathcal{E}$ is denoted by $\Omega(\mathcal{E})$.

*Mazurkiewicz traces:* let $A$ be a finite alphabet of letters and $\diamond \subseteq A \times A$ a symmetric and irreflexive relation called independence. The relation $\diamond$ induces an equivalence relation $\equiv_\diamond$ over $A^*$. Two words $\sigma$ and $\sigma'$ are equivalent ($\sigma \equiv_\diamond \sigma'$) if there exists a sequence $\sigma_1 \dots \sigma_k$ of words such that $\sigma = \sigma_1, \sigma' = \sigma_k$ and for all $1 \leq i \leq k$ there exists words $\sigma_i', \sigma_i''$ and letters $a_i, b_i$ satisfying

$$\sigma_i = \sigma_i' a_i b_i \sigma_i'', \qquad \sigma_{i+1} = \sigma_i' b_i a_i \sigma_i'', \quad \text{and } (a_i, b_i) \in \diamond$$

Thus, two words are equivalent by $\equiv_\diamond$ if one can be obtained from the other by successive commutation of neighboring independent letters. For a word $\sigma \in A^*$ the equivalence class of $\sigma$ under $\equiv_\diamond$ is called a Mazurkiewicz trace [6].

We now describe the problem tackled in this paper, one of the main challenges in the *process mining* field [1].

*Process Discovery:* a log $\mathcal{L}$ is a finite set of traces over an alphabet $A$ representing the footprints of the real process executions of a system $\mathcal{S}$ that is only (partially) visible through these runs. Process discovery techniques aim at extracting from

a log $\mathcal{L}$ a process model $\mathcal{M}$ (e.g., a Petri net) with the goal to elicit the process underlying in $\mathcal{S}$. By relating the behaviors of $\mathcal{L}$, $obs(\mathcal{M})$ and $\mathcal{S}$, particular concepts can be defined [11]. A log is *incomplete* if $\mathcal{S} \backslash \mathcal{L} \neq \varnothing$. A model $\mathcal{M}$ *fits* log $\mathcal{L}$ if $\mathcal{L} \subseteq obs(\mathcal{M})$. A model is *precise* in describing a log $\mathcal{L}$ if $obs(\mathcal{M}) \backslash \mathcal{L}$ is small. A model $\mathcal{M}$ represents a *generalization* of log $\mathcal{L}$ with respect to system $\mathcal{S}$ if some behavior in $\mathcal{S} \backslash \mathcal{L}$ exists in $obs(\mathcal{M})$. Finally, a model $\mathcal{M}$ is *simple* when it has the minimal complexity in representing $obs(\mathcal{M})$, i.e., the well-known *Occam's razor principle*. It is widely acknowledged that the size of a process model is the most important simplicity indicator. Let $\mathcal{U}^{\mathcal{N}}$ be the universe of nets, we define a function $\hat{c} : \mathcal{U}^{\mathcal{N}} \to \mathbb{N}$ to measure the simplicity of a net by counting the number of some of its elements, e.g., its transitions and/or places.

## 3 Independence-Preserving Discovery

Let $\mathcal{S}$ be a system whose set of actions is $A$. Given two actions $a, b \in A$ and one state $s$ of $\mathcal{S}$, we say that $a$ and $b$ *commute* at $s$ when

- if $a$ can fire at $s$ and its execution reaches state $s'$, then $b$ is possible at $s$ iff it is possible at $s'$; and
- if both $a$ and $b$ can fire at $s$, then firing $ab$ and $ba$ reaches the same state.

Commutativity of actions at states identifies an equivalence relation in the set of executions of the system $\mathcal{S}$; it is a *ternary* relation, relating two transitions with one state.

Since asking the expert to provide the commutativity relation of $\mathcal{S}$ would be difficult, we restrict ourselves to unconditional independence, i.e., a conservative overapproximation of the commutativity relation that is a sole property of transitions, as opposed to transitions and states. An *unconditional independence* relation of $\mathcal{S}$ is any *binary*, symmetric, and irreflexive relation $\diamond \subseteq A \times A$ satisfying that if $a \diamond b$ then $a$ and $b$ commute at *every reachable state* of $\mathcal{S}$. If $a, b$ are not independent according to $\diamond$, then they are dependent, denoted by $a \circledast b$.

In this section, given a log $\mathcal{L} \subseteq A^*$, representing some behaviors of $\mathcal{S}$, and an arbitrary unconditional independence $\diamond$ of $\mathcal{S}$, provided by the expert, we construct an occurrence net whose executions contain $\mathcal{L}$ together with all sequences in $A^*$ which are $\equiv_\diamond$-equivalent to some sequence in $\mathcal{L}$.

If commuting actions are not declared independent by the expert (i.e., $\diamond$ is smaller than it could be), then $\mathcal{M}$ will be more sequential than $\mathcal{S}$; if some actions that did not commute are marked as independent, then $\mathcal{M}$ will not be a truthful representation of $\mathcal{S}$. The use of expert knowledge in terms of an independence relation is a novel feature not considered before in the context of process discovery. We believe this is a powerful way to fight with the problem of log incompleteness in a practical way since it is only needed to observe in the log one trace representative of a class in $\equiv_\diamond$ to include the whole set of traces of the class in the process model's executions.

Our final goal is to generate a Petri net that represents the behavior of the underlying system. We start by translating $\mathcal{L}$ into a collection of partial orders whose shape depends on the specific definition of $\diamond$.

**Definition 1.** *Given a sequence $\sigma \in A^*$ and an independence relation $\diamond \subseteq A \times A$, we associate to $\sigma$ a labeled partial order $lpo_\diamond(\sigma)$ inductively defined by:*

1. *If $\sigma = \varepsilon$, then let $\bot := \langle \tau, \varnothing \rangle$ and set $lpo_\diamond(\sigma) := (\{\bot\}, \varnothing)$.*
2. *If $\sigma = \sigma' a$, then let $lpo_\diamond(\sigma') := (E', \leq')$ and let $e := \langle a, H \rangle$ be the single event such that $H$ is the unique $\subseteq$-minimal, causally-closed set of events in $E'$ satisfying that for any event $e' \in E'$, if $\lambda(e') \otimes a$, then $e' \in H$. Then set $lpo_\diamond(\sigma) := (E, \leq)$ with $E := E' \cup \{e\}$ and $\leq := \leq' \cup (H \times \{e\})$.*

Since a system rarely generates a single observation, we need a compact way to model all the possible observations of the system. We represent all the partially ordered executions of a system with an event structure.

**Definition 2.** *Given a set of partial orders $S := \{(E_i, \leq_i) \mid 1 \leq i \leq n\}$, we define $ES(S) := (E, \leq, \#)$ where:*

1. *$E := \bigcup_{1 \leq i \leq n} E_i$,*
2. *$\leq := (\bigcup_{1 \leq i \leq n} \leq_i)^*$, and*
3. *for $e := \langle a, H \rangle$ and $e' := \langle b, H' \rangle$, we have that $e \#_d e'$ (read: $e$ and $e'$ are in direct conflict) iff $e' \notin H, e \notin H'$ and $a \otimes b$. The conflict relation $\#$ is the smallest relation that includes $\#_d$ and is inherited w.r.t. $\leq$, i.e., for $e \# e'$ and $e \leq f$, $e' \leq f'$, one has $f \# f'$.*

Given a set of finite partial orders $S$, we now show that $S$ is included in the configurations of the event structure obtained by Definition 2. This means that our event structure is a fitting representation of $\mathcal{L}$.

**Proposition 1.** *If $S$ is finite, then $S \subseteq \Omega(ES(S))$.*

Since we want to produce a Petri net, we now need to "*attach conditions*" to the result of Definition 2. Event structures and occurrence nets are conceptually very similar objects so this might seem very easy for the acquainted reader. However, this definition is crucial for the success of the subsequent folding step (Section 4), as we will be constrained to merge conditions in the preset and postset of an event when we merge the event. As a result, the conditions that we produce now should constraint as little as possible the future folding step.

**Definition 3.** *Given an event structure $\mathcal{E} := (E, \leq, \#)$ we construct the occurrence net $\beta := (B, E \backslash \{\bot\}, F)$ in two steps*

1. *Let $G := (V, A)$ be a graph where $V := E$ and $(e_1, e_2) \in A$ iff $e_1 \#_d e_2$. For each clique (maximal complete subgraph) $K := \{e_1, \ldots, e_n\}$ of $G$, let $C_K := [e_1] \cap \cdots \cap [e_n]$ and $e_K \in \max(C_K)$. We add a condition $b$ to $B$ and set $b \in e_K{}^\bullet$ and $b \in {}^\bullet e_i$ for $i = 1 \ldots n$.*
2. *For each $e \in E$, let $G_e := (V_e, A_e)$ be a graph where $V_e := \{e' \in E \mid e \lessdot e'\}$ and $(e_1, e_2) \in A_e$ iff $\lambda(e_1) \otimes \lambda(e_2)$. For each clique $K_e := \{e_1, \ldots, e_n\}$ of $G_e$, we add a condition $b$ to $B$ and set $b \in e^\bullet$ and $b \in {}^\bullet e_i$ for $i = 1 \ldots n$.*

Definition 3.1. adds a condition for every set of pairwise direct conflicting events; the condition is generated by some event $e_K$ which is in the past of every conflicting event and consumed by all of them; by the latter the conflict of the event structure is preserved in the occurrence net. For each event and its immediate successors, Definition 3.2. adds conditions between them to preserve causality. To minimize the number of conditions, for the successor events having dependent labels only one condition is generated. This step does not introduce new conflicts in the occurrence net since the events have dependent labels and none is in the past of the other, then by Definition 2 they are also in conflict in the event structure.

We note that Winskel already explained, in categorical terms, how to relate an event structure with an occurrence net [12]. However, his definition is of interest only in that context, while ours focus on a practical and efficient translation.

Given a log $\mathcal{L}$ and an independence relation $\diamond$, the net obtained applying Definitions 1, 2 and 3, in this order, is denoted by $\beta_{\mathcal{L},\diamond}$. Since every trace in $\mathcal{L}$ is a linearization of some of the partial orders in the set $S$ obtained by Definition 1 and these partial orders are included by Proposition 1 in the configurations of $ES(S)$ (which are the same as the configurations in $\beta_{\mathcal{L},\diamond}$), the obtained net is fitting.

**Proposition 2.** *Let $\mathcal{L}$ be a log and $\diamond$ an independence relation, for every $\sigma \in \mathcal{L}$ we have $\sigma \in obs(\beta_{\mathcal{L},\diamond})$.*

It is worth noticing that the obtained net generalizes the behavior of the model, but in a controlled manner imposed by the independence relation. For instance, if $\mathcal{L} := \{ab\}$ and $a \diamond b$, then $ba \in obs(\beta_{\mathcal{L},\diamond})$, even if this behavior was not present in the log. If the expert rightly declared $a$ and $b$ independent (i.e., if they commute at all states of $\mathcal{S}$), then necessarily $ba$ is a possible observation of $\mathcal{S}$, even if it is not in $\mathcal{L}$. The extra information provided by the expert allows us to generalize the discovered model in a provably sound manner, thus coping with the log incompleteness problem.

The independence relation between labels gives rise to an arbitrary relation between transitions of a net (not necessarily an independence relation):

**Definition 4.** *Let $\diamond \subseteq A \times A$ be an independence relation, $\mathcal{N} := (P, T, F, \lambda, M_0)$ a net, and $\lambda{:}T \to A$. We define relation $\diamond_N \subseteq T \times T$ between transitions of $N$ as*

$$t \diamond_N t' \Leftrightarrow \lambda(t) \diamond \lambda(t').$$

In the next section we will define an approach to fold $\beta_{\mathcal{L},\diamond}$ into a Petri net whose natural independence relation equals $\diamond$. To formalize our approach we first need to define such natural independence.

**Definition 5.** *Let $N := (P, T, F)$ be a net. We define the* natural independence *relation $\natural_N \subseteq T \times T$ on $N$ as*

$$t \natural_N t' \Leftrightarrow {}^\bullet t \cap {}^\bullet t' = \varnothing \wedge t^\bullet \cap {}^\bullet t' = \varnothing \wedge {}^\bullet t \cap t'^\bullet = \varnothing.$$

In fact, one can prove that when $N$ is safe, then $\natural_N$ is the notion of independence underlying the unfolding semantics of $N$. In other words, the equivalence

classes of $\equiv_{\#_N}$ are in bijective correspondence with the configurations in the unfolding of $N$. The following result shows that the natural independence on the discovered occurrence net corresponds to the relation provided by the expert, when both we restrict to the set of co-enabled transitions.

**Theorem 1.** *Let $\beta_{\mathcal{L},\diamond}$ be the occurrence net from the log $\mathcal{L}$ with $\diamond$ as the independence relation, then*

$$\diamond_{\beta_{\mathcal{L},\diamond}} \cap coe(\beta_{\mathcal{L},\diamond}) = \#_{\beta_{\mathcal{L},\diamond}} \cap coe(\beta_{\mathcal{L},\diamond})$$

## 4 Introducing Generalization

The construction described in the previous section guarantees that the unfolding obtained is fitting (see Proposition 1). However, the difference between $\mathcal{S}$ and $\mathcal{L}$ may be significant (e.g., $\mathcal{S}$ can contain cyclic behavior that can be instantiated an arbitrary number of times whereas only finite traces exist in $\mathcal{L}$) and the unfolding may be poor in generalization. The goal of this section is to generalize $\beta_{\mathcal{L},\diamond}$ in a way that the right patterns from $\mathcal{S}$, partially observed in $\mathcal{L}$ (e.g., loops), are incorporated in the generalized model. To generalize, we fold the discovered occurrence net. This folding is driven by an equivalence relation $\sim$ on $E \cup B$ that dictates which events merge into the same transition, and analogously for conditions; events cannot be merged with conditions. We write $[x]_\sim := \{x' \mid x \sim x'\}$ for the equivalence class of node $x$. For a set $X$, $[X]_\sim := \{[x]_\sim \mid x \in X\}$ is a set of equivalence classes.

**Definition 6 (Folded net [13]).** *Let $\beta := (B, E, F)$ be an occurrence net and $\sim$ a equivalence relation on the nodes of $\beta$. The folded Petri net (w.r.t. $\sim$) is defined as $\beta^\sim := (P_\sim, T_\sim, F_\sim, M_{0_\sim})$ where*

$$P_\sim := \{[b]_\sim \mid b \in B\}, \qquad F_\sim := \{([x]_\sim, [y]_\sim) \mid (x, y) \in F\},$$
$$T_\sim := \{[e]_\sim \mid e \in E\}, \qquad M_{0_\sim}([b]_\sim) := |\{b' \in [b]_\sim \mid {}^\bullet b' = \varnothing\}|.$$

Notice that the initial marking of the folded net is not necessarily safe. Safeness of the net depends on the chosen equivalence relation (see Proposition 3).

### 4.1 Language-Preserving Generalization

Different folding equivalences guarantee different properties on the folded net. From now on we focus our attention on three interesting classes of folding equivalences. The first preserves sequential executions of $\beta_{\mathcal{L},\diamond}$.

**Definition 7 (Sequence-preserving folding equivalence).** *Let $\beta$ be an occurrence net; an equivalence relation $\sim$ is called a sequence preserving (SP) folding equivalence iff $e_1 \sim e_2$ implies $\lambda(e_1) = \lambda(e_2)$ and $[{}^\bullet e_1]_\sim = [{}^\bullet e_2]_\sim$ for all events $e_1, e_2 \in E$.*

From the definition above it follows that $e_1 \sim e_2$ implies $\forall b \in {}^\bullet e_1 : \exists b' \in {}^\bullet e_2$ with $b \sim b'$. Since for every folded net obtained from a SP folding equivalence only equally labeled events are merged; we define then $\lambda([e]_\sim) := \lambda(e)$.

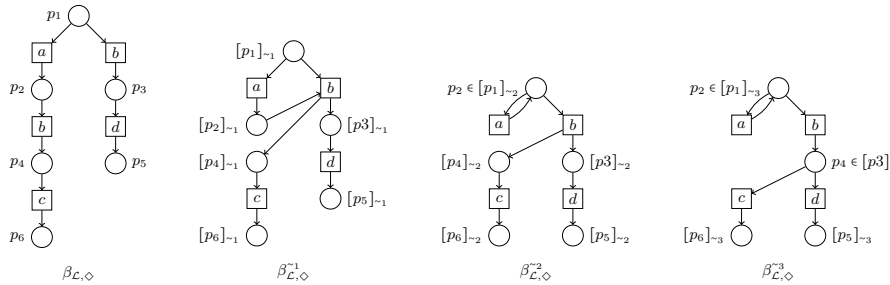


**Fig. 2.** Folding equivalences and folded nets.

*Example 1.* Consider the log $\mathcal{L} = \{abc, bd\}$ and the independence relation $\diamond = \varnothing$. Fig. 2 shows the obtained unfolding $\beta_{\mathcal{L},\diamond}$ (left) and three of its folded nets. The equivalence relation $\sim_1$ merges events labeled by $b$, but it does not merge their presets, i.e. is not a SP folding equivalence. It can be observed that $bd$ is not fireable in $\beta^{\sim_1}_{\mathcal{L},\diamond}$. Whenever two events are merged, their preconditions need to be merged to preserved sequential executions. The equivalence relation $\sim_2$ does not only merge events labeled by $b$, but it also sets $p_1 \sim_2 p_2$ and is a SP folding equivalence. The folded net $\beta^{\sim_2}_{\mathcal{L},\diamond}$ can replay every trace in the log $\mathcal{L}$, but it also adds new traces of the form $a^*, a^*b, a^*bc, a^*bd, a^*bcd$ and $a^*bdc$.

Given an unfolding, every SP folding equivalence generates a net that preserves its sequential executions.

**Theorem 2.** *Let $\beta$ be an occurrence net and $\sim$ a SP folding equivalence, then every fireable sequence $M_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} M_n$ from $\beta$ generates a fireable sequence $[M_0]_\sim \xrightarrow{[e_1]_\sim} \dots \xrightarrow{[e_n]_\sim} [M_n]_\sim$ from $\beta^\sim$.*

As a corollary of the result above and Proposition 2, the folded net obtained from $\beta_{\mathcal{L},\diamond}$ with a SP folding equivalence is fitting.

**Corollary 1.** *Let $\mathcal{L}$ be a log, $\diamond$ an independence relation and $\sim$ a SP folding equivalence, then for every $\sigma \in \mathcal{L}$ we have $\sigma \in obs(\beta^\sim_{\mathcal{L},\diamond})$.*

*Example 2.* We saw in Example 1 that every trace from $\mathcal{L}$ can be replayed in $\beta^{\sim_2}_{\mathcal{L},\diamond}$, but (as expected) the net accepts more traces. However this net also adds some independence between actions of the system: after firing $b$ the net puts tokens at $[p_3]_{\sim_2}$ and $[p_4]_{\sim_2}$ and the reached marking enables concurrently actions $c$ and $d$ which contradicts $c \otimes d$ (the independence relation $\diamond = \varnothing$ implies $c \otimes d$). In order to avoid this extra independence, we now consider the following class of equivalences.

**Definition 8 (Independence-preserving folding equivalence).** *Let $\beta$ be an occurrence net and $\diamond$ an independence relation; an equivalence relation $\sim$ is called an independence preserving (IP) folding equivalence iff*

1. *$\sim$ is a SP folding equivalence,*

2. $\lambda(e_1) \diamond \lambda(e_2) \Leftrightarrow [{}^\bullet e_1]_\sim \cap [{}^\bullet e_2]_\sim = \varnothing \wedge [{}^\bullet e_1]_\sim \cap [e_2{}^\bullet]_\sim = \varnothing \wedge [e_1{}^\bullet]_\sim \cap [{}^\bullet e_2]_\sim = \varnothing$ *for all events* $e_1, e_2 \in E$.
3. $b_1$ ***co*** $b_2$ *implies* $b_1 \not\prec b_2$ *for all conditions* $b_1, b_2 \in B$.

IP folding equivalences not only preserve the sequential behavior of $\beta$, but also ensure that $\beta^\sim$ and $\beta$ exhibit the same natural independence relation.

The definition above differs from the folding equivalence definition given in [13]; they consider occurrence nets coming from an unfolding procedure which takes as an input a net. This procedure generates a mapping between conditions and events of the generated occurrence net and places and transitions in the original net. Such mapping is necessary to define their folding equivalence. In our setting, the occurrence net does not come from a given net and therefore the mapping is not available.

*Example 3.* The equivalence $\sim_2$ from Fig. 2 is not an IP folding equivalence since the intersection of the equivalent classes of the preset of $c$ and $d$ is empty ($[{}^\bullet c]_{\sim_2} = \{[p_4]_{\sim_2}\}, [{}^\bullet d]_{\sim_2} = \{[p_3]_{\sim_2}\}$ and $\{[p_4]_{\sim_2}\} \cap \{[p_3]_{\sim_2}\} = \varnothing$), but $c$ and $d$ are not independent. Consider the equivalence relation $\sim_3$ which merges events labeled by $b$ and it sets $p_1 \sim_3 p_2$ and $p_3 \sim_3 p_4$; this relation is an IP folding equivalence. It can be observed in the net $\beta^{\sim_3}_{\mathcal{L},\diamond}$ of Fig. 2 that all the traces from the log can be replayed, but new independence relations are not introduced.

The occurrence net $\beta_{\mathcal{L},\diamond}$ is clearly safe. We show that $\beta^\sim_{\mathcal{L},\diamond}$ is also safe when $\sim$ is an IP folding equivalence. In this work, we constraint IP equivalences to generate safe nets because their natural independence relation is well understood (Definition 5), thus allowing us to assign a solid meaning to the class IP. It is unclear what is the natural unconditional independence of an unsafe net, and extending our definitions to such nets is subject of future work.

**Proposition 3.** *Let* $\beta_{\mathcal{L},\diamond}$ *be the unfolding obtained from the log* $\mathcal{L}$ *with* $\diamond$ *as the independence relation and* $\sim$ *an IP folding equivalence. Then* $\beta^\sim_{\mathcal{L},\diamond}$ *is safe.*

Theorem 1 shows that the structural relation between events of the unfolding and the relation generated by the independence given by the expert coincide (when we restrict to co-enabled events); the result also holds for the folded net when an IP folding equivalence is used.

**Theorem 3.** *Let* $\beta_{\mathcal{L},\diamond}$ *be the unfolding obtained from the log* $\mathcal{L}$ *with* $\diamond$ *as the independence relation and* $\sim$ *an IP folding equivalence, then* $\diamond_{\beta^\sim_{\mathcal{L},\diamond}} = \natural_{\beta^\sim_{\mathcal{L},\diamond}}$.

### 4.2 Controlling Generalization via Negative Information

We have shown that IP folding equivalences preserve independence. However, they could still introduce new unintended behaviour not present in $\mathcal{S}$. In this section we limit this phenomena by considering *negative information*, denoted by traces that should not be allowed by the model. Concretely, we consider negative information which is also given in the form of sequences $\sigma \in \mathcal{L}^- \subseteq A^*$. Negative information is often provided by an expert, but it can also be obtained automatically by recent methods [14]. Very few techniques in the literature use negative information in process discovery [5]. In this work, we assume a minimality criterion on the negative traces used:

**Assumption 1** *Let $\mathcal{L} := \mathcal{L}^+ \uplus \mathcal{L}^-$ be a pair of positive and negative logs and $\diamond$ the independence relation given by the expert. Any negative trace $\sigma \in \mathcal{L}^-$ corresponds to the local configuration of some event $e_\sigma$ in $\beta_{\mathcal{L},\diamond}$.*

This assumption implies that each negative trace is of the form $\sigma' a$ where $\sigma'$ only contains the actions that are necessarily to fire $a$. If $a$ can happen without them, they should not be consider part of $\sigma$. By removing all events $e_\sigma$ from $\beta_{\mathcal{L},\diamond}$ (one for each negative trace $\sigma \in \mathcal{L}^-$), we obtain a new occurrence net denoted by $\beta_{\mathcal{L},\diamond,*}$. The goal of this section is to fold this occurrence net without re-introducing the negative traces in the generalization step. If the expert is unable to provide negative traces satisfying this assumption, the discovery tool can always let him/her choose $e_\sigma$ from a visual representation of the unfolding.

**Definition 9 (Removal-aware folding equivalence).** *Let $\beta := (B, E, F)$ be an occurrence net and $\mathcal{L}^-$ a negative log; an equivalence relation $\sim$ is called removal aware (RA) folding equivalence iff*

1. *$\sim$ is a SP folding equivalence, and*
2. *for every $\sigma \in \mathcal{L}^-$ and $e' \in E$ we have $\lambda(e') = \lambda(e_\sigma)$ implies $[{}^\bullet e']_\sim \notin [{}^\bullet e_\sigma]$.*

The folded net obtained from $\beta_{\mathcal{L},\diamond,*}$ with a RA folding equivalence does not contain any of the negative traces.

**Theorem 4.** *Let $\beta_{\mathcal{L},\diamond,*}$ be the unfolding obtained from the log $\mathcal{L} := \mathcal{L}^+ \uplus \mathcal{L}^-$ with $\diamond$ as the independence relation after removing the corresponding event of each negative trace and $\sim$ a RA folding equivalence,*[5] *then*

$$obs(\beta^{\sim}_{\mathcal{L},\diamond,*}) \cap \mathcal{L}^- = \varnothing$$

## 5 Computing Folding Equivalences

Section 3 presents a discovery algorithm that generates fitting occurrence nets and Section 4 defines three classes of folding criteria, SP, IP, and RA, that ensure various properties. This section proposes an approach to synthesize SP, IP and RA folding equivalences using SMT.

### 5.1 SMT Encoding

We use an SMT encoding to find folding equivalences generating a net $\beta^{\sim}$ satisfying specific metric properties. Specifically, given a measure $\hat{c}$ (cf., Section 2), decidable in polynomial time, and a number $k \in \mathbb{N}$, we generate an SMT formula which is satisfiable iff there exists a folding equivalence $\sim$ such that $\hat{c}(\beta^{\sim}) = k$. We consider the number of transitions in the folded net as the measure $\hat{c}$, however, theoretically, any other measure that can be computed in polynomial time could be used. As explained in Section 2 simple functions like counting the number of nodes/arcs provide in practice reasonable results.

[5] Since Definition 9 refers to the events that generates the local configurations of the negative traces, the folding equivalence must be defined over the nodes of $\beta_{\mathcal{L},\diamond}$ and not those of $\beta_{\mathcal{L},\diamond,*}$.

Given an occurrence net $\beta := (B, E, F)$, for every event $e \in E$ and condition $b \in B$ we have integer variables $v_e$ and $v_b$. The key intuition is that two events (conditions) whose variables have equal number are equivalent and will be merged into the same transition (place). The following formulas state, respectively, that every element of a set $X$ is related with at least one element of a set $Y$, and that every element of $X$ is not related with any element of $Y$:

$$\phi^{sub}_{X,Y} := \bigwedge_{x \in X} \bigvee_{y \in Y} (v_x = v_y) \qquad \phi^{disj}_{X,Y} := \bigwedge_{x \in X, y \in Y} (v_x \neq v_y)$$

We force any satisfying assignment to represent an SP folding equivalence (Definition 7) with the following two constraints:

$$\phi^{SP}_{\beta} := \phi^{lab}_{\beta} \wedge \phi^{pre}_{\beta}.$$

Formulas $\phi^{lab}_{\beta}$ and $\phi^{pre}_{\beta}$ impose that only equally labeled events should be equivalent and that if two events are equivalent, then their presets should generate the same equivalence class:

$$\phi^{lab}_{\beta} := \bigwedge_{\substack{e,e' \in E \\ \lambda(e) \neq \lambda(e')}} (v_e \neq v_{e'}) \qquad \phi^{pre}_{\beta} := \bigwedge_{e,e' \in E} (v_e = v_{e'} \Rightarrow (\phi^{sub}_{{}^\bullet e, {}^\bullet e'} \wedge \phi^{sub}_{{}^\bullet e', {}^\bullet e}))$$

In addition to the properties encoded above, an IP folding equivalence (Definition 8) should satisfy some other restrictions:

$$\phi^{IP}_{\beta} := \phi^{SP}_{\beta} \wedge \phi^{ind}_{\beta} \wedge \phi^{co}_{\beta}$$

where $\phi^{ind}_{\beta}$ imposes that the presets and postsets of events with independent labels should generate equivalence classes that do not intersect and $\phi^{co}_{\beta}$ forbids concurrent conditions to be merged:

$$\phi^{ind}_{\beta} := \bigwedge_{e,e' \in E} (\lambda(e) \diamond \lambda(e') \Leftrightarrow (\phi^{disj}_{{}^\bullet e, {}^\bullet e'} \wedge \phi^{disj}_{{}^\bullet e, e'^\bullet} \wedge \phi^{disj}_{e, {}^\bullet e'^\bullet})) \qquad \phi^{co}_{\beta} := \bigwedge_{\substack{b,b' \in B \\ b \ \mathbf{co} \ b'}} (v_b \neq v_{b'})$$

Given a negative log $\mathcal{L}^-$, to encode a RA folding equivalence (Definition 9) we define:

$$\phi^{RA}_{\beta,\mathcal{L}^-} := \phi^{SP}_{\beta} \wedge \Big( \bigwedge_{\substack{\sigma \in \mathcal{L}^-, e' \in E \\ \lambda(e') = \lambda(e_\sigma)}} \neg \phi^{sub}_{{}^\bullet e', {}^\bullet e_\sigma} \Big)$$

where the right part of the conjunction imposes that for every $e_\sigma$ generated by a negative trace and any other event with the same label, their presets cannot generate the same equivalence class.

We now encode the optimality (w.r.t. the number of transitions) of the mined net. Given an occurrence net $\beta := (B, E, F)$, each event $e \in E$ generates a transition $v_e$ in the folded net $\beta^{\sim}$. To impose that the number of transitions in $\beta^{\sim}$ should be at most $k \in \mathbb{N}$, we define:

$$\phi^{MET}_{\beta,k} := \bigwedge_{e \in E} (1 \leq v_e \leq k)$$

To find an IP and RA folding equivalence that generates a net with at most $k$ transitions we propose the following encoding:

$$\phi^{OPT}_{\beta,\mathcal{L}^-,k} := \phi^{IP}_{\beta} \wedge \phi^{RA}_{\beta,\mathcal{L}^-} \wedge \phi^{MET}_{\beta,k}$$

**Theorem 5.** *Let $\mathcal{L} := \mathcal{L}^+ \uplus \mathcal{L}^-$ be a set of positive and negative logs, $\diamond \subseteq A \times A$ and independence relation and $k \in \mathbb{N}$. The formula $\phi^{OPT}_{\beta,\mathcal{L}^-,k}$ is satisfiable iff there exists an IP and RA folding equivalence $\sim$ such that $\tilde{\beta}_{\mathcal{L},\diamond,*}$ contains at most $k$ transitions.*

### 5.2 Finding an Optimal Folding Equivalence

Section 5.1 explains how to compute a folding equivalence that generates a folded net with a bounded number of transitions; this section explain how to obtain the optimal folded net, i.e the one with minimal number of transitions satisfying the properties of Theorem 3 and Theorem 4.

Iterative calls to the SMT solver can be done for a binary search with $k$ between $min_k$ and $max_k$; since only equally labeled events can be merged by the folding equivalence, the minimal number of transitions in the folded net is $min_k := |A|$; in the worst case, when events cannot be merged, $max_k := |E|$.

As a side remark, we have noted that the optimal folding equivalence can be encoded as a MaxSMT problem [15] where some clauses which are called hard must be true in a solution (in our case $\phi^{IP}_{\beta}$ and $\phi^{RA}_{\beta,\mathcal{L}^-}$) and some soft clauses may not ($\phi^{MET}_{\beta,k}$ for $|A| \leq k \leq |E|$); a MaxSMT solver maximizes the number of soft clauses that are satisfiable and thus it obtains the minimal $k$ generating thus the optimal folded net.

## 6 Experiments

As a proof of concept, we implemented our approach into a new tool called Pod (Partial Order Discovery).[6] It supports synthesis of SP and IP folding equivalences using a restricted form of our SMT encoding. In particular Pod merges all events with equal label, in contrast to the encoding in Section 5 which may in general yield more than one transition per log action. While this ensures a minimum (optimal as per Section 5.2) number of folded transitions, the tool could sometimes not find a suitable equivalence (unsatisfiable SMT encoding). Since the number of transitions in the folded net is fixed, it turns out that the quality of the mined model increases as we increase the number of folded places, as we show below. Using Pod we evaluate the ability of our approach to rediscover the original process model, given its independence relation and a set of logs. For this we have used standard benchmarks from the verification and process mining literature [16,17].

In our experiments, Table 1, we consider a set of original processes faithfully modelled as safe Petri nets. For every model $\mathcal{S}$ we consider a log $\mathcal{L}$, i.e. a subset of its traces. We extract from $\mathcal{S}$ the (best) independence relation $\natural_{\mathcal{S}}$ that an expert could provide. We then provide $\mathcal{L}$ and $\natural_{\mathcal{S}}$ to Pod and find an SP folding equivalence with the largest number of places (cols. "max. places") and with 60% of the places of $\mathcal{S}$ (last group of cols.), giving rise to two different mined models. All three models, original plus mined ones, have perfect fitness but varying levels of precision, i.e. traces of the model not present in the log. For

[6] Tool and benchmarks: http://lipn.univ-paris13.fr/~rodriguez/exp/atva15/.

| Original | | | Pod (max. places) | | | | Pod (60% places) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | $\|T\|$ | $\|P\|$ | $r_{\mathcal{S}\subseteq\mathcal{M}}$ | $r_{\mathcal{M}\subseteq\mathcal{S}}$ | %Prec. | $\|P\|$ | $r_{\mathcal{S}\subseteq\mathcal{M}}$ | $r_{\mathcal{M}\subseteq\mathcal{S}}$ | %Prec. | $\|P\|$ |
| A(22) | 22 | 20 | 0.99 | 1.00 | 0.77 | 19 | 0.57 | 1.00 | 0.22 | 11 |
| A(32) | 32 | 32 | 1.00 | 1.00 | 0.80 | 32 | 0.46 | 1.00 | 0.19 | 19 |
| A(42) | 42 | 47 | 0.98 | 1.00 | 0.54 | 40 | 0.79 | 1.00 | 0.21 | 28 |
| T(32) | 33 | 31 | 1.00 | 1.00 | 0.88 | 31 | 0.54 | 1.00 | 0.19 | 18 |
| Angio(1) | 64 | 39 | 0.39 | 0.94 | 0.18 | 21 | 0.10 | 0.92 | 0.06 | 13 |
| Complex | 19 | 13 | 0.98 | 1.00 | 0.62 | 12 | 0.62 | 1.00 | 0.39 | 7 |
| ConfDimB | 11 | 10 | 1.00 | 1.00 | 1.00 | 10 | 0.62 | 1.00 | 0.39 | 6 |
| Cycles(5) | 20 | 16 | 1.00 | 1.00 | 1.00 | 16 | 0.60 | 1.00 | 0.40 | 6 |
| DbMut(2) | 32 | 38 | 0.98 | 0.98 | 0.94 | 32 | 0.76 | 0.98 | 0.21 | 19 |
| Dc | 32 | 35 | 0.99 | 0.99 | 0.77 | 27 | 0.84 | 0.99 | 0.38 | 21 |
| Peters(2) | 126 | 102 | 0.45 | 1.00 | 0.07 | 51 | 0.30 | 1.00 | 0.05 | 30 |

**Table 1.** Experimental results.

the mined models, we report (cols. "%Prec.") on the ratio between their precision and the precision of the original model $\mathcal{S}$. All precisions were estimated using the technique from [18]. All Pod running times were below 10s.

Additionally, we measure how much independence of the original model is preserved in the mined ones. For that, we define the ratios $r_{\mathcal{S}\subseteq\mathcal{M}} := |\natural_{\mathcal{S}}\cap\natural_{\mathcal{M}}|/|\natural_{\mathcal{S}}|$ and $r_{\mathcal{M}\subseteq\mathcal{S}} := |\natural_{\mathcal{S}}\cap\natural_{\mathcal{M}}|/|\natural_{\mathcal{M}}|$. The closer $r_{\mathcal{S}\subseteq\mathcal{M}}$ is to 1, the larger is the number of pairs in $\natural_{\mathcal{S}}$ also contained in $\natural_{\mathcal{M}}$ (i.e., the more independence was preserved), and conversely for $r_{\mathcal{M}\subseteq\mathcal{S}}$ (the less independence was "*invented*"). Remark that $\natural_{\mathcal{S}} = \natural_{\mathcal{M}}$ iff $r_{\mathcal{S}\subseteq\mathcal{M}} = r_{\mathcal{M}\subseteq\mathcal{S}} = 1$.

In 7 out of the 11 benchmarks in Table 1 our proof-of-concept tool rediscovers the original model or finds one with only minor differences. This is even more encouraging when considering that we only asked Pod to find SP equivalences which, unlike IP, do not guarantee preservation of independence. In 9 out of 11 cases both ratios $r_{\mathcal{S}\subseteq\mathcal{M}}$ and $r_{\mathcal{M}\subseteq\mathcal{S}}$ are above 98%, witnessing that independence is almost entirely preserved. Concerning the precision, we observe that it is mostly preserved for these 9 models. We observe a clear correlation between the number of discovered places and the precision of the resulting model. The running times of Pod on all benchmarks in Table 1 were under few seconds.

In Peters(2) and Angio(1) our tool could not increase the number of places in the folded net, resulting in a significant loss of independence and precision. We tracked the reason down to (a) the additional restrictions on the SMT encoding imposed by our implementation and (b) the algorithm for transforming event structures into unfoldings (i.e., introducing conditions). We plan to address this in future work. This also prevented us from of employing IP equivalences instead of SP for these experiments: Pod could find IP equivalences for only 5 out of 11 cases. Nonetheless, as we said before, in 9 out of 11 the found SP equivalences preserved at least 98% of the independence.

Finally, we instructed Pod to synthesize SP equivalences folding into an arbitrarily chosen low number of places (60% of the original). Here we observe a large reduction of precision and significant loss of independence (surprisingly only $r_{\mathcal{S}\subseteq\mathcal{M}}$ drops, but not $r_{\mathcal{M}\subseteq\mathcal{S}}$). This witnesses a strong dependence between

the number of discovered places and the ability of our technique to preserve independence.

## 7 Related Work

To the best of our knowledge, there is no technique in the literature that solves the particular problem we are considering in this paper: given a set of positive and negative traces and an independence relation on events, derive a Petri net that both preserves the independence relation and satisfies the quality dimensions enumerated in Section 2. However, there is related work that intersects partially with the techniques of this paper. We now report on it.

Perhaps the closest work is [13], where the simplification of an initial process model is done by first unfolding the model (to derive an overfitting model) and then folding it back in a controlled manner, thus generalizing part of the behavior. The approach can only be applied for fitting models, which hampers its applicability unless alignment techniques [19] are used. The folding equivalences presented in this paper do not consider a model and therefore are less restrictive than the ones presented in [13].

*Synthesis* is a problem different from discovery: in synthesis, the underlying system is given and therefore one can assume $\mathcal{S} = \mathcal{L}$. Considering a synthesis scenario, Bergenthum *et al.* have investigated the synthesis of a p/t net from partial orders [20]. The class of nets considered in this paper (safe Petri nets) is less expressive than p/t nets, which in practice poses no problems in the context of business processes. The algorithms in [20] are grounded in the *theory of regions* and split the problem into two steps *(i)* the p/t net $\mathcal{M}$ is generated which, by construction, satisfies $\mathcal{L} \subseteq obs(\mathcal{M})$, and *(ii)* it is checked whereas $\mathcal{L} = obs(\mathcal{M})$. Actually, by avoiding *(ii)*, a discovery scenario is obtained where the generalization feature is not controlled, in contrast to the technique of this paper. With the same goal but relying on ad-hoc operators tailored to compose lpos (choice, sequentialization, parallel compositions and repetition), a discovery technique is presented in [21]. Since the operators may in practice introduce wrong generalizations, a domain expert is consulted for the legality of every extra run.

## 8 Conclusions

A fresh look at process discovery is presented in this paper, which establishes theoretical basis for coping with some of the challenges in the field. By automating the folding of the unfolding that covers traces in the log but also combinations thereof derived from the input independence relation, problems like log incompleteness and noise may be alleviated. The approach has been implemented and the initial results show the potential of the technique in rediscovering a model, even for the simplest of the folding equivalences described in this paper.

Next steps will focus on implementing the remaining folding equivalences, and in general improving the SMT constraints for computing folding equivalences. Also, incorporating the notion of trace frequency in the approach will be considered, to guide the technique to focus on principal behavior. This will allow to also test the tool in presence of incomplete or noisy logs.

## References


1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van der Aalst, W.M.P.: On the representational bias in process mining. In: 20th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2011, France, 2011. (2011) 2–7
3. 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
4. Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. Inductive Logic Programming (2008) 132–146
5. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research **10** (2009) 1305–1340
6. Mazurkiewicz, A.W.: Trace theory. In: Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 1986. (1986) 279–324
7. Dumas, M., García-Bañuelos, L.: Process mining reloaded: Event structures as a unified representation of process models and event logs. In: Application and Theory of Petri Nets and Concurrency (ICATPN'15). Volume 9115 of LNCS., Springer (2015) 33–48
8. Ponce-de-León, H., Rodríguez, C., Carmona, J., Heljanko, K., Haar, S.: Unfolding-based process discovery. CoRR **abs/1507.02744** (2015)
9. Esparza, J., Römer, S., Vogler, W.: An improvement of McMillan's unfolding algorithm. Formal Methods in System Design **20**(3) (2002) 285–310
10. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part I. Theoretical Computer Science **13** (1981) 85–108
11. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. Int. J. Cooperative Inf. Syst. **23**(1) (2014)
12. Winskel, G.: Categories of models for concurrency. In: Seminar on Concurrency, Carnegie-Mellon University, Pittsburg, PA, USA, July 9-11, 1984. (1984) 246–267
13. Fahland, D., van der Aalst, W.M.P.: Simplifying discovered process models in a controlled manner. Inf. Syst. **38**(4) (2013) 585–605
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. Nieuwenhuis, R., Oliveras, A.: On SAT modulo theories and optimization problems. In: Theory and Applications of Satisfiability Testing - SAT 2006. (2006) 156–169
16. The Model Checking Contest: Website. http://mcc.lip6.fr/.
17. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. In: Proc. of ICATPN'08. (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. Adriansyah, A.: Aligning observed and modeled behavior. PhD thesis, Technische Universiteit Eindhoven (2014)
20. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of petri nets from finite partial languages. Fundam. Inform. **88**(4) (2008) 437–468
21. Bergenthum, R., Desel, J., Mauser, S., Lorenz, R.: Construction of process models from example runs. T. Petri Nets and Other Models of Concurrency **2** (2009) 243–259