Learning Hybrid Process Models From Events

Process Discovery Without Faking Confidence

Wil M.P. van der Aalst[1,2] and Riccardo De Masellis[2] and Chiara Di Francescomarino[2] and Chiara Ghidini[2]

[1] Eindhoven University of Technology, PO Box 513, Eindhoven, The Netherlands
[2] FBK-IRST, Via Sommarive 18, 38050 Trento, Italy

Abstract. Process discovery techniques return process models that are either formal (precisely describing the possible behaviors) or informal (merely a "picture" not allowing for any form of formal reasoning). Formal models are able to classify traces (i.e., sequences of events) as fitting or non-fitting. Most process mining approaches described in the literature produce such models. This is in stark contrast with the over 25 available commercial process mining tools that only discover informal process models that remain *deliberately vague* on the precise set of possible traces. There are two main reasons why vendors resort to such models: *scalability* and *simplicity*. In this paper, we propose to combine the best of both worlds: discovering *hybrid process models* that have formal and informal elements. As a proof of concept we present a discovery technique based on *hybrid Petri nets*. These models allow for formal reasoning, but also reveal information that cannot be captured in mainstream formal models. A novel discovery algorithm returning hybrid Petri nets has been implemented in ProM and has been applied to several real-life event logs. The results clearly demonstrate the advantages of remaining "vague" when there is not enough "evidence" in the data or standard modeling constructs do not "fit". Moreover, the approach is scalable enough to be incorporated in industrial-strength process mining tools.

Key words: Process mining, Process discovery, Petri nets, BPM

1 Introduction

The increased interest in process mining illustrates that Business Process Management (BPM) is rapidly becoming more data-driven [1]. Evidence-based BPM based on process mining helps to create a common ground for business process improvement and information systems development. The uptake of process mining is reflected by the growing number of commercial process mining tools available today. There are over 25 commercial products supporting process mining (Celonis, Disco, Minit, myInvenio, ProcessGold, QPR, etc.). All support process discovery and can be used to improve compliance and performance problems. For example, without any modeling, it is possible to learn process models clearly showing the main bottlenecks and deviating behaviors.

These commercial tools are based on variants of techniques like the heuristic miner [17] and the fuzzy miner [8] developed over a decade ago [1]. All return process models that *lack formal semantics* and thus cannot be used as a *classifier* for traces. Classifying traces into *fitting* (behavior allowed by the model) and *non-fitting* (not possible according to the model) is however important for more advanced types of process mining. Informal models ("boxes and arcs") provide valuable insights, but cannot be used to draw reliable conclusions. Therefore, most discovery algorithms described in the literature (e.g., the α-algorithm [3], the region-based approaches [6, 15, 18], and the inductive mining approaches [11, 12, 13]) produce formal models (Petri nets, transition systems, automata, process trees, etc.) having clear semantics.

So why did vendors of commercial process mining tools opt for informal models? Some of the main drivers for this choice include:

– *Simplicity*: Formal models may be hard to understand. End-users need to be able to interpret process mining results: Petri nets with smartly constructed places and BPMN with many gateways are quickly perceived as too complex.
– *Vagueness*: Formal models act as binary classifiers: traces are fitting or non-fitting. For real-life processes this is often not so clear cut. The model capturing 80 percent of all traces may be simple and more valuable than the model that allows for all outliers and deviations seen in the event log. Hence, "vagueness" may be desirable to show relationships that cannot be interpreted in a precise manner.
– *Scalability*: Commercial process mining tools need to be able to handle logs with millions of events and still be used in an interactive manner. Many of the more sophisticated discovery algorithms producing formal models (e.g., region-based approaches [6, 15, 18]) do not scale well.

The state-of-the-art commercial products show that simplicity, vagueness and scalability can be combined effectively. Obviously, vagueness and simplicity may also pose problems. People may not trust process mining results when a precise interpretation of the generated model is impossible. When an activity has multiple outgoing arcs, i.e., multiple preceding activities, one would like to know whether these are concurrent or in a choice relation. Which combinations of output arcs can be combined? Showing frequencies on nodes (activities) and arcs may further add to the confusion when "numbers do not add up".

We propose *hybrid process models* as a way to combine the best of both worlds. Such models show informal dependencies (like in commercial tools) that are deliberately vague and at the same time provide formal semantics for the parts that are clear-cut. Whenever there is enough structure and evidence in the data, explicit routing constructs are used. If dependencies are weak or too complex, then they are not left out, but depicted in an informal manner.

We use *hybrid Petri nets*, a new class for Petri nets with informal annotations, as a concrete representation of hybrid process models. However, the ideas, concepts, and algorithms are generic and could also be used in the context of BPMN, UML activity diagrams, etc. Our proposed *discovery technique* has two phases. First we discover a *causal graph* based on the event log. Based on different (threshold) parameters we scan the event log for possible causalities. In the second phase we try to learn places based on explicit quality criteria. Places added can be interpreted in a precise manner and have a guaranteed quality. Causal relations that cannot or should not be expressed in terms of places are added as sure or unsure arcs. The resulting hybrid Petri net can be used as a starting point for other types of process mining.

The approach has been implemented in *ProM* and has been tested on various event logs and processes. These applications of our approach show that hybrid process models are useful and combine the best of both worlds: *simplicity, vagueness, and scalability can be combined with partly formal models that allow for reasoning and provide formal guarantees*.

The remainder is organized as follows. We first present a running example (Sect. 2) and some preliminaries (Sect. 3). Sect. 4 defines hybrid Petri nets. The actual two-phase discovery approach is presented in Sect. 5. Sect. 6 describes the *ProM* plug-ins developed to support the discovery of hybrid process models. Sect. 7 evaluates the approach. Sect. 8 discusses related work and Sect. 9 concludes the paper.

2 Motivating Example

Figure 1 illustrates the trade-offs using example data from an order handling process. All five models have been produced for the same event log containing 12,666 cases, 80,609 events, and eight unique activities. Each case has a corresponding trace, i.e., a sequence of events. Models (a), (b), and (c) are expressed in terms of a Petri net and have formal semantics. Model (a) was created using the ILP miner with default settings; it is precise and each of the 12,666 cases perfectly fits the model. However, model (a) is difficult to read. For larger event logs, having more activities and infrequent paths, the ILP miner is not able to produce meaningful models (the approach becomes intractable and/or produces incomprehensible models). Models (b) and (c) were created using the inductive miner (IMf [12]) with different settings for the noise threshold (0.0 respectively 0.2). Model (b) is underfitting, but able to replay all cases. Model (c) focuses on the mainstream behavior only, but only 9,440 of the 12,666 cases fit perfectly. In 3,189 cases there are multiple reminders and in 37 cases the payment is done before sending the invoice. All other cases conform to model (c). Models (d) and (e) were created using the commercial process mining tool *Disco* (Fluxicon) using different settings. These models are informal. Model (d) shows only the most frequent paths and

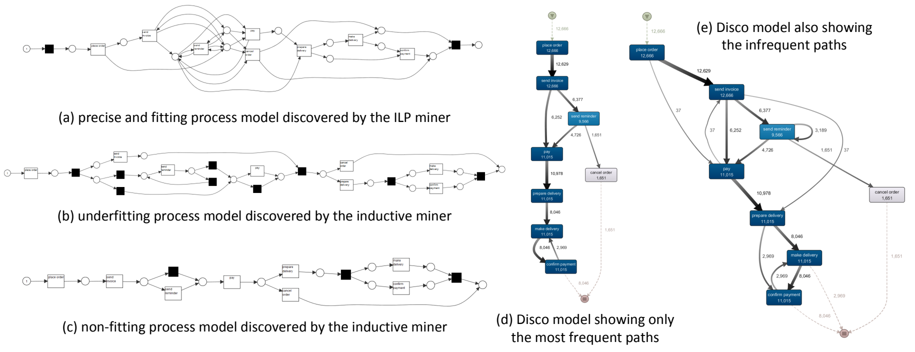
(a) precise and fitting process model discovered by the ILP miner
(b) underfitting process model discovered by the inductive miner
(c) non-fitting process model discovered by the inductive miner
(d) Disco model showing only the most frequent paths
(e) Disco model also showing the infrequent paths

Fig. 1. Five process models discovered for an event log recorded for 12,666 orders (labels are not intended to be readable).

model (e) shows all possible paths. For such informal models it is impossible to determine the exact nature of splits and joins. Commercial tools have problems dealing with loops and concurrency. For example, for each of the 12,666 cases, activities *make delivery* and *confirm payment* happened at most once, but not in a fixed order. However, these concurrent activities are put into a loop in models (d) and (e). This problem is not specific for *Disco* or this event log: all commercial tools suffer from this problem.

We would like to combine the left-hand side and the right-hand side of Figure 1 by using formal semantics when the behavior is clear and easy to express and resorting to informal annotations when things are blurry or inexact.

3 Preliminaries

In this section we introduce basic concepts, including multisets, operations on sequences, event logs and Petri nets.

$\mathcal{B}(A)$ is the set of all multisets over some set A. For some multiset $X \in \mathcal{B}(A)$, $X(a)$ denotes the number of times element $a \in A$ appears in X. Some examples: $X = [\,]$, $Y = [x, x, y]$, and $Z = [x^3, y^2, z]$ are multisets over $A = \{x, y, z\}$. X is the empty multiset, Y has three elements ($Y(x) = 2$, $Y(y) = 1$, and $Y(z) = 0$), and Z has six elements. Note that the ordering of elements is irrelevant.

$\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in A^*$ denotes a sequence over A. $\sigma(i) = a_i$ denotes the i-th element of the sequence. $|\sigma| = n$ is the length of σ and $dom(\sigma) = \{1, \ldots, |\sigma|\}$ is the domain of σ. $\langle\,\rangle$ is the empty sequence, i.e., $|\langle\,\rangle| = 0$ and $dom(\langle\,\rangle) = \emptyset$. $\sigma_1 \cdot \sigma_2$ is the concatenation of two sequences.

Let A be a set and $X \subseteq A$ one of its subsets. $\upharpoonright_X \in A^* \to X^*$ is a projection function and is defined recursively: $\langle\,\rangle\upharpoonright_X = \langle\,\rangle$ and for $\sigma \in A^*$ and $a \in A$: $(\langle a \rangle \cdot \sigma)\upharpoonright_X = \sigma\upharpoonright_X$ if $a \notin X$ and $(\langle a \rangle \cdot \sigma)\upharpoonright_X = \langle a \rangle \cdot \sigma\upharpoonright_X$ if $a \in X$. For example, $\langle a, b, a \rangle\upharpoonright_{\{a,c\}} = \langle a, a \rangle$. Projection can also be applied to multisets of sequences, e.g., $[\langle a, b, a \rangle^5, \langle a, d, a \rangle^5, \langle a, c, e \rangle^3]\upharpoonright_{\{a,c\}} = [\langle a, a \rangle^{10}, \langle a, c \rangle^3]$.

Starting point for process discovery is an event log where events are grouped into cases. Each case is represented by a trace, e.g., $\langle \triangleright, a, b, c, d, \square \rangle$.

Definition 1 (Event Log). *An* event log $L \in \mathcal{B}(A^*)$ *is a non-empty multiset of traces over some activity set A. A trace $\sigma \in L$ is a sequence of activities. There is a special start activity $\triangleright$ and a special end activity $\square$. We require that $\{\triangleright, \square\} \subseteq A$ and each trace $\sigma \in L$ has the structure $\sigma = \langle \triangleright, a_1, a_2, \ldots, a_n, \square \rangle$ and $\{\triangleright, \square\} \cap \{a_1, a_2, \ldots, a_n\} = \emptyset$. $\mathcal{U}_L$ is the set of all event logs satisfying these requirements.*

An event log captures the observed behavior that is used to learn a process model. An example log is $L_1 = [\langle \triangleright, a, b, c, d, \square \rangle^{45}, \langle \triangleright, a, c, b, d, \square \rangle^{35}, \langle \triangleright, a, e, d, \square \rangle^{20}]$ containing 100 traces ($|L_1| = 100$) and 580 events ($\sum_{\sigma \in L_1} |\sigma| = 580$). In reality, each event has a timestamp and may have any number of additional attributes. For example, an event may refer to a customer, a product, the person executing the event, associated costs, etc. Here we abstract from these notions and simply represent an event by its activity name.

A *Petri net* is a bipartite graph composed of places (represented by circles) and transitions (represented by squares).

Definition 2 (Petri Net). *A Petri net is a tuple $N = (P, T, F)$ with P the set of places, T the set of transitions, $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation.*

Transitions represent activities and places are added to model causal relations. $\bullet x = \{y \mid (y, x) \in F\}$ and $x\bullet = \{y \mid (x, y) \in F\}$ define input and output sets of places and transitions. Places can be used to causally connect transitions as is reflected by relation $\widehat{F}$: $(t_1, t_2) \in \widehat{F}$ if t_1 and t_2 are connected through a place p, i.e., $p \in t_1\bullet$ and $p \in \bullet t_2$.

Definition 3 ($\widehat{F}$). *Let $N = (P, T, F)$ be a Petri net. $\widehat{F} = \{(t_1, t_2) \in T \times T \mid \exists_{p \in P}\ \{(t_1, p), (p, t_2)\} \subseteq F\}$ are all pairs of transitions connected through places.*

The state of a Petri net, called *marking*, is a multiset of places indicating how many *tokens* each place contains. Tokens are shown as block dots inside places.

Definition 4 (Marking). *Let $N = (P, T, F)$ be a Petri net. A marking M is a multiset of places, i.e., $M \in \mathcal{B}(P)$.*

A transition $t \in T$ is *enabled* in marking M of net N, denoted as $(N, M)[t\rangle$, if each of its input places ($p \in \bullet t$) contains at least one token. An enabled transition t may *fire*, i.e., one token is removed from each of the input places ($p \in \bullet t$) and one token is produced for each of the output places ($p \in t\bullet$).

$(N, M)[t\rangle(N, M')$ denotes that t is enabled in M and firing t results in marking M'. Let $\sigma = \langle t_1, t_2, \ldots, t_n\rangle \in T^*$ be a sequence of transitions, sometimes referred to as a *trace*. $(N, M)[\sigma\rangle(N, M')$ denotes that there is a set of markings $M_0, M_1, \ldots, M_n$ such that $M_0 = M$, $M_n = M'$, and $(N, M_i)[t_{i+1}\rangle(N, M_{i+1})$ for $0 \leq i < n$.

A *system net* has an initial and a final marking. The *behavior* of a system net corresponds to the set of traces starting in the initial marking M_{init} and ending in the final marking M_{final}.

Definition 5 (System Net Behavior). *A system net is a triplet $SN = (N, M_{init}, M_{final})$ where $N = (P, T, F)$ is a Petri net, $M_{init} \in \mathcal{B}(P)$ is the initial marking, and $M_{final} \in \mathcal{B}(P)$ is the final marking. $behav(SN) = \{\sigma \mid (N, M_{init})[\sigma\rangle(N, M_{final})\}$ is the set of traces possible according to the model.*

Note that a system net classifies traces σ into *fitting* ($\sigma \in behav(SN)$) and *non-fitting* ($\sigma \notin behav(SN)$).

4 Hybrid Petri Nets

A *formal process model* is able to make firm statements about the inclusion or exclusion of traces, e.g., trace $\langle \triangleright, a, b, c, d, \square\rangle$ fits the model or not. *Informal process models* are unable to make such precise statements about traces. Events logs only show example behavior: (1) logs are typically incomplete (e.g., the data only shows a fraction of all possible interleavings, combinations of choices, or unfoldings) and (2) logs may contain infrequent exceptional behavior where the model should abstract from. Therefore, it is impossible to make conclusive decisions based on event logs. More observations may lead to a higher certainty and the desire to make a formal statement (e.g., "after a there

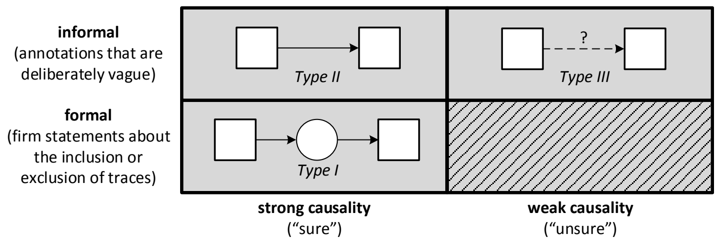
informal
(annotations that are deliberately vague)
formal
(firm statements about the inclusion or exclusion of traces)
Type II
?
Type III
Type I
strong causality
("sure")
weak causality
("unsure")

Fig. 2. The strength of a causality and the formality of a modeling construct are orthogonal. However, it makes less sense to express a weak causality in a formal manner.

is a choice between b and c"). However, fewer observations and complex dependencies create the desire to remain "vague". Models (a), (b) and (c) in Figure 1 have formal semantics as described in Definition 5. (The initial and final markings are defined but not indicated explicitly: the source places are initially marked and the sink places are the only places marked in the final markings.) Models (d) and (e) in Figure 1 are informal and therefore unable to classify traces into fitting and non-fitting.

In essence process models describe *causalities* between activities. Depending on the evidence in the data these causalities can be seen as stronger ("sure") or weaker ("unsure"). The strength of a causal relation expresses the level of confidence. A strong causality between two activities a and b suggests that one is quite sure that activity a causes activity b to happen later in time. This does not mean that a is always followed by b. The occurrence of b may depend on other factors, e.g., b requires c to happen concurrently or a only increases the likelihood of b.

The strength of a causality and the formality of a modeling construct are orthogonal as shown in Figure 2. Even when one is not sure, one can still use a formally specified modeling construct. Moreover, both notions may be local, e.g., parts of the process model are more certain or modeled precisely whereas other parts are less clear and therefore kept vague.

As Figure 2 suggests it seems undesirable to express a weak causality using a formal construct. Moreover, depending on the *representational bias* of the modeling notation, strong causalities may not be expressed easily. The modeling notation may not support concurrency, duplicate activities, unstructured models, long-term dependencies, OR-joins, etc. Attempts to express behavior incompatible with representational bias of the modeling notation in a formal model are doomed to fail. Hence, *things that cannot be expressed easily in an exact manner can only be captured using annotations that are deliberately vague and non-executable.* Instead, we aim to combine the best of both worlds, i.e., marrying the left-hand side and the right-hand side of Figure 1 by combining both formal and informal notations.

Although the ideas are *generic* and also apply to other notations (BPMN, UML activity diagrams, etc.), we operationalize the notion of hybrid process models by defining and using so-called *hybrid Petri nets*. Unlike conventional Petri nets, we use different types of arcs to indicate the level of *certainty*.

Figure 3 shows an example of a hybrid Petri net discovered based on the event log also used to create the models in Figure 1. Strong causalities are expressed through conventional places and arcs and *sure arcs* (arcs directly connecting transitions). Weak causalities are expressed using *unsure arcs* (dashed arcs with a question mark). Figure 2 shows the three types of arcs.

Definition 6 (Hybrid Petri Net). *A* hybrid Petri net *is a tuple $HPN = (P, T, F_1, F_2, F_3)$ where (P, T, F_1) is a Petri net, $F_2 \subseteq T \times T$, and $F_3 \subseteq T \times T$ such that $\widehat{F_1}$, F_2, and F_3 are pairwise disjoint. Arcs of Type I ($(p,t) \in F_1$ or $(t,p) \in F_1$) are the normal arcs connecting a place to a transition or vice versa. Arcs of Type II ($(t_1,t_2) \in F_2$) are arcs indicating a strong causality between two transitions (sure arcs). Arcs of Type III ($(t_1,t_2) \in F_3$) are arcs indicating a weak causality between two transitions (unsure arcs).*

Transitions, places, and normal (*Type I*) arcs have formal semantics as defined in Sect. 3. Again we define an initial and final marking to reason about the set of traces possible. Therefore, we define the notion of a *hybrid system net*.

Definition 7 (Hybrid System Net). *A* hybrid system net *is a triplet $HSN = (HPN, M_{init}, M_{final})$ where $HPN = (P, T, F_1, F_2, F_3)$ is a hybrid Petri net, $M_{init} \in \mathcal{B}(P)$ is the initial marking, and $M_{final} \in \mathcal{B}(P)$ is the final marking. $\mathcal{U}_{HSN}$ is the set of all possible hybrid system nets. $behav(HSN)$ is defined as in Definition 5 while ignoring the sure and unsure arcs (i.e., remove F_2 and F_3).*

Only normal (*Type I*) arcs have formal semantics; the other two types of arcs are informal and do not include or exclude traces. Recall that Petri net without any places allows for any behavior and adding a place can only restrict behavior. A sure arc $(t_1, t_2) \in F_2$ should be interpreted as a strong causal relationship that cannot be expressed (easily) in terms of a place connecting t_1 and t_2. An unsure arc $(t_1, t_2) \in F_3$ is a suspected causal relationship that is too weak to justify a place connecting t_1 and t_2.

The role of sure and unsure arcs will become clearer when presenting the discovery technique in the next section. Figure 3 also uses special symbols for the start and end activities ($\triangleright$ and $\square$) as introduced in Definition 1, but the semantics of *HSN* do not depend on this.

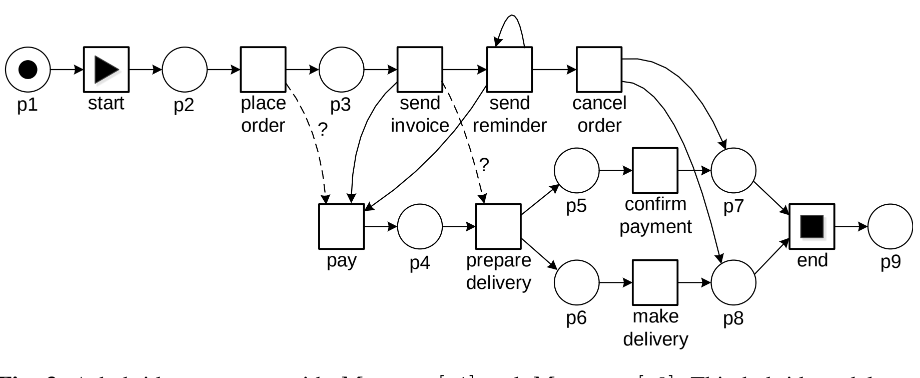

Fig. 3. A hybrid system net with $M_{init} = [p1]$ and $M_{final} = [p9]$. This hybrid model was discovered using the approach presented in Sect. 5.

We would like to stress that only the places in a hybrid system net HSN provide formal semantics. Behavioral quality measures such as fitness and precision will be based solely on the places in HSN (see definition $behav(HSN)$). Sure arcs (F_2) and unsure arcs (F_3) carry important information but cannot be used for such quality measures.

5 Discovering Hybrid Process Models

We aim to discover hybrid process models. As a target format we have chosen hybrid system nets that have three types of arcs. We use a *two-step approach*. First, we discover a *causal graph* (Sect. 5.1). Based on the causalities identified, we generate candidate places. These places are subsequently evaluated using replay techniques (Sect. 5.2). Strong causalities that cannot be expressed in terms of places are added to the *hybrid system net* as sure arcs. Moreover, the resulting hybrid model may also express weak causal relations as unsure arcs.

5.1 Discovering Causal Graphs

A causal graph is a directed graph with activities as nodes. There is always a unique start activity ($\triangleright$) and end activity ($\square$). There are two kinds of causal relations: *strong* and *weak*. These correspond to the two columns in Figure 2.

Definition 8 (Causal Graph). *A* causal graph *is a triplet $G = (A, R_S, R_W)$ where A is the set of activities including start and end activities (i.e., $\{\triangleright, \square\} \subseteq A$), $R_S \subseteq A \times A$ is the set of strong causal relations, $R_W \subseteq A \times A$ is the set of weak causal relations, and $R_S \cap R_W = \emptyset$ (relations are disjoint). $\mathcal{U}_G$ is the set of all causal graphs.*

Figure 4 shows a causal graph derived from the event log also used to discover the models in Figure 1. The dashed arcs with question marks correspond to weak causal relations. The other arcs correspond to strong causal relations.

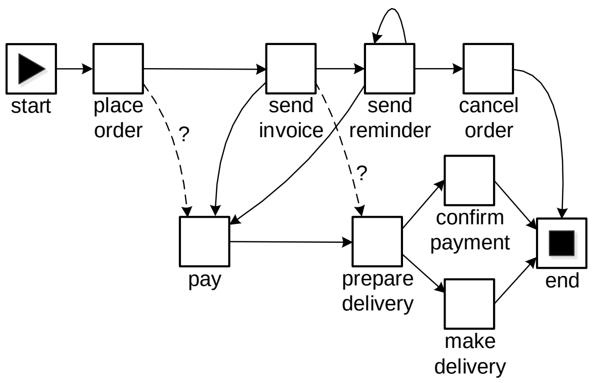

Fig. 4. A causal graph: nodes correspond to activities and arcs correspond to causal relations.

Definition 9 (Causal Graph Discovery). *A causal graph discovery function $disc_{cg} \in \mathcal{U}_L \to \mathcal{U}_G$ is a function that constructs a causal graph $disc_{cg}(L) = (A, R_S, R_W)$ for any event log $L \in \mathcal{U}_L$ over A.*

There are many algorithms possible to construct a causal graph from a log. As an example, we use a variant of the approach used by the heuristic miner [1, 17]. We tailored the approach to hybrid discovery (i.e., different types of arcs) while aiming for parameters that are intuitive and can be used interactively (e.g., thresholds can be changed seamlessly while instantly showing the resulting graph). Note that we clearly separate the identification of causalities from the discovery of process logic (see Sect. 5.2).

Definition 10 (Log-Based Properties). *Let $L \in \mathcal{U}_L$ be an event log over A and $\{a, b\} \subseteq A$.*

- $\#(a, L) = \sum_{\sigma \in L} |\{i \in dom(\sigma) \mid \sigma(i) = a\}|$ *counts the number of a's in log L.*
- $\#(X, L) = \sum_{x \in X} \#(x, L)$ *counts the number of $X \subseteq A$ activities in L.*
- $\#(a, b, L) = \sum_{\sigma \in L} |\{i \in dom(\sigma) \setminus \{|\sigma|\} \mid \sigma(i) = a \wedge \sigma(i+1) = b\}|$ *counts the number of times a is directly followed by b in event log L.*
- $\#(*, b, L) = \sum_{\sigma \in L} |\{i \in dom(\sigma) \setminus \{|\sigma|\} \mid \sigma(i+1) = b\}|$ *counts the number of times b is preceded by some activity.*
- $\#(a, *, L) = \sum_{\sigma \in L} |\{i \in dom(\sigma) \setminus \{|\sigma|\} \mid \sigma(i) = a\}|$ *counts the number of times a is succeeded by some activity.*
- $Rel1(a, b, L) = \dfrac{\#(a, b, L) + \#(a, b, L)}{\#(a, *, L) + \#(*, b, L)}$ *counts the strength of relation (a, b) relative to the split and join behavior of activities a and b.*
- $Rel2_c(a, b, L) = \begin{cases} \frac{\#(a,b,L) - \#(b,a,L)}{\#(a,b,L) + \#(b,a,L) + c} & \text{if } \#(a, b, L) - \#(b, a, L) > 0 \\ \frac{\#(a,b,L)}{\#(a,b,L) + c} & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$
  counts the strength of relation (a, b) taking into account concurrency and loops using parameter $c \in \mathbb{R}^+$ (default $c = 1$).
- $Caus_{c,w}(a, b, L) = w \cdot Rel1(a, b, L) + (1 - w) \cdot Rel2_c(a, b, L)$ *takes the weighted average of both relations where $w \in [0, 1]$ is a parameter indicating the relative importance of the first relation. If $w = 1$, we only use $Rel1(a, b, L)$. If $w = 0$, we only use $Rel2_c(a, b, L)$. If $w = 0.5$, then both have an equal weight.*

$Rel1(a, b, L)$, $Rel2_c(a, b, L)$, and $Caus_{c,w}(a, b, L)$ all produce values between 0 (weak) and 1 (strong). Using the properties in Definition 10, we define a concrete function $disc_{cg}$ to create causal graphs. All activities that occur at least t_{freq} times in the event log are included as nodes. The strength of relations between remaining activities (based on $Caus_{c,w}$) are used to infer causal relations. t_{R_S} and t_{R_W} are thresholds for strong respectively weak causal relations. Parameter w determines the relative importance of $Rel1$ and $Rel2_c$. Parameter c is typically set to 1.

Definition 11 (Concrete Causal Graph Discovery Technique). *Let $L \in \mathcal{U}_L$ be an event log over A and let $t_{freq} \in \mathbb{N}^+$, $c \in \mathbb{R}^+$, $w \in [0, 1]$, $t_{R_S} \in [0, 1]$, $t_{R_W} \in [0, 1]$ be parameters such that $t_{R_S} \geq t_{R_W}$. The corresponding causal graph is $G = disc_{cg}(L) = (A', R_S, R_W)$ where*

– $A' = \{a \in A \mid \#(a, L) \geq t_{freq}\} \cup \{\triangleright, \square\}$ *is the set of activities that meet the threshold (the start and end activities are always included).*
– $R_S = \{(a, b) \in A' \times A' \mid Caus_{c,w}(a, b, L\upharpoonright_{A'}) \geq t_{R_S}\}$ *is the set of strong causal relations.*
– $R_W = \{(a, b) \in A' \times A' \mid t_{R_S} > Caus_{c,w}(a, b, L\upharpoonright_{A'}) \geq t_{R_W}\}$ *is the set of weak causal relations.*

Figure 4 shows a causal graph constructed using parameters $t_{freq} = 1000$, $c = 1$, $w = 0.2$, $t_{R_S} = 0.8$, and $t_{R_W} = 0.75$.

5.2 Discovering Hybrid System Nets

In the second step of the approach we use the causal graph to create a hybrid system net (that turns strong causalities into formal constraints if possible).

Definition 12 (Hybrid System Net Discovery). *A hybrid system net discovery function $disc_{hsn} \in (\mathcal{U}_L \times \mathcal{U}_G) \to \mathcal{U}_{HSN}$ is a function that for any event log L and causal graph G discovers a hybrid system net $disc_{hsn}(L, G) \in \mathcal{U}_{HSN}$.*

Just like there are many algorithms possible to create a causal graph, there are also multiple ways to construct a hybrid system net from an event log and causal graph. The minimal consistency requirements can be defined as follows.

Definition 13 (Consistent). *Let $L \in \mathcal{U}_L$ be an event log, let $G = (A, R_S, R_W) \in \mathcal{U}_G$ be a causal graph, and let $HSN = (HPN, M_{init}, M_{final}) \in \mathcal{U}_{HSN}$ with $HPN = (P, T, F_1, F_2, F_3)$ be a hybrid system net. L, G, and SN are* consistent *if and only if: $T = A \subseteq \bigcup_{\sigma \in L}\{a \in \sigma\}$, $\{p_\triangleright, p_\square\} \subseteq P$, $F_1 \cap ((\{p_\triangleright, p_\square\} \times T) \cup (T \times \{p_\triangleright, p_\square\})) = \{(p_\triangleright, \triangleright), (\square, p_\square)\}$, $M_{init} = [p_\triangleright]$ and $M_{final} = [p_\square]$, for all $p \in P \setminus \{p_\triangleright, p_\square\}$: $\bullet p \neq \emptyset$ and $p\bullet \neq \emptyset$, $R_S = \widehat{F_1} \cup F_2$, $\widehat{F_1} \cap F_2 = \emptyset$, and $R_W = F_3$.*

An event log L, causal graph G, and hybrid system net HSN are consistent if (1) L and G refer to the same set of activities all appearing in the event log, (2) there is a source place $p_\triangleright$ marked in the initial place and enabling start activity $\triangleright$, (3) there is a sink place $p_\square$ marked in the final marking and connected to end activity $\square$, (4) all other places connect activities, (5) there is a one-to-one correspondence between strong causal relations (R_S) and connections through places ($\widehat{F_1}$) or sure arcs (F_2), and (6) there is a one-to-one correspondence between weak causal relations (R_W) and unsure arcs (F_3).

Consider two activities $a_1, a_2 \in A$ that are frequent enough to be included in the model. These can be related in three different ways: $(a_1, a_2) \in \widehat{F_1}$ if there is a place connecting a_1 and a_2, $(a_1, a_2) \in F_2$ if there is no place connecting a_1 and a_2 but there is a strong causal relation between a_1 and a_2 (represented by a sure arc), $(a_1, a_2) \in F_3$ if there is a weak causal relation between a_1 and a_2 (represented by an unsure arc).

Any discovery function $disc_{hsn} \in (\mathcal{U}_L \times \mathcal{U}_G) \to \mathcal{U}_{HSN}$ should ensure consistency. In fact, Definition 13 provides hints on how to discover a hybrid system net.

Assume a place $p = (I, O)$ with input transitions $\bullet p = I$ and output transitions $p\bullet = O$ is added. $R_S = \widehat{F_1} \cup F_2$ implies that $\widehat{F_1} \subseteq R_S$. Hence, $I \times O \subseteq R_S$, i.e., place $p = (I, O)$ can only connect transitions having strong causal relations. Moreover, I and O should not be empty. These observations based on Definition 13 lead to the following definition of candidate places.

Definition 14 (Candidate Places). *Let $G = (A, R_S, R_W) \in \mathcal{U}_G$ be a causal graph. The candidate places based on G are: $candidates(G) = \{(I, O) \mid I \neq \emptyset \ \wedge \ O \neq \emptyset \ \wedge \ I \times O \subseteq R_S\}$.*

Given a candidate place $p = (I, O)$ we can check whether it allows for a particular trace.

Definition 15 (Replayable trace). *Let $p = (I, O)$ be a place with input set $\bullet p = I$ and output set $p\bullet = O$. A trace $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in A^*$ is perfectly replayable with respect to place p if and only if*

- *for all $k \in \{1, 2, \ldots, n\}$: $|\{1 \leq i < k \mid a_i \in I\}| \geq |\{1 \leq i \leq k \mid a_i \in O\}|$ (place p cannot "go negative" while replaying the trace) and*
- *$|\{1 \leq i \leq n \mid a_i \in I\}| = |\{1 \leq i \leq n \mid a_i \in O\}|$ (place p is empty at end).*

We write $\checkmark(p, \sigma)$ if σ is perfectly replayable with respect to place $p = (I, O)$. $act(p, \sigma) = \exists_{a \in \sigma} \ a \in (I \cup O)$ denotes whether place $p = (I, O)$ has been activated, i.e., a token was consumed or produced for it in σ.

Note that $\checkmark(p, \sigma)$ if σ is a trace of the system net having only one place p. To evaluate candidate places one can define different scores.

Definition 16 (Candidate Place Scores). *Let $L \in \mathcal{U}_L$ be an event log. For any candidate place $p = (I, O)$ with input set $\bullet p = I$ and output set $p\bullet = O$, we define the following scores:*

- *$score_{freq}(p, L) = \frac{|\,[\sigma \in L | \checkmark(p,\sigma)]\,|}{|L|}$ is the fraction of fitting traces,*
- *$score_{rel}(p, L) = \frac{|\,[\sigma \in L | \checkmark(p,\sigma) \ \wedge \ act(p,\sigma)]\,|}{|\,[\sigma \in L | act(p,\sigma)]\,|}$ is the fraction of fitting traces that have been activated, and*
- *$score_{glob}(p, L) = 1 - \frac{|\,\#(I,L) - \#(O,L)\,|}{max(\#(I,L), \#(O,L))}$ is a global score only looking at the aggregate frequencies of activities.*

To explain the three scoring functions consider again $L_1 = [\langle \triangleright, a, b, c, d, \square \rangle^{45}, \langle \triangleright, a, c, b, d, \square \rangle^{35}, \langle \triangleright, a, e, d, \square \rangle^{20}]$. Let us consider place $p_1 = (I_1, O_1)$ with $I_1 = \{a\}$ and $O_2 = \{b\}$. $score_{freq}(p_1, L_1) = score_{rel}(p_1, L_1) = {}^{80}/_{100} = 0.8$ and $score_{glob}(p_1, L_1) = 1 - {}^{|100-80|}/_{max(100,80)} = 0.8$. For place $p_2 = (I_2, O_2)$ with $I_2 = \{a\}$ and $O_2 = \{b, e\}$: $score_{freq}(p_2, L_1) = score_{rel}(p_2, L_1) = score_{glob}(p_2, L_1) = 1$. Hence, all three scoring functions agree and show that the second place is a better candidate. Note that if the candidate place p does not inhibit any of the traces in the log, then all scores are 1 by definition.

Let us now consider event log $L_2 = [\langle c, d \rangle^{1000}, \langle a, b \rangle^{100}, \langle b, a \rangle^{10}, \langle a, a, a, a, \ldots, a \rangle]$ (with the last trace containing 1000 a's) and candidate place $p_1 = (I_1, O_1)$ with $I_1 = \{a\}$ and $O_2 = \{b\}$. $score_{freq}(p_1, L_2) = {}^{1100}/_{1111} = 0.99$, $score_{rel}(p_1, L_2) = {}^{100}/_{111} = 0.90$, $score_{glob}(p_1, L_2) = 1 - {}^{|1110-110|}/_{max(1110,110)} = 0.099$. Now the values are very different. Interpreting the scores reveals that $score_{freq}$ is too optimistic. Basically one can add any place connected to low frequent activities, without substantially lowering the $score_{freq}$ score. Hence, $score_{rel}$ is preferable over $score_{freq}$. $score_{glob}$ can be computed very efficiently because traces do not need to be replayed. It can be used to quickly prune the set of candidate places, but the last example shows that one needs to be careful when traces are unbalanced (i.e., I or O activities occur many times in a few traces).

Based on the above discussion we use scoring function $score_{rel}$ in conjunction with a threshold t_{replay}. The causal graph, a set of candidate places, and this threshold can be used to discover a hybrid system net.

Definition 17 (Concrete Discovery Technique). *Let $L \in \mathcal{U}_L$ be an event log and let $G = (A, R_S, R_W) \in \mathcal{U}_G$ be a causal graph. t_{replay} is the threshold for the fraction of fitting traces that have been activated. The discovered hybrid system net $disc_{hsn}(L, G) = (HPN, M_{init}, M_{final})$ with $HPN = (P, T, F_1, F_2, F_3)$ is constructed as follows*

- *$Q = \{p \in candidates(G) \mid score_{rel}(p, L\lceil_A) \geq t_{replay}\}$ is the set of internal places (all candidate places meeting the threshold),*
- *$P = \{p_\triangleright, p_\square\} \cup Q$ is the set of places ($\{p_\triangleright, p_\square\} \cap Q = \emptyset$),*
- *$T = A$ is the set of transitions,*
- *$F_1 = \{(p_\triangleright, \triangleright), (\square, p_\square)\} \cup \{(t, (I, O)) \in T \times Q \mid t \in I\} \cup \{((I, O), t) \in Q \times T \mid t \in O\}$ is the set of normal arcs,*
- *$F_2 = R_S \setminus \widehat{F_1}$ is the set of sure arcs, and*
- *$F_3 = R_W$ is the set of unsure arcs.*

It is easy to check that this concrete $disc_{hsn}$ function indeed ensures consistency. The construction of the discovered hybrid system net is guided by the causal graph. We can construct hybrid system net $disc_{hsn}(L, disc_{cg}(L))$ for any event log L using parameters t_{freq}, c, w, t_{R_S}, t_{R_W}, and t_{replay}. For example, the hybrid model shown in Figure 3 was discovered using $t_{freq} = 1000$, $c = 1$, $w = 0.2$, $t_{R_S} = 0.8$, $t_{R_W} = 0.75$, and $t_{replay} = 0.9$. Our discovery approach is highly configurable and also provides formal guarantees (e.g., $t_{replay} = 1$ ensures perfect fitness). When there is not enough structure or evidence in the data, the approach is not coerced to return a model that suggests a level of confidence that is not justified.

6 Implementation

Two novel *ProM* plug-ins have been created to support the approach described in this paper.[1] The *Causal Graph Miner* plug-in is used to create a causal graph using the approach described in Definition 11. The user can control the parameters w, t_{freq}, t_{R_S}, and t_{R_W} through sliders and directly see the effects in the resulting graph. The *Hybrid Petri Net Miner* plug-in implements Definition 17 and takes as input an event log and a causal graph. The plug-in returns a discovered hybrid system net. Only places that meet the t_{replay} threshold are added. The replay approach has been optimized to stop replaying a trace when it does not fit.

[1] Install *ProM* and the package *HybridMiner* from `http://www.promtools.org`.

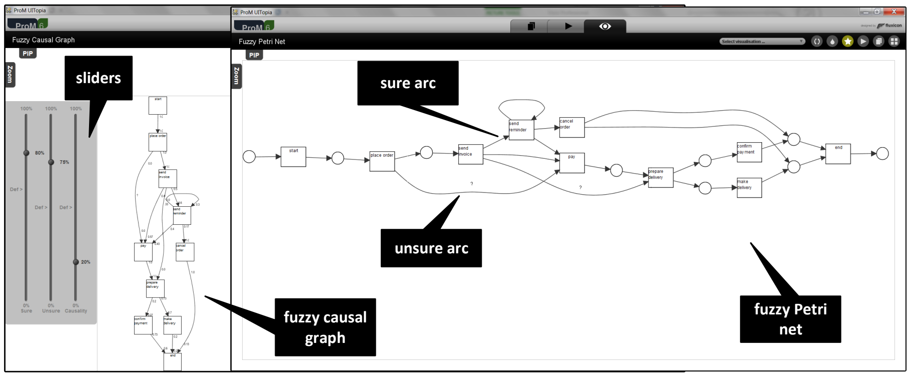

Fig. 5. Screenshots of the *Causal Graph Miner* (left) and the *Hybrid Petri Net Miner* (right) analyzing the running example with parameter settings $t_{freq} = 1000$, $c = 1$, $w = 0.2$, $t_{R_S} = 0.8$, $t_{R_W} = 0.75$, and $t_{replay} = 0.9$.

Figure 5 shows the two plug-ins in action for the event log containing 12,666 cases and 80,609 events. The results returned correspond to the causal graph depicted in Figure 4 and the hybrid system net depicted in Figure 3. *Both were computed in less than a second on a standard laptop.* Activity *send reminder* may occur repeatedly (or not) after sending the invoice but before payment or cancellation. However, payments may also occur before sending the invoice. The hybrid system net in Figure 5 (also see Figure 3 which is better readable) clearly differentiates between (1) the behavior which is dominant and clear and (2) the more vague behavior that cannot be captured formally or is not supported by enough "evidence". The example illustrates the scalability of the approach while supporting simplicity and deliberate vagueness.

7 Evaluation

Table 1. Six data sets used.

Log	Cases	Events	Activities
BPI-2011	1143	150291	624
BPI-2012	13087	164506	23
BPI-2014	46616	466737	39
BPI-2015	1199	52217	398
BPI-2016	557	286075	312
BPI-2017	31509	475306	24

Process discovery techniques can be evaluated using a range of indicators referring to *fitness* (ability to replay the observed behavior), *precision* (avoiding underfitting), *generalization* (avoiding overfitting), and *simplicity* (is the model easy to understand) [1]. Existing indicators are less suitable for the evaluation of hybrid models explicitly capturing vagueness. Criteria involving fitness, precision, and generalization can also not be measured for the informal models produced by existing commercial process mining tools. When computing traditional quality measures for hybrid system nets we basically ignore the sure and unsure arcs.

We applied our approach to a large number of real-life events logs and analyzed the effects of the different parameters (t_{freq}, c, w, t_{R_S}, t_{R_W}, and t_{replay}) on the resulting

Table 2. Parameters used to create the base models and their characteristics.

Log	t_{freq}	t_{R_S}	t_{R_W}	w	t_{replay}	$\|T\|$	$\|P\|$	$\|\widehat{F_1}\|$	$\|F_2\|$	$\|F_3\|$	**Fitness**	**Precision**	**Time (ms)**
BPI-2011	343	0.81	0.80	0.10	0.80	38	6	4	200	6	0.84	0.04	11772
BPI-2012	3926	0.90	0.89	0.10	0.80	14	8	7	20	1	0.90	0.26	12414
BPI-2014	13985	0.90	0.90	0.10	0.80	10	5	3	13	0	0.93	0.54	21233
BPI-2015	360	0.45	0.40	0.50	0.80	59	26	24	145	75	0.74	0.05	7055
BPI-2016	445	0.50	0.50	0.10	0.80	12	2	0	31	0	0.83	0.10	31428
BPI-2017	9453	0.51	0.50	0.50	0.80	22	8	7	36	12	0.95	0.12	24772

models. In this section, we report on our findings using six data sets taken from the well-known *BPI Challenges* [16].[2]

Table 1 shows the basic characteristics of the six event logs used: *BPI-20XX* refers to the year of the corresponding BPI challenge [16] and the number of cases, events, and unique activities (event classes) are shown. For *BPI-2011*, *BPI-2012*, and *BPI-2017* we used the full data set. For *BPI-2014* we used the event log for incidents, for *BPI-2015* we used the data of the first municipality, and for *BPI-2016* we used the event log with click data. These selections were made to focus on a particular process or organization.

We first selected initial parameters for each of the six event logs in Table 1 to create six "reasonable" base models. To create the base models we interactively set the thresholds in such a way that the underlying graph is connected. t_{replay} was set in such way that a reasonable number of places remained. Table 2 shows the settings used and some of the characteristics of the resulting hybrid process models.

Obviously different parameter settings lead to different models. For example, if t_{replay} is set to 1, then (by definition) the fitness will be 1. Similarly, the number of unsure arcs is directly affected by t_{R_W}. If $t_{R_W} = t_{R_S}$, then (by definition) there will be no unsure arcs. Column $|T|$ shows the number of retained activities. Columns $|P|$ and $|\widehat{F_1}|$ provide insights in the dominant and clear behavior captured in terms of normal arcs. Columns $|F_2|$ and $|F_3|$ indicate the number of sure and unsure arcs. These numbers give insights in the complexity of the models (simplicity dimension). Fitness and precision are computed using the techniques from [4] and [5] while ignoring the sure and unsure arcs (i.e., only considering the normal places and arcs).

The fitness values in Table 2 are as expected. It is possible to improve fitness at the cost of having fewer places. The precision values in Table 2 vary widely. Precision is very low for *BPI-2011* and *BPI-2015*. However, for these models there are many sure arcs showing the added value of our hybrid approach. Things that cannot be expressed in terms of reasonable places ($t_{replay} = 0.8$) can still be expressed. Traditional approaches would be forced to accept places that have a lower quality or ignore the causalities observed. For example, the inductive miner would generate underfitting models or models focusing on the mainstream behavior only.

The computation times (last column in Table 2) are in milliseconds. Clearly, the size of the event log and computation time positively correlate. Moreover, the fewer candidate places the faster the second step is performed. These numbers show that the approach is already quite fast compared to other approaches returning a formal model

[2] The reader is invited to redo the experiments using the latest version of *ProM* , the *Hybrid-Miner* package (`promtools.org`), and the publicly available data sets used here [16].

(all models are computed in less than 25 seconds). Implementation-wise there is ample room for improvement, showing that the approach itself is highly scalable.

For each of the six event logs, we used the baseline values for w, t_{R_S}, t_{R_W}, t_{replay}, and t_{freq} (Table 2) as a starting point. (We fixed the value of c to its default value: $c = 1$.) Next, we varied some of the key parameters one-by-one while keeping the baseline values for the other parameters fixed: $t_{replay} \in \{0.7, 0.8, 0.9, 1.0\}$, $w \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$, $t_{R_S} \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, and $t_{R_W} \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ (such that $t_{R_S} \geq t_{R_W}$). These results are discussed in detail in a technical report [2]. Given the limited space, we only summarize the main findings here.

– Increasing the value of t_{replay} improves fitness of the model because places that are not perfectly fitting are removed. The precision of the model typically decreases when t_{replay} goes up. Moreover, the removal of places leads to an increase in sure arcs.
– Increasing the value of w has a marginal effect on fitness and precision. For some of the event logs, precision is better for lower values of w (i.e., more weight is given to $Rel2_c(a, b, L)$).
– Increasing the value of t_{R_S} leads to fewer connections through places and sure arcs. This can only improve fitness. However, the effect is moderate and heavily depends on t_{replay}. Precision tends to go down when t_{R_S} goes up.
– Increasing the value of t_{R_W}, by definition, has no effect on precision and fitness and only affects the number of unsure arcs.

In summary, the discovery approach works in a predictable manner. Using the parameters the analyst can influence the characteristics of the discovered model in a fast and reliable manner. It is possible to express "vagueness" in terms of sure and unsure arcs. If there is not enough evidence in the data to justify the addition of many "good" places, then the resulting model will have a low precision. Fitness can be controlled directly by t_{replay}. We refer to the technical report for detailed experimental results [2], but acknowledge that additional evaluations are needed (involving new metrics and groups of users).

8 Related Work

The work reported in this paper was inspired by the work of Herrmann et al. [9, 10] who argue that modeling "requires the representation of those parts of knowledge which cannot be stated definitely and have to be modeled vaguely". They propose annotations to make vagueness explicit. In [9, 10] the goal is to *model* vagueness, but we aim to automatically *discover* hybrid models supporting both vagueness and formal semantics.

Hybrid process models are related to the partial models considered in software engineering [7, 14]. These partial models can be completed into formal models and do not consider data-driven uncertainty. In fact, these partial models are closer to configurable process models representing sets of concrete models

In literature one can find a range of process discovery approaches that produce formal models [1]. The α-algorithm [3] and its variants produce a Petri net. Approaches based on state-based regions [15] and language-based regions [6, 18] also discover Petri nets. The more recently developed inductive mining approaches produce process trees that can be easily converted to Petri nets or similar [11, 12, 13].

Commercial process mining tools typically produce informal models. These are often based on the first phases of the heuristic miner [17] (dependency graph) or the fuzzy miner [8] (not allowing for any form of formal reasoning).

It is impossible to give a complete overview of all discovery approaches here. However,as far as we know there exist on other discovery approaches that return hybrid models having both formal and informal elements.

9 Conclusion

In this paper we advocated the use of *hybrid models* to combine the best of two worlds: commercial tools producing informal models and discovery approaches providing formal guarantees. We provided a concrete realization of our hybrid discovery approach using *hybrid Petri nets*. The ideas are not limited to Petri nets and could be applied to other types of process models (e.g., BPMN models with explicit gateways for the clear and dominant behavior and additional arcs to capture complex or less dominant behavior). Unlike existing approaches there is no need to straightjacket behavior into a formal model that suggests a level of confidence that is not justified. The explicit representation of vagueness and uncertainty in hybrid process models is analogous to the use of confidence intervals and box-and-whisker diagrams in descriptive statistics.

The approach has been fully implemented and tested on numerous real-life event logs. The results are very promising, but there are still many open questions. In fact, the paper should be seen as the starting point for a new branch of research in BPM and process mining. To evaluate differences between informal, formal, and hybrid models from a user perspective, we need new evaluation criteria taking understandability and perceived complexity into account. Future work will also include "hybrid BPMN and UML activity diagrams" focusing on different model constructs (gateways, swimlanes, artifacts, etc.). Existing techniques (also supported by *ProM*) can already be used to map compliance and performance indicators onto causalities expressed in terms of explicit places. We would like to also provide approximative compliance and performance indicators for sure and unsure arcs. Note that commercial tools show delays and frequencies on arcs, but these indicators may be very misleading as demonstrated in Sect. 11.4.2 of [1]. Finally, we would like to improve performance. The approach has already a good performance. Moreover, there are several ways to further speed-up analysis (e.g., pruning using $score_{glob}$ or user-defined preferences). Moreover, computation can be distributed in a straightforward manner (e.g., using MapReduce).

References

1. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.
2. W.M.P. van der Aalst, R. De Masellis, C. Di Francescomarino, and C. Ghidini. Learning Hybrid Process Models From Events: Process Discovery Without Faking Confidence (Experimental Results). *ArXiv e-prints 1703.06125*, `arxiv.org/abs/1703.06125`, 2017.

3. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
4. A. Adriansyah. *Aligning Observed and Modeled Behavior*. Phd thesis, Eindhoven University of Technology, April 2014.
5. A. Adriansyah, J. Munoz-Gama, J. Carmona, B.F. van Dongen, and W.M.P. van der Aalst. Measuring Precision of Modeled Behavior. *Information Systems and e-Business Management*, 13(1):37–67, 2015.
6. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process Mining Based on Regions of Languages. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 375–383. Springer-Verlag, Berlin, 2007.
7. M. Famelis, R. Salay, and M. Chechik. Partial Models: Towards Modeling and Reasoning with Uncertainty. In *International Conference on Software Engineering (ICSE 2012)*, pages 573–583. IEEE Computer Society, 2012.
8. C.W. Günther and W.M.P. van der Aalst. Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer-Verlag, Berlin, 2007.
9. T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured Models are Surprisingly Useful for User-Centered Design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.
10. T. Herrmann and K.U. Loser. Vagueness in Models of Socio-technical Systems. *Behaviour and Information Technology*, 18(5):313–323, 1999.
11. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer-Verlag, Berlin, 2013.
12. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer-Verlag, Berlin, 2014.
13. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, pages 1–33, 2016. doi:10.1007/s10270-016-0545-x.
14. R. Salay, M. Chechik, J. Horkoff, and A. Di Sandro. Managing Requirements Uncertainty with Partial Models. *Requirements Engineering*, 18(2):107–128, 2013.
15. M. Sole and J. Carmona. Process Mining from a Basis of Regions. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets 2010*, volume 6128 of *Lecture Notes in Computer Science*, pages 226–245. Springer-Verlag, Berlin, 2010.
16. B.F. van Dongen. BPI Challenges (2011-2017), Real life Event Logs Collection, `data.4tu.nl/repository/collection:event_logs`, 2017.
17. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
18. J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens, and A. Serebrenik. Process Discovery using Integer Linear Programming. *Fundamenta Informaticae*, 94:387–412, 2010.