Learning Generalized Stochastic Petri Nets From Event Data

Wil M.P. van der Aalst$^{[0000-0002-0955-6940]}$ and Sander J.J. Leemans$^{[0000-0002-5201-7125]}$

RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany
wvdaalst@pads.rwth-aachen.de,s.leemans@bpm.rwth-aachen.de

Abstract. Generalized Stochastic Petri Nets (GSPNs) are an established tool for representing and analyzing concurrency, timing, synchronization, precedence, and priority in processes. GSPNs emerged in the 1980s as the de facto standard for modeling stochastic processes using Petri nets, supported by tools such as GreatSPN. However, traditional applications of this technology assume that GSPNs are created manually. Given the widespread availability of event data in information systems, this seems sub-optimal. This explains the uptake of process mining, which starts from event data instead of manually created process models. There are dozens of techniques to discover basic (i.e., non-stochastic) Petri nets given an event log. However, there is an increasing interest in not just discovering control flow, but also learning the stochastic behavior based on event data. Therefore, we take GSPNs as the target representation for process discovery. Since there are numerous techniques to discover the control flow, we focus on the extensions provided by GSPNs. These include priorities, blocking, probabilities, and rates. In this paper, we sketch a concrete approach to discover probabilities and rates from event data. This is done by translating to GSPNs to Markov chains to which parameter synthesis is applied. Since priorities and blocking can be added without limiting GSPN-based performance analysis, we advocate the development of control-flow discovery techniques incorporating these features. Having GSPNs learned from event data, we can support more forward-looking forms of process mining.

Keywords: Petri nets · Process mining · Generalized Stochastic Petri Nets

1 Introduction

Initially, Petri nets could not be used to model time and probabilities. In the 1970s, several researchers proposed ways of adding *time* to Petri nets [40,37]. *Stochastic Petri Nets* (SPNs) were introduced in 1980s [22,38]. In SPNs, all transitions are assumed to be atomic having exponentially distributed firing times using a so-called "race execution" policy. With the aim of extending the expressiveness of SPNs, *Generalized Stochastic Petri Nets* (GSPNs) were introduced in [36]. GSPNs have two types of transitions: exponentially distributed *timed* transitions (modeling durations) and *immediate* transitions (i.e., transitions that fire immediately without any delay). This leads to the concept of "vanishing" states and "tangible" states. By definition, the time spent in vanishing states is zero (i.e., a transition fires immediately). Time progresses only in tangible states waiting for a time transition to fire. Over time, GSPNs were extended with known concepts such as inhibitor arcs and priorities [17,6,23]. *GreatSPN* is an example of a tool that supports the modeling, validation, and performance evaluation of GSPNs that has been around for almost four decades [9]. Next to simulation, GreatSPN and other tools support the automatic translation of GSPNs to Continuous-Time Markov Chains (CTMCs). These CTMCs can be used to give exact answers to questions ranging from the utilization of resources to flow times and throughput.

Joost-Pieter Katoen has been (and still is) one of the leading researchers in formal methods combining model checking, concurrency theory, and probabilistic reasoning and programming. The first author vividly remembers his invited talk at Petri Nets 2012 and ACSD 2012 in Hamburg where he talked about “GSPNs Revisited: Simple Semantics and New Analysis Algorithms” [25,21]. Together with colleagues, he provided alternative semantics for GSPNs addressing the well-known confusion problem in non-free-choice nets. Joost-Pieter Katoen also applied GSPN to fault trees and Markov decision problems [24]. In recent work, he also exploits probabilistic model checking and parameter synthesis techniques for parametric Markov chains to find the right probabilities [43,19,39]. This is highly related to the topic of this paper.

Process mining is an “evidence-driven” approach to analyzing, visualizing, and improving business processes based on event data readily available in modern information systems [1,2]. Unlike classical approaches using GSPNs, the starting point is not the modeled behavior, but the actual behavior recorded in the systems. The adoption of process mining has been steadily growing across various industries as organizations realize that there is often a mismatch between the actual process and the assumed or desired process. It is trivial to construct a so-called Directly-Follows Graph (DFG) by simply counting how often one activity is followed by another activity. The real challenge is to discover concurrency, long-term dependencies, and other advanced control-flow constructs. During the last two decades, many process discovery approaches have been developed doing precisely that [1,10,11,29,46]. Most of them discover models that can be mapped onto Petri nets. There have been many attempts to extend the discovered models with time and probabilities. Most approaches create a simulation model [42,41]. Also, conformance-checking techniques have been adapted to include stochastics [26,30], time [41], data [35,34] and history [31].

However, we are not aware of approaches to systematically discover GSPNs that can be analyzed using Continuous-Time Markov Chains (CTMCs) and related techniques instead of simulation. These are vital to allow for forward-looking forms of process mining.

In this paper, we establish a link between GSPNs and probabilistic parameter synthesis. We translate the problem of finding a GSPN from an event log into two main problems: (1) discovering the control flow, including silent transitions, inhibitor arcs, and priorities, and (2) learning probabilities and firing delays. Existing control-flow discovery techniques partially solve the first problem. Mature techniques exist to discover labeled Petri nets with silent transitions. However, inhibitor arcs and priorities are rarely discovered. This paper addresses the second problem by translating GSPNs to Markov chains, to which parameter synthesis is applied to find transition weights for immediate transitions and transition rates for timed transitions. Note that we picked GSPNs as our target representation due to the availability of tools and techniques to conduct performance analysis.

In the remainder, we first introduce preliminaries such as event data and Petri nets (Section 2). Section 3 introduces Generalized Stochastic Petri Nets (GSPNs) tailored towards process mining and making the connection to CTMCs. Section 4 introduces a two-step approach where the first step discovers the process structure using existing approaches, and the second step discovers weights and rates to determine the stochastic and temporal behavior. Section 5 concludes the paper.

2 Event Data and Petri Nets

In this section, we introduce some preliminaries assuming that the reader has a basic understanding of process mining and Petri nets. We refer to [1,2] for more extensive introductions.

Events may have many attributes and refer to multiple objects of different types. However, here we assume that each *event* refers to just a *case*, an *activity*, and has a *timestamp*. These are the minimal requirements for process mining. By ordering the events using the timestamp per case and considering only the activity label, we can simplify this into a *multiset of traces* where each trace is simply a sequence of activities.

$\mathcal{B}(A)$ is the set of all *multisets* over some set A. For multiset $B \in \mathcal{B}(A)$, $B(a)$ denotes the number of times element $a \in A$ appears in B. Some examples of multisets over $A = \{x, y, z\}$: $B_1 = [\,]$, $B_2 = [x, x, y]$, $B_3 = [x, y, z]$, $B_4 = [x, x, y, x, y, z]$, and $B_5 = [x^3, y^2, z]$. Note that $B_4 = B_5$. The standard set operators can be extended to multisets, e.g., $x \in B_2$, $B_2 \uplus B_3 = B_4$, $B_5 \setminus B_2 = B_3$, $|B_5| = 6$, etc. Also, sets can be interpreted as multisets.

Definition 1 (Event Log). *$\mathcal{U}_{act}$ is the universe of activity names. An event log $L \in \mathcal{B}({\mathcal{U}_{act}}^*)$ is a multiset of traces. A trace $\sigma = \langle a_1, a_2, \ldots a_n \rangle \in {\mathcal{U}_{act}}^*$ is a sequence of activities. $L(\sigma)$ is the number of times trace σ appears in event log L.*

For example, $L = [\langle a, b, c, e \rangle^5, \langle a, c, b, e \rangle^5, \langle a, b, c, d, c, b, e \rangle^2]$ is an event log describing 12 cases and $5 \times 4 + 5 \times 4 + 2 \times 7 = 54$ events.

Definition 2 (Labeled Petri Net). *A labeled Petri net with priorities and inhibitor arcs is a tuple $N = (P, T, F, H, pr, lb)$ with P the set of places, T the set of transitions, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ the flow relation, $H \subseteq (P \times T)$ the inhibitor arcs, $pr \in T \to \mathbb{N}$ the priorities, and $lb \in T \not\to \mathcal{U}_{act}$ a labeling function. We write $lb(t) = \tau$ if $t \in T \setminus dom(lb)$ (i.e., t is a silent transition that cannot be observed).*

We refer to a labeled Petri net with priorities and inhibitor arcs as simply a Petri net. Note that we do not use arc weights because, in our setting, a Petri net describes the life-cycle of a single case. Technically, we can add arc weights without problems, but it complicates explanations and is rarely used in the context of process mining, workflow/process modeling, and business process management. Figure 1 shows an example of a Petri net.

$$\begin{aligned}
AN &= (N, M_{init}, M_{\mathit{final}})\\
M_{init} &= [p_0]\\
M_{\mathit{final}} &= [p_2]\\
N &= (P, T, F, H, pr, lb)\\
P &= \{p_0, p_1, p_2\}\\
T &= \{t_1, a_2, a_3, t_4, b_5\}\\
F &= \{(p_0, a_3), (p_0, t_4), (a_3, p_1),\\
&\qquad (t_4, p_1), (t_1, p_0), (p_1, t_1),\\
&\qquad (a_2, p_0), (p_1, a_2), (p_1, b_5),\\
&\qquad (b_5, p_2)\}\\
H &= \emptyset\\
pr &= t_1 \to 1, a_2 \to 1, a_3 \to 1, t_4 \to 1, b_5 \to 1\\
lb &= t_1 \to \tau, a_2 \to a, a_3 \to a, t_4 \to \tau, b_5 \to b
\end{aligned}$$

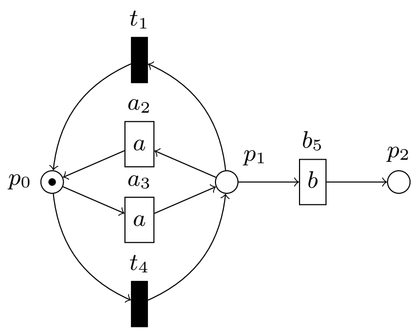

Fig. 1: A Petri net with a visual representation.

As usual, a marking $M \in \mathcal{B}(P)$ is represented as a multiset of places. The marking describes the state of the Petri net. Transitions have input and output places described by $F \subseteq (P \times T) \cup (T \times P)$. For any $x \in P \cup T$, $\bullet x = \{y \mid (y, x) \in F\}$ denotes the set of input nodes and $x\bullet = \{y \mid (x, y) \in F\}$ denotes the set of output nodes.

A transition $t \in T$ is *enabled* in marking M of net N, denoted as $(N, M)[t\rangle$ if and only if

- $\bullet t \sqsubseteq M$, i.e., each of its input places $\bullet t$ contains at least one token,
- $\{p \in M \mid (p, t) \in H\} = \emptyset$, i.e., none of the inhibitor places $\bullet t$ is enabled,
- there is no other transition t' that satisfies the two previous requirements and has a higher priority, i.e., $\forall_{t' \in T}\ (\bullet t' \sqsubseteq M\ \wedge\ \{p \in M \mid (p, t') \in H\} = \emptyset) \Rightarrow pr(t') \leq pr(t)$.

The set of *enabled transitions* in marking M is $en(N, M) = \{t \in T \mid (N, M)[t\rangle\}$. An enabled transition t may *fire*, i.e., one token is removed from each of the input places $\bullet t$ and one token is produced for each of the output places $t\bullet$. Formally: $M' = (M \setminus \bullet t) \uplus t\bullet$ is the marking resulting from firing enabled transition t in marking M of Petri net N. $(N, M)[t\rangle(N, M')$ denotes that t is enabled in M and firing t results in marking M'.

Let $\sigma = \langle t_1, t_2, \dots, t_n\rangle \in T^*$ be a sequence of transitions. $(N, M)[\sigma\rangle(N, M')$ denotes that there is a set of markings $M_1, M_2, \dots, M_{n+1}$ ($n \geq 0$) such that $M_1 = M$, $M_{n+1} = M'$, and $(N, M_i)[t_i\rangle(N, M_{i+1})$ for $1 \leq i \leq n$. A marking M' is *reachable* from M if there exists a *firing sequence* σ such that $(N, M)[\sigma\rangle(N, M')$. $R(N, M) = \{M' \in \mathcal{B}(P) \mid \exists_{\sigma \in T^*}\ (N, M)[\sigma\rangle(N, M')\}$ is the set of all reachable markings.

In process mining, we track cases or objects that have life-cycles with a clear start and end. Therefore, we focus on *accepting* Petri nets with an initial state and a final state.

Definition 3 (Accepting Petri Net). *An accepting Petri net is a triplet $AN = (N, M_{init}, M_{final})$ where $N = (P, T, F, H, pr, lb)$ 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. $\mathcal{U}_{AN} \subseteq \mathcal{U}_M$ is the set of accepting Petri nets.*

The behavior described by an accepting Petri net is the collection of all traces (i.e., firing sequences projected on the activity label). Therefore, we are interested in *sound* nets where it is always possible to reach the final marking.

Definition 4 (Sound Accepting Petri Net). *An accepting Petri net $AN = (N, M_{init}, M_{final})$ is sound if, from any reachable marking, it is possible to reach M_{final} and M_{final} is a dead marking, i.e., $\forall_{M \in R(N, M_{init})} \, M_{final} \in R(N, M) \wedge R(N, M_{final}) = \emptyset$.*

The behavior of a sound accepting Petri net can be represented as a *marking graph*, also referred to as its *reachability graph*. Later we will add time and probabilities to the marking graph.

Definition 5 (Marking Graph). *Let $AN = (N, M_{init}, M_{final})$ be a sound accepting Petri net. The marking graph of AN is $MG(AN) = (S, R, M_{init}, M_{final})$ with $S = R(N, M_{init})$ the set of states, $R = \{(M, (t, a), M') \in S \times T \times (\mathcal{U}_{act} \cup \{\tau\}) \times S \mid (N, M)[t\rangle(N, M') \wedge lb(t) = a\}$ the state transitions, M_{init} the initial state, and M_{final} the final state.*

Note that any sound accepting Petri net defines a marking graph and also a set of *possible traces*. For any $\sigma \in T^*$, we can apply the partial labeling function lb mapping the visible transitions (i.e., non-τ) onto the corresponding activity labels, i.e., $lb(\sigma) \in \mathcal{U}_{act}{}^*$. Hence, for any $AN = (N, M_{init}, M_{final})$, $trace(AN) = \{lb(\sigma) \mid \exists_\sigma (N, M)[\sigma\rangle(N, M')\}$ is the set of possible traces. Note that trace $\sigma = \langle a_1, a_2, \dots a_n \rangle \in trace(AN)$ if there a path from M_{init} to M_{final} in the marking graph considering only the labels of visible transitions.

Process mining techniques focus on the relation between event data and process models. *Process discovery* techniques aim to discover a process model (e.g., an accepting Petri net as defined in Def. 3) from example behavior (e.g., an event log as defined in Def. 1). The model should allow for as much of the observed behavior without underfitting the data (recall versus precision). *Conformance checking* takes as input a process model and event data and compares both to identify commonalities and discrepancies.

Note that an accepting Petri net and the corresponding marking graph do not specify probabilities and temporal behavior. GSPNs add these additional perspectives without restricting the set of possible traces, i.e., the marking graph already describes the structure of the corresponding Continuous-Time Markov Chain (CTMC).

3 Generalized Stochastic Petri Nets

After introducing the basics of event data, Petri nets, and process mining, we focus on *Generalized Stochastic Petri Nets* (GSPNs) that add time and probabilities. In a GSPN, we split the set of transitions into the set of immediate transitions T_i and the set of timed transitions T_t. The immediate transitions take no time and have priority over time transitions. If multiple immediate transitions are enabled with the same maximal priority, a weight function $wt \in T_i \to \mathbb{R}^+$ is used to resolve the conflict. Let, T_{max} be the non-empty set of enabled immediate transitions having the highest priority in marking M. The probability that $t \in T_{max}$ occurs in marking M is $\frac{wt(t)}{\sum_{t' \in T_{max}} wt(t')}$. Markings where there is at least one enabled immediate transition are called *vanishing*, because the time spent in such states is zero. If only timed transitions are enabled in M, function $rt \in T_t \to \mathbb{R}^+$ is used to resolve the choice. Such an M is called a *tangible* marking because a non-zero time is spent in such states. Function rt assigns a *transition rate* to each timed transition. In GSPNs, it is assumed that times are sampled from a negative exponential distribution described by the transition rate. (Note that phase-type distributions could be used as well, but here we stick to the basic variant.) Let T_{timed} be the non-empty set of enabled timed transitions in marking M (no immediate transitions are enabled). The probability that $t \in T_{timed}$ occurs first in marking M is $\frac{rt(t)}{\sum_{t' \in T_{timed}} rt(t')}$.

The memoryless property of the negative exponential probability distribution allows for different interpretations. For example, we can use the "race with age memory" interpretation where time transitions compete for firing and the competition is won by the transition that samples the shortest delay [17,6,23]. Recall that the probability density function of an exponential distribution with rate λ is $pdf(x) = \lambda e^{\lambda x}$ (with $x \geq 0$). The mean value is $\frac{1}{\lambda}$ and the variance is $\frac{1}{\lambda^2}$.

Definition 6 (Generalized Stochastic Petri Net). *An accepting generalized stochastic Petri net is a tuple $GSPN = (AN, T_i, T_t, wt, rt)$ where $AN = (N, M_{init}, M_{final})$ is a sound accepting Petri net with $N = (P, T, F, H, pr, lb)$, T_i is the set of immediate transitions (for any $t \in T_i$: $pr(t) > 0$ and $lb(t) = \tau$), T_t is the set of timed transitions (for any $t \in T_t$: $pr(t) = 0$), $wt \in T_i \to \mathbb{R}^+$ assigns a weight to each immediate transitions, and $rt \in T_t \to \mathbb{R}^+$ assigns a transition rate to each timed transition.*

Note that immediate transitions have priority over timed transitions because all timed transitions have a priority of zero and all immediate transitions have a non-zero priority. Therefore, there are two types of markings: *vanishing* (only immediate transitions are enabled) and *tangible* (only timed transitions are enabled). Note that the final marking is neither vanishing nor tangible because no transition is enabled.

Note that we assume that *only timed transitions can have a label*, i.e., we assume that immediate transitions are *not* logged in an event log and are only there for routing purposes. Traces in the event log correspond to accepting traces of the GSPN. This was the reason to focus on accepting Petri nets and abstract from arc weights. Note that most GSPN examples used in traditional literature are cyclic and cannot be linked to traces in event logs.

Interestingly, the addition of T_i, T_t, wt, and rt does *not* change the sets of accepting traces and reachable markings. They are only added to describe the *temporal* and *stochastic behavior*. Therefore, we can extend the marking graph with probabilities and firing rates without having to change the structure.

Definition 7 (Annotated Marking Graph). *Let $GSPN = (AN, T_i, T_t, wt, rt)$ be an accepting generalized stochastic Petri net with $AN = (N, M_{init}, M_{final})$ and $N =$*

(P, T, F, H, pr, lb). *The annotated marking graph of GSPN is* $MG(GSPN) = (S, R, M_{init}, M_{final}, S_v, S_t, rate, prob)$ *with*

- $MG(AN) = (S, R, M_{init}, M_{final})$ *the marking graph of* AN*,*
- $S_v = \{M \in S \mid \exists_{t \in T_i}\ (N, M)[t\rangle\}$ *the set of vanishing states,*
- $S_t = \{M \in S \mid \exists_{t \in T_t}\ (N, M)[t\rangle\}$ *the set of tangible states,*[1]
- $rate \in S_t \to \mathbb{R}^+$ *indicating the time spent in each tangible state with* $rate(M) = \sum_{t \in en(N,M)} rt(t)$ *for* $M \in S_t$*,*
- $prob \in R \to [0, 1]$ *indicating transition probabilities such that for any* $r = (M, (t, a), M') \in R$*:* $prob(r) = \frac{wt(t)}{\sum_{t' \in en(N,M)} wt(t')}$ *if* $M \in S_v$ *and* $prob(r) = \frac{rt(t)}{rate(M)}$ *if* $M \in S_t$*.*

The set of states is partitioned into S_v, S_t, and the final marking M_{final}. The time spent in vanishing states is zero. The time spent in a tangible state M is $\frac{1}{rate(M)}$ where $rate(M)$ is the sum of the rates of the enabled time transitions. (Recall that the minimum of a set of independent exponentially distributed random variables with rates $\lambda_1, \lambda_2, \ldots, \lambda_k$ is the exponentially distributed random variable with rate $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_k$.) Given a state $M \in S$, the probability of moving to M' via $r = (M, (t, a), M')$ is given by $prob(r)$.

The removal of vanishing markings yields a Continuous-Time Markov Chain (CTMC) that can be analyzed to answer a wide variety of performance questions (e.g., flow times). This is quite standard, and therefore, we do not elaborate on this [17,6,23].

4 Discovering Generalized Stochastic Petri Nets

After introducing GSPNs from a process-mining viewpoint, we now show how GSPNs can be discovered from event data.

4.1 Discovering Process Structure

Process-discovery techniques can be split into bottom-up approaches and top-down approaches. Bottom-up approaches like the Alpha algorithm [5], heuristic mining [45], and region-based approaches [4,12,18,33,44] try to find local connections between different activities. Top-down approaches include the family of inductive mining techniques [27,28,29] where recursively, the event log is split into smaller event logs using operators such as sequence, parallel, choice, and loop.

It is impossible to give a complete overview of all approaches. For example, the Split-Miner uses a combination of techniques [11]. See [10] for a survey.

Discovery techniques often use Petri nets as a target representation. Also, techniques that use other representations, such as process trees and causal nets, can be trivially mapped onto Petri nets. Therefore, we limit ourselves to the Petri-net representation.

In the first step, we discover a sound accepting Petri net as introduced in definitions 3 and 4. Note that such a model does *not* specify time and probabilities, these are discovered later. However, we allow for labeled transitions, inhibitor arcs, and priorities.

[1] Note that $S_v \cap S_t = \emptyset$, $M_{final} \notin S_v$, $M_{final} \notin S_t$, and $S = S_v \cup S_t \cup \{M_{final}\}$.

Classifying existing techniques using such Petri nets as a target notation, we identify the following dimensions:

- *Silent activities.* Is it possible to have a transition without a visible label (also called τ or invisible transitions)? For example, the Alpha algorithm requires all transitions to have a visible unique label.
- *Duplicate activities.* Is it possible to have multiple transitions having the same label? For example, state-based region approaches may use label-splitting.
- *Block-structuredness and other structural limitations.* Are there limitations regarding the Petri net structure? Many approaches produce only free-choice Petri net structures or impose a hierarchical structure. For example, all inductive approaches produce Petri nets that are block-structured and free choice.
- *Soundness.* Are models free of deadlocks and livelocks? Recall that there are different notions of soundness, e.g., relaxed soundness only requires that the final marking is reachable. In this paper, we assume that the accepting Petri net is sound as defined in Def. 4 (unless stated differently).

GSPNs allow for non-block-structured and non-free-choice models having both silent and duplicate activities. We limit ourselves to sound GSPNs. Interestingly, GSPNs allow for inhibitor arcs and priorities. These extensions significantly extend the expressive power of Petri nets. For example, many of the workflow patterns or business rules described in literature can only be expressed using inhibitor arcs or priorities. Inhibitor arcs and priorities can be added without jeopardizing Markovian-based performance analysis. Therefore, it is interesting to think of control-flow discovery techniques using inhibitor arcs or priorities.

There exist synthesis techniques for Petri nets with inhibitor arcs [16] or even the combination of reset and inhibitor arcs[20]. However, these do *not* start form event logs, but from labeled transition systems that are assumed to isomorphic to the reachability graph. Therefore, they cannot be used in any realistic process-mining setting. Through this paper we would like to encourage the development of such techniques. The complicating factor is that inhibitor arcs and priorities remove behavior and are therefore only observable through the absence of behavior in the event log.

It is important to note that all visible transitions corresponding to activities recorded in the event log correspond to timed transitions and can only happen in tangible markings. The time spent in a vanishing marking is zero. Therefore, it is natural to assume that "real-world events" occur in tangible markings. This matches the earlier requirement for GSPNs that for any $t \in T_i$: $pr(t) > 0$ and $lb(t) = \tau$ and for any $t \in T_t$: $pr(t) = 0$. Visible transitions need to be timed, but timed transitions do not need to be visible.

When mapping a GSPN onto Markov chains, these extensions do not cause any additional difficulty. Therefore, there are opportunities to enhance discovery techniques. Inhibitor arcs and priorities only restrict behavior. Therefore, they can be used to make discovered process models more precise. However, such extensions may also introduce deadlocks or livelocks.

In the remainder, we assume that we have discovered a sound accepting labeled Petri net with priorities and inhibitor arcs. Therefore, we just need to determine T_i, T_t, wt, and rt to obtain a complete GSPN.

$$\begin{tikzcd}
\text{model} \arrow[dr, bend right] & & & & \epsilon \arrow[d, bend left, "\text{synthesis}"] & \\
\log L \arrow[r, "\text{remove}", "\text{non-fitting traces}"'] & \log L' \arrow[r] & \text{Markov chain} \arrow[r] & \text{parameterised Markov chain} \arrow[r] & \text{query} \arrow[u, bend left, "\text{parameter}"] \arrow[r] & \text{GSPN}
\end{tikzcd}$$

Fig. 2: Overview of our approach to discover GSPNs from a log and a control-flow model.

4.2 Discovering Weights and Rates

There are currently no techniques that can, given an event log L and a sound accepting labeled Petri net with priorities and inhibitor arcs, obtain a complete GSPN. That is, assume $AN = (N, M_{init}, M_{final})$ and $N = (P, T, F, H, pr, lb)$ to be fixed (control-flow discovery). Now, what is left is to determine T_i, T_t, wt, and rt in $GSPN = (AN, T_i, T_t, wt, rt)$.

The problem of automatically adding probabilities and durations to a Petri net was first addressed in [42], where so-called Colored Petri Nets (CPNs) are discovered. Token-based replay was used to relate the event data to the control-flow model. In [41], a similar approach was followed but using alignments (to relate event data and model) and GSPNs as a target language. These classical approaches did not consider priorities or inhibitor arcs and used heuristics to estimate parameters.

More recent work aims to optimize for well-defined stochastic log-quality measures [15,32,8], e.g., estimating weights [13,30] or focusing on block-structured models [14]. However, these approaches only consider probabilities and do not consider time durations, priorities, or inhibitor arcs.

Other approaches do not consider concurrency, using sequential models such as stochastic directed action graphs [7].

Therefore, there is no approach supporting the discovery of transition weights and rates for arbitrary GSPNs using a well-defined quality measure. To this end, we translate the problem of finding weights and rates of GSPN given its control-flow structure to leverage a parameter synthesis technique by Joost-Pieter Katoen [19,39]. Figure 2 shows an overview, and here we provide the steps in detail:

1. AN defines a control-flow model that expresses a language of traces. The first step is to filter the event log L by removing all traces that cannot be replayed on AN (that is, that are not part of the language of AN). By definition, traces that do not fit AN have a probability of 0. Let L' be this filtered log.
2. Second, we consider that each trace σ in L' has a potentially infinite set of paths in AN that project to σ. For instance, consider the trace $\langle a, a, b\rangle$ and our Petri net in Figure 1. The trace is in the language of the net, however there are infinitely many paths that project to this trace, as the silent transitions t_1 and t_4 form a loop that can be taken arbitrarily often.
   To capture these potentially infinitely many paths, we create an absorbing Markov chain $(\mathcal{S}, \mathcal{T}, \mathcal{P})$. The states of the chain represent the combination of a marking and (a prefix of) a trace. The transitions in the Markov chain are transitions of the Petri

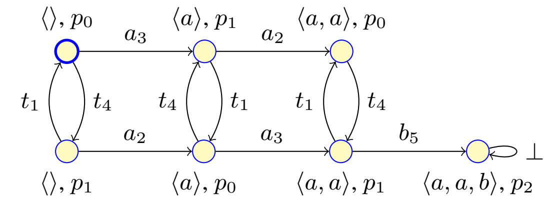

Fig. 3: Absorbing Markov chain of trace $\langle a, a, b\rangle$ and the Petri net of Figure 1.

net. Figure 3 shows the structure of the absorbing Markov chain of our example. Formally, the structure of the absorbing Markov chain is the combination of the smallest sets of states $\mathcal{S}$ and transitions $\mathcal{T}$ such that:

- The initial state is the combination of the initial marking and the empty trace:

$$(\langle\rangle, M_{init}) \in \mathcal{S} \tag{1}$$

- If in the net a silent transition is enabled, then that transition can be followed in $\mathcal{T}$ and $\mathcal{S}$:

$$\begin{aligned}(\sigma, M) \in \mathcal{S} \wedge (N, M)[t\rangle(N, M') \wedge lb(t) = \tau \\ \Rightarrow (\sigma, M) \xrightarrow{t} (\sigma, M') \in \mathcal{T} \wedge (\sigma, M') \in \mathcal{S}\end{aligned} \tag{2}$$

- If in the net a labeled transition is enabled, and there is a trace in L' that can follow that transition's label, then that transition can be followed in $\mathcal{T}$ and $\mathcal{S}$:

$$\begin{aligned}(\sigma, M) \in \mathcal{S} \wedge (N, M)[t\rangle(N, M') \wedge \sigma \cdot \langle lb(t), \ldots\rangle \in L' \\ \Rightarrow (\sigma, M) \xrightarrow{t} (\sigma \cdot \langle lb(t)\rangle, M') \in \mathcal{T} \wedge (\sigma \cdot \langle lb(t)\rangle, M') \in \mathcal{S}\end{aligned} \tag{3}$$

- As in a sound Petri net the final marking is a deadlock, every corresponding state is an absorbing state:

$$(\sigma, M_{final}) \in \mathcal{S} \Rightarrow (\sigma, M_{final}) \xrightarrow{\perp} (\sigma, M_{final}) \in \mathcal{T} \tag{4}$$

Figure 3 shows an example of our example Petri net and log $[\langle a, b\rangle, \langle a, a, b\rangle]$. Please note that through Equation (4), each trace of the log has a unique absorbing state. We refer to this state as $\mathcal{S}_{absorb}(\sigma)$.

3. Next, we assign the probabilities in the Markov chain in the function $\mathcal{P}$, using parameters for weights wt and preliminary rates rt'.
For every immediate transition t such that $(\sigma, M) \xrightarrow{t} (\sigma', M') \in \mathcal{T}$ (added through Equation (2)), we assign a probability that is relative to the weight of t compared to the sum of the weights of the enabled transitions in M:

$$\mathcal{P}((\sigma, M) \xrightarrow{t} (\sigma', M')) = \frac{wt(t)}{\sum_{t' \in en(N,M)} wt(t')} \tag{5}$$

For every timed transition t such that $(\sigma, M) \xrightarrow{t} (\sigma', M') \in \mathcal{T}$ (added through equations (2) and (3)), we assign a probability that is relative to the rate of its transition compared to the sum of the rates of the enabled transitions in M:

$$\mathcal{P}((\sigma, M) \xrightarrow{t} (\sigma', M')) = \frac{rt'(t)}{\sum_{t' \in en(N,M)} rt'(t')} \tag{6}$$

Every transition added through Equation (4) is assigned a probability of 1:

$$\mathcal{P}((\sigma, M_{\mathit{final}}) \xrightarrow{\perp} (\sigma, M_{\mathit{final}})) = 1 \tag{7}$$

By construction, for each state, the sum of probabilities on outgoing transitions is 1. As the Petri net is sound, it has no livelocks, and therefore the Markov chain is absorbing.

4. Observe that we now have a parameterized absorbing Markov chain that represents the traces of the log, the control-flow structure of the model, and the – parameterized – weights and rates of the transitions in the model. Next, we formulate a probabilistic query stating that for a given small ϵ, we can assign weights such that for each trace in the log, the absolute difference the probability of that trace in the log and the probability of that trace in the model is at most ϵ. That is, for a given small ϵ, whether there is an assignment to the weight and rate parameters $wt(t)$ and $rt'(t)$ such that for all traces σ, the probability that the Markov chain ends up in $\mathcal{S}_{absorb}(\sigma)$ is ϵ-close to the probability observed in the event log L':

$$\forall_{\sigma \in L'} \left| P(\mathcal{S}_{absorb}(\sigma)) - \frac{|[\sigma \mid \sigma \in L']|}{|L'|} \right| \leq \epsilon \tag{8}$$

5. We can answer this query for a given ϵ using parameter synthesis [19,39]. That is, we look whether there is a region of weight and rate parameter assignments for which the query returns true. Next, we perform a binary search to find the lowest ϵ for which the query holds. Once we are happy with the reached precision, we can pick an arbitrary point from the feasible region, which directly yields the weight parameters wt for the silent transitions.
6. For the timed transitions, the feasible region provides only preliminary values, which we refer to as rt'. This stems from the fact that the rates of dependent transitions can be scaled together by an arbitrary factor without changing the stochastic behavior of the GSPN. As such, the rates of (Petri net) transitions can be scaled to yield a different time perspective without changing the stochastic perspective. Two (Petri net) transitions are dependent if they are both enabled in at least one state in $\mathcal{S}$ [31]. Formally, a dependent set of transitions T' is a smallest set containing at least one transition such that:

$$\forall_{t \in T, t' \in T'} \Big((\sigma, M) \xrightarrow{t'} (\sigma', M')\Big) \in \mathcal{T} \wedge \Big((\sigma, M) \xrightarrow{t} (\sigma'', M'')\Big) \in \mathcal{T} \Rightarrow t \in T' \tag{9}$$

Dependence is a transitive property, and thus every transition is part of exactly one such set. Our remaining problem is thus to scale the rates of each set of dependent

transitions by a factor, such that the rates best match the times observed in the event log.
Let $T' \subseteq T$ be a set of dependent (Petri net) transitions, such that there is no dependent set $T'' \supset T'$. Let $\mathcal{M}$ be the set of all markings in which any transition of T' was enabled. By definition, no transitions outside of T' can be enabled in any $M \in \mathcal{M}$. Then, we can easily observe in the event log L' how long the system spent on average in each marking $M \in \mathcal{M}$; let $o(M)$ be this average.
Then, we aim to find a scaling factor $x_{T'}$ such that

$$\begin{aligned} \text{let } \forall_{M \in \mathcal{M}} \frac{1}{o(M)} \cdot \frac{1}{x_{T'}} + \epsilon_M &= rt' \\ \text{find } x_{T'} & \\ \text{while minimizing } \sum_{t \in T'} |\epsilon_M| & \end{aligned} \tag{10}$$

This is a linear problem, as $|a| \leq b$ can be written as $a \leq b \wedge -a \leq b$. Solving this linear problem yields a value for $x_{T'}$ for every maximal independent set of transitions $T' \subseteq T$. The preliminary rates rt' can then be scaled: for each t from each such T', $rt(t) = rt'(t) \cdot x_{T'}$.

5 Conclusion

Joost-Pieter Katoen is one of the most renowned scientists in formal methods, with seminal contributions to model checking, concurrency theory, and probabilistic reasoning. He has worked on the analysis of Generalized Stochastic Petri Nets (GSPNs) and parameter synthesis. Therefore, we related GSPNs and parameter synthesis to process discovery based on event data. We introduced (GSPNs) from a process-mining perspective and presented a two-step approach to discover GSPNs from event data. We showed that it is possible to optimize the transition weights and rates constrained by a fixed control-flow structure discovered in the first phase. Obviously, the approach has several limitations. The most important one is that time distributions are assumed to be negative exponential. Future work aims to implement and evaluate the approach. We would also like to experiment with extensions of GSPN using phase-type distributions [6], stochastic process trees using discrete time [3], and various approximations. Moreover, we would like to explore new discovery techniques that allow for inhibitor arcs and priorities. These extensions only limit the behavior of the process model and can therefore only be discovered through the absence of behavior in the event log. This provides opportunities to discover more precise models while still leveraging the GSPN-based performance analysis techniques.

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 and J. Carmona, editors. *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*. Springer-Verlag, Berlin, 2022.
3. W.M.P. van der Aalst, K.M. van Hee, and H.A. Reijers. Analysis of Discrete-time Stochastic Petri Nets. *Statistica Neerlandica*, 54(2):237–255, 2000.
4. 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.
5. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. QUT Technical report, FIT-TR-2003-03, Queensland University of Technology, Brisbane, 2003. (Accepted for publication in IEEE Transactions on Knowledge and Data Engineering.).
6. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.
7. H. Alkhammash, A.Polyvyanyy, and A. Moffat. Stochastic directly-follows process discovery using grammatical inference. In *CAiSE*, volume 14663 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2024.
8. H. Alkhammash, A. Polyvyanyy, A. Moffat, and L. García-Bañuelos. Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. *Inf. Syst.*, 107:101922, 2022.
9. E.G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis. 30 Years of GreatSPN. In L. Fiondella and A. Puliafito, editors, *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*, pages 27–254. Springer-Verlag, Berlin, 2016.
10. A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F.M. Maggi, A. Marrella, M. Mecella, and A. Soo. Automated Discovery of Process Models from Event Logs: Review and Benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):686–705, 2019.
11. A. Augusto, R. Conforti, M. Marlon, M. La Rosa, and A. Polyvyanyy. Split Miner: Automated Discovery of Accurate and Simple Business Process Models from Event Logs. *Knowledge Information Systems*, 59(2):251–284, 2019.
12. 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.
13. A. Burke, S.J.J. Leemans, and M.T. Wynn. Stochastic process discovery by weight estimation. In *ICPM Workshops*, volume 406 of *Lecture Notes in Business Information Processing*, pages 260–272. Springer, 2020.
14. A. Burke, S.J.J. Leemans, and M.T. Wynn. Discovering stochastic process models by reduction and abstraction. In *Petri Nets*, volume 12734 of *Lecture Notes in Computer Science*, pages 312–336. Springer, 2021.
15. A.T. Burke, S.J.J. Leemans, M.T. Wynn, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Stochastic Process Model-Log Quality Dimensions: An Experimental Study. In A. Burattin, A. Polyvyanyy, and B. Weber, editors, *International Conference on Process Mining (ICPM 2022)*, pages 80–87. IEEE, 2022.
16. N. Busi and G.M. Pinna. Synthesis of nets with inhibitor arcs. In A.W. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*, volume 1243 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 1997.
17. J. Campos, M.A. Marsan, G. Balbo, and G. Conte. Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications. *IEEE Transactions on Software Engineering*, 19(2):89–107, 1993.

18. J. Carmona, J. Cortadella, and M. Kishinevsky. A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In *Business Process Management (BPM 2008)*, pages 358–373, 2008.
19. C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, J.P. Katoen, E. Ábrahám, and H. Bruintjes. Parameter synthesis for probabilistic systems. In *MBMV*, pages 72–74. Albert-Ludwigs-Universität Freiburg, 2016.
20. R.R. Devillers and R. Tredup. Synthesis of Inhibitor-Reset Petri Nets: Algorithmic and Complexity Issues. In L. Bernardinello and L. Petrucci, editors, *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*, volume 13288 of *Lecture Notes in Computer Science*, pages 213–235. Springer, 2022.
21. C. Eisentraut, H. Hermanns, J.P. Katoen, and L. Zhang. A Semantics for Every GSPN. In J.M. Colom and J. Desel, editors, *Application and Theory of Petri Nets and Concurrency - 34th International Conference (Petri Nets 2013)*, volume 7927 of *Lecture Notes in Computer Science*, pages 90–109. Springer-Verlag, Berlin, 2013.
22. G. Florin and S. Natkin. Evaluation based upon Stochastic Petri Nets of the Maximum Throughput of a Full Duplex Protocol. In C. Girault and W. Reisig, editors, *Application and theory of Petri nets: selected papers from the first and the second European workshop*, volume 52 of *Informatik Fachberichte*, pages 280–288, Berlin, 1982. Springer-Verlag, Berlin.
23. P.J. Haas. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer Series in Operations Research. Springer-Verlag, Berlin, 2002.
24. S. Junges, J.P. Katoen, M. Stoelinga, and M. Volk. One Net Fits All - A Unifying Semantics of Dynamic Fault Trees Using GSPNs. In V. Khomenko and O.H. Roux, editors, *Application and Theory of Petri Nets and Concurrency (Petri Nets 2018)*, volume 10877 of *Lecture Notes in Computer Science*, pages 272–293. Springer-Verlag, Berlin, 2018.
25. J.P. Katoen. GSPNs Revisited: Simple Semantics and New Analysis Algorithms. In J. Brandt and K. Heljanko, editors, *12th International Conference on Application of Concurrency to System Design (ACSD 2012)*, pages 6–11. IEEE Computer Society, 2012.
26. S.J.J. Leemans, W.M.P. van der Aalst, T. Brockhoff, and A. Polyvyanyy. Stochastic Process Mining: Earth Movers' Stochastic Conformance. *Information Systems*, 102:101724, 2021.
27. 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.
28. 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.
29. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, 17(2):599–631, 2018.
30. S.J.J. Leemans, T. Li, M. Montali, and A. Polyvyanyy. Stochastic process discovery: Can it be done optimally? In *CAiSE*, volume 14663 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2024.
31. S.J.J. Leemans, L.L. Mannel, and N. Sidorova. Significant stochastic dependencies in process models. *Information Systems*, 118:102223, 2023.
32. S.J.J. Leemans, A.F. Syring, and W.M.P. van der Aalst. Earth movers' stochastic conformance checking. In *BPM Forum*, volume 360 of *Lecture Notes in Business Information Processing*, pages 127–143. Springer, 2019.

33. L. Mannel and W.M.P. van der Aalst. Finding Complex Process-Structures by Exploiting the Token-Game. In S. Donatelli and S. Haar, editors, *Applications and Theory of Petri Nets 2019*, volume 11522 of *Lecture Notes in Computer Science*, pages 258–278. Springer-Verlag, Berlin, 2019.
34. F. Mannhardt, S.J.J. Leemans, C.T. Schwanen, and M. de Leoni. Modelling data-aware stochastic processes - discovery and conformance checking. In L. Gomes and R. Lorenz, editors, *Application and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings*, volume 13929 of *Lecture Notes in Computer Science*, pages 77–98. Springer, 2023.
35. F. Mannhardt, M. de Leoni, H.A. Reijers, and W.M.P. van der Aalst. Balanced Multi-Perspective Checking of Process Conformance. *Computing*, 98(4):407–437, 2016.
36. M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalised Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.
37. P. Merlin and D.J. Faber. Recoverability of communication protocols. *IEEE Transactions on Communication*, 24(9):1036–1043, Sept 1976.
38. M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. PhD thesis, University of California, Los Angeles, 1981.
39. T. Quatmann, C. Dehnert, N. Jansen, S. Junges, and J.P. Katoen. Parameter synthesis for markov models: Faster than ever. In *ATVA*, volume 9938 of *Lecture Notes in Computer Science*, pages 50–67, 2016.
40. C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 1973.
41. A. Rogge-Solti, W.M.P. van der Aalst, and M. Weske. Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs. 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 15–27. Springer-Verlag, Berlin, 2014.
42. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Simulation Models. *Information Systems*, 34(3):305–327, 2009.
43. B. Salmani and J.P. Katoen. Automatically Finding the Right Probabilities in Bayesian Networks. *Journal of Artificial Intelligence Research*, 77:1637–1696, 2023.
44. M. Solé and J. Carmona. Process Mining from a Basis of State 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.
45. 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.
46. 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.