Discovering the "Glue" Connecting Activities
Exploiting Monotonicity to Learn Places Faster

Wil M.P. van der Aalst

Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
E:wvdaalst@pads.rwth-aachen.de W:www.vdaalst.com

Abstract. Process discovery, one of the key areas within process mining, aims to derive behavioral models from event data. Since event logs are inherently incomplete (containing merely example behaviors) and unbalanced, this is often challenging. Different target languages can be used to capture sequential, conditional, concurrent, and iterative behaviors. In this paper, we assume that a process model is merely a set of places (like in Petri nets). Given a particular behavior, a place can be "fitting", "underfed" (tokens are missing), or "overfed" (tokens are remaining). We define a partial order on places based on their connections. Then we will show various monotonicity properties that can be exploited during process discovery. If a candidate place is underfed, then all "lighter" places are also underfed. If a candidate place is overfed, then all "heavier" places are also overfed. This allows us to prune the search space dramatically. Moreover, we can further reduce the search space by not allowing conflicting or redundant places. These more foundational insights can be used to develop fast process mining algorithms producing places with a guaranteed quality level.

Keywords: Process mining, Process discovery, Petri nets, BPM

1 Introduction

It is a pleasure to contribute to this Festschrift honoring Farhad Arbab's contributions to computer science. Farhad worked on different topics in the broader field of formal methods and software engineering. However, he is best known for his work in coordination models and languages. Often concurrency and composition played an important role in his work. The Reo coordination language is the piece de resistance of Farhad's work. Reo is a channel-based coordination model wherein complex coordinators, called connectors, are composed from simpler ones [10]. The language has been mapped to many other languages [16], including zero-safe nets (a variant for Petri nets) and constraint automata. There is also work on the synthesis of Reo circuits from scenario specifications [21]. Unfortunately, mining techniques to learn Reo models from event data are still missing.

In process mining, typically representations such as Petri nets, workflow nets, causal nets, process trees, transition systems, statecharts, and BPMN models are

used [3]. Rather than coordinating complex components, these models merely coordinate activities derived from event data.

The goal of Reo is to provide the "glue" between different software components. *In the same way, one could view places in a Petri net as the "glue" between transitions representing activities.* In this sense, places in a Petri net can be viewed as a simple coordination layer. The goal of this paper is to discover sets of places modeling the underlying process such that (1) this can be done quickly (handling event logs with millions of events) and (2) that places have a well-defined minimal quality level.

Event data are collected in logistics, manufacturing, finance, healthcare, customer relationship management, e-learning, e-government, and many other domains. The events found in these domains typically refer to activities executed by resources at particular times and for a particular case (i.e., process instances). Process mining techniques are able to exploit such data. Here, we focus on *process discovery*, but process mining also includes conformance checking, performance analysis, decision mining, organizational mining, predictions, recommendations, etc.

Over the last two decades, hundreds of process discovery techniques have been proposed [3]. Many of the initial techniques could not cope with infrequent behavior and made very strong assumptions about the completeness of the event log. For example, traditional region-based techniques assume that all possible behavior has been observed (i.e., the log is complete) and that all observed traces are equally important. State-based regions were introduced by Ehrenfeucht and Rozenberg in 1989 and generalized by Cortadella et al. [14, 12]. Various authors used state-based regions for process discovery [7, 22]. Also, language-based regions have been used for this purpose [11, 25]. Over time attention shifted to approaches able to deal with noise and infrequent behavior. Early approaches include heuristic mining, fuzzy mining, and various genetic process mining approaches [24, 15]. Since 2010 the speed at which new process discovery techniques are proposed is accelerating. As an example consider the family of inductive mining techniques [17–19].

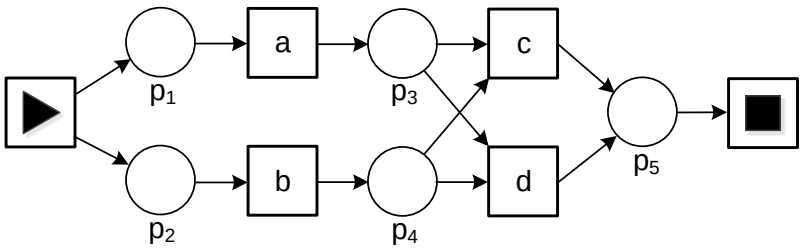

Fig. 1. Process model $P_1 = \{(\{\blacktriangleright\}, \{a\}), (\{\blacktriangleright\}, \{b\}), (\{a\}, \{c, d\}), (\{b\}, \{c, d\}), (\{c, d\}, \{\blacksquare\})\}$ composed of five places discovered from event log $L_1 = [\langle \blacktriangleright, a, b, c, \blacksquare\rangle^{31}, \langle \blacktriangleright, b, a, c, \blacksquare\rangle^{27}, \langle \blacktriangleright, a, b, d, \blacksquare\rangle^{23}, \langle \blacktriangleright, b, a, d, \blacksquare\rangle^{19}]$.

This paper provides a fresh look at places in a Petri net seen from the viewpoint of process discovery. Each place can be viewed as a *constraint*, limiting the behavior of the Petri net. We use a so-called *open-world assumption*: Any behavior is possible unless explicitly forbidden by one of the places in the model. Consider the process model shown in Figure 1 which is composed of five places. Transition ▶ models the start of the process and transition ■ marks the end. Place p_1 specifies that activity a can only happen after ▶. Moreover, at the end, the number of occurrences of a should match the number of occurrences of ▶. Since ▶ happens once, also a should also happen precisely once. Place p_3 specifies that activity c and activity d can only happen after a occurred. At the end, the number of occurrences of c and d should match the number of occurrences of a. The goal is to discover models merely composed of places from event data. Each event refers to a case (process instance), activity, and a timestamp. We can group events based on cases and sort events within a case based on the timestamps. This way each case can be presented by a trace $\langle \blacktriangleright, a, b, c, \blacksquare \rangle$, i.e., a sequence of activities. An event log is a multiset of such traces. The caption of Figure 1 shows event log L_1 consisting of 100 cases and 500 events referring to six unique activities.

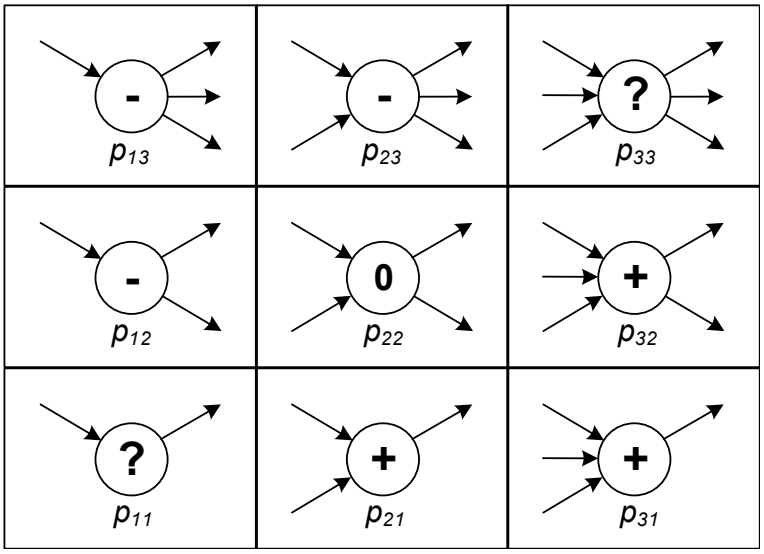

Fig. 2. During replay each of the places p_{32}, p_{21}, and p_{31} will always have at least the number of tokens in p_{22}. Similarly, places p_{13}, p_{23}, and p_{12} will always have at most the number of tokens in p_{22}. Places p_{11} and p_{33} may have more or fewer tokens.

It is far from trivial to discover places from larger event logs referring to many activities. The number of possible places grows exponentially in the number of activities and to evaluate a place one needs to traverse the whole event log. A

naive algorithm would need to replay the event log for every possible candidate place. This can be very time-consuming. Moreover, places may be redundant or conflicting. Therefore, we explore relationships among (sets of) places and present several *monotonicity* results. To do this, we define new notions such as "underfed" and "overfed" places and partial orders on (sets of) places based on their input and output transitions.

Figure 2 shows the basic idea. If we replay a trace on a particular place, there could be two problems:

- At some stage, a transition needs to remove a token from the place, but the place is already empty (the place is "underfed").
- At the end of the trace, tokens remain in the place (the place is "overfed").

Note that a place can be "overfed" and "underfed" at the same time. Assume now that we have a place p_{22} with two input transitions and two output transitions (Figure 2 only shows the corresponding arcs). If this place is perfectly fitting some trace σ (the place is not "underfed" and not "overfed"), then we know that adding an input arc and/or removing an output arc can only make the place "heavier" (indicated by the + sign in Figure 2). Moreover, removing an input arc and/or adding an output arc can only make the place "lighter" (indicated by the − sign in Figure 2). We can exploit this simple observation. If place p_{22} is already overfed, then we know that the places p_{32}, p_{21}, and p_{31} also need to be overfed. If place p_{22} is already underfed, then we know that the places p_{13}, p_{23}, and p_{12} also need to be underfed. These monotonicity properties allow us to prune the search space of candidate places. In fact, *the monotonicity results can be exploited by discovery algorithms to speed-up discovery while still producing all places that meet predefined quality criteria.* This paper focuses on the formal foundation of such approaches without providing a specific process discovery technique. Nevertheless, it is quite straightforward to see how the results can be used to speed-up process discovery.

The remainder is organized as follows. Section 2 provides the formal setting by defining behaviors, event logs, process models, and their semantics. In Section 3 we relate places using partial orders and prove the first monotonicity results. This is then lifted to quality scores for places (Section 4 and Section 5). We briefly discuss how these monotonicity results can be used for process discovery (Section 6). To further prune the set of candidate places we define redundancy and conflict (Section 7). Section 8 discusses implications for conformance checking. Section 9 concludes the paper.

2 Behaviors, Event Logs, and Models

To be able to discuss the monotonicity results that can be exploited by process discovery approaches we define key notions such as behaviors, event logs, and places. We also define the semantics of process models based on places using an open-world assumption.

2.1 Behaviors

First, we introduce some basic mathematical notations.

$\mathcal{P}(X) = \{Y \mid Y \subseteq X\}$ is the powerset of set X. $\mathcal{B}(X) = X \to \mathbb{N}$ is the set of all multisets over some set X. For any $B \in \mathcal{B}(X)$: $B(x)$ denotes the number of times element $x \in X$ appears in B. $B_1 = [\,]$, $B_2 = [a, a, b]$, and $B_3 = [a^3, b^2, c]$ are multisets over $X = \{a, b, c\}$. B_1 is the empty multiset, B_2 has three elements, and B_3 has six elements. Note that the ordering of elements is irrelevant. Union ($B_1 \cup B_2$), intersection ($B_1 \cap B_2$), and difference ($B_1 \setminus B_2$) are defined as usual. All operators for sets are generalized to multisets, e.g., $\sum_{x \in [a,b,b,a,c]} x = 2a + 2b + c$.

$\sigma = \langle x_1, x_2, \ldots, x_n \rangle \in X^*$ denotes a sequence over X. $\sigma(i) = a_i$ 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.

Based on the preliminaries we can define the notion of *behavior*. $\langle \blacktriangleright, a, b, c, \blacksquare \rangle$ is an example behavior, i.e., a sequence of activities starting with $\blacktriangleright$ and ending with $\blacksquare$.

Definition 1 (Activities and Behaviors). *$\mathbb{A}$ is the universe of activities (actions, tasks, operations, transaction types, etc.). There are two special activities: $\{\blacktriangleright, \blacksquare\} \subseteq \mathbb{A}$. $\blacktriangleright$ is the unique start activity and $\blacksquare$ is the unique end activity. A behavior $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in \mathbb{A}^*$ is a sequence of activity names such that $n \geq 2$, $a_1 = \blacktriangleright$, $a_n = \blacksquare$, and for all $1 < i < n$: $a_i \in \mathbb{A} \setminus \{\blacktriangleright, \blacksquare\}$. $\mathbb{B}$ is the set of all possible behaviors.*

In this paper, $\mathbb{A} = \{\blacktriangleright, \blacksquare, a, b, c, d, \ldots\}$ and $\mathbb{B} = \{\langle \blacktriangleright, \blacksquare \rangle, \langle \blacktriangleright, a, \blacksquare \rangle, \langle \blacktriangleright, b, \blacksquare \rangle, \ldots \langle \blacktriangleright, a, b, c, c, a, d, \blacksquare \rangle, \ldots\}$.

2.2 Event Logs

An *event log* can be defined as a multiset of behaviors. Elements of such a multiset are called *traces* and refer to *cases* (i.e., process instance).

Definition 2 (Event Log). *An event log L is a multiset of behaviors, i.e., $L \in \mathcal{B}(\mathbb{B})$. $\sigma \in L$ is called a trace.*

$L_1 = [\langle \blacktriangleright, a, b, c, \blacksquare \rangle^{31}, \langle \blacktriangleright, b, a, c, \blacksquare \rangle^{27}, \langle \blacktriangleright, a, b, d, \blacksquare \rangle^{23}, \langle \blacktriangleright, b, a, d, \blacksquare \rangle^{19}]$ is an example of an event log with 100 traces. For example, 31 cases exhibit the behavior $\langle \blacktriangleright, a, b, c, \blacksquare \rangle$. Typically, an event log has more information. For example, events may have a timestamp, refer to resources, locations, customers, costs, etc. Since we focus on the discovery of the "control-flow backbone" of a process, we can abstract from these optional attributes.

2.3 Using Places To Constrain Behavior

In the context of process mining, a wide variety of modeling languages are used ranging from Petri nets, workflow nets, causal nets, process trees and transition

systems to statecharts and BPMN models. In this paper, we use a very "lean" modeling language based on places and an open-world assumption. First, we define $\mathbb{P}$ and $\mathbb{P}^!$ as the set of all possible (through) places.

Definition 3 (Places). $\mathbb{P} = \mathcal{P}(\mathbb{A}) \times \mathcal{P}(\mathbb{A})$ *is the set of all possible places. For any* $p = (I, O) \in \mathbb{P}$, $\bullet p = I$ *is the set of input activities and* $p\bullet = O$ *is the set of output activities.* $\mathbb{P}^! = (\mathcal{P}(\mathbb{A}) \setminus \{\emptyset\}) \times (\mathcal{P}(\mathbb{A}) \setminus \{\emptyset\})$ *is the set of through places, i.e., places having non-empty sets of input and output activities.*

Note that places do not have names, they are fully identified by the input and output activities. Therefore, for any p_1 and p_2, if $\bullet p_1 = \bullet p_2$ and $p_1\bullet = p_2\bullet$, then $p_1 = p_2$. A process model is simply a set of places.

Definition 4 (Process Model). *A set of places* $P \subseteq \mathbb{P}$ *defines a process model.*

Figure 1 shows the process model $P_1 = \{(\{\blacktriangleright\}, \{a\}), (\{\blacktriangleright\}, \{b\}), (\{a\}, \{c, d\}), (\{b\}, \{c, d\}), (\{c, d\}, \{\blacksquare\})\}$.

Unlike conventional Petri nets, there is (1) no initial marking and (2) not an explicit set of transitions. We do not need an initial marking because behaviors start with the unique start activity $\blacktriangleright$. $T = \bigcup_{p \in P} \bullet p \cup p\bullet$ is the implicit set of transitions (corresponding to the inputs and output of places). However, because we use an open-world assumption, we allow for activities not mentioned in the process model. Places only constrain the behavior of the activities explicitly mentioned. For example, $\langle \blacktriangleright, a, d, d, d, b, e, e, c, \blacksquare \rangle$ is a behavior allowed by process model P_1 (simply ignore activities d and e). In the remainder, we will use the terms transition and activity interchangeably. Whereas the term transition is common in the context of Petri nets, event logs refer to occurrences of activities rather than model elements.

2.4 Behavior Defined By Places

To formalize the semantics of a process model $P \subseteq \mathbb{P}$ we define "underfed", "overfed", and "fitting" places. Given a behavior $\sigma \in \mathbb{B}$, place p is underfed if during the replay of the trace place p "goes negative", i.e., a token needs to be consumed while it has not been produced (yet). Place p is overfed if at the end of replaying a trace, tokens remain in p. Place p is fitting if it is not underfed and not overfed, i.e., place p does not "go negative" and at the end no tokens remain.

Definition 5 (Underfed, Overfed, and Fitting Places). *Let* $p \in \mathbb{P}$ *be a place and* $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in \mathbb{B}$ *a behavior.*

- $\bigtriangledown_\sigma(p)$ *if and only if* $|\{1 \leq i < k \mid a_i \in \bullet p\}| < |\{1 \leq i \leq k \mid a_i \in p\bullet\}|$ *for some* $k \in \{1, 2, \ldots, n\}$ *(place* p *is "underfed"),*
- $\bigtriangleup_\sigma(p)$ *if and only if* $|\{1 \leq i \leq n \mid a_i \in \bullet p\}| > |\{1 \leq i \leq n \mid a_i \in p\bullet\}|$ *(place* p *is "overfed"), and*
- $\square_\sigma(p)$ *if and only if* $\not\bigtriangledown_\sigma(p)$ *and* $\not\bigtriangleup_\sigma(p)$ *(place* p *is "fitting", i.e., not "underfed" and not "overfed").*

Consider trace $\sigma = \langle \blacktriangleright, a, b, c, d, \blacksquare \rangle$ and the five places in Figure 1. Places p_1 and p_2 are fitting, p_3 and p_4 are underfed (because d occurs when these places empty), and p_5 is overfed (because two tokens are produced and only one is consumed).

As mentioned before, activities not in $\bullet p \cup p\bullet$ have no effect on the evaluation. If $\sigma = \langle \blacktriangleright, a, e, e, b, f, c, f, d, e, \blacksquare \rangle$, then p_1 and p_2 are still fitting, p_3 and p_4 are still underfed, and p_5 is still overfed.

A place can be both underfed and overfed. Consider trace $\sigma = \langle \blacktriangleright, c, a, a, b, b, \blacksquare \rangle$ and the five places in Figure 1. Places p_1 and p_2 are underfed, p_3 and p_4 are both underfed and overfed (tokens are missing when c occurs and at the end tokens remain), and p_5 is fitting.

The above notions can be generalized to sets of places. Therefore, it is possible to say that a model $P \subseteq \mathbb{P}$ is fitting ($\square_\sigma(P)$), underfed ($\bigtriangledown_\sigma(P)$), or overfed ($\bigtriangleup_\sigma(P)$).

Definition 6. *Let $P \subseteq \mathbb{P}$ be a set of places and $\sigma \in \mathbb{B}$ a behavior.*

- $\bigtriangledown_\sigma(P)$ *if and only if there exists a place $p \in P$ such that $\bigtriangledown_\sigma(p)$,*
- $\bigtriangleup_\sigma(P)$ *if and only if there exists a place $p \in P$ such that $\bigtriangleup_\sigma(p)$, and*
- $\square_\sigma(P)$ *if and only if $\square_\sigma(p)$ for all $p \in P$.*

Using $\bigtriangledown_\sigma(P)$, $\bigtriangleup_\sigma(P)$, and $\square_\sigma(P)$ we can compute all fitting, underfed, and overfed behaviors. The set of fitting behaviors $\mathit{fit}(P)$ precisely defines the semantics of a process model $P \subseteq \mathbb{P}$.

Definition 7 (Model Behavior). *Let $P \subseteq \mathbb{P}$ be a set of places.*

- $\mathit{fit}(P) = \{\sigma \in \mathbb{B} \mid \square_\sigma(P)\}$ *is the set of fitting behaviors,*
- $\mathit{neg}(P) = \{\sigma \in \mathbb{B} \mid \bigtriangledown_\sigma(P)\}$ *is the set of underfed behaviors, and*
- $\mathit{pos}(P) = \{\sigma \in \mathbb{B} \mid \bigtriangleup_\sigma(P)\}$ *is the set of overfed behaviors.*

We use the following shorthands: $\mathit{fit}(p) = \mathit{fit}(\{p\})$, $\mathit{neg}(p) = \mathit{neg}(\{p\})$, $\mathit{pos}(p) = \mathit{pos}(\{p\})$ for any place p. Note that $\mathit{fit}(P) = \mathbb{B} \setminus (\mathit{neg}(P) \cup \mathit{pos}(P))$.

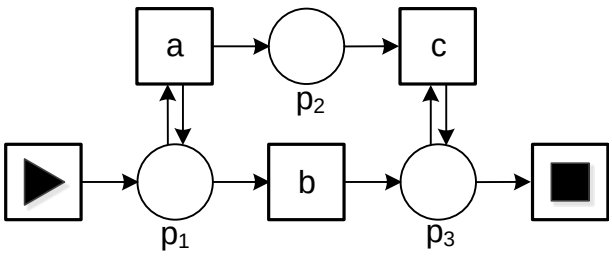

Fig. 3. Process model $P_2 = \{(\{\blacktriangleright, a\}, \{a, b\}), (\{a\}, \{c\}), (\{b, c\}, \{c, \blacksquare\})\}$ composed of three places discovered from event log $L_1 = [\langle \blacktriangleright, b, \blacksquare \rangle^{49}, \langle \blacktriangleright, a, b, c, \blacksquare \rangle^{31}, \langle \blacktriangleright, a, a, b, c, c, \blacksquare \rangle^{12}, \langle \blacktriangleright, a, a, a, b, c, c, c, \blacksquare \rangle^{5}, \langle \blacktriangleright, a, a, a, a, b, c, c, c, c, \blacksquare \rangle^{1}, \langle \blacktriangleright, a, a, a, a, a, b, c, c, c, c, c, \blacksquare \rangle^{2}]$.

Figure 3 shows another example illustrating the declarative nature of places. $P_2 = \{(\{\blacktriangleright, a\}, \{a, b\}), (\{a\}, \{c\}), (\{b, c\}, \{c, \blacksquare\})\}$ has three places allowing for any behavior satisfying the following constraints: (1) b occurs precisely once, (2) a occurs any number of times, but only before b, (3) c occurs any number of times, but only after b, and (4) a and c occur the same number of times. Note that the model in Figure 3 only constrains activities a, b, and c and therefore also allows for behaviors like $\langle \blacktriangleright, d, b, e, f, \blacksquare \rangle$ and $\langle \blacktriangleright, a, d, a, d, b, e, c, c, e, \blacksquare \rangle$.

2.5 Mapping to Petri nets

Traditional *Petri nets* are described by tuple $N = (S, T, F)$ where S is the set of places, T is the set of transitions, and $F \subseteq (S \times T) \cup (T \times S)$ the set of arcs [13]. A *system net* $SN = (S, T, F, M_{init}, M_{final})$ has an initial and a final marking [1]. 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} [1]. The models used in this paper can be converted to a system net using the following conversion. Given a set of places $P \subseteq \mathbb{P}$ (in the sense of Definition 4), we construct the system net $SN = (S, T, F, M_{init}, M_{final})$ with: $S = P \cup \{i, q, f\}$, $T = \mathbb{A}$, $F = \{(i, \blacktriangleright), (\blacksquare, f)\} \cup \{(t, p) \in T \times P \mid t \in {\bullet p}\} \cup \{(p, t) \in P \times T \mid t \in p{\bullet}\} \cup \{(t, q) \mid t \in T \setminus \{\blacksquare\}\} \cup \{(q, t) \mid t \in T \setminus \{\blacktriangleright\}\}$, $M_{init} = [i]$, and $M_{final} = [f]$. The set of traces starting in M_{init} and ending in M_{final} is precisely the set $\mathit{fit}(P)$ (see Definition 7). Moreover, note that SN is a so-called *workflow net* [5]. The workflow net does not need to be sound, but we only consider firing sequences starting in marking $[i]$ and ending in marking $[f]$.

It is also possible to translate any system net (including workflow nets) with initial and final markings into an equivalent model $P \subseteq \mathbb{P}$.

We use the simple representation using merely places and no initial and final markings to be able to succinctly express a range of properties and monotonicity results without considering markings.

3 Relating Places and Monotonicity

The ultimate goal is to discover places from event logs. However, the goal of this paper is not to propose a concrete discovery approach. Instead, we reason about properties of (sets of) places that can be exploited by discovery techniques.

Definition 8 (Place Notations). *Let $p_1 = (I_1, O_1) \in \mathbb{P}$ and $p_2 = (I_2, O_2) \in \mathbb{P}$ be two places. These places can be combined to form new places:*

- $p_1 \sqcap p_2 = (I_1 \cap I_2, O_1 \cap O_2) \in \mathbb{P}$,
- $p_1 \sqcup p_2 = (I_1 \cup I_2, O_1 \cup O_2) \in \mathbb{P}$,
- $p_1 \otimes p_2 = ((I_1 \cup I_2) \setminus (I_1 \cap I_2), (O_1 \cup O_2) \setminus (O_1 \cap O_2)) \in \mathbb{P}$.

Places p_1 and p_2 can be related in different ways:

- $p_1 = p_2$ if and only if $I_1 = I_2$ and $O_1 = O_2$ (equality),

- $p_1 \parallel p_2$ if and only if $p_1 \sqcap p_2 = (\emptyset, \emptyset)$ (non-overlapping),
- $p_1 \sqsubset p_2$ if and only if $I_1 \subseteq I_2$, $O_1 \subseteq O_2$, and $p_1 \neq p_2$ (proper subset), and
- $p_1 \div p_2$ if and only if $p_1 \neq p_2$, $p_1 \not\parallel p_2$, $p_1 \not\sqsubset p_2$ and $p_1 \not\sqsupset p_2$ (incomparable).

We would like to avoid discovering places that are a combination of places already in the model. Consider for example adding place $p_r = (\{\blacktriangleright, c, d\}, \{a, \blacksquare\})$ to the five places in Figure 1. This place would be *redundant*, because $p_r = p_1 \sqcup p_5$. Indeed, adding p_r would not change the set of fitting behaviors and only complicate the model. A set of places is *non-redundant* if none of its places can be derived from the rest.

Definition 9 (Redundant). *Place $p \in \mathbb{P}$ is redundant with respect to a set of places $P \subseteq \mathbb{P}$ (notation $P \Rightarrow p$) if there is a non-empty subset $P' = \{p_1, p_2, \ldots, p_n\} \subseteq P$ such that $p_i \parallel p_j$ for any $1 \leq i < j \leq n$ and $p = (p_1 \sqcup p_2 \sqcup \ldots \sqcup p_n)$.*

For two sets of places $P_1 \subseteq \mathbb{P}$ and $P_2 \subseteq \mathbb{P}$: $P_1 \Rightarrow P_2$ if and only if $\forall_{p_2 \in P_2}\ P_1 \Rightarrow p_2$ (i.e., P_2 is "implied" by P_1).

A set of places $P \subseteq \mathbb{P}$ is non-redundant if and only if it is impossible to split P in two disjoint non-empty subsets P_1 and P_2 such that $P_1 \Rightarrow P_2$.

Adding input transitions to a place can only lead to more tokens in the place. Therefore, a place that is overfed by many traces in the event log will also be overfed by these traces after adding the input transitions. Adding output transitions to a place can only lead to fewer tokens in the place. Therefore, a place that is underfed by many traces in the event log will also be underfed by these traces after adding the output transitions. This information can be used to prune the search space of discovery algorithms. Therefore, we define a partial order on places and use this to prove monotonicity results that can be exploited during process discovery.

Definition 10 (Weighing Places). *Let $p_1 = (I_1, O_1) \in \mathbb{P}$ and $p_2 = (I_2, O_2) \in \mathbb{P}$ be two places.*

- *$p_1 \preceq p_2$ if and only if $I_1 \subseteq I_2$ and $O_2 \subseteq O_1$ (i.e., p_1 is at least as "light" as p_2) and*
- *$p_1 \succeq p_2$ if and only if $I_2 \subseteq I_1$ and $O_1 \subseteq O_2$ (i.e., p_1 is at least as "heavy" as p_2).*

Note that $p_1 \preceq p_2$ if and only if $p_2 \succeq p_1$. It is easy to see that $\preceq$ defines a partial order. The relation is reflexive ($p \preceq p$), antisymmetric ($p_1 \preceq p_2$ and $p_2 \preceq p_1$ implies $p_1 = p_2$), and transitive ($p_1 \preceq p_2$ and $p_2 \preceq p_3$ implies $p_1 \preceq p_3$).

Definition 11 (Weighing Sets of Places). *Let $P_1 \subseteq \mathbb{P}$ and $P_2 \subseteq \mathbb{P}$ be two sets of places.*

- *$P_1 \preceq P_2$ if and only if $\forall_{p_1 \in P_1}\ \exists_{p_2 \in P_2}\ p_1 \preceq p_2$ (i.e., P_1 is at least as "light" as P_2) and*
- *$P_1 \succeq P_2$ if and only if $\forall_{p_1 \in P_1}\ \exists_{p_2 \in P_2}\ p_1 \succeq p_2$ (i.e., P_1 is at least as "heavy" as P_2).*

Note that $P_1 \preceq P_2$ is *not* equivalent to $P_2 \succeq P_1$. Let $P_1 = \{(\{a\}, \{b, c\})\}$ and $P_2 = \{(\{a\}, \{b\}), (\{a\}, \{d\})\}$. $P_1 \preceq P_2$ because $(\{a\}, \{b, c\}) \preceq (\{a\}, \{b\})$. However, $P_2 \not\succeq P_1$, because $(\{a\}, \{d\}) \not\succeq (\{a\}, \{b, c\})$. Both $\preceq$ and $\succeq$ (for sets of places) are reflexive and transitive, but not antisymmetric. Consider $P_3 = \{(\{a, c\}, \{b\}), (\{a\}, \{b\}), (\{a\}, \{b, d\})\}$ and $P_4 = \{(\{a, c\}, \{b\}), (\{a\}, \{b, d\})\}$. $P_3 \preceq P_4$ and $P_4 \preceq P_3$, but $P_3 \neq P_4$. Also, $P_3 \succeq P_4$ and $P_4 \succeq P_3$, but $P_3 \neq P_4$. Hence, $\preceq$ and $\succeq$ are not antisymmetric.

The above notations and insights allow us to provide very general monotonicity results.

Theorem 1 (Monotonicity Results). *Let $P_1 \subseteq \mathbb{P}$ and $P_2 \subseteq \mathbb{P}$ be two sets of places.*

- *$P_1 \preceq P_2$ implies $pos(P_1) \subseteq pos(P_2)$,*
- *$P_1 \succeq P_2$ implies $neg(P_1) \subseteq neg(P_2)$,*
- *$P_1 \Rightarrow P_2$ implies $fit(P_1) \subseteq fit(P_2)$.*

Proof. If $p_1 \preceq p_2$, then while replaying a trace σ, p_1 cannot have more tokens than p_2, but p_2 can have more tokens than p_1 if the right transitions are activated. Therefore, if $\triangle_\sigma(p_1)$, then $\triangle_\sigma(p_2)$, and if $\triangledown_\sigma(p_2)$, then $\triangledown_\sigma(p_1)$.

Using this insight we prove that $P_1 \preceq P_2$ implies $pos(P_1) \subseteq pos(P_2)$. Assume $P_1 \preceq P_2$, i.e., $\forall_{p_1 \in P_1}\ \exists_{p_2 \in P_2}\ p_1 \preceq p_2$. We need to prove that for any $\sigma \in \mathbb{B}$: $\exists_{p_1 \in P_1}\ \triangle_\sigma(p_1)$ implies $\exists_{p_2 \in P_2}\ \triangle_\sigma(p_2)$. Take a p_1 such that $\triangle_\sigma(p_1)$. There exists a $p_2 \in P_2$ such that $p_1 \preceq p_2$. Place p_2 can only have more tokens than p_1 (and not fewer). Hence, $\triangle_\sigma(p_2)$.

Similarly, we can prove that $P_1 \succeq P_2$ implies $neg(P_1) \subseteq neg(P_2)$.

$P_1 \Rightarrow P_2$ means that all places in P_2 correspond to combinations of places in P_1. Therefore, adding these places does not change the behavior, i.e., $fit(P_1) = fit(P_1 \cup P_2)$. Removing places from $P_1 \cup P_2$ can only result in more fitting traces. Hence, $fit(P_1) = fit(P_1 \cup P_2) \subseteq fit(P_2)$. □

4 Scoring Places

Theorem 1 can be exploited by process discovery algorithms. If a place is underfed (overfed), it does not make sense to consider lighter (heavier) places. Therefore, monotonicity results allow for quickly pruning the search space. To illustrate this, we define concrete quality characteristics for individual places.

One could simply count the fraction of cases having problems. However, some activities may occur infrequently. Places that are only connected to these low-frequency activities have many fitting traces by definition (the place is rarely involved in the execution of a case). In other words, "random places" only connected to low-frequency activities will always have a good score. Therefore, we also consider the "relative" scores of places by only considering traces that actually consume/produce tokens from/for the place under investigation. A trace "activates" place p if it contains an activity in $\bullet p \cup p\bullet$.

Definition 12 (Activation). *Let $p \in \mathbb{P}$ be a place.*

- $act_\sigma(p) = \exists_{a \in \sigma}\ a \in (\bullet p \cup p\bullet)$ *denotes whether the place has been activated in a trace* $\sigma \in \mathbb{B}$*, i.e., a token was consumed or produced for* p *in* σ*.*
- $act_L(p) = \exists_{\sigma \in L}\ act_\sigma(p)$ *denotes whether place* p *has been activated in an event log* $L \in \mathcal{B}(\mathbb{B})$*.*

Definition 13 (Place Scores). *Let* $L \in \mathcal{B}(\mathbb{B})$ *be an event log and* $\tau \in [0,1]$ *a threshold. For any place* $p \in \mathbb{P}$ *such that* $act_L(p)$*, we define the following scores:*

- $\#^{\bigtriangledown}_{freq,L}(p) = \frac{|\,[\sigma \in L | \bigtriangledown_\sigma(p)]\,|}{|L|}$ *is the fraction of traces for which* p *is underfed,*
- $\#^{\bigtriangleup}_{freq,L}(p) = \frac{|\,[\sigma \in L | \bigtriangleup_\sigma(p)]\,|}{|L|}$ *is the fraction of traces for which* p *is overfed,*
- $\#^{\square}_{freq,L}(p) = \frac{|\,[\sigma \in L | \square_\sigma(p)]\,|}{|L|}$ *is the fraction of fitting traces,*
- $\#^{\bigtriangledown}_{rel,L}(p) = \frac{|\,[\sigma \in L | \bigtriangledown_\sigma(p)\ \wedge\ act_\sigma(p)]\,|}{|\,[\sigma \in L | act_\sigma(p)]\,|} = \frac{|\,[\sigma \in L | \bigtriangledown_\sigma(p)]\,|}{|\,[\sigma \in L | act_\sigma(p)]\,|}$ *is the fraction of activating traces for which* p *is underfed,*
- $\#^{\bigtriangleup}_{rel,L}(p) = \frac{|\,[\sigma \in L | \bigtriangleup_\sigma(p)\ \wedge\ act_\sigma(p)]\,|}{|\,[\sigma \in L | act_\sigma(p)]\,|} = \frac{|\,[\sigma \in L | \bigtriangleup_\sigma(p)]\,|}{|\,[\sigma \in L | act_\sigma(p)]\,|}$ *is the fraction of activating traces for which* p *is overfed,*
- $\#^{\square}_{rel,L}(p) = \frac{|\,[\sigma \in L | \square_\sigma(p)\ \wedge\ act_\sigma(p)]\,|}{|\,[\sigma \in L | act_\sigma(p)]\,|}$ *is the fraction of activated traces that are also fitting,*
- $\bigtriangledown^\tau_{freq,L}(p)$ *if and only if* $\#^{\bigtriangledown}_{freq,L}(p) > \tau$*,*
- $\bigtriangleup^\tau_{freq,L}(p)$ *if and only if* $\#^{\bigtriangleup}_{freq,L}(p) > \tau$*,*
- $\square^\tau_{freq,L}(p)$ *if and only if* $\#^{\square}_{freq,L}(p) \geq \tau$*,*
- $\bigtriangledown^\tau_{rel,L}(p)$ *if and only if* $\#^{\bigtriangledown}_{rel,L}(p) > \tau$*,*
- $\bigtriangleup^\tau_{rel,L}(p)$ *if and only if* $\#^{\bigtriangleup}_{rel,L}(p) > \tau$*,*
- $\square^\tau_{rel,L}(p)$ *if and only if* $\#^{\square}_{rel,L}(p) \geq \tau$*.*

For a discovered place we would like $\#^{\square}_{freq,L}(p)$ and $\#^{\square}_{rel,L}(p)$ to be as high as possible. A place p is perfectly fitting log L if $\#^{\square}_{freq,L}(p) = \#^{\square}_{rel,L}(p) = 1$. If $\square^{0.95}_{rel,L}(p)$, then at least 95% of all traces that activate place p are fitting. If a discovery algorithm only adds places for which $\square^{0.95}_{rel,L}(p)$, then all places have a minimal quality level interpretable by end users (unlike existing approaches that do not provide such a guarantee or "only in the limit").

5 Monotonicity of Place Scores

Since we are interested in places of a certain quality level, e.g., places for which $\square^{0.95}_{rel,L}(p)$ holds, we would like to avoid spending time on the evaluation of places that do not meet the desired quality level. We would like to use Theorem 1 to quickly prune the set of candidate places. We start by listing several observations that directly follow from earlier definitions.

Lemma 1. *Let* $L \in \mathcal{B}(\mathbb{B})$ *be an event log,* $\sigma \in \mathbb{B}$ *a trace, and* $\tau \in [0,1]$ *a threshold. For any place* $p \in \mathbb{P}$ *such that* $act_L(p)$*:*

- $\bigtriangledown_\sigma(p)$ *implies* $act_\sigma(p)$*,*

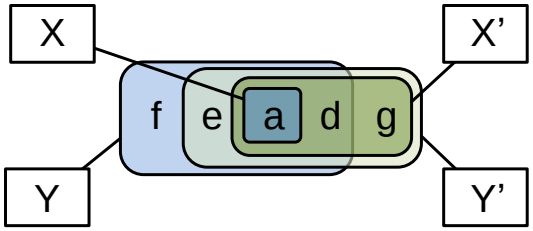

Fig. 4. Visualization of the sets used in Lemma 2. In Theorem 2: $X = [\sigma \in L \mid \triangle_\sigma(p_1)]$, $Y = [\sigma \in L \mid act_\sigma(p_1)]$, $X' = [\sigma \in L \mid \triangle_\sigma(p_2)]$, and $Y' = [\sigma \in L \mid act_\sigma(p_2)]$.

- $\triangle_\sigma(p)$ *implies* $act_\sigma(p)$,
- $\#^{\square}_{freq,L}(p) \leq 1 - \#^{\triangledown}_{freq,L}(p)$,
- $\#^{\square}_{freq,L}(p) \leq 1 - \#^{\triangle}_{freq,L}(p)$,
- $\#^{\square}_{freq,L}(p) \geq 1 - (\#^{\triangledown}_{freq,L}(p) + \#^{\triangle}_{freq,L}(p))$,
- $\#^{\square}_{rel,L}(p) \leq 1 - \#^{\triangledown}_{rel,L}(p)$,
- $\#^{\square}_{rel,L}(p) \leq 1 - \#^{\triangle}_{rel,L}(p)$,
- $\#^{\square}_{rel,L}(p) \geq 1 - (\#^{\triangledown}_{rel,L}(p) + \#^{\triangle}_{rel,L}(p))$,
- $\square^{\tau}_{freq,L}(p)$ *implies* $\not\triangle^{1-\tau}_{freq,L}(p)$,
- $\square^{\tau}_{freq,L}(p)$ *implies* $\not\triangledown^{1-\tau}_{freq,L}(p)$,
- $\square^{\tau}_{rel,L}(p)$ *implies* $\not\triangle^{1-\tau}_{rel,L}(p)$,
- $\square^{\tau}_{rel,L}(p)$ *implies* $\not\triangledown^{1-\tau}_{rel,L}(p)$,
- $\not\triangledown^{\tau}_{freq,L}(p)$ *and* $\not\triangle^{\tau}_{freq,L}(p)$ *implies* $\square^{1-2\times\tau}_{freq,L}(p)$, *and*
- $\not\triangledown^{\tau}_{rel,L}(p)$ *and* $\not\triangle^{\tau}_{rel,L}(p)$ *implies* $\square^{1-2\times\tau}_{rel,L}(p)$.

Proof. Note that $\triangledown_\sigma(p)$ implies $\not\triangle_\sigma(p)$, $\triangle_\sigma(p)$ implies $\not\triangle_\sigma(p)$, and $\square_\sigma(p)$ implies $\not\triangledown_\sigma(p)$ and $\not\triangle_\sigma(p)$. These insights can be used to verify the properties listed.

Consider for example the last property. Assume $\not\triangledown^{\tau}_{rel,L}(p)$ and $\not\triangle^{\tau}_{rel,L}(p)$. Since $\#^{\triangledown}_{rel,L}(p) \leq \tau$ and $\#^{\triangle}_{rel,L}(p) \leq \tau$, we know $\#^{\square}_{rel,L}(p) \geq 1 - (\#^{\triangledown}_{rel,L}(p) + \#^{\triangle}_{rel,L}(p)) \geq 1 - (\tau + \tau)$. Hence, $\square^{1-2\times\tau}_{rel,L}(p)$. □

Before we show monotonicity with respect to the place scores, we first prove the following lemma.

Lemma 2. *Let X, Y, X', and Y' be sets such that $Y \neq \emptyset$, $Y' \neq \emptyset$, $X \subseteq Y$, $X' \subseteq Y'$, $X \subseteq X'$, and $Y' \setminus Y \subseteq X'$.*

$$\frac{|X|}{|Y|} \leq \frac{|X'|}{|Y'|}$$

Proof. Let $A = Y \cup Y'$, $a = |X \cap X'|$, $b = |X \cap (Y' \setminus X')|$, $c = |X \cap (A \setminus Y')|$, $d = |(Y \setminus X) \cap X'|$, $e = |(Y \setminus X) \cap (Y' \setminus X')|$, $f = |(Y \setminus X) \cap (A \setminus Y')|$, $g =$

$|(A \setminus Y) \cap X'|$, $h = |(A \setminus Y) \cap (Y' \setminus X')|$, and $i = |(A \setminus Y) \cap (A \setminus Y')|$ (see Figure 4). Because $X \subseteq X'$, $b = c = 0$. Because $Y' \setminus Y \subseteq X'$, $h = 0$. Also $i = 0$. Hence, $|X| = a$, $|Y| = a + d + e + f$, $|X'| = a + d + g$, $|Y'| = a + d + e + g$.

$$\frac{|X|}{|Y|} = \frac{a}{a+d+e+f} \leq \frac{a}{a+d+e} \leq \frac{a+g}{a+d+e+g} \leq \frac{a+d+g}{a+d+e+g} = \frac{|X'|}{|Y'|}$$

Note that $\frac{a}{a+d+e} \leq \frac{a+g}{a+d+e+g}$ because $a(a+d+e+g) = a^2 + ad + ae + ag \leq a^2 + ad + ae + ag + dg + eg = (a+g)(a+d+e)$. □

Recall that our goal is to dismiss candidate places that are overfed or underfed as soon as possible. Given a threshold τ we would like to avoid checking the quality of places for which $\triangle^{\tau}_{freq,L}(p)$, $\triangledown^{\tau}_{freq,L}(p)$, $\triangle^{\tau}_{rel,L}(p)$, or $\triangledown^{\tau}_{rel,L}(p)$. Using the partial order on places, we can exploit the following monotonicity result.

Theorem 2 (Monotonicity). *Let $L \in \mathcal{B}(\mathbb{B})$ be a non-empty event log and let $\tau \in [0,1]$ be some threshold. For any two places $p_1, p_2 \in \mathbb{P}$ such that $p_1 \preceq p_2$:*

- *If $\triangle^{\tau}_{freq,L}(p_1)$, then $\triangle^{\tau}_{freq,L}(p_2)$.*
- *If $\triangledown^{\tau}_{freq,L}(p_2)$, then $\triangledown^{\tau}_{freq,L}(p_1)$.*

Moreover, if $act_L(p_1)$ and $act_L(p_2)$, then these findings also apply to the relative notion.

- *If $\triangle^{\tau}_{rel,L}(p_1)$, then $\triangle^{\tau}_{rel,L}(p_2)$.*
- *If $\triangledown^{\tau}_{rel,L}(p_2)$, then $\triangledown^{\tau}_{rel,L}(p_1)$.*

Proof. Assume $\triangle^{\tau}_{freq,L}(p_1)$. Hence, $\#^{\triangle}_{freq,L}(p_1) = \frac{|[\sigma \in L | \triangle_\sigma(p_1)]|}{|L|} \geq \tau$. Since $\triangle_\sigma(p_1)$ implies $\triangle_\sigma(p_2)$ for any $\sigma \in L$, $\frac{|[\sigma \in L | \triangle_\sigma(p_2)]|}{|L|} \geq \frac{|[\sigma \in L | \triangle_\sigma(p_1)]|}{|L|}$. Hence, $\triangle^{\tau}_{freq,L}(p_2)$. Similarly, we can show that $\triangledown^{\tau}_{freq,L}(p_2)$ implies $\triangledown^{\tau}_{freq,L}(p_1)$.

Assume $\triangle^{\tau}_{rel,L}(p_1)$. Hence, $\#^{\triangle}_{rel,L}(p_1) = \frac{|[\sigma \in L | \triangle_\sigma(p_1)]|}{|[\sigma \in L | act_\sigma(p_1)]|} \geq \tau$. Using Lemma 2 we show that $\#^{\triangle}_{rel,L}(p_2) = \frac{|[\sigma \in L | \triangle_\sigma(p_2)]|}{|[\sigma \in L | act_\sigma(p_2)]|} \geq \tau$. Let $X = [\sigma \in L \mid \triangle_\sigma(p_1)]$, $Y = [\sigma \in L \mid act_\sigma(p_1)]$, $X' = [\sigma \in L \mid \triangle_\sigma(p_2)]$, and $Y' = [\sigma \in L \mid act_\sigma(p_2)]$. X, X', Y, and Y' are multisets. However, for simplicity assume that each case is uniquely identifiable so that we can treat these as sets. One can use case identifiers to identify traces even when they are identical. To apply Lemma 2, we first check the conditions: $Y \neq \emptyset$ because $act_L(p_1)$, $Y' \neq \emptyset$ because $act_L(p_2)$, $X \subseteq Y$ because $\triangle_\sigma(p_1)$ implies $act_\sigma(p_1)$, $X' \subseteq Y'$ because $\triangle_\sigma(p_2)$ implies $act_\sigma(p_2)$, $X \subseteq X'$ because $\triangle_\sigma(p_1)$ implies $\triangle_\sigma(p_2)$, and $Y' \setminus Y \subseteq X'$, because if $act_\sigma(p_2)$ and not $act_\sigma(p_1)$, then $\triangle_\sigma(p_2)$. The last observation holds because $p_1 \preceq p_2$, $act_\sigma(p_2)$ and not $act_\sigma(p_1)$ implies that a token was put in p_2 and it was not removed. Note that all output arcs of p_2 are also output arcs of p_1. Hence, the token put in p_2 cannot be removed. Therefore, $\triangle_\sigma(p_2)$. Applying Lemma 2 shows that $\#^{\triangle}_{rel,L}(p_1) \leq \#^{\triangle}_{rel,L}(p_2)$, proving that $\triangle^{\tau}_{rel,L}(p_2)$. Similarly, we can show that $\triangledown^{\tau}_{rel,L}(p_2)$ implies $\triangledown^{\tau}_{rel,L}(p_1)$. □

6 Exploiting Monotonicity During Discovery

The goal of this paper is not to provide a particular discovery algorithm. However, Theorem 2 provides the basis for Apriori-style algorithms [8, 9, 20]. Such algorithms are used in frequent item-set mining, association rule learning, sequence mining, and episode mining. The basic idea of such algorithms is to avoid spending time on "hopeless candidate patterns" by dramatically pruning the search space. For example, in a supermarket, the number of customers buying products A, B, and C is smaller than (1) the number of the customers buying products A and B, (2) the number of the customers buying products A and C, and (3) the number of the customers buying products B and C. Hence, if one of the latter three groups ($\{A, B\}$, $\{A, C\}$, or $\{B, C\}$) is infrequent, then by definition also the former group ($\{A, B, C\}$) is infrequent. Obviously, we can use the monotonicity results presented in this paper in a similar fashion.

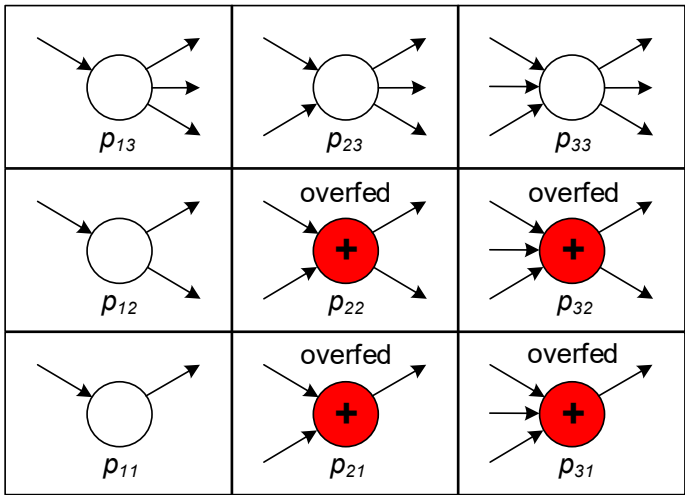

Fig. 5. If place p_{22} is already overfed, then we know that the places p_{32}, p_{21}, and p_{31} also need to be overfed.

Figure 5 sketches the situation where we have evaluated a place p_{22} and the place turned out to be overfed, i.e., at the end of a trace tokens remain. Obviously, the place remains overfed when we remove an output arc or add an input arc. Therefore, by definition, p_{32}, p_{21}, and p_{31} also need to be overfed. Figure 6 describes to opposite situation. If place p_{22} "goes negative" when replaying a trace (i.e., the place is underfed), then the place remains underfed when we remove an input arc or add an output arc. Hence, p_{13}, p_{23}, and p_{12} also need to be underfed.

We can use these insights to prune the search space of candidate places. Assume we have 80 candidate places as shown in Figure 7(a). We can pick

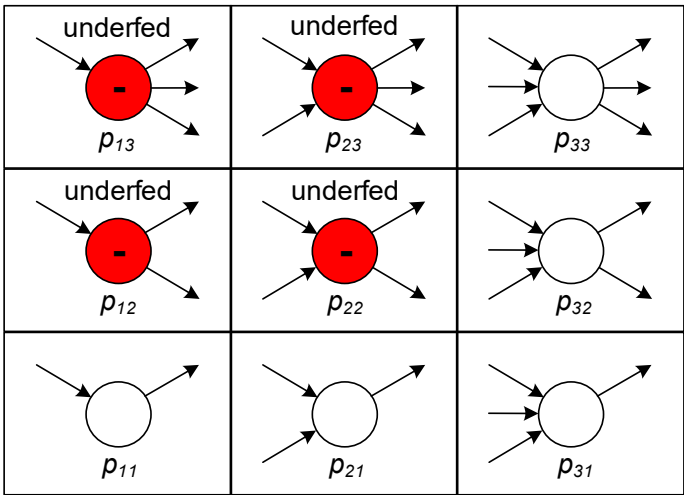

Fig. 6. If place p_{22} is already underfed, then we know that the places p_{13}, p_{23}, and p_{12} also need to be underfed.

a random candidate place, say Place 1. If this place is underfed according to some criterion (e.g., in more than 10% of the traces the place does not have enough tokens at some stage), then we can identify lighter places that must have the same problem. Assume that Place 1 is indeed underfed and has the lighter neighboring places highlighted in Figure 7(b). As a result, we can remove 16 candidate places by just evaluating Place 1. Then we pick the next random candidate place, say Place 2. If this place is overfed (e.g., in more than 10% of the traces Place 2 was not empty at the end), then we can identify all heavier places that must have the same problem. These are removed. Figure 7(c) shows that we can remove 15 candidate places by just evaluating Place 2. The next randomly selected candidate place turns out to be fitting (e.g., in 90% of the traces Place 3 was empty at the end and in 90% of the traces there were sufficient tokens), i.e., Place 3 is not underfed and not overfed. In the next step, we remove another 16 candidate places because Place 4 is overfed (Figure 7(d)). Then we remove another set of places because Place 5 is underfed (Figure 7(e)). We can repeat the process until there no candidate places left. Figure 7(f) shows the remaining three places. Note that an evaluated place can be both underfitting and overfitting. When encountering such places, the search space can be pruned in two directions (remove all lighter and heavier places). As sketched in Figure 7, it will often be the case that only a fraction of the candidate places needs to be evaluated using replay techniques.

Figure 7 only sketches the idea and places are selected randomly. One can also think of smarter strategies. For example, one can start with places having just a few connections. Let p_c be a candidate place having n input and output activities, i.e., $n = |\bullet p_c| + |p_c\bullet|$ is the number of arcs. Let $A^{\triangle} = \{p \in \mathbb{P} \mid |\bullet p| + |p\bullet| < n \ \wedge \ p \succeq p_c\}$ and $A^{\triangledown} = \{p \in \mathbb{P} \mid |\bullet p| + |p\bullet| < n \ \wedge \ p \preceq p_c\}$ be the

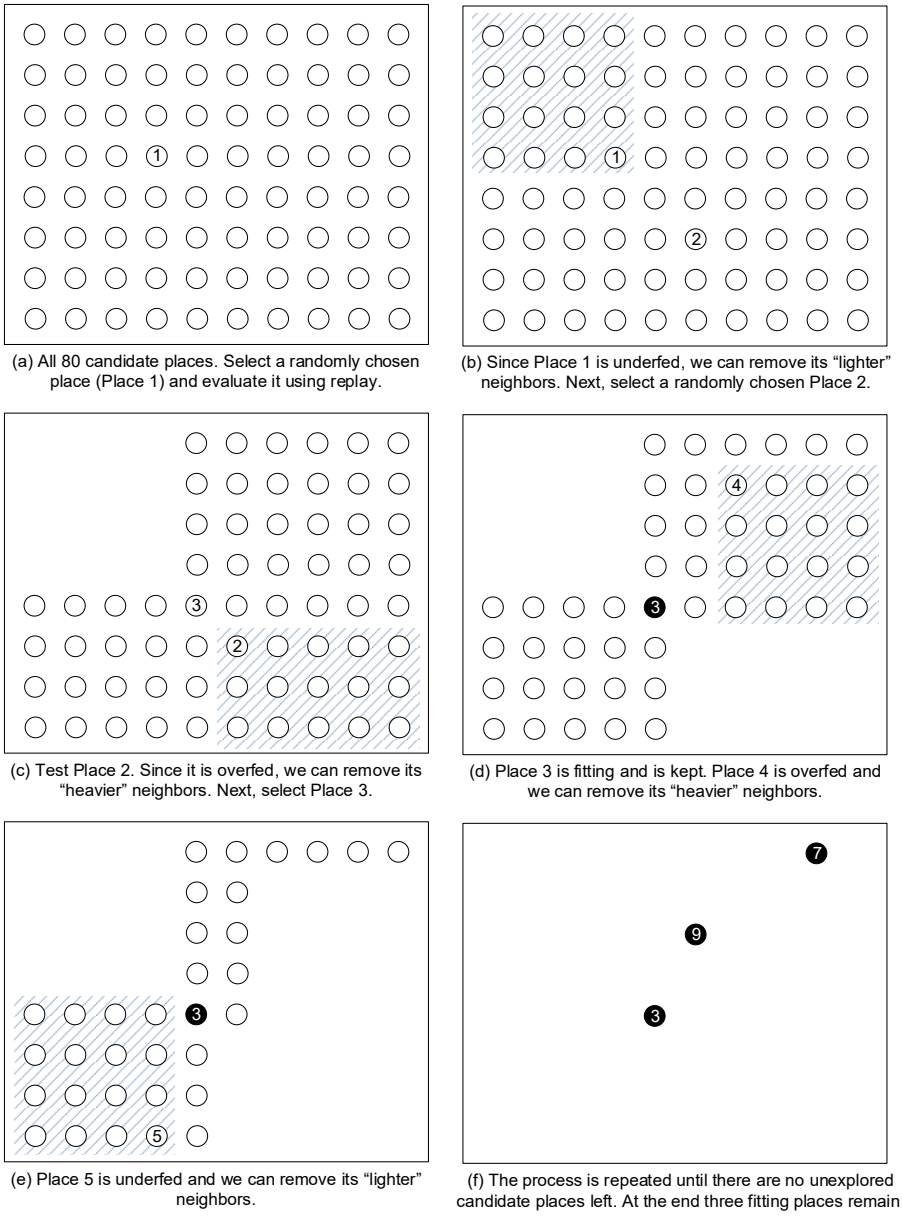

Fig. 7. A process discovery algorithm could simply evaluate all places and only keep those that are fitting (i.e., meet certain quality criteria). However, the monotonicity results in Theorem 2 show that there are many candidate places that we do not need to check. This is the key to discovering places efficiently.

heavier and lighter ancestors of p_c. If for any $p \in A^{\triangle}$, $\bigtriangledown^{\tau}_{freq,L}(p)$ or $\bigtriangledown^{\tau}_{rel,L}(p)$, then we know that $\bigtriangledown^{\tau}_{freq,L}(p_c)$ or $\bigtriangledown^{\tau}_{rel,L}(p_c)$. If for any $p \in A^{\bigtriangledown}$, $\bigtriangleup^{\tau}_{freq,L}(p)$ or $\bigtriangleup^{\tau}_{rel,L}(p)$, then we know that $\bigtriangleup^{\tau}_{freq,L}(p_c)$ or $\bigtriangleup^{\tau}_{rel,L}(p_c)$. These properties can be used to avoid certain checks.

There are many more ways to speedup the search process further:

- Suppose that we consider a place to be overfed when at least 10% of the traces have remaining tokens. This means that we can abort the place evaluation when we have found 10% of traces having problems (for poorly fitting places this may be reached quickly).
- A similar strategy can be used for underfed places. Moreover, the replay of a trace can be aborted when the first problem is encountered.
- If we know the frequencies of all activities in the log, we can do an initial check to see whether the sum of the frequencies of the input activities approximately matches the sum of the frequencies of the input activities (this is also used in [6]). Such aggregate information can be used to guide the pruning process. There can even be guarantees, provided that we can make assumptions about the distribution of activities over traces or bound the trace length.

In short, there are many ways to exploit the monotonicity results provided in this paper.

7 Further Pruning of the Search Space

Next to avoiding checks for places that are obviously "too light" or "too heavy", we would also like to avoid adding redundant and conflicting places.

In Theorem 1, we showed that $P_1 \Rightarrow P_2$ implies $\mathit{fit}(P_1) \subseteq \mathit{fit}(P_2)$. Hence, adding redundant places does not limit the set of fitting traces and therefore only complicates the process model. This can be exploited while constructing a process model. If two non-overlapping places p_1 and p_2 have been added, one should not consider adding place $p = p_1 \sqcup p_2$.

Moreover, we would also like to avoid adding conflicting places. If two places are in conflict (notation $p_1 \# p_2$), then there are traces that could never fit both places.

Definition 14 (Conflict). *Let $p_1, p_2 \in \mathbb{P}$ be two places. $p_1 \prec p_2$ if and only if $p_1 \preceq p_2$ and $p_1 \neq p_2$. $p_1 \succ p_2$ if and only if $p_1 \succeq p_2$ and $p_1 \neq p_2$. $p_1 \# p_2$ if and only if $p_1 \prec p_2$ and $p_1 \succ p_2$.*

Theorem 3. *Let $p_1, p_2 \in \mathbb{P}$ be two places and $\sigma \in \mathbb{B}$ a trace. If $\square_\sigma(p_1)$, $p_1 \# p_2$, and $act_\sigma(p_1 \otimes p_2)$, then $\not\triangle_\sigma(p_2)$.*

Proof. Assume $\square_\sigma(p_1)$, $p_1 \# p_2$, and $act_\sigma(p_1 \otimes p_2)$. Hence, $p_1 \prec p_2$ or $p_1 \succ p_2$. If $p_1 \prec p_2$, then the number of tokens in p_2 is always at least the number of tokens in p_1 for any sequence, including σ. In fact, in σ there is at least one additional token produced or a token was not consumed (because $act_\sigma(p_1 \otimes p_2)$). Because

p_1 ends empty, p_2 must have a remaining token at the end. Hence, σ cannot be fitting ($\not\triangleq_\sigma(p_2)$). If $p_1 \succ p_2$, then the number of tokens in p_2 is always at most the number of tokens in p_1. In fact, there is at least one additional token consumed or a token was not produced (because $act_\sigma(p_1 \otimes p_2)$). Because p_1 ends empty, p_2 must have a missing token at the end. Hence, σ cannot be fitting. □

Consider places $p_1 = (\{a\}, \{b, c\})$ and $p_2 = (\{a, d\}, \{b\})$. Obviously, $p_1 \prec p_2$. For a trace involving c and/or d activities, it can never be the case that both places are fitting. Since p_1 and p_2 disagree on the allowed behavior, one would not like to add both to the same process model. Also this property can be exploited during discovery.

8 How About Conformance Checking?

Process mining is not limited to process discovery and includes conformance checking, model repair, performance analysis, decision mining, and organizational mining. Moreover, also predictive and prescriptive analytics are supported by process mining tools and techniques [3]. Viewing a process model as merely a collection of independent places may also help to expedite these other analysis tasks. Consider for example conformance checking which involves detecting and diagnosing both differences and commonalities between an event log and a process model [4]. Typically, four dimensions are distinguished: fitness, precision, generalization, and simplicity. State-of-the-art techniques use so-called alignments or token-based replay [3]. However, these techniques often do not have the monotonicity properties one expects [23]. For example, removing a place should never result in a better precision or lower fitness. In [2] a probabilistic angle is added to these questions, also revealing obvious problems related to existing conformance measures. The monotonicity results presented in this paper may provide a fresh look on conformance problems. First of all, it could be good to check places individually. Second, there are ways to quickly analyze whether a place is overfed and/or underfed.

9 Conclusion

For this Festschrift celebrating Farhad Arbab's achievements in coordination models and languages, I decided to focus on the discovery of a very simple coordination model: "places". We like to learn such coordination structures from observed behaviors.

The process models in this paper are fully described by places. For the semantics, we employ an open-world assumption and special start and end activities. This yields a representation very suitable for process mining. Given a particular behavior, a place can be "fitting", "underfed" (tokens are missing), or "overfed" (tokens are remaining). We would like to discover fitting places from event data satisfying predefined quality criteria. This is not a trivial task and for large event

logs this easily becomes intractable. Therefore, we studied monotonicity properties in the context of event logs. For example, if place p_1 is "lighter" than place p_2 (i.e., $p_1 \preceq p_2$) and 5% of the activated traces produce too few tokens for p_2 ($\bigtriangledown^{0.05}_{rel,L}(p_2)$), then the same traces also produce too few tokens for p_1 (i.e., $\bigtriangledown^{0.05}_{rel,L}(p_1)$). This helps to prune the set of candidate places. Moreover, also notions like redundancy and conflict can be used to reduce the search space further. These properties allow for new Apriori-style algorithms. The insights could also be used to speed-up the discovery of hybrid process models [6].

This contribution did not show how to synthesize Reo circuits from event data. However, this remains an interesting question and I encourage the Reo community to look into this. Finally, I would like to wish Farhad all the best and hope that he will remain working on the "science of coordination" after his "coordination of science" activities at CWI have ended.

References

1. W.M.P. van der Aalst. Decomposing Petri Nets for Process Mining: A Generic Approach. *Distributed and Parallel Databases*, 31(4):471–507, 2013.
2. W.M.P. van der Aalst. Mediating Between Modeled and Observed Behavior: The Quest for the "Right" Process. In *IEEE International Conference on Research Challenges in Information Science (RCIS 2013)*, pages 31–43. IEEE Computing Society, 2013.
3. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.
4. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
5. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
6. 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. In J. Carmona, G. Engels, and A. Kumar, editors, *International Conference on Business Process Management (BPM 2017)*, volume 10445 of *Lecture Notes in Computer Science*, pages 59–76. Springer-Verlag, Berlin, 2017.
7. W.M.P. van der Aalst, V. Rubin, H.M.W. Verbeek, B.F. van Dongen, E. Kindler, and C.W. Günther. Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. *Software and Systems Modeling*, 9(1):87–111, 2010.
8. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499, Santiago de Chile, Chile, 1994. Morgan Kaufmann Publishers Inc.
9. R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE Computer Society, 1995.
10. F. Arbab. Reo: A Channel-Based Coordination Model for Component Composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

11. 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.
12. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.
13. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
14. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.
15. 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.
16. S.S.T.Q. Jongmans and F. Arbab. Overview of Thirty Semantic Formalisms for Reo. *Scientific Annals of Computer Science*, 22(1):201–251, 2012.
17. 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.
18. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Incomplete Event Logs. In G. Ciardo and E. Kindler, editors, *Applications and Theory of Petri Nets 2014*, volume 8489 of *Lecture Notes in Computer Science*, pages 91–110. Springer-Verlag, Berlin, 2014.
19. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery with Guarantees. In K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma, editors, *Enterprise, Business-Process and Information Systems Modeling (BPMDS 2015)*, volume 214 of *Lecture Notes in Business Information Processing*, pages 85–101. Springer-Verlag, Berlin, 2015.
20. H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
21. S. Meng, F. Arbab, and C. Baier. Synthesis of Reo Circuits From Scenario-Based Interaction Specifications. *Science of Computer Programming*, 76(8):651–680, 2011.
22. 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.
23. N. Tax, X. Lu, N. Sidorova, D. Fahland, and W.M.P. van der Aalst. The Imprecisions of Precision Measures in Process Mining. *Information Processing Letters*, 135:1–8, 2018.
24. 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.
25. 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.