Process Science

RESEARCH Open Access

Abstract-and-compare stochastic conformance checking

Eduardo Goulart Rocha[1,2*], Sander J. J. Leemans[2,3] and Wil M. P. van der Aalst[1,2]

*Correspondence:
e.goulartrocha@celonis.com

[1] Celonis Labs GmbH, Munich, Germany
[2] RWTH Aachen University, Aachen, Germany
[3] Fraunhofer FIT, Aachen, Germany

Abstract

Conformance checking is a key area of process mining focused on identifying and quantifying deviations between observed and modeled behavior. Traditional techniques only consider the frequency information in event logs, leading to one-sided analyses. In recent years, stochastic conformance checking, which considers the stochastic perspective of process models, has gained attention. However, state-of-the-art stochastic conformance checking techniques are often either not robust to partial mismatches or cannot be efficiently computed for broad classes of process models or large input sets, limiting their practical applicability. To address these challenges, we propose an abstract-and-compare approach, where stochastic languages are first abstracted and then compared using existing stochastic conformance checking techniques. Specifically, we introduce the stochastic Markovian abstraction, based on expected subtrace frequencies, and demonstrate how to compute it for bounded, livelock-free stochastic labeled Petri nets- a widely used stochastic process model formalism. The Markovian abstraction defines a stochastic language and can be combined with state-of-the-art stochastic conformance measures to obtain "new" derived measures. We evaluate two such derived measures on synthetic and real-world datasets, showing that the abstraction can effectively approximate model behavior, is computationally efficient, and improves robustness to partial mismatches. Additionally, we demonstrate how to obtain stochastic conformance diagnostics for the derived measures.

Keywords: Conformance checking, Stochastic, Subtraces, N-grams, Markovian

Introduction

Conformance checking is the field of process mining concerned with quantifying and diagnosing deviations between the as-is behavior of a process, usually captured as an event log, and its ideal should-be behavior, expressed as a process model. Traditional conformance checking techniques only consider the control-flow perspective of process models, leading to one-sided analysis where the frequency information in event logs cannot be properly compared. Consider the example from Fig. 1, which presents two event logs and an idealized process model relating to a purchase process. The model mandates that one or more approvals must happen at the beginning, after which point the payment is booked and the goods are collected.

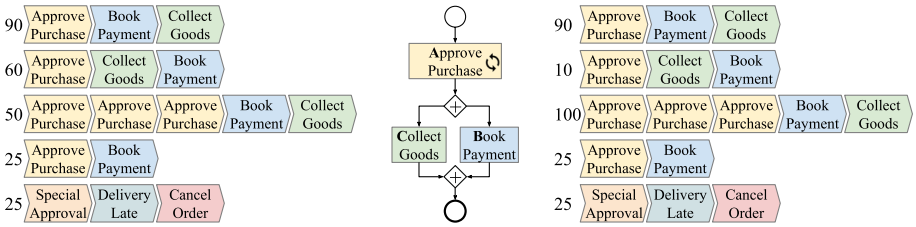


Fig. 1 Two event logs related to a purchase process and a reference process model

In Fig. 1, both logs contain the same variants and the same alignment-based fitness and precision. Therefore, traditional conformance checking techniques treat both logs as equally conforming. However, from a practical standpoint, the behaviors they represent are not equally desirable. Repeated approvals introduce delays and require extra effort and should be minimized. Furthermore, depending on business rules, it might be preferred to issue a payment only after the goods were received.

In recent years, the field of stochastic process mining has gained relevance. In comparison with traditional conformance checking, stochastic conformance checking considers the model's stochastic behavior. Therefore, one can compare not only whether a process follows its prescribed control flow, but also whether its stochastic behavior is as expected. Current state-of-the-art stochastic conformance checking approaches face several challenges, often exhibiting one or more of the following limitations:

1. They are not robust to partial mismatches in the log and model's languages (Li et al. 2024; Leemans and Polyvyanyy 2023; Alkhammash et al. 2022; Leemans et al. 2024b)
2. They can only be computed for a limited class of process models (Leemans and Polyvyanyy 2023; Bergami et al. 2021)
3. They cannot be computed analytically and/or rely on model sampling (Leemans et al. 2021; Bergami et al. 2021)

This happens because these are three competing requirements. To appropriately consider partial mismatches, one must consider the model's full language, which is not feasible for complex models, such as non-deterministic models and models containing parallelism or loops; therefore, one is forced to either restrict the class of process models that can be analyzed, or sample behavior from the model.

However, sacrificing any of these three aspects is not practical. In real-world scenarios, traces that are not fully conforming oftentimes still present a high degree of overlap with the model behavior, e.g. we should impose a less severe penalty for variant $\langle$*Approve Purchase*, *Book Payment*$\rangle$, which only has one missing event, than for $\langle$*Special Approval*, *Delivery Late*, *Cancel Order*$\rangle$, which only contains unmapped events. Furthermore, the class of process models that are "tractable", e.g. acyclic or deterministic models, is not expressive enough to model complex process behavior. At the same time, sampling techniques do not scale for large models, i.e. it is not always possible to sample a representative number of traces from the model.

In this paper, we propose using a suitable abstraction of log and model behavior that helps mitigate the aforementioned issues. The abstraction defines a stochastic language and can be combined with any existing stochastic conformance checking technique to

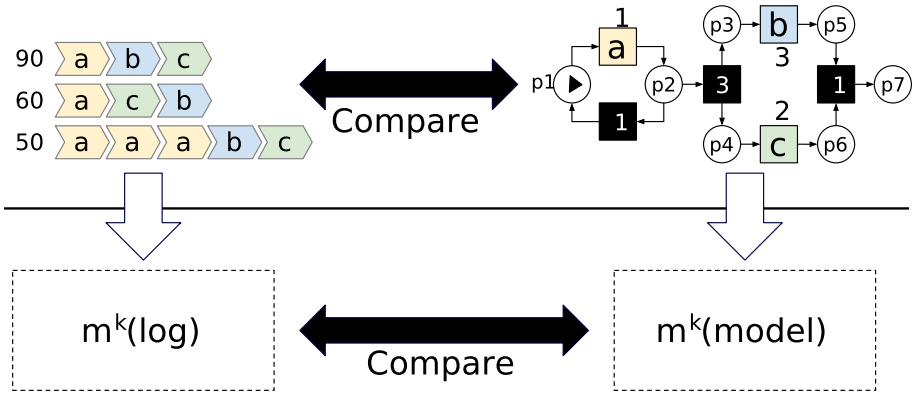


Fig. 2 Overview of the proposed abstract-and-compare framework applied to the example event log L_0 and stochastic labeled Petri net SN_0

obtain "new" derived conformance measures. Figure 2 illustrates the approach. Instead of directly comparing two objects, we compare their language abstractions.

The proposed abstraction essentially encodes the expected subtrace frequency of each language. We show that the abstraction helps mitigate robustness issues caused by partial mismatches, and that it can be efficiently computed in an exact manner for bounded, livelock-free stochastic Petri nets, a widely adopted class of stochastic process models in process mining.

This is a revised and extended version of a conference paper (Rocha et al. 2024). Compared to Rocha et al. (2024), we:

1. Included an evaluation of the expressive power of the abstraction
2. Extended the evaluation with additional derived conformance measures
3. Discussed conformance diagnostics methods for the derived measures

The remainder of this paper is structured as follows: "Preliminaries" section establishes notation and introduces key concepts in stochastic conformance checking. "The stochastic markovian abstraction" section presents the stochastic Markovian abstraction and its properties, while "Derived stochastic measures" section introduces two stochastic conformance measures derived from it (the m^k measures). In "Quantitative evaluation" section, we evaluate the abstraction and its derived m^k measures, and in "Stochastic conformance diagnostics" section, we demonstrate how to generate conformance diagnostics based on the derived m^k measures. Related work is discussed in "Related work" section. Finally, "Conclusions, discussion and future work" section concludes the paper and outlines directions for future research.

Preliminaries

This section fixes the notation and introduces the basic definitions used throughout the rest of this paper. $\mathbb{N}_0$ and $\mathbb{R}_{\geq 0}$ denote the set of natural numbers including 0 and the non-negative real numbers. Given a set S, the indicator function $\mathbf{1}_S(s)$ returns 1 if $s \in S$, and 0 otherwise. Let Σ be an alphabet of activities, we define $\tau \notin \Sigma$ as the *invisible label* and define $\Sigma_\tau = \Sigma \cup \{\tau\}$. $\Sigma^{\leq k}$ denotes the set of all traces over Σ of

length less than or equal to k, and Σ^* denotes the set of all finite traces over Σ. Given a trace $\sigma = \langle a_1, a_2, \cdots a_n \rangle \in \Sigma^*$, $\sigma[i] = a_i$ denotes the i-th entry in the trace and $\sigma^{i \to j} = \langle a_i, a_{i+1}, \cdots a_j \rangle, 1 \leq i \leq j \leq n$ denotes the subtrace of σ of length $j - i + 1$ starting at index i.

A *multiset* B over a set S is a function $B : S \to \mathbb{N}_0$ where $B(s)$ returns the frequency of s in B for each $s \in S$. Multisets are written using bracket notation $B = [s_1^{B(s_1)}, s_2^{B(s_2)}, \cdots]$. We omit the superscript if $B(s_i) = 1$ and we do not write s_i if $B(s_i) = 0$, e.g. for $S = \{s_1, s_2, s_3\}, [s_1^2, s_2]$ is the multiset containing 2 copies of s_1, one copy of s_2, and no copy of s_3. A multiset's *support* defined as $supp(B) = \{s \in S \mid B(s) > 0\}$ is the set of entries in B occurring at least once. A multiset's cardinality $|B|$ is defined as $|B| = \sum_{s \in S} B(s)$. Furthermore, $\mathcal{B}(S)$ denotes the set of all multisets over S. We define the union and difference multisets as $(A \uplus B)(s) = A(s) + B(s)$ and $(A \setminus B)(s) = A(s) - B(s)$ (the latter is only defined if $A(b) \geq B(b)$ for every $b \in supp(B)$). Last, we abuse notation and write $x < \infty$ and $x = \infty$ to denote that an expression/function x is bounded/defined, respectively unbounded/undefined. Stochastic languages are the central concept in this work:

Definition 1 (Stochastic Language) Given an alphabet Σ, a stochastic language l is a function $l : \Sigma^* \to [0, 1]$ such that $\sum_{\sigma \in \Sigma^*} l(\sigma) = 1$, i.e. l defines a probability distribution over Σ^*.

We use the same notation as multisets for stochastic languages, e.g. we represent the stochastic language $l : \Sigma^* \to [0, 1]$ as $l = [\sigma_1^{l(\sigma_1)}, \sigma_2^{l(\sigma_2)}, \cdots]$, $\sigma_i \in \Sigma^*$. An event log is a collection of events grouped into traces describing the system's observed behavior.

Definition 2 (Event Log) Let Σ be a set of activity labels, an *event log* is a multiset of traces $L \in \mathcal{B}(\Sigma^*)$.

Figure 2 shows the example event log $L_0 = [\langle a, b, c \rangle^{90}, \langle a, c, b \rangle^{60}, \langle a, a, a, b, c \rangle^{50}]$. An event log L implicitly defines a stochastic language l_L as $l_L(\sigma) = \frac{L(\sigma)}{|L|}$, e.g. $l_{L_0} = [\langle a, b, c \rangle^{0.45}, \langle a, c, b \rangle^{0.3}, \langle a, a, a, b, c \rangle^{0.25}]$. In contrast to event logs, process models often include an infeasibly large, or even infinite number of traces due to parallelism and loops. Therefore, process models are preferably expressed using compact representations such as stochastic labeled Petri nets (SLPN):

Definition 3 (Stochastic Labeled Petri Nets (SLPN)) A stochastic labeled Petri net (SLPN) is a septuple $SN = (P, T, F, \Sigma, \lambda, M_0, w)$ where P is the set of places, T is the set of transitions s.t. $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, Σ is the set of activity labels s.t. $\tau \notin \Sigma$, $\lambda : T \to \Sigma_\tau$ is a transition labeling function, $M_0 \in \mathcal{B}(P)$ is the initial marking, and $w : T \to \mathbb{R}^+$ is a weight function.

SLPNs extend labeled Petri nets with the stochastic perspective using transition weights.[1] States in a SLPN are represented as multisets of places $M \subseteq \mathcal{B}(P)$ called *markings*. For each transition $t \in T$, we define $\bullet t = \{p \in P \mid (p, t) \in F\}$ and

[1] Our definition differs from the reference in Marsan et al. (1995) by adding transition labels and ignoring time.

$t\bullet = \{p \in P \mid (t,p) \in F\}$ as the transition's pre-/post sets. A transition $t \in T$ is *enabled* at marking M (written $M[t\rangle$) if $\bullet t \leq M$. We write $enabled_{SN}(M) = \{t \in T \mid M[t\rangle\}$ as the set of enabled transitions in M. A marking is a *deadlock marking* if $enabled_{SN}(M) = \emptyset$, i.e. no transition is enabled at M. Firing a transition $t \in enabled_{SN}(M)$ changes the state of the Petri net from M to $M' = (M \setminus \bullet t) \uplus t\bullet$ (we write $M[t\rangle M'$).

A sequence of transitions $\eta = \langle t_1, t_2, \cdots t_n\rangle \in T^*$ is enabled in marking M (written $M[\eta\rangle$) if there exists $M_1, M_2, \cdots M_n \in \mathcal{B}(P)$ such that $M[t_1\rangle M_1, M_1[t_2\rangle M_2, \cdots, M_{n-1}[t_n\rangle M_n$ (compactly, $M[\eta\rangle M_n$). If $M = M_0$, then η is a *run*. A marking M' is *reachable* from M if there exists a run $\eta \in T^*$ such that $M[\eta\rangle M'$. We define $\mathcal{R}(SN, M) = \{M' \in \mathcal{B}(P) \mid \exists \eta \in T^*,\ M[\eta\rangle M']\}$ as the set of markings reachable from M. For this work, we consider all deadlock markings as the net's accepting states. If M_n is an accepting state, then η is an *accepting run*. A *livelock* is a marking from which no accepting state can be reached, i.e. there are no deadlock states in $\mathcal{R}(SN, M)$. A SLPN is *livelock-free* if $\mathcal{R}(SN, M_0)$ contains no livelock and it is *bounded* if it has finitely many reachable states from M_0, i.e. $|\mathcal{R}(SN, M_0)| < \infty$.

In a SLPN, a transition $t \in enabled_{SN}(M)$ fires with probability $p_{SN}(M,t) = w(t) / \sum_{t' \in enabled_{SN}(M)} w(t')$, i.e. transitions in a marking fire with a probability based on their relative weight to the total weight of all transitions enabled in that marking. The probability of an accepting run η is defined as $p_{SN}(\eta) = \prod_{0 \leq i < n} p_{SN}(M_i, t_{i+1})$, i.e. the product of its transition probabilities at each intermediate marking. A bounded livelock-free SLPN SN generates a stochastic language l_{SN} defined as $l_{SN}(\sigma) = \sum_{\eta \in T^*, \lambda(\eta) = \sigma} p_{SN}(\eta)$. SLPNs have a close link to Stochastic Non-Deterministic Finite Automata (SNFA), defined below:

Definition 4 (Stochastic Non-Deterministic Finite Automaton (SNFA)) A stochastic non-deterministic finite automaton (SNFA) is a tuple $(Q, \Sigma, \delta, q_0, p_f)$ where Q is a set of states, Σ is an alphabet s.t. $\tau \notin \Sigma$, $\delta : Q \times \Sigma_\tau \times Q \to [0,1]$ is the transition probability function, q_0 is the initial state and $p_f : Q \to [0,1]$ is the function defining the final probability of each state, where $\forall q \in Q : \sum_{(a,q') \in (\Sigma_\tau \times Q)} \delta(q,a,q') + p_f(q) = 1$.

We define the set of SNFA edges $E = (Q \times \Sigma_\tau \times Q)$. For an edge $e = (q,a,q') \in E$, we define $src(e) = q$, $\lambda(e) = a$, $tgt(e) = q'$. Notice that in an SNFA, each state may have multiple outgoing edges with the same label (including τ). Given a SNFA $N = (Q, \Sigma, \delta, q_0, p_f)$, a *path* π in N is a pair $\pi = (q, \rho) \in (Q \times E^*)$ where q is its *root* and $\rho = \langle e_1, e_2, \cdots, e_n\rangle \in E^*$ is a sequence of edges such that ρ is empty or $src(e_1) = q$, and $tgt(e_i) = src(e_{i+1})$ for every $1 \leq i < n$. We define $src(\pi) = q$ and $tgt(\pi) = tgt(e_n)$ (q if ρ is empty). We denote by $\Theta_N(q)$ the set of all paths in N such that $src(\pi) = q$. Similarly, $\Theta_N(q,q')$ is the set of paths in $\Theta_N(q)$ such that $tgt(\pi) = q'$, and $\Theta_N(q,q',n)$ is the set of paths in $\Theta_N(q,q')$ with length n. We extend δ to paths as $\delta(\pi) = \prod_{e_i \in \rho} \delta(e_i)$ (1 if π is empty). The labeling of a path $\pi = (q,\rho) \in (Q \times E^*)$ is the trace $\lambda(\pi)$ obtained by concatenating the *visible* labels $\lambda(e_i)$ of ρ. The accepting probability of a path π is $P_N(\pi) = \delta(\rho) \cdot p_f(tgt(\rho))$. The probability of N generating a trace $\sigma \in \Sigma^*$ is defined as the sum of the probabilities of all paths rooted in q_0 labeled with σ, i.e. $\mathbb{P}_N(\sigma) = \sum_{\rho \in \Theta_N(q_0), \lambda(\pi) = \sigma} P_N(\pi)$.

A state q in N is *accessible* if it can be reached from the starting state with non-zero probability and *useful* if it is visited by at least one path $\pi \in \Theta_N(q_0)$ such that $P_N(\pi) > 0$,

i.e. it is always possible to reach a final state from q with non-zero probability. Non-useful states are either inaccessible, or are in a livelock. N is *consistent* if all of its accessible states are useful (Vidal et al. 2005a), i.e. one never reaches a livelock from q_0. Last, a SNFA is *trimmed* if all of its states are accessible. A SNFA can be trimmed in linear time by removing all of its inaccessible states.

If N is consistent, then $\mathbb{P}_N : \Sigma^* \to [0, 1]$ defines a stochastic language (Vidal et al. 2005a). The class of *stochastic regular languages* is the class of stochastic languages that can be generated by a consistent SNFA (Vidal et al. 2005a).[2] This paper only considers trimmed, consistent SNFAs.[3] These can be directly obtained from bounded, livelock-free SLPNs.

Definition 5 (The Embedded SNFA of a Bounded, Livelock-Free SLPN) Let $SN = (P, T, F, \Sigma, \lambda, M_0, w)$ be a bounded, livelock-free SLPN. Its *embedded* SNFA is the SNFA $N = (\mathcal{R}(SN, M_0), \Sigma, \delta, M_0, p_f)$ such that:

$$\delta(M, a, M') = \frac{\sum_{t \in enabled(M), M[t\rangle M', \lambda(t)=a} w(t)}{\sum_{t \in enabled(M)} w(t)}$$

and

$$p_f(M) = \begin{cases} 1 & M \text{ is a deadlock marking} \\ 0 & \text{otherwise} \end{cases}$$

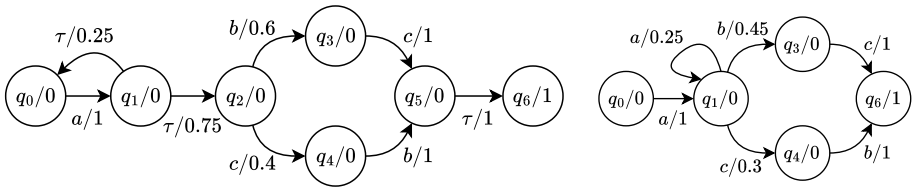
By construction, N generates the same stochastic language as SN. Figure 3a shows the embedded SNFA from the SLPN of Fig. 2. Notice that it contains τ transitions. A SNFA is *τ-free* if for every $q_i, q_j \in Q : \delta(q_i, \tau, q_j) = 0$, i.e. τ transitions have weight 0. Given an arbitrary SNFA N, one can always convert it into a τ-free SNFA N' by τ-removal (Hanneforth and de la Higuera 2010). Figure 3b shows SNFA N'_0 obtained from N_0 after τ -removal. N'_0 is coincidentally also *deterministic*. Stochastic deterministic regular languages can be expressed using a stochastic deterministic finite automaton:

Definition 6 (Stochastic Deterministic Finite Automaton (SDFA)) A stochastic deterministic finite automaton (SDFA) is an SNFA $D = (Q, \Sigma, \delta, q_0, p_f)$ with the additional restrictions that: $\delta(q, \tau, q') = 0 \; \forall \, (q, q') \in Q \times Q$ (no τ transitions) and $|\{q' \mid \delta(q, l, q') > 0\}| \leq 1 \; \forall \, (q, l) \in Q \times \Sigma$ (edges leaving each state are uniquely labeled).

In our running example, SNFA N'_0 in Fig. 3b is deterministic. Observe that while N_0 could be converted into an equivalent SDFA, this is not always possible as SDFAs are strictly less expressive than SNFAs (Dupont et al. 2005). Still, SDFAs and other restrictive classes of stochastic models are attractive as they are easier to analyze, e.g. trace probabilities can be computed in linear time (Vidal et al. 2005a) and weight estimation can be achieved using Maximum Likelihood Estimation (Dupont et al. 2005). This work

[2] Notice that the language induced by an SNFA has regular support, i.e. $supp(\mathbb{P}_N)$ is regular. However, not all stochastic languages with regular support can be represented by an SNFA Dupont et al. (2005).

[3] This is not a limitation since inconsistent SNFAs do not define a stochastic language and any consistent SNFAs can be trimmed in linear time.



(a) SN_0's embedded SNFA N_0 (b) Equivalent τ-free SNFA N_0'

Fig. 3 SNFAs N_0 and N_0' generating the same stochastic language as SN_0. N_0' is obtained from N_0 after τ -removal (the state names are kept). Termination probabilities are annotated in each state

has a close relation to stochastic k-testable languages (García and Vidal 1990), which we define using k-testable machines.

Definition 7 (K-Testable Machine (adapted from Chu and Bonsangue (2020))) Let Σ be an alphabet and $k > 1$. A *k-testable machine* is a 5-tuple $Z^k = (\Sigma, Pref, Suff, Inf, Short)$ where $Pref, Suff \subseteq \Sigma^{k-1}$ are the language's set of prefixes/suffixes of length $k-1$, $Inf \subseteq \Sigma^k$ is the set of infixes of length k of the language, and $Short \subseteq \Sigma^{<k-1}$ is the set of short traces in the language.[4]

Z^k accepts the language $\mathcal{L}(Z^k) = ((Pref\,\Sigma^* \cap \Sigma^* Suff) \setminus \Sigma^*(\Sigma^k \setminus Inf)^*\Sigma^*) \cup Short$, i.e. all the short sequences and all the finite sequences of activities that start with one of the allowed prefixes, contains only the allowed infixes and ends with one of the allowed suffixes. For example, the k-testable machines $Z^2 = (\{a,b,c\}, \{\langle a\rangle\}, \{\langle b\rangle, \langle c\rangle\}, \{\langle a,a\rangle, \langle a,b\rangle, \langle a,c\rangle, \langle b,c\rangle, \langle c,b\rangle\}, \emptyset)$ and $Z^3 = (\{a,b,c\}, \{\langle a,a\rangle, \langle a,b\rangle, \langle a,c\rangle\}, \{\langle b,c\rangle, \langle c,b\rangle\}, \{\langle a,a,a\rangle, \langle a,a,b\rangle, \langle a,a,c\rangle, \langle a,b,c\rangle, \langle a,c,b\rangle\}, \emptyset)$ accept languages $\mathcal{L}(Z^2) = \{\langle a,b\rangle, \langle a,a,b\rangle, \langle a,c\rangle, \langle a,b,c\rangle, \langle a,c,b\rangle, \langle a,a,b,c,b\rangle, \cdots\}$ and $\mathcal{L}(Z^3) = \{\langle a,b,c\rangle, \langle a,c,b\rangle, \langle a,a,b,c\rangle, \langle a,a,c,b\rangle, \cdots\}$, respectively.

The parameter k controls the machine's memory size and, consequently, its expressive power. For a fixed k, the set of languages recognizable by a k-testable machine forms the class of k-testable languages. This class represents a strictly less expressive subclass of regular languages. Nevertheless, they can approximate certain regular languages with increasing accuracy as k grows. For example, $\mathcal{L}(Z^2)$ and $\mathcal{L}(Z^3)$ are the smallest 2/3-testable languages that include $supp(SN_0)$ (Fig. 2). Notice that $supp(SN_0) \subset \mathcal{L}(Z^2)$ and $supp(SN_0) = \mathcal{L}(Z^3)$, i.e. Z^2 does not perfectly describe $supp(SN_0)$, while Z^3 does. Finally, we define stochastic k-testable languages (*k-TSSL*) as the set of stochastic regular languages with k-testable support. Similarly, stochastic k-testable languages form a proper subclass of stochastic deterministic regular languages (García and Vidal 1990). Figure 4 shows an overview of the classes of stochastic languages considered in this work.

[4] We slightly differ from the definition in Chu and Bonsangue (2020) by considering $Short \subseteq \Sigma^{<k-1}$ instead of $\subseteq \Sigma^{<k}$ to ensure that $Short \cap Pref\,\Sigma^* \cap \Sigma^* Suff = \emptyset$.

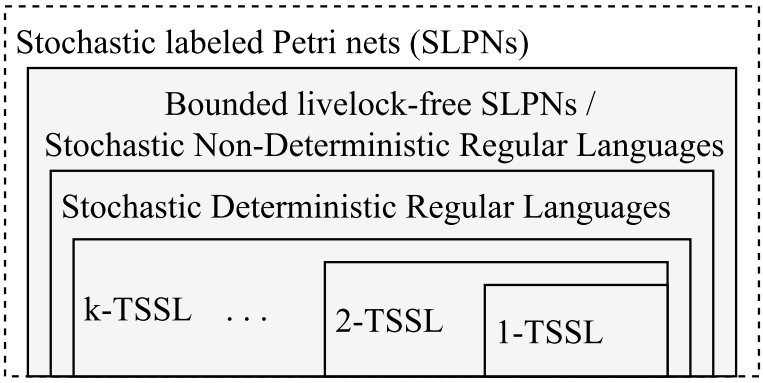


Fig. 4 Hierarchy of classes of stochastic languages. The classes treated by this paper are colored gray. Stochastic Deterministic Regular Languages can be approximated by a k-TSSL for a sufficiently large k

Table 1 Example reallocation function from the example event log L_0 to the process model SN_0. No probability is reallocated from L_0's $\langle a,b,c\rangle$ and $\langle a,c,b\rangle$. All the remaining traces in the model are covered by $\langle a,a,a,b,c\rangle$

		Model (SN_0)					
		$\langle a,b,c\rangle$	$\langle a,c,b\rangle$	$\langle a,a,b,c\rangle$	$\langle a,a,c,b\rangle$	$\langle a,a,a,b,c\rangle$	$\cdots$
Log (L_0)	**Prob.**	**0.45**	**0.3**	**0.1125**	**0.075**	**0.028125**	$\cdots$
$\langle a,b,c\rangle$	0.45	0.45	0.0	0.0	0.0	0.0	$\cdots$
$\langle a,c,b\rangle$	0.3	0.0	0.3	0.0	0.0	0.0	$\cdots$
$\langle a,a,a,b,c\rangle$	0.25	0.0	0.0	0.1125	0.075	0.028125	$\cdots$

Earth mover's stochastic conformance measure

This work is heavily based on the earth mover's stochastic conformance measure (Leemans et al. 2021), which we detail below. In order to introduce it, we must first introduce the notion of a reallocation function.

Definition 8 (Reallocation Function) Let Σ be an alphabet and $l_1, l_2 : \Sigma^* \to [0,1]$ be stochastic languages. A reallocation function is a function $r_{l_1,l_2} : \Sigma^* \times \Sigma^* \to [0,1]$ such that:

$$\forall \sigma \in \Sigma^*,\ l_1(\sigma) = \sum_{\sigma' \in \Sigma^*} r_{l_1,l_2}(\sigma,\sigma') \text{ and } \forall \sigma' \in \Sigma^*,\ l_2(\sigma') = \sum_{\sigma \in \Sigma^*} r_{l_1,l_2}(\sigma,\sigma') \tag{1}$$

A reallocation function defines an assignment from a source probability distribution to a target probability distribution. Table 1 shows a possible reallocation function from the event log L_0 to the process model SN_0 from Fig. 2.

The earth mover's stochastic conformance measure searches for an optimal reallocation with respect to a given distance measure.

Definition 9 (Earth Mover's Stochastic Conformance Measure (EMSC)) Let Σ be an alphabet, $l_1, l_2 : \Sigma^* \to [0,1]$ stochastic languages, and $\mathcal{R}_{l_1,l_2}$ the set of all possible reallocation functions from l_1 to l_2. Let $c : \Sigma^* \times \Sigma^* \to [0,1]$ be a trace distance function. The

Earth Mover's Stochastic Conformance measure (EMSC) with cost function c is defined as:

$$EMSC(l_1, l_2, c) = 1 - \min_{r \in \mathcal{R}_{l_1,l_2}} \sum_{(\sigma,\sigma') \in \Sigma^* \times \Sigma^*} r(\sigma, \sigma') c(\sigma, \sigma') \tag{2}$$

Two cost functions are commonly used in process mining. The normalized string edit (Levenshtein) distance (Leemans et al. 2021) (without replacements) and the unit distance between traces $u : \Sigma^* \times \Sigma^* \to [0, 1]$ defined as $u(\sigma, \sigma') = 1 - \mathbf{1}_{\{\sigma\}}(\sigma')$, defining the EMSC and uEMSC stochastic conformance measures respectively.

Both measures have significant weaknesses. The Levenshtein distance is robust to partial mismatches. Because of that, the EMSC is often a robust measure for comparing stochastic languages. However, the EMSC cannot be computed if one of the languages has infinite support (which is the common case in process mining), therefore, an approximation computed by sampling the model behavior is used instead, yielding the *truncated* EMSC measure (tEMSC). The tEMSC measure requires sampling a representative probability mass from the model. In the example of Table 1, we could truncate SN_0's language to consider only its five most frequent model traces, which represent approximately 97% of the model's total behavior. While this may suffice for this simple process model, sampling enough behavior is difficult for models with a high degree of parallelism or with loops. Furthermore, even if the sampling step succeeds, solving the associated minimization problem requires solving an optimal transportation problem (Leemans et al. 2021), which is computationally expensive.

In contrast, the uEMSC can be computed in an exact manner (Leemans et al. 2024b) and is efficient in practice, however, the unit cost function severely penalizes partial trace mismatches. For example, $u(\langle a, b, c\rangle, \langle a, a, b, c\rangle) = 1$, indicating that these are totally different traces, even though they only differ in one step. Therefore, the tEMSC and the uEMSC measures compromise on one of the quality criteria listed in "Introduction" section. In the next section, we see how the Markovian abstraction can be used to alleviate these issues.

The stochastic markovian abstraction

This section presents the stochastic Markovian abstraction, a natural extension of the Markovian abstraction from Augusto et al. (2022) to stochastic languages. We first define the abstraction. Then, we show how to compute it for stochastic regular languages and discuss some of its properties, including a necessary and sufficient condition for its existence. The abstraction is based on the multiset of k-trimmed subtraces:

Definition 10 (Multiset of K-Trimmed Subtraces) Let $\sigma \in \Sigma^*$ and $k \geq 1$. The multiset of k-trimmed subtraces S^k_σ is recursively defined as:

$$S^k_\sigma = \begin{cases} [\sigma] & \text{if } |\sigma| \leq k \\ \left[\sigma^{1 \to k}\right] \uplus S^k_{\sigma^{2 \to |\sigma|}} & \text{otherwise} \end{cases}$$

Let $\sigma = \langle a, a, b, c\rangle$, then $S^1_\sigma = [\langle a\rangle^2, \langle b\rangle, \langle c\rangle]$, $S^2_\sigma = [\langle a, a\rangle, \langle a, b\rangle, \langle b, c\rangle]$, $S^3_\sigma = [\langle a, a, b\rangle, \langle a, b, c\rangle]$, and $S^k_\sigma = [\langle a, a, b, c\rangle]$ for all $k \geq 4$. The k-th order multiset Makorvian abstraction of a trace (defined below) is similar to the multiset of k-trimmed

subtraces but with special start/end markers $+/-$ to track the language's prefixes and suffixes.

Definition 11 (K-th Order Multiset Markovian Abstraction of a Trace) Let Σ be an alphabet, $+/-$ be special start/end markers, with $+, - \notin \Sigma$ and $k \geq 2$. The k-th order multiset Markovian abstraction of a trace $\sigma \in \Sigma$ is defined as:

$$M^k_\sigma = S^k_{+\sigma-}$$

From now on, we abbreviate $\Sigma \cup \{+, -\} = \Sigma_\pm$. Consider again $\sigma = \langle a, a, b, c\rangle$, then $M^2_\sigma = [\langle +, a\rangle, \langle a, a\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle c, -\rangle]$, $M^3_\sigma = [\langle +, a, a\rangle, \langle a, a, b\rangle, \langle a, b, c\rangle, \langle b, c, -\rangle]$, $M^4_\sigma = [\langle +, a, a, b\rangle, \langle a, a, b, c\rangle, \langle a, b, c, -\rangle]$, $M^5_\sigma = [\langle +, a, a, b, c\rangle, \langle a, a, b, c, -\rangle]$, and $M^k_\sigma = [+\sigma-]$ for $k \geq 6$. Definition 11 is naturally extended to stochastic languages by considering trace probabilities as follows:

Definition 12 (K-th Order Stochastic Markovian Abstraction) Let Σ be an alphabet, $l : \Sigma^* \to [0, 1]$ a stochastic language, and $k \geq 2$. Define $f^k_l : \Sigma^*_\pm \to \mathbb{R}_{\geq 0}$ as:

$$f^k_l(\gamma) = \sum_{\sigma \in \Sigma^*} l(\sigma) \cdot M^k_\sigma(\gamma) \tag{3}$$

If the sum $\sum_{\gamma' \in \Sigma^*_\pm} f^k_l(\gamma')$ is finite and non-zero, the k-th order stochastic Markovian abstraction of l is the function $m^k_l : \Sigma^*_\pm \to [0, 1]$ defined as:

$$m^k_l(\gamma) = \frac{f^k_l(\gamma)}{\sum_{\gamma' \in \Sigma^*_\pm} f^k_l(\gamma')} \tag{4}$$

Otherwise, m^k_l is undefined.

Function f^k_l returns the expected number of occurrences of l's k-trimmed subtraces, with special start and end markers to track its prefixes and suffixes. The k-th order stochastic Markovian abstraction m^k_l is the normalization of f^k_l, yielding a probability distribution over subtraces, provided the total expected count is finite. This is a natural generalization of the non-stochastic version from Goulart Rocha and van der Aalst (2023). The use of prefix/suffix markers $+/-$ is for practical reasons as they help distinguish between prefixes, suffixes, infixes, and short traces, which makes it easier to define operations on this set, as done in Goulart Rocha and van der Aalst (2023).

For the example event log L_0 from Fig. 2, we obtain that $f^2_{l_{L_0}}(\langle +, a\rangle) = \Big(l_{L_0}(\langle a, b, c\rangle) * M^2_{\langle a,b,c\rangle}(\langle +, a\rangle)\Big) + \Big(l_{L_0}(\langle a, c, b\rangle) * M^2_{\langle a,c,b\rangle}(\langle +, a\rangle)\Big) + \Big(l_{L_0}(\langle a, a, a, b, c\rangle) * M^2_{\langle a,a,a,b,c\rangle}(\langle +, a\rangle)\Big) = \frac{9}{20} * 1 + \frac{6}{20} * 1 + \frac{5}{20} * 1 = \frac{20}{20}$ and (in general), $f^2_{l_{L_0}} = [\langle +, a\rangle^{\frac{20}{20}}, \langle a, b\rangle^{\frac{14}{20}}, \langle b, c\rangle^{\frac{14}{20}}, \langle c, -\rangle^{\frac{14}{20}}, \langle a, a\rangle^{\frac{10}{20}}, \langle a, c\rangle^{\frac{6}{20}}, \langle c, b\rangle^{\frac{6}{20}}, \langle b, -\rangle^{\frac{6}{20}}]$. And so $\sum_{\gamma \in \Sigma^*_\pm} f^k_l(\gamma) = \frac{90}{20}$. Therefore, $m^2_{l_{L_0}} = [\langle +, a\rangle^{\frac{20}{90}}, \langle a, b\rangle^{\frac{14}{90}}, \langle b, c\rangle^{\frac{14}{90}}, \langle c, -\rangle^{\frac{14}{90}}, \langle a, a\rangle^{\frac{10}{90}}, \langle a, c\rangle^{\frac{6}{90}}, \langle c, b\rangle^{\frac{6}{90}}, \langle b, -\rangle^{\frac{6}{90}}]$.

For finite stochastic languages, m^k can be computed by counting the subtrace frequencies and normalizing them. Next, we show a procedure to compute m^k for

stochastic regular languages represented by a SNFA. But first, we must introduce patched SNFAs and an SNFA's transition matrix.

Definition 13 (Patched SNFA) Let $N = (Q, \Sigma, \delta, q_0, p_f)$ be an SNFA. Its patched SNFA is the SNFA $\hat{N} = (Q \cup \{q_+, q_-\}, \Sigma_\pm, \hat{\delta}, q_+, \hat{p}_f)$, where $\hat{p}_f = \mathbf{1}_{\{q_-\}}$, and

$$\hat{\delta}(q, a, q') = \begin{cases} p_f(q) & \text{if } q' = q_- \text{ and } a = - \\ 1 & \text{if } q = q_+, a = +, \text{ and } q' = q_0 \\ \delta(q, a, q') & \text{if } q, q' \in Q, a \in \Sigma \cup \{\tau\} \\ 0 & \text{otherwise} \end{cases}$$

Figure 5a shows the SNFA $\hat{N}'_0$ obtained by patching N'_0. $\hat{N}'_0$ adds a source state q_+ and a sink state q_- to N'_0. It accepts the stochastic language $\hat{l} : \Sigma^*_\pm \to [0, 1]$ equal to language l concatenated with prefix $+$ and suffix $-$, i.e. $l(\sigma) = \hat{l}(+\sigma-)\ \forall\, \sigma \in supp(l)$.

Definition 14 (SNFA's Transition Matrix) Let $N = (Q, \Sigma, \delta, q_0, p_f$ be an SNFA. Its *transition matrix* is the matrix $\Delta = [\Delta_{i,j}] \in \mathbb{R}^{|Q|\times|Q|}_{\geq 0}$, where $\Delta_{ij} = \sum_{l\in\Sigma} \delta(q_i, l, q_j)$.

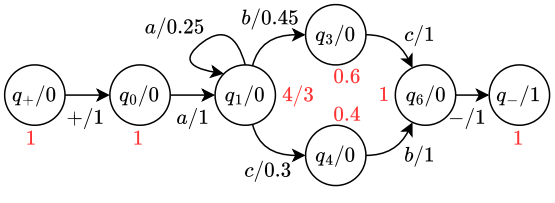
Figure 5b shows the transition matrix $\hat{\Delta}$ of the patched SNFA $\hat{N}'_0$. Each entry $\hat{\Delta}_{ij}$ represents the probability of transitioning from state q_i to q_j in $\hat{N}'_0$ via any edge. The lemma below shows how this can be used to compute m^k of a stochastic language.

Lemma 1

(*Computing f^k_l of a Stochastic Regular Language l*) *Let Σ be an alphabet, $N = (Q, \Sigma, \delta, q_0, p_f)$ be a trimmed, consistent τ-free SNFA generating the stochastic language $l : \Sigma^* \to [0, 1]$, and $k \geq 2$. Let $\hat{N} = (\hat{Q}, \Sigma_\pm, \hat{\delta}, q_+, \hat{p}_f)$ be N's patched SNFA generating language $\hat{l}$. For $\gamma \in supp(f^k_l)$, it holds that*:

$$f^k_l(\gamma) = \sum_{q\in\hat{Q}} \mathbf{x}[q] \cdot \hat{\phi}(\gamma|q) \tag{5}$$

Where $\mathbf{x} = [x_{q_+}\ x_{q_0} \cdots x_{q_-}]$ is the solution to the equation $\mathbf{x}(I - \hat{\Delta}) = [1\, 0 \cdots 0]$ ($\hat{\Delta}$ is $\hat{N}$'s transition matrix) and $\hat{\phi}(\gamma|q) = \sum_{\pi_\gamma \in \Theta_{\hat{N}}(q), \lambda(\pi_\gamma)=\gamma} \hat{\delta}(\pi_\gamma)$ is the probability of generating the sequence γ starting from state q in N.



(a)

$$\hat{\Delta} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0.45 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(b)

Fig. 5 $\hat{N}'_0$ (5a), obtained by patching N'_0 from Fig. 3a, and its transition matrix $\hat{\Delta}$ (5b). The solution to $\mathbf{x}(I - \hat{\Delta}) = [1\, 0 \cdots 0]$ is annotated in red

Proof

See Appendix A. The idea is to first prove that $f_l^k(\gamma) = \sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta)$ *for* $\gamma \in supp(f_l^k)$ *and then prove that* $\sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = \sum_{q' \in \hat{Q}} x_{q'} \cdot \hat{\phi}(\gamma|q')$. □

Here, $\mathbf{x}[q]$ represents the expected number of visits to state q of a random walk over $\hat{N}$ starting from state q_+ before absorption in q_- (Grinstead and Snell 2003) and $\hat{\phi}(\gamma|q)$ the replay probability of subtrace γ from state q. The formula from Eq. 5 directly provides a method to compute m_l^k given a trimmed, consistent SNFA N accepting l. First, we transform N into a τ-free SNFA N' with a τ-removal step (Hanneforth and de la Higuera 2010) and patch N' by adding the $+/-$ markers, obtaining $\hat{N}'$. Then $\mathbf{x}$ can be computed by solving the system of linear equations $\mathbf{x}(I - \hat{\Delta}) = [1\ 0 \cdots 0]$, and $\hat{\phi}(\gamma|q)$ can be computed by path enumeration over $\hat{N}'$ starting from q for each required γ. The algorithm works by summing each subtrace's replay probabilities starting from each state and adding up their contributions, weighted by the respective expected number of state visits. Notice that this only works because N' is τ-free and trimmed.

Figure 5b shows the transition matrix $\hat{\Delta}$ of $\hat{N}_0'$ (Fig. 5a). Solving $\mathbf{x}(I - \hat{\Delta}) = [1\ 0 \cdots 0]$ for $\mathbf{x}$, we obtain $\mathbf{x} = [1\ 1\ 4/3\ 0.6\ 0.4\ 1\ 1]$, (annotated in red in Fig. 5a). From that, it is possible to compute $f_l^k(\gamma)$ as $\sum_{q_i \in \hat{Q}} x_{q_i} \cdot \hat{\phi}(\gamma|q_i)$. For example, $\hat{\phi}(\langle a, b\rangle|q_0) = 1 * 0.45 = 0.45$, $\hat{\phi}(\langle a, b\rangle|q_1) = 0.25 * 0.45 = 0.1125$, and $\hat{\phi}(\langle a, b\rangle|q_i) = 0$ for the other states q_i. Therefore, $f_l^2(\langle a, b\rangle) = 1 * 0.45 + 4/3 * 0.1125 = 0.6$.

Runtime

The m^k abstraction of a finite stochastic language (like an event log) can be computed via single iteration over the traces. Let SN be a SLPN. Let $N = (Q, \Sigma, \delta, q_0, p_f)$ be its embedded SNFA (Definition 5) and $N' = (Q', \Sigma, \delta', q_0', p_f')$ a τ-free SNFA generating the same language as N and $\hat{N}' = (\hat{Q}, \Sigma_\pm, \hat{\delta}, q_+, \hat{p}_f)$ its patched SNFA (Definition 13). Then $|Q| = |\mathcal{R}(SN, M_0)|$, where $|\mathcal{R}(SN, M_0)|$ can be exponential in the number of places in SN. $|Q'| \leq |Q|$ (Mohri 2011), but N' has higher edge density than N.

The computation of Lemma 1 occurs in two steps: First, solve the linear system $\mathbf{x}(I - \hat{\Delta}) = [1\ 0 \cdots 0]$. Then, compute $\hat{\phi}(\gamma|q)$ for each $\gamma \in supp(f_l^k)$ and $q \in Q$. For the first step, $I - \hat{\Delta}$ is a $|\hat{Q}| \times |\hat{Q}|$ matrix ($|\hat{Q}| = |Q'| + 2$). Oftentimes, e.g. if the model is deterministic, $I - \hat{\Delta}$ is very sparse, such that the system can be efficiently solved. The second step has runtime in $\mathcal{O}(|\hat{Q}| * |\Sigma_\pm|^k)$, which is the same as the method for its non-stochastic version m^k (Augusto et al. 2022). Despite the relatively high exponents, "Quantitative evaluation" section shows that the computation is still feasible.

The stochastic K-testable language induced by m^k

For a given stochastic language l with well-defined m^k, $supp(m_l^k)$ implicitly defines a k-testable machine $Z_l^k = (\Sigma, Pref, Suff, Inf, Short)$, where the prefixes/suffixes are obtained from subtraces containing only the start/end marker, the infixes are obtained from subtraces containing neither markers, and the short traces are the subtraces containing both. Furthermore, it holds that Z_l^k is the smallest k-testable machine such that

$supp(l) \subseteq \mathcal{L}(Z_l^k)$ (Chu and Bonsangue 2020). Moreover, we can create a stochastic DFA D generating the stochastic k-testable language $TSSL_l^k$ such that $supp(TSSL_l^k) = \mathcal{L}(Z_l^k)$ and $m^k_{TSSL_l^k} = m_l^k$, i.e. m^k of $TSSL_l^k$ is the original m_l^k (see Chu and Bonsangue (2020) for the construction and Vidal et al. (2005b) for the proofs). We say that $TSSL_l^k$ is the stochastic language *induced* by m_l^k. In the limit, as k increases, $TSSL_l^k$ approaches l. In the "Approximation quality of the markovian abstraction" section, we will use $TSSL_l^k$ to measure how well m^k represents l.

Well-definedness beyond stochastic regular languages

A consequence of Lemma 1 is that m_l^k of stochastic regular languages is well-defined. This is not all too surprising considering existing work on island probability computation for stochastic regular (Fred 2000) and context-free grammars (Corazzat et al. 1990). For arbitrary stochastic languages, this might not be the case. Consider $l : \{a\}^* \to [0,1]$ defined as $l(\sigma) = \begin{cases} \frac{1}{2^{n+1}} & \text{if } \sigma = \langle \underbrace{a, a, \cdots a}_{2^n+1 \text{ times}} \rangle, \text{for some } n \in \mathbb{N}_0 \\ 0 & \text{otherwise} \end{cases}$. Then $l = \{\langle a,a\rangle^{\frac{1}{2}}, \langle a,a,a\rangle^{\frac{1}{4}}, \langle a,a,a,a,a\rangle^{\frac{1}{8}} \cdots\}$. l is a stochastic language since $\sum_{\sigma\in\{a\}^*} l(\sigma) = 1$. However, $f_l^2(\langle a,a\rangle) = 1\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 4\cdot\frac{1}{8} + \cdots = \infty$, i.e. f_l^2 is not defined. The following lemma shows that a stochastic language's expected trace length being finite is a necessary and sufficient condition for $f_l^k < \infty$.

Lemma 2

(*A Necessary and Sufficient Condition for Well-Defined* f_l^k) *Given a stochastic language* $l : \Sigma^* \to [0,1]$. *Then for any* $k \geq 2$

$$f_l^k(\gamma) < \infty \; \forall \gamma \in \Sigma_{\pm}^{\leq k} \iff \sum_{\sigma\in\Sigma^*} l(\sigma)|\sigma| < \infty \tag{6}$$

Proof

See Appendix A. Use the inequality $|\sigma|/k \leq |S_\sigma^k| \leq |\sigma| + 1$. □

It follows directly from $supp(f_l^k) \subseteq \Sigma_{\pm}^{\leq k}$ that if f_l^k is well-defined, then so is m^k. We can use Lemma 2 to show that there exist unbounded livelock-free SLPNs for which m_l^k is well-defined. For example, consider the net from Fig. 6.

The net generates language $l : \{a,b,c\}^* \to [0,1]$. Observe that $supp(l) = \{\sigma \in \{a,b,c\}^* \mid |\sigma_{\{a\}}| = |\sigma_{\{b\}}| = |\sigma_{\{c\}}|\}$, where σ_X is the projection of σ in X, i.e. $supp(l)$ is the set of all sequences containing an equal number of occurrences of a, b, and c. Traces in $supp(l)$ have size $3n$ for $n \in \mathbb{N}_0$ and $\sum_{\sigma\in supp(l),|\sigma|=3n} = \frac{1}{2^{n+1}}$, i.e. the sum of the probabilities of all traces of length $3n$ is $2^{-(n+1)}$. Thus, $\sum_{\sigma\in\Sigma^*} |\sigma| l(\sigma) = \sum_{n=0}^{\infty} \frac{3n}{2^{n+1}} < \infty$. Therefore (Lemma 2), m_l^k is well-defined for the net above. The question arises whether m_l^k is well-defined for all livelock-free SLPNs. Which we could neither prove nor disprove.

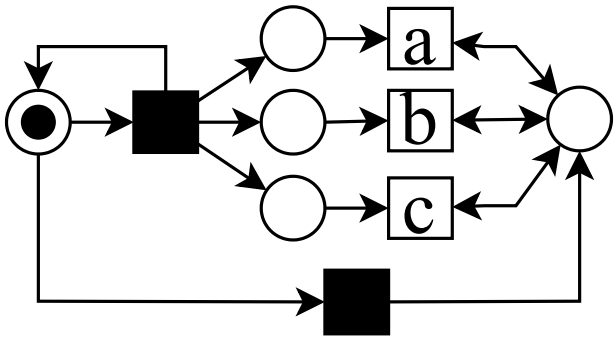


Fig. 6 An unbounded, livelock-free SLPN generating stochastic language l with finite expected trace length, ensuring m_l^k is well-defined. All transitions have a weight of 1

Derived stochastic measures

Definition 12 defines a stochastic language. Hence, given two stochastic languages l_1 and l_2, their respective abstractions, $m_{l_1}^k$ and $m_{l_2}^k$, can be compared using any established stochastic conformance measure. This paper considers the two variations of the EMSC measure (uEMSC and EMSC) introduced in the "Earth mover's stochastic conformance measure" section to obtain the "new" derived m^k-EMSC and m^k-uEMSC measures defined as $m^k\text{-EMSC}(l_1, l_2) = \text{EMSC}(m_{l_1}^k, m_{l_2}^k)$ and $m^k\text{-uEMSC}(l_1, l_2) = \text{uEMSC}(m_{l_1}^k, m_{l_2}^k)$.

The m^k-EMSC measure inherits the "good" quality characteristics from the original EMSC measure, namely its robustness to partial mismatches. At the same time the Markovian abstraction requires no sampling, which makes it scalable for complex models where it is not possible to sample enough behavior. However, a significant disadvantage of the m^k-EMSC measure, also inherited directly from the base EMSC measure, is that m^k-EMSC still requires computing the optimal reallocation by solving an optimal transport problem, which can be computationally expensive.

In contrast, the m^k-uEMSC measure can be computed using the formula $m^k\text{-}uEMSC(l_1, l_2) = 1 - \sum_{\gamma \in \Sigma_{\pm}^{\leq k}} max(m_{l_1}^k(\gamma) - m_{l_2}^k(\gamma), 0)$ (Leemans et al. 2021), which can be computed in a linear pass over $m_{l_1}^k$ and $m_{l_2}^k$. By using the stochastic Markovian abstraction, m^k-uEMSC compensates for the rigidity of the uEMSC measure regarding partially matching traces. Intuitively, m^k-uEMSC compares the expected frequency of subtraces in both languages. m^k-uEMSC is more scalable than m^k-EMSC and should be used if m^k-EMSC cannot be computed efficiently.

Note that m_l^k approaches $\hat{l}$ as $k \to \infty$. Therefore, for larger ks, $m^k\text{-}EMSC(l_1, l_2)$ approaches $EMSC(\hat{l}_1, \hat{l}_2)$ and $m^k\text{-}uEMSC(l_1, l_2)$ approches $uEMSC(\hat{l}_1, \hat{l}_2) = uEMSC(l_1, l_2)$. This implies that for large ks, m^k measures will increasingly share the same limitations and characteristics as their respective base measures (EMSC and uEMSC).

Quantitative evaluation

This section evaluates the abstraction and its derived measures on a series of synthetic and real-world datasets. We investigate how well the abstraction represents the original languages, and whether it can mitigate the weaknesses of their base measures.

We evaluate the approach on a collection of real-world event logs including event logs taken from multiple editions of the Business Process Intelligence Challenges (Steeman 2013; van Dongen 2014, 2015; Khayatbashi et al. 2023; van Dongen 2020) (BPI), the italian road fines traffic management event log (de Leoni and Mannhardt 2015) (RF), the hospital billing event log (Mannhardt 2017) (HB), and the sepsis event log (Mannhardt 2016) (SEPSIS). The BPI datasets are split into sub-logs according to their sub-processes. The BPI 2015 log is prepared by merging the data of all of its municipalities. Furthermore, we select only its five largest sub-processes (by number of events) to avoid a single dataset from dominating the results.

For each event log, we discover multiple stochastic process models by mining control flow models and combining them with a weight estimation method from Burke et al. (2021). For the control-flow perspective, we consider the Split Miner (Augusto et al. 2019) with default parameters (SM), the Inductive Miner infrequent variant (Leemans et al. 2014) with a noise threshold of 0.2 (IMf02), the Directly-Follows Miner (Leemans et al. 2019) with a frequency threshold of 0.8, and the Flower Miner (FM). As weight estimators, we use the frequency-based estimator (FBE), the Bill Clinton estimator (BCE) (fork distribution estimator in Burke et al. (2021)), and the alignment-based estimator (ABE).

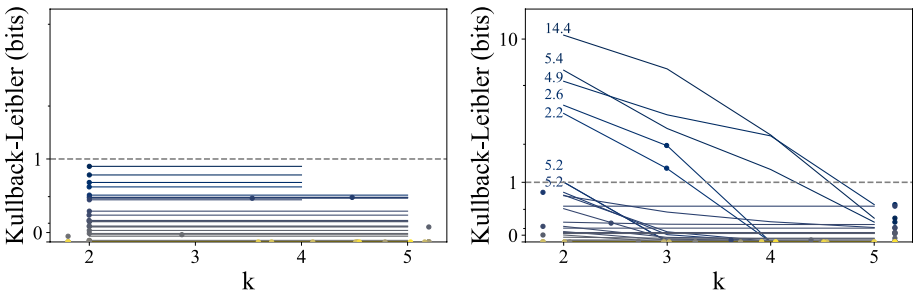
The weight estimators are assigned a single-threaded maximum execution time of two hours. We discard unsound control-flow models. We further discard trivial control-flow models that consist of a single trace. Furthermore, for the directly-follows and flower miners, the BCE and FBE estimators return models with equal weights, therefore, we discard the BCE estimator for these models. In the end, our dataset consists of 20 event logs spawning seven different process types, 76 unique control-flow models, and 190 unique stochastic models.

Approximation quality of the markovian abstraction

We start by evaluating how effectively the proposed abstraction approximates its original stochastic language l. To this end, we compare the Kullback-Leibler (KL) divergence (in bits) between the stochastic languages l and m_l^k's induced stochastic k-testable language $TSSL_l^k$.[5] As discussed in the section "The stochastic K-testable language induced by m^k", we expect that $TSSL_l^k$ will converge to l as k increases, i.e. we expected $KL(l, TSSL_l^k)$ to approach 0 as k increases.

Interpreting the value of the Kullback-Leibler divergence can be challenging. Therefore, we provide context by also considering entropy of the original language $H(l)$ (annotated on the left, only for languages with KL higher than one). If one interprets the KL divergence as the "excess entropy" incurred when encoding l using $TSSL_l^k$, then we say that the approximation is "good" if the divergence is relatively small compared to l's entropy. Furthermore, the effectiveness of the abstraction diminishes if k becomes excessively large compared to the typical length of traces in l; in such cases, the abstraction is exhaustively enumerating traces of l, rather than providing a compact representation of l. To monitor this, we mark each language's expected trace length on the plots (indicated

[5] The Kullback-Leibler is suitable in this setup since $supp(l) \subseteq supp(TSSL_l^k)$



(a) DFM and FM models. (b) IMf02 and SM models.

Fig. 7 Approximation quality of Markovian abstraction for the discovered stochastic models, measured as $KL(l, TSSL_l^k)$. We only consider models for which the sampled behavior covers more than 90% of model behavior. Dots indicate each model's expected trace length. Models where $KL(l, TSSL_l^2) > 1$are annotated with the model's entropy. Some lines terminate before $k = 5$. These corresponds to points where m^k could not be computed

by dots). We consider the abstraction tp successfully approximate l if $KL(l, TSSL_l^k)$ is relatively small compared to $H(l)$ for a value of k that is less than the expected trace length of l.

We start the evaluation by considering the stochastic languages derived from the discovered models. We estimate the KL divergence via sampling-based approach. For that, we sample two million model traces via unbiased sampling using the Ebi tool (Leemans et al. 2024a). We limit the sampling time to two hours and 64 GB of RAM. The reliability of the computed KL divergence heavily depends on how well these samples represent the true model behavior. Therefore, we discard models for which the sample covers less than 90% of the model behavior, as conclusions drawn from these are deemed unreliable. Finally, we split our models into two groups based on their control-flow miners: the DFM/FM models and the IMf02/SM models. The former are based solely on the directly-follows relation, therefore we expect m^2 to perfectly approximate their languages. The results of this analysis are presented in Fig. 7.

As expected, we see that the abstraction can approximate the DFM and FM models well and that the divergence does not change as k increases (Fig. 7a). For the IMf02 and SM models (Fig. 7b), the results confirm the trend that $TSSL_l^k$ approaches the original language l as k grows. In fact, at $k = 2$, most models already achieve $KL < 1$. To those for which that isn't the case, KL quickly decreases as k increases, falling

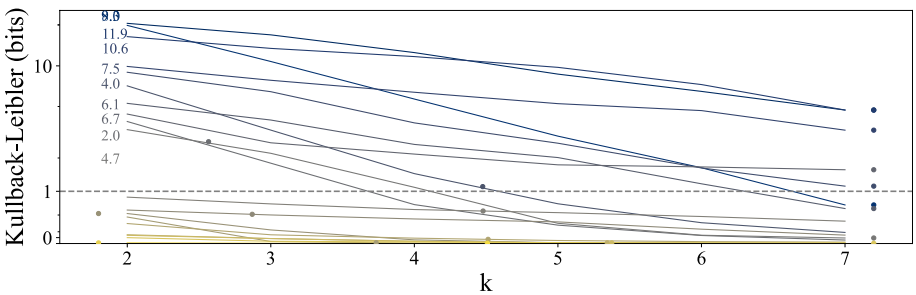


Fig. 8 Approximation quality for the event logs

below a bit for all models at $k = 5$. This is despite most models having expected trace length greater than 5.

Next, we analyze how well the abstraction can approximate the event logs. These can be particularly challenging since, as opposed to our considered control-flow models, they contain stochastic long-term dependencies, which our abstraction cannot properly handle. The results are summarized in Fig. 8. Here, the performance is more varied. While the abstraction adequately approximates most of the logs, three specific logs- BPI-2014, BPI-2015-Workflow, and SEPSIS - maintain relatively high KL divergence values even at $k = 7$. This reduced accuracy is likely caused by the presence of significant long-term dependencies within these log.

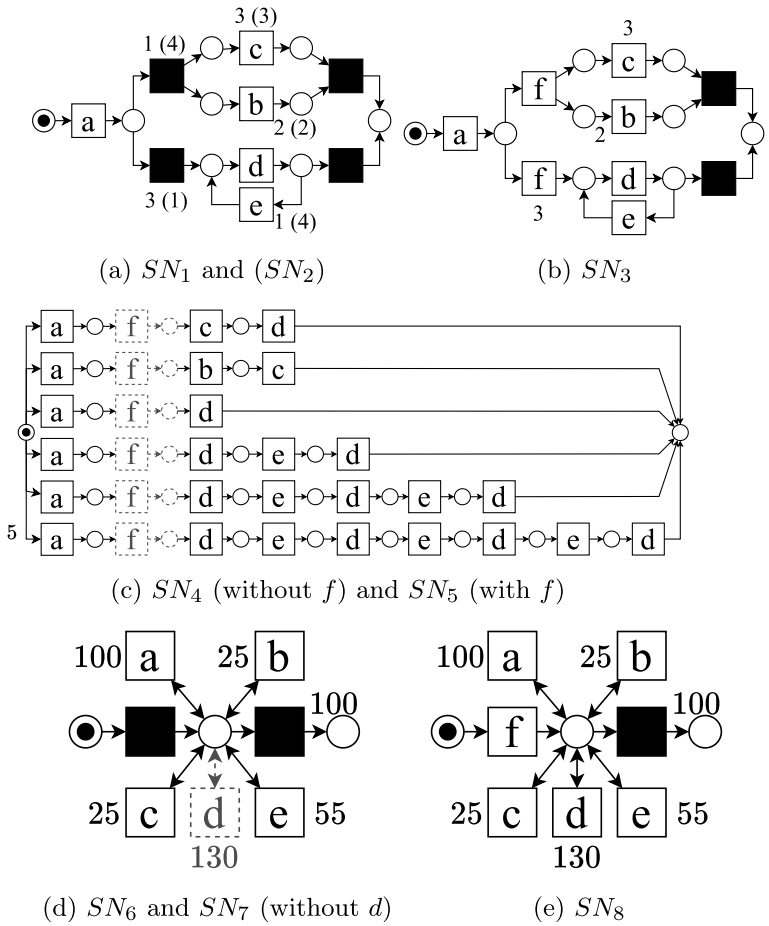
In summary, the experiments in this section demonstrate that the abstraction effectively approximates the stochastic languages derived from various discovered process models. However, its accuracy is lower for certain complex event logs, likely due to the influence of long-term dependencies inherent in the real-world processes they represent. Nevertheless, as will be shown in the following sections, the abstractions can still be applied for stochastic conformance checking.

mk-EMSC and mk-uEMSC measures

We now evaluate the derived conformance measures from the "Derived stochastic measures" section. For that, we consider the tEMSC, and the uEMSC measures and their derived m^k measures, with k varying from 2 to 4. For tEMSC, the sampling process terminates when either 99% of the probability mass is covered, 50,000 model simulation runs are completed, or 100 seconds have elapsed, whichever occurs first. When enough behavior can be sampled from the model, the tEMSC measure provides what we consider to be the reference value for the induced ranks. The m^k computation is implemented in pure Python, except for the solution of the linear system $\mathbf{x}(I - \hat{\Delta}) = [1\ 0 \cdots 0]$, which leverages native linear algebra libraries (NumPy and SciPy). Compared to our work in Rocha et al. (2024), we replaced ProM's reference tEMSC implementation with a highly optimized custom implementation that uses Python for sampling the model and C++ for solving the EMD problem. All experiments are run single-threaded on an Intel Xeon Platinum 8468 CPU running Ubuntu 24.04.

Synthetic datasets

We start with a controlled experiment to evaluate the sanity of the derived measures. For that, we construct a synthetic dataset consisting of an event log $L = [\langle a, c, b\rangle^{10}, \langle a, b, c\rangle^{15}, \langle a, d\rangle^{40}, \langle a, d, e, d\rangle^{20}, \langle a, d, e, d, e, d\rangle^{10}, \langle a, d, e, d, e, d, e, d\rangle^{5}]$ and a set of manually designed process models (displayed in Fig. 9). The models are constructed to test how the measures handle partial mismatches in the control flow and mismatches in the stochastic perspective. SN_1 represents the target model that best represents the log without overfitting it. SN_2 has the same control flow structure as SN_1, but its weights are changed to make the branch containing activities b and c more frequently and the loop on e more likely. SN_3 represents the target model with an added activity f after a. SN_4 is the *trace model*, that represents the same stochastic language as the event log (by overfitting it). SN_5 is the trace model with extra activity f added after a. Model



(a) SN_1 and (SN_2) (b) SN_3

(c) SN_4 (without f) and SN_5 (with f)

(d) SN_6 and SN_7 (without d) (e) SN_8

Fig. 9 8 synthetic stochastic models. Unless marked, transition weights = 1. This dataset has been adapted from Leemans and Polyvyanyy (2020)

SN_6 is the flower model obtained by assigning the frequency of each transition as transition weight. SN_7 equals SN_6 with activity d removed and SN_8 equals SN_6 with activity f added at the start of each trace. It is not possible to provide a reference rank of all models, but intuitively we expect to observe the following phenomena:

Obs. 1 $conf(SN_4) = 1.0$ for all conformance measures. (Perfect conformance)

Obs. 2. SN_1 closely resembles the underlying process and should have the second-highest conformance value, with $conf(SN_1)$ close to 1.0.

Obs. 3. SN_8 adds an extra unmapped activity f to SN_6 without modifying the underlying distribution. Therefore, we expect $conf(SN_6) > conf(SN_8)$.

Obs. 4. SN_3 and SN_5 have a high conformance rate since all log traces only deviate from them in activity f.

Obs. 5. SN_2's probability distribution significantly differs from the log. Therefore, we expect $conf(SN_1) > conf(SN_3) > conf(SN_2)$.

Obs. 6. The flower models are the least conforming and should have a comparatively lower conformance.

Table 2 Measure value (model ranking) for log L_1. Mismatches with respect to tEMSC's rankings are

	tEMSC				uEMSC			
	orig	m^2	m^3	m^4	orig	m^2	m^3	m^4
SN_1	0.97 (2)	0.95 (2)	0.94 (2)	0.93 (2)	0.95 (2)	0.94 (2)	0.92 (2)	0.90 (2)
SN_2	0.58 (5)	0.80 (3)	0.76 (3)	0.79 (3)	0.37 (3)	0.68 (3)	0.57 (3)	0.54 (3)
SN_3	0.73 (4)	0.77 (4)	0.72 (4)	0.70 (4)	0.00 (8)	0.64 (4)	0.35 (4)	0.27 (4)
SN_4	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)
SN_5	0.74 (3)	0.76 (5)	0.70 (5)	0.67 (5)	0.00 (8)	0.63 (5)	0.31 (5)	0.22 (5)
SN_6	0.42 (6)	0.66 (6)	0.59 (6)	0.55 (6)	0.017 (4)	0.32 (6)	0.09 (6)	0.02 (6)
SN_7	0.28 (8)	0.610 (8)	0.52 (8)	0.47 (8)	0.001 (5)	0.221 (8)	0.03 (8)	0.005 (8)
SN_8	0.37 (7)	0.611 (7)	0.54 (7)	0.51 (7)	0.00 (8)	0.222 (7)	0.05 (7)	0.01 (7)

Table 2 shows the measure values for each log-model combination. We notice that **Obs. 1**, **Obs. 2**, and **Obs. 3** are satisfied by all measures. We see that **Obs. 4** is only satisfied for the tEMSC measures (original and m^k-based). Meanwhile, the *uEMSC* measure assigns these models a conformance of zero because the addition of f makes all log traces unfitting. This highlights the issue of the unit-cost function for partially matching traces. The severity of this issue can be seen in the example of model SN_5. While *tEMSC* ranks this model as the third best model, *uEMSC* assigns it a conformance of 0. For the $m^k - uEMSC$ measures, **Obs. 4** holds for low values of k. While for increasing values of k, the measure converges to the *uEMSC*.

We see that **Obs. 5** is only satisfied by the original *tEMSC* measure. However, we see that the $m^k - tEMSC$ assigns close conformance values for SN_2 and SN_3, while the differences between the measure values are comparably high for the *uEMSC* measures (original and m^k-based). Finally, **Obs. 6** is also satisfied by all measures. However, the *uEMSC* measures tend to be too strict and assign conformance values close to zero, while the *tEMSC* measures assign higher values, with $m^k - tEMSC$ being the most permissive of all measures.

Real-world datasets

We now assess the performance and behavior of the proposed measures using real-world event logs. We begin by examining the computational feasibility of each measure. Table 3 provides a summary of failure rates (timeouts and out-of-memories) for the m^k measures. We categorize when each situation occurred according to how other measures coped with the dataset. A measure fails if it times out or goes out of memory. Sometimes, tEMSC can be computed, but with a non-representative sampling of the model behavior (we assume this to be the case if less than 50% of the model's probability is sampled). The uEMSC can be computed for all but one model. However, its value is oftentimes too close to zero. We consider models for which the uEMSC measure value is less than 10^{-4} to be affected by issues with partial mismatches.

Overall, we see that the m^k measures scale to the real-world models. Except for the m^4-EMSC measure, which fails even more frequently than tEMSC. These failures are mostly caused by out of memories when trying to construct the reallocation matrix between two very large abstractions. The m^4-uEMSC goes out of memory for 5 models (groups 3. and 4.). This can be explained by the method being implemented in Python, where small strings consume orders of magnitude more space than in compiled languages. We believe this problem would not happen if the method was implemented in a

Table 3 Number of failed computations for the m^k measures, grouped by how the reference measures coped with the same model, including whether tEMSC sampled sufficient behavior (p< 50%) and uEMSC returned a negligible value (< 10^{-4})

Group	tEMSC		uEMSC	Σ	m^k-EMSC failed			m^k-uEMSC failed		
	failed	p<50%	<10^{-4}		$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
1.	Y	-	Y	7	0	0	0	0	0	0
2.	Y	-	N	6	0	0	1	0	0	0
3.	N	Y	Y	14	0	0	6	0	0	4
4.	N	Y	N	16	0	0	4	0	0	1
5.	N	N	Y	32	0	0	3	0	0	0
6.	N	N	N	109	0	2	12	0	0	0
All	-	-	-	184	0	2	26	0	0	5

compiled language. Groups 1. and 3. are particularly interesting. These contain 21 models for which uEMSC is negligible and tEMSC either failed, or did not sample representative behavior. For these 21 models, the m^k measures are arguably the only measures that could process them.

Figure 10 displays the distribution of computation times for each measure. This analysis includes only experiments that either completed successfully or timed out, excluding those that failed due to out-of-memory errors.

The first thing to notice is that tEMSC and uEMSC are respectively the slowest and fastest measures to compute, with a difference of two orders of magnitude in their median computation times. While the m^k measures fall between these extremes.

Comparing the m^k measures, we see that the m^k-uEMSC measures demonstrate better scalability than their m^k-EMSC counterparts. They exhibit lower median computation times and fewer outliers in their runtime distributions, particularly when $k=4$. For smaller values of k ($k=2,3$), the runtime of the m^k measures is primarily dominated by the linear pass over the event log. However, for larger values of k, the computation of each subtrace probability and the solution to the optimal transport problem (in the case of m^k-EMSC) become the main factors influencing runtime.

Finally, it is important to note the difference in implementation languages: the m^k measures are partially implemented in Python, while the uEMSC measure is fully implemented in Java. We believe that implementing the m^k measures in a compiled language could close the performance gap between these two approaches.

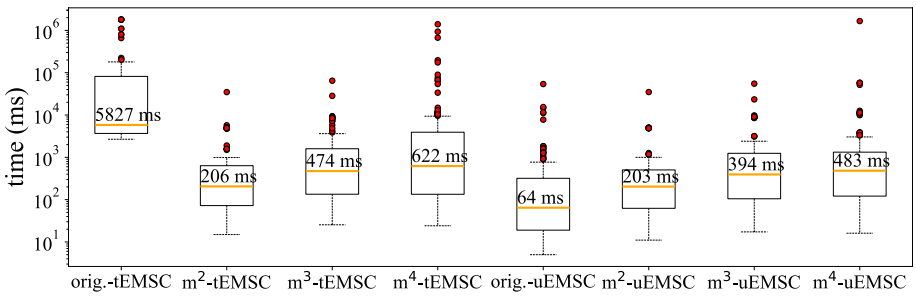
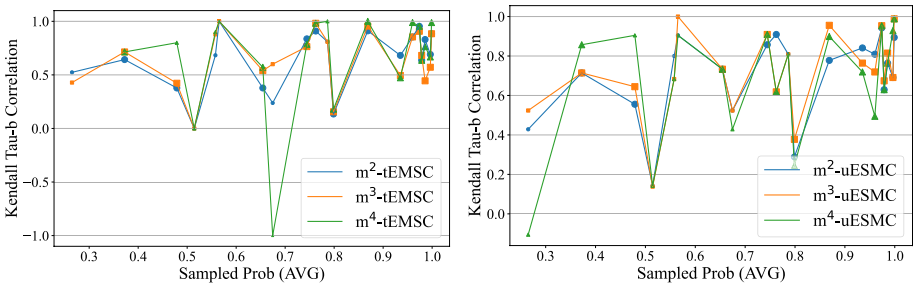


Fig. 10 Runtime distribution for each measure. The median is indicated in orange



(a) Correlation for m^k-EMSC measures (b) Correlation for m^k-uEMSC measures

Fig. 11 Kendall Tau-b rank correlation between model rankings produced by the m^k measures and the tEMSC baseline, plotted against tEMSC's average sampling probability. Larger points indicate correlations calculated from a higher number of successful computations (fewer timeouts/errors)

Next, we evaluate the rankings produced by the derived measures. For each event log, we examine the rankings of its associated process models induced by each measure. We then compare these rankings against the ranking produced by tEMSC, which we consider to provide the reference rankings, particularly when it can sample a sufficient amount of behavior from the models. The comparison uses the Kendall Tau-b rank correlation coefficient, which is computed based on the relative number of swaps on the orderings of both rankings. A coefficient of 1 indicates perfect agreement, -1 indicates perfect disagreement (inverse ranking), and 0 indicates no correlation.

Figure 11 shows the correlation between the rankings produced by the m^k measures and tEMSC. Since tEMSC's reliability depends on sampling enough behavior, and sampling for some measures timed out, our analysis considers these factors by plotting the correlation coefficient as a function of tEMSC's average sampled probability mass in Fig. 11. Furthermore, since many experiments went out-of-memory, we adjust the size of the points on the plot to indicate the proportion of successful computations. The correlation coefficient is computed based on the successful computations. Smaller points represent cases where few entries were used to compute the coefficient, suggesting that these values are less significant.

The m^k-uEMSC and m^k-EMSC measures show strong positive correlations with tEMSC, often exceeding 0.6. This positive correlation is particularly evident when enough behavior can be sampled and enough computations succeed (larger dots). For m^4-EMSC and $m4$-uEMSC, two points with negative correlation stand out. These correspond to the BPI 2015 (subprocess 8) and BPI 2014 datasets, respectively. For these specific cases, only 5 and 2 (out of 10) models yielded successful computations, which indicates that the correlation coefficient is less reliable.

We further evaluate the measures based on their ability to rank different weight estimators according to an expected, albeit idealized, order of conformance. Given a fixed control-flow model, we generally expect the alignment-based estimator (ABE) to produce the most conforming stochastic model, followed by the Bill-Clinton estimator (BCE) if available, and finally the frequency-based estimator (FBE) producing the least conforming model. It is important to notice that this idealized ordering may not hold in all situations, as various factors can influence an estimator's performance.

Figure 12 visualizes the rankings assigned to the estimators (ABE, BCE, FBE) by each measure for each of the 74 control-flow models evaluated. In the figure, for each control-flow model (represented in the Y-axis), we order the estimators according to their relative rankings from the most to the least conforming.

The results show that tEMSC and the m^k measures consistently rank ABE as the best estimator. Most of these measures assigned ABE the top rank at leat 65 out of 74 cases (the exception being m^4-tEMSC, for which many experiments failed). For the m^k measures, most deviations from the expected ranking occurred with DFM and FM models. This might be explained by the fact that, for these model types, the ABE and FBE estimators produce very similar abstractions, that only differ in the relative frequency of their suffixes/prefixes. Consequently, the m^k measures assign very close scores to models derived using these two estimators. In contrast, the uEMSC measure frequently ranked BCE or FBE highest, especially for SM and FM models.

Regarding the FBE estimator, while tEMSC and the m^k measures generally rank it as the least conforming, the results are less consistent than for ABE. Notably, there are control flow models where the measures significantly disagree on the specific ranking assigned to FBE.

In summary, the experiments on real-world datasets demonstrate that the stochastic Markovian abstraction helps alleviate issues related to sampling and partial mismatches of existing stochastic measures. The derived m^k measures were shown to produce model rankings that are consistent with state-of-the-art techniques and (guarded some limitations) with the expected ranking of estimators. However, for larger values of k, we found that the m^k measures shared the same limitations as their base measures: m^4-EMSC became very slow, and m^4-uEMSC less robust to partial mismatches. Therefore, in real-world settings, users must choose the measure, along with an appropriate value of k, that best balances speed and reliability for their specific needs.

Stochastic conformance diagnostics

So far, we have only considered the computation of a single number, which allows us to assess the degree of conformance. However, oftentimes users would like to obtain diagnostics on which parts of the process are non-conforming. This section discusses how to provide diagnostics for the derived m^k measures. This can be achieved by comparing the (finite) stochastic languages of both m^ks (similar to the log-log comparison presented in

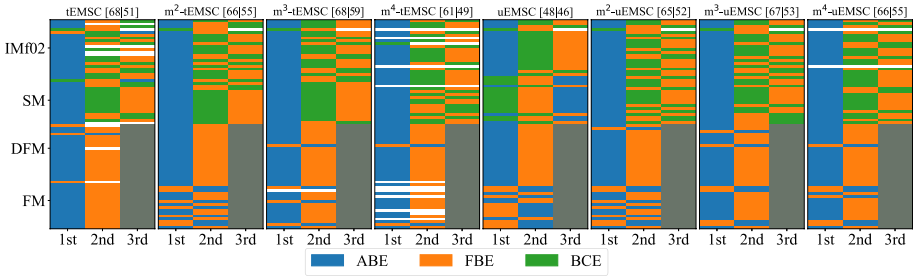


Fig. 12 Estimator rankings induced by each measure for each of the 74 control-flow models. White spaces represent timeouts/out-of-memory. Gray boxes correspond to FM/DFM models with BCE estimator, which are discarded. Subplots are annotated with the number of times where the measure assigned ABE as the most conforming estimator and FBE as the least conforming estimator

Leemans et al. (2021)). The results of the subtrace comparison can be projected onto the event log/the process model to obtain similar log-model diagnostics as in Leemans et al. (2021). We demonstrate the diagnostics based on the Italian Road Fines event log, which is compared with its RF-IMf02-ABE stochastic model. The choice of parameter k affects the quality of the analysis and must be chosen carefully by the user. As an illustrative example, we consider $k = 3$. Throughout the analysis, we discuss the effects of varying this parameter.

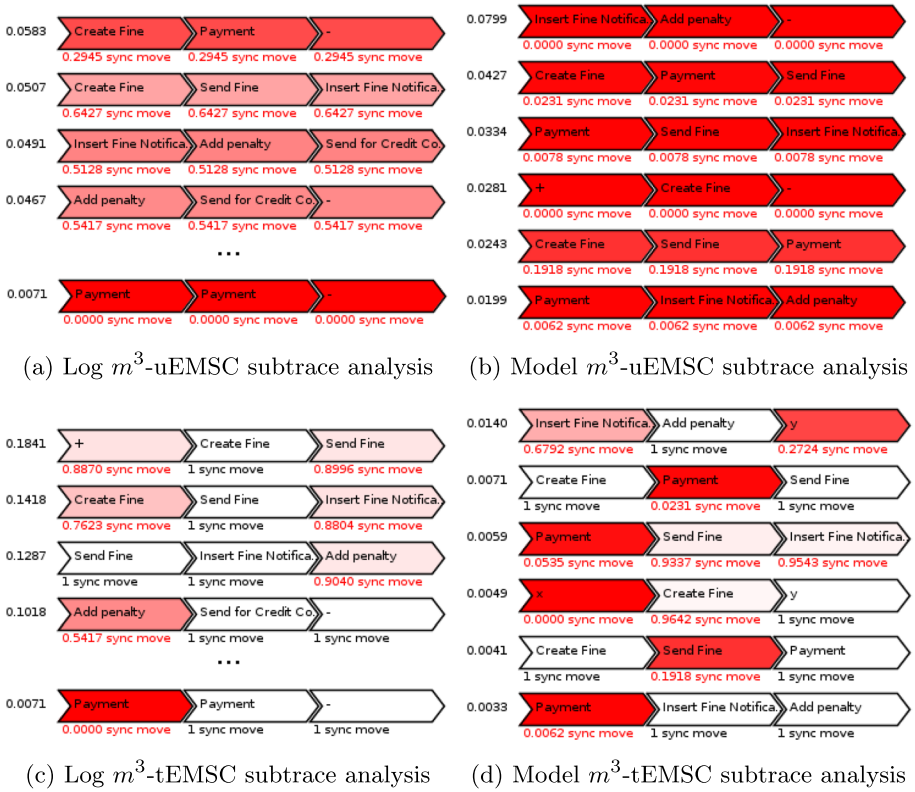
Subtrace analysis

The diagnostics method presented in Leemans et al. (2021) consists of projecting the solution of the optimal transport problem onto both languages using the resulting stochastic trace alignments.[6] Given a stochastic language l, we obtain an *event synchronous-likelihood* (Leemans et al. 2021) function $P^l_{sync} : \Sigma^* \times \mathbb{N} \to [0, 1]$ where $P^l_{sync}(\gamma, i)$ returns the likelihood of the $i - th$ event of trace $\gamma \in \Sigma^*$ synchronizing in the stochastic trace alignments (the function equals 0 if $\gamma \notin supp(l)$ and undefined if $i > |\gamma|$). Function P^l_{sync} can be visualized on top of the language's trace variants as done in Leemans et al. (2021). Figure 13 shows snippets of the visualization when applied to the log and the model's m^3 abstractions:

Despite the rigidity of the unit-cost function, it is still possible to derive insights from the m^3-uEMSC projections (Fig. 13a and b). The first thing that stands out in the visualization is that the moves of some subtraces are assigned a synchronous move probability of 0.0 such as $\langle$*Payment*, *Payment*, $-\rangle$ in the log projection and $\langle$*Insert Fine Notification*, *Add Penalty*, $-\rangle$ in the model projection. This indicates a mismatch in the control flow, i.e. subtrace $\langle$*Payment*, *Payment*, $-\rangle$ is not present in the model and $\langle$*Insert Fine Notification*, *Add Penalty*, $-\rangle$ is not present in the log. It is also possible to identify mismatches in the stochastic perspective. For example, the subtrace $\langle$*Create Fine*, *Payment*, $-\rangle$ is identified as the most prominent issue in the log, i.e. it happens more frequently in the log than the model allows. Similarly, the sequence $\langle$*Create Fine*, *Payment*, *Send Fine*$\rangle$ is identified as the most prominent issue in the model, i.e. the model prescribes that it should happen more frequently than was observed in the log.

A similar analysis can be performed for m^3-tEMSC measure (Fig. 13c and d), whereas using the normalized Levenshtein cost function leads to more precise diagnostics. For example, we can infer from the log move in $\langle$*Payment*, *Payment*, $-\rangle$ in Fig. 13c that *Payment* should not be executed a second time before finishing. Similarly, we can infer from the model move in $\langle$*Create Fine*, *Payment*, *Send Fine*$\rangle$ in Fig. 13d that *Payment* is oftentimes skipped between *Create Fine* and *Send Fine*. Some subtraces are more difficult to interpret. For example, the model move on $+$ in subtrace $\langle +$, *Create Fine*, $-\rangle$ in Fig. 13d indicates that the subtrace is aligned to other subtraces where *Create Fine* is not the first activity. However, in the event log, all traces begin with *Create Fine*. This discrepancy suggests that traces in the model

[6] Notice that the work in Leemans et al. (2021) does not explicitly discuss a diagnostics method tailored for the uEMSC measure. Nevertheless, it is possible to extrapolate the diagnostic method presented for the tEMSC measure to the uEMSC measure by considering the alignment between mismatching traces to be the "trivial alignment" that contains no synchronous moves.



(a) Log m^3-uEMSC subtrace analysis (b) Model m^3-uEMSC subtrace analysis

(c) Log m^3-tEMSC subtrace analysis (d) Model m^3-tEMSC subtrace analysis

Fig. 13 Subtrace analysis for m^3-EMSC measures for the RF-IMf02-ABE dataset. Subtraces are annotated on the left with their absolute contribution to non-conformance. Red displays the probability that an event in the subtrace does not synchronize

might actually be shorter, causing the start marker (+) to appear relatively more frequently in the model.

In summary, we show how the subtrace analysis can provide diagnostics on the nature of the deviations. As k increases, the abstractions converges to the log/model's stochastic languages and the analysis becomes similar to Leemans et al. (2021), including the same runtime and robustness issues. For low values of k, the subtraces' short context window might limit the insights to be gained. In the next section, we show how to contextualize the deviations by projecting the subtraces onto the log/model.

Log and model projections

Given an event log and a process model, we want to compute functions $P_{sync}(\sigma, i)$ and $P_{sync}(t)$ indicating respectively the probability of an event/a transition in the log/model synchronizing. This can be computed by projecting the synchronization probabilities of subtrace activities ("Subtrace analysis" section) to events/transitions in the log/model that contribute to the subtrace. The definitions below formalize this concept:

Definition 15 (m^k Log Projection) Let Σ be an alphabet, $l : \Sigma^* \to [0, 1]$ be a stochastic language, $k \geq 2$. Let $P^{m^k_l}_{sync} : \Sigma^* \times \mathbb{N} \to [0, 1]$ be m^k_l's event synchronous-likelihood (Leemans et al. 2021). Then:

$$P^{l}_{sync}(\sigma, n) = \frac{\sum_{i,j \in \Gamma_{\sigma,n}} \left(P^{m^k_l}_{sync}(\sigma^{i \to j}, n - i + 1) \cdot m^k_l(\sigma^{i \to j}) \right)}{\sum_{i,j \in \Gamma_{\sigma,n}} m^k_l(\sigma^{i \to j})}$$

Where $\Gamma_{\sigma,n} = \{(i,j) \mid \sigma^{i \to j} \in M^k_\sigma \ \wedge \ i \leq n \leq j\}$ the set of subtrace indexes of σ containing event $\sigma[n]$.

Definition 15 adds up the synchronous probability of all subtraces γ containing event $\sigma[n]$, weighted by $m^k_l(\gamma)$. Notice that as $k \to \infty$, then $\Gamma_{\sigma,n} = \{(1, |\sigma|)\}$ and $P^{l}_{sync}(\sigma, n) = P^{m^k_l}_{sync}(\sigma, n)$; i.e. as m^k_l approaches l, then Definition 15 converges to the subtrace projection. For the model projection, we first assume that the Petri net is uniquely-labeled.

Definition 16 (m^k Model Projection for Uniquely-Labeled Models) Let Σ be an alphabet, $SN = (P, T, F, \Sigma, \lambda, M_0, w)$ a bounded livelock-free SLPN. Let SN be uniquely-labeled, i.e. λ^{-1} is defined for visible transitions. And let $l : \Sigma^* \to [0,1]$ be its generated stochastic language. Let $k \geq 2$, and $P^{m^k_l}_{sync} : \Sigma^* \times \mathbb{N} \to [0,1]$. Then:

$$P^{l}_{sync}(t) = \frac{\sum_{\gamma \in supp(m^k_l)} \left(m^k_l(\gamma) \sum_{1 \leq i \leq |\gamma|} \mathbf{1}_{\{t\}}(\lambda^{-1}(\gamma[i])) P^{m^k_l}_{sync}(\gamma, i) \right)}{\sum_{\gamma \in supp(m^k_l)} \left(m^k_l(\gamma) \sum_{1 \leq i \leq |\gamma|} \mathbf{1}_{\{t\}}(\lambda^{-1}(\gamma[i])) \right)}$$

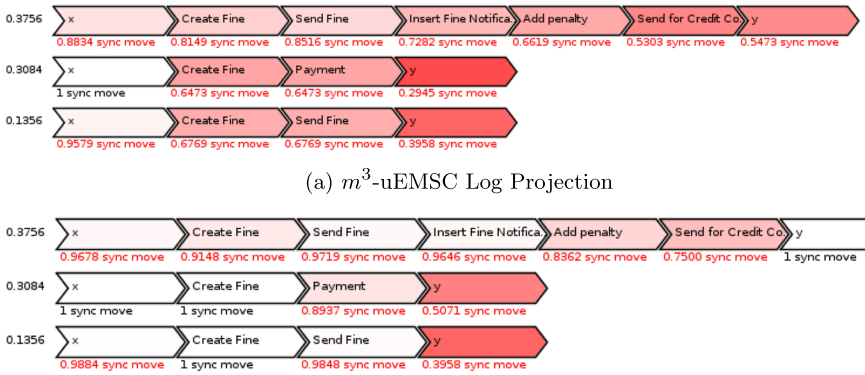
Definition 16 averages the synchronization likelihood of each transition, weighted by how often each transition happens in the abstraction. If SN is not uniquely labeled, the numerator must be adjusted to consider the relative likelihood of all subpaths ρ that contribute to subtrace γ.

Figure 14 shows the log and model projections for the m^3-uEMSC (Fig. 14a and c) and m^3-tEMSC measures (Fig. 14b and d). The first thing to notice is that both measures agree on the regions of the log responsible for the deviations. As expected, the m^3-tEMSC projection offers more localized diagnostics, while the m^3-uEMSC projection spreads the diagnostics over a larger region. Both diagnostics hint at mismatches in the stochastic perspective. When examining the deviation for the most frequent trace, we see that *Send for Credit Collection* has a particularly high occurence of log moves, indicating that fines should be sent for credit collection less frequently after a penalty is added. In comparison, the 2nd and 3rd traces present a high occurrence of log moves in the end marker (−), which indicates that the model expects further activities after *Payment* and *Send Fine*.

For the model projections, we cannot derive as many insights. The m^3-uEMSC model projection (Fig. 14c) colors almost all transitions with the darkest shade of red, indicating total non-conformance. We believe that this can be attributed to the model being imprecise from a control-flow perspective. Essentially, the model contains too many subtraces that are not present in the log (this can also be observed in Fig. 13b), which skews the analysis. Therefore, the model projection cannot be used to diagnose stochastic deviations in this situation and one must first adjust the model's control flow to make it more precise.

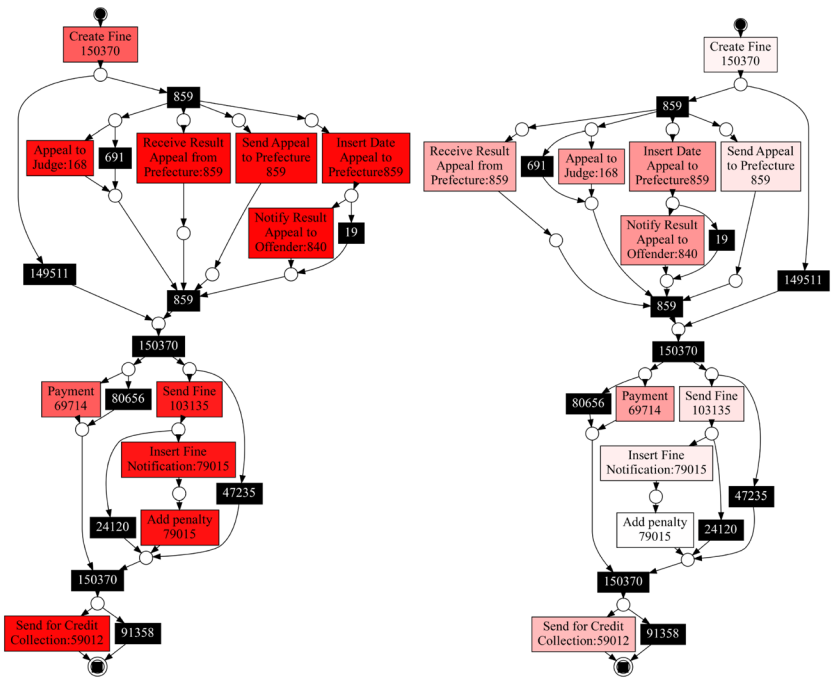
In contrast, the m^3-tEMSC model projection (Fig. 14d) is more robust to mismatches and is able to highlight the regions of the model responsible for the deviations. One can see that activities *Appeal to Judge*, *Insert Date Appeal to Prefecture*, and *Notify Result Appeal to Offender* have a high rate of model moves. However, without inspecting the corresponding alignments, it is not possible to conclude whether this is caused by control-flow deviations or mismatches in the stochastic behavior.

In summary, this section shows how to derive stochastic conformance diagnostics using subtrace analysis and log/model projection. Although our analysis is limited to the m^k-uEMSC and m^k-tEMSC measures, we believe that a similar approach to piggybacking on the diagnostic technique of the original measure can be used to provide



(a) m^3-uEMSC Log Projection

(b) m^3-tEMSC Log Projection



(c) m^k-uEMSC Model Projection (d) m^k-tEMSC Model Projection

Fig. 14 Log and model projections for m^3-EMSC measures for the RF_IMf02_ABE dataset. Traces are annotated on the left with their probabilities. Petri net transitions are annotated with their weights. For the log and model projections, red displays the intensity of non-conformance (fully red means $P_{sync} = 0$)

diagnostics for other derived measures such as (Leemans and Polyvyanyy 2023; Alkhammash et al. 2022; Li et al. 2024).

Related work

In process mining, a wide range of non-stochastic conformance measures has been introduced (Adriansyah et al. 2011; Augusto et al. 2022; Leemans et al. 2016). This work builds upon the Markovian-based abstraction first presented in Augusto et al. (2022) and later redefined in Goulart Rocha and van der Aalst (2023). The original motivation for extending it to the stochastic perspective was to address the issue of vanishing precision observed in Goulart Rocha and van der Aalst (2023) as k increases. As shown in the "Quantitative evaluation" section, this might still happen for m^k-uEMSC, albeit in a more attenuated form, and is now linked to partial mismatches in the subtraces.

This same abstraction has also been informally introduced in other works. In Chapela-Campa et al. (2023), the abstraction is used to measure the quality of simulation models. The abstractions are compared using the unit-cost Earth Mover's distance. In Burke et al. (2024), sets of subtraces are compared using the Gower's similarity. Both works rely on sampling traces from the process model. Our work differs in that we provide a formal definition of the abstraction, discuss its properties, and most importantly present an exact computation for stochastic languages with infinite support, as commonly the case in process mining.

Projecting on smaller subsets of activities is a common approach in process mining to deal with partial mismatches. In Brockhoff et al. (2024), an approach for event log comparison based on activity projection is introduced. It differs from the Markovian abstraction in that it considers non-consecutive sequences of activities, which is a similar idea as introduced in Leemans et al. (2016). Activity projections are naturally better at capturing violations in long-term dependencies. It is an interesting line of future work to adapt the method from Leemans et al. (2016) to a stochastic setting.

As far as we are aware, the EMSC measure (Leemans et al. 2021) was the first conformance checking technique to consider stochastic process models. Later, an exact method to compute it for the unit-cost distance by exploiting the link between SLPNs and absorbing Markov chains was presented in Leemans et al. (2024b). In Brockhoff et al. (2020), the EMSC measure is extended to consider the time perspective in the task of concept-drift detection.

Multiple stochastic conformance measures are based on the notion of entropy (Leemans and Polyvyanyy 2023; Alkhammash et al. 2022; Li et al. 2024). In Leemans and Polyvyanyy (2023), stochastic precision and recall measures based on the notion of projection and gain entropy are introduced. While the authors show that these measures satisfy a series of quality axioms, they suffer from two main drawbacks: first, the measures are not robust to partial mismatches between both languages. Second, they can only be computed for stochastic deterministic regular languages. Also, the projection precision and recall do not properly consider the stochastic information of both artifacts. If the log and model share the same support language, then the projection entropy (Leemans and Polyvyanyy 2023) returns fitness and precision of 1, even if their probability distributions differ. The entropic relevant (Alkhammash et al. 2022) is another entropy-based conformance measure. In essence, it measures the expected number of bits needed to encode the log using the model. The closer the log and model's distributions,

the lower the description. However, the entropic relevance measure does not consider model traces that are not in the log. Recent work proposed using the Jensen-Shannon Li et al. (2024) measure for stochastic conformance checking. The measure uses the average language between two stochastic languages and thus successfully considers both object's perspectives. However, the measure is also not robust to partial mismatches and requires sampling for model-model comparisons.

Last, probabilistic trace alignments have been proposed in Bergami et al. (2021) as an extension to traditional trace alignments that balances the probability of an alignment and its cost. The method returns the most likely traces within a neighborhood of the investigated trace. By design, this technique is robust to partial mismatches. However, it requires sampling the underlying process model, a procedure that, as previously discussed, does not scale with increasing model complexity. Moreover, the technique focuses on generating conformance diagnostics, and no conformance measure based on probabilistic alignments is proposed.

Finally, there exist methods to compute substring probabilities given a stochastic regular (Fred 2000) or context-free (Corazza et al. 1992) grammar. In comparison, we provide a method for computing substring probabilities from an SNFA. We believe it is possible to convert the τ-free SNFA into a stochastic regular grammar and use the method in Fred (2000) to extract the subtraces. The method in Fred (2000) might be more efficient for very dense SNFAs, while our method is expected to be more efficient for sparse SNFAs.

Conclusions, discussion and future work

This work extends (Rocha et al. 2024) by evaluating the expressive power of Markovian abstraction, introducing and evaluating new derived measures (m^k-tEMSC), and presenting conformance diagnostics methods based on this abstraction. The experiments in the "Quantitative evaluation" section demonstrate that the abstraction can suitably approximate classes of stochastic process models commonly used in process mining. For event logs, the abstraction may not always be suitable due to long-term dependencies. Nevertheless, when used to measure the conformance between two objects and generate diagnostics, the abstraction mitigates robustness and scalability issues in existing measures while still producing model rankings that align with a reference measure.

The "Stochastic conformance diagnostics" section demonstrates how the proposed abstraction can be combined with existing stochastic conformance diagnostics methods, either directly through subtrace comparison ("Subtrace analysis" section), or indirectly via log/model projections ("Log and model projections" section). A key limitation is that interpreting the resulting diagnostics requires in-depth understanding of the underlying conformance checking technique. This limitation is inherited from the base technique (Leemans et al. (2021)) and is not exacerbated by using the abstraction. This issue reflects a broader challenge of understandability present in state-of-the-art conformance diagnostics methods. It is an interesting direction of future research to develop newer, more easily understandable methods. However, this is not so simple since in stochastic conformance checking, one must not only consider the non-determinism of replay methods (e.g., trace alignments) but also the non-locality of the firing rule of SLPNs, which inevitably complicates the interpretation of results.

Threats to validity

The process models considered in the "Quantitative evaluation" section do not contain long-term dependencies. This affects the generalizability of the results to broader classes of models because the Markovian abstraction is vulnerable to long-term dependencies. Our experiments were limited by a lack of process discovery methods that can generate such models as they are not as widely studied in process mining. State-of-the-art control-flow discovery methods that produce long-term dependency models often produced unsound or unrealistically complex models, or do not scale for large event logs.

In the "Real-world datasets" section, the derived measures are validated based on how their induced model rankings compare with a reference measure. Nevertheless, this test might not be challenging enough, since even seemingly trivial measures such as the uEMSC seem to pass it. Therefore, we also included a controlled experiment ("Synthetic datasets" section) to investigate whether the derived measures match one's intuition of a "good" measure.

Future work

In future work, we would like to refine the diagnostic techniques presented in the "Stochastic conformance diagnostics" section to better suit the stochastic Markovian abstraction. One approach is to develop adaptive values of k to mitigate the imprecision of the m^k-uEMSC diagnostics. If we can identify a deviation in shorter subtraces, then we do not need to consider deviations in longer subtraces containing them. This might also help improve the usability of the technique. With an adaptive method, users can set the parameter k to be as high as their runtime budget allows. Another promising direction is to separate the control flow and pure stochastic diagnostics. Control flow deviations can be effectively analyzed using non-stochastic techniques. Therefore, stochastic conformance diagnostics should focus on issues related to the stochastic perspective.

On the theoretical side, we aim to expand the class of models for which this abstraction can be computed, which may lead to new insights into stochastic conformance checking methods. Additionally, we plan to investigate efficient computation techniques for restricted classes of models, such as stochastic process trees (Burke et al. 2024) and Petri nets without silent loops (Bergami et al. 2021).

Last, there is potential for cross-fertilization with fields that have also studied n-gram models such as natural language processing and pattern recognition. For instance, smoothing techniques (Ney et al. 1994; Brants et al. 2007) may inspire the development of adaptive values of k. However, we notice that stochastic conformance checking focuses on language comparison and diagnostics, which differs from traditional language modeling tasks.

Appendix A Proofs of Lemmata 1 and 2

Lemma 1 (*Computing f_l^k of a Stochastic Regular Language l*) *Let Σ be an alphabet, $N = (Q, \Sigma, \delta, q_0, p_f)$ be a trimmed, consistent τ-free SNFA generating the stochastic language $l : \Sigma^* \to [0, 1]$, and $k \geq 2$. Let $\hat{N} = (\hat{Q}, \Sigma_\pm, \hat{\delta}, q_+, \hat{p}_f)$ be N's patched SNFA generating language $\hat{l}$. For $\gamma \in supp(f_l^k)$, it holds that*:

$$f_l^k(\gamma) = \sum_{q \in \hat{Q}} \mathbf{x}[q] \cdot \hat{\phi}(\gamma|q) \tag{7}$$

Where $\mathbf{x} = [x_{q_+}\ x_{q_0} \cdots\ x_{q_-}]$ *is the solution to the equation* $\mathbf{x}(I - \hat{\Delta}) = [1\,0 \cdots 0]$ ($\hat{\Delta}$ *is* $\hat{N}$*'s transition matrix) and* $\hat{\phi}(\gamma|q) = \sum_{\pi_\gamma \in \Theta_{\hat{N}}(q), \lambda(\pi_\gamma)=\gamma} \hat{\delta}(\pi_\gamma)$ *is the probability of generating the sequence* γ *starting from state* q *in* N.

Proof We first prove that $f_l^k(\gamma) = \sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta)$ for $\gamma \in supp(f_l^k)$. From Definition 12, $f_l^k(\gamma) = \sum_{\sigma \in \Sigma^*} l(\sigma) M_\sigma^k(\gamma)$. We distinguish between two cases:

Case 1 ($|\gamma| < k$) In this case, $M_\sigma^k(\gamma) = 0$, unless $\gamma = +\sigma-$, in which case $M_\sigma^k(\gamma) = 1$. Thus $f_l^k(\gamma) = \mathbb{P}_{\hat{N}}(\gamma)$. Furthermore, $|\gamma| < k$ implies $+, -$ in γ. Thus, $\mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = 0$ for $\alpha, \beta \neq \langle\rangle \Rightarrow \sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = \mathbb{P}_{\hat{N}}(\gamma)$.

Case 2 ($|\gamma| = k$) Define the boundary-affixes set $\mathcal{A}_\sigma(\gamma) = \{(\alpha, \beta) \mid \alpha, \beta \in \Sigma_\pm^*, +\sigma- = \alpha\gamma\beta\}$ which defines all prefixes/suffices in σ associated to each occurrence of γ in σ. Therefore, $M_\sigma^k(\gamma) = |\mathcal{A}_\sigma(\gamma)|$. Therefore:

$$f_l^k(\gamma) = \sum_{\sigma \in \Sigma^*} l(\sigma) M_\sigma^k(\gamma) = \sum_{\sigma \in \Sigma^*} \sum_{\alpha,\beta \in \mathcal{A}_\sigma(\gamma)} \mathbb{P}_{\hat{N}}(+\sigma-) = \sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta)$$

(To check the last step, expand $\mathcal{A}_\sigma(\gamma)$ and reorganize the terms).

Now, we show that $\sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = \sum_{q' \in \hat{Q}} x_q \cdot \hat{\phi}(\gamma|q')$. First, observe that since $\hat{N}$ is τ-free, then every path $\pi_{\alpha\gamma\beta}$ with $\lambda(\pi_{\alpha\gamma\beta}) = \alpha\gamma\beta$ can be *uniquely* divided into subpaths $\pi_{\alpha\gamma\beta} = \pi_\alpha \pi_\gamma \pi_\beta$ and $\lambda(\pi_\alpha) = \alpha$, $\lambda(\pi_\gamma) = \gamma$, and $\lambda(\pi_\beta) = \beta$. Therefore:

$$\sum_{\alpha,\beta \in \Sigma_\pm^*} \mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = \sum_{\substack{\pi_\alpha\pi_\gamma\pi_\beta \in \Theta_{\hat{N}}(q_+) \\ \lambda(\pi_\gamma) = \gamma}} P_{\hat{N}}(\pi_\alpha\pi_\gamma\pi_\beta) \tag{8}$$

The right side of Eq. 8 computes the sum of the probabilities of all passes over all subpaths labeled γ in the set of all accepting paths of $\hat{N}$. Now, we rewrite the term as:

$$= \sum_{\substack{\pi_\alpha\pi_\gamma \in \Theta_{\hat{N}}(q_+) \\ \lambda(\pi_\gamma) = \gamma}} \left(\hat{\delta}(\pi_\alpha\pi_\gamma) \sum_{\pi_\beta \in \Theta_{\hat{N}}(tgt(\pi_\alpha\pi_\gamma))} P_{\hat{N}}(\pi_\beta) \right) \tag{9}$$

$tgt(\pi_\alpha\pi_\gamma)$ is useful, therefore $\sum_{\pi_\beta \in \Theta_{\hat{N}}(tgt(\pi_\alpha\pi_\gamma))} P_{\hat{N}}(\pi_\beta) = 1$, so:

$$= \sum_{\substack{\pi_\alpha\pi_\gamma \in \Theta_{\hat{N}}(q_+) \\ \lambda(\pi_\gamma) = \gamma}} \hat{\delta}(\pi_\alpha\pi_\gamma) = \sum_{\pi_\alpha \in \Theta_{\hat{N}}(q_+)} \left(\hat{\delta}(\pi_\alpha) \cdot \sum_{\substack{\pi_\gamma \in \Theta_{\hat{N}}(tgt(\pi_\alpha)) \\ \lambda(\pi_\gamma) = \gamma}} \hat{\delta}(\pi_\gamma) \right) \tag{10}$$

$$= \sum_{q \in \hat{Q}} \sum_{\pi_\alpha \in \Theta_{\hat{N}}(q_+, q)} \left(\hat{\delta}(\pi_\alpha) \cdot \hat{\phi}(\gamma|q) \right) \tag{11}$$

Now define $\psi(q_i, q_j, n) = \sum_{\pi \in \Theta_{\hat{N}}(q_i, q_j, n)} \hat{\delta}(\pi)$. Then $\sum_{\pi_\alpha \in \Theta_{\hat{N}}(q_+, q)} \hat{\delta}(\pi_\alpha) = \sum_{n=0}^{\infty} \psi(q_+, q, n)$. The following relation holds:

$$\psi(q_i, q_j, n) = \begin{cases} \mathbf{1}_{\{q_i\}}(q_j) & n = 0 \\ \sum_{q_k \in \hat{Q}} \psi(q_i, q_k, n-1) \hat{\Delta}[k, j] & n > 0 \end{cases}$$

In matrix form, $\sum_{n=0}^{\infty} \psi(q+, q, n) = (I + \hat{\Delta} + \hat{\Delta}^2 + \cdots)_{q_+, q} = (I - \hat{\Delta})^{-1}[q_+, q] = \mathbf{x}[q]$ (Grinstead and Snell (2003)). □

Lemma 2 (A Necessary and Sufficient Condition for Well-Defined f_l^k) *Given a stochastic language* $l : \Sigma^* \to [0, 1]$. *Then for any* $k \geq 2$

$$f_l^k(\gamma) < \infty \; \forall \gamma \in \Sigma_{\pm}^{\leq k} \iff \sum_{\sigma \in \Sigma^*} l(\sigma)|\sigma| < \infty \tag{12}$$

Proof We first show that $|\sigma|/k \leq |S_\sigma^k| \leq |\sigma| + 1$ for every k by using induction on $|\sigma|$. The induction's base-case ($|\sigma| \leq k$) can be easily verified. For every $\sigma \in \Sigma^*$, $|\sigma| > k$, $S_\sigma^k = \{\sigma^{1 \to k}\} \uplus S_{\sigma^{2 \to |\sigma|}}^k$. So $|S_\sigma^k| = 1 + |S_{\sigma^{2 \to |\sigma|}}^k| \geq 1 + \frac{|\sigma|-1}{k} \geq |\sigma|/k$ and $|S_\sigma^k| = 1 + |S_{\sigma^{2 \to |\sigma|}}^k| \leq 1 + (|\sigma^{2 \to |\sigma|}| + 1) \leq 1 + ((|\sigma| - 1) + 1) = 1 + |\sigma|$.

($\Rightarrow$) Since $\Sigma_{\pm}^{\leq k}$ is finite, then $f_l^k(\gamma) < \infty \; \forall \gamma$ implies $\sum_{\gamma \in \Sigma_{\pm}^{\leq k}} f_l^k(\gamma) < \infty$. Furthermore, from Definition 11, observe that $supp(M_\sigma^k) \subseteq \Sigma_{\pm}^{\leq k}$. Then:

$$\sum_{\gamma \in \Sigma_{\pm}^{\leq k}} f_l^k(\gamma) = \sum_{\gamma \in \Sigma_{\pm}^{\leq k}} \left(l(\sigma) \sum_{\sigma \in \Sigma^*} M_\sigma^k(\gamma) \right) = \sum_{\sigma \in \Sigma^*} \left(l(\sigma) \sum_{\gamma \in \Sigma_{\pm}^{\leq k}} M_\sigma^k(\gamma) \right) = \sum_{\sigma \in \Sigma^*} l(\sigma)|M_\sigma^k| < \infty$$

(the sums can be swapped since the sum is finite). Since $|M_\sigma^k| = |S_{+\sigma-}^k| \geq \frac{|\sigma|}{k}$, then:

$$\sum_{\sigma \in \Sigma^*} \frac{l(\sigma)|\sigma|}{k} < \infty \Rightarrow \sum_{\sigma \in \Sigma^*} l(\sigma)|\sigma| < \infty$$

($\Leftarrow$) From $|S_\sigma^k| \leq |\sigma| + 1$ it follows $|M_\sigma^k| = |S_{+\sigma-}^k| \leq |\sigma| + 3$. Thus:

$$\begin{aligned} f_l^k(\gamma) &= \sum_{\sigma \in \Sigma^*} l(\sigma) M_\sigma^k(\gamma) \leq \sum_{\sigma \in \Sigma^*} l(\sigma)(|\sigma| + 3) \\ &= \sum_{\sigma \in \Sigma^*} l(\sigma)|\sigma| + 3 \cdot \sum_{\sigma \in \Sigma^*} l(\sigma) = \sum_{\sigma \in \Sigma^*} l(\sigma)|\sigma| + 3 \end{aligned}$$

Which is $< \infty$ if $\sum_{\sigma \in \Sigma^*} l(\sigma)|\sigma| < \infty$. □

Acknowledgements
We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Special thanks also go to Tobias Brockhoff for the insightful discussions, which helped improve this paper.

Authors'contributions
E.G.R.- Conceptualization, Methodology, Software, Validation, Investigation, Writing- Original Draft S.J.J.L.- Conceptualization, Methodology, Software, Validation, Writing- Review & Editing W.M.P.vdA.- Conceptualization, Methodology, Validation, Writing- Review & Editing, Supervision.

Funding
This work was partially supported by the Alexander von Humboldt (AvH) Stiftung. Eduardo is fully funded by Celonis.

Data availability
Data is provided within the manuscript or supplementary information files.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 19 December 2024 Accepted: 27 April 2025
Published online: 11 June 2025

References

Adriansyah A, van Dongen B, van der Aalst WMP (2011) Conformance checking using cost-based fitness analysis. In: 15th International Enterprise Distributed Object Computing Conference, pp 55–64. https://doi.org/10.1109/EDOC.2011.12

Alkhammash H, Polyvyanyy A, Moffat A et al (2022) Entropic relevance: a mechanism for measuring stochastic process models discovered from event data. Inf Syst 107:101922. https://doi.org/10.1016/J.IS.2021.101922

Augusto A, Armas-Cervantes A, Conforti R, et al (2022) Measuring fitness and precision of automatically discovered process models: A principled and scalable approach. IEEE Trans Knowl Data Eng 34(4). https://doi.org/10.1109/TKDE.2020.3003258

Augusto A, Conforti R, Dumas M et al (2019) Split miner: automated discovery of accurate and simple business process models from event logs. Knowl Inf Syst 59(2):251–284. https://doi.org/10.1007/S10115-018-1214-X

Bergami G, Maggi FM, Montali M, et al (2021) Probabilistic trace alignment. In: 2021 3rd International Conference on Process Mining (ICPM), pp 9–16. https://doi.org/10.1109/ICPM53251.2021.9576856

Brants T, Popat AC, Xu P, et al (2007) Large language models in machine translation. In: Eisner J (ed) EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic. ACL, pp 858–867. https://aclanthology.org/D07-1090/

Brockhoff T, Uysal MS, van der Aalst WMP (2020) Time-aware concept drift detection using the earth mover's distance. In: van Dongen BF, van Dongen M and Wynn MT (eds) 2nd International Conference on Process Mining, {ICPM} 2020, Padua,Italy, October 4-9, 2020. IEEE, Padua, pp 33–40. https://doi.org/10.1109/ICPM49681.2020.00016

Brockhoff T, Uysal MS, van der Aalst WMP (2024) Process comparison based on selection-projection structures. In: Guizzardi G, Santoro FM, Mouratidis H, et al (eds) Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings, Lecture Notes in Computer Science, vol 14663. Springer, pp 20–35. https://doi.org/10.1007/978-3-031-61057-8_2

Burke A, Leemans SJJ, Wynn MT (2021) Stochastic process discovery by weight estimation. In: Leemans SJJ and Leopold H (eds) Process Mining Workshops - {ICPM} 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers. Lecture notes in business information processing, vol 406, Springer, Padua, pp 260–272. https://doi.org/10.1007/978-3-030-72693-5_20

Burke AT, Leemans SJJ, Wynn MT et al (2024) A chance for models to show their quality: Stochastic process model-log dimensions. Inf Syst 124:102382. https://doi.org/10.1016/J.IS.2024.102382

Chapela-Campa D, Benchekroun I, Baron O, et al (2023) Can I trust my simulation model? measuring the quality of business process simulation models. In: Di Francescomarino C, Burattin A, Janiesch C and Sadiq S (eds) Business Process Management - 21st International Conference, (BPM) 2023, Utrecht, The Netherlands, September 11-15, 2023, Proceedings. Lecture notes in computer science, vol 14159, Springer, Utrecht, pp 20–37. https://doi.org/10.1007/978-3-031-41620-0_2

Chu W, Bonsangue MM (2020) Learning probabilistic languages by k-testable machines. In: Aoki T, Li Q (eds) International Symposium on Theoretical Aspects of Software Engineering, TASE 2020, Hangzhou, China, December 11-13, 2020. IEEE, pp 129–136. https://doi.org/10.1109/TASE49443.2020.00026

Corazza A, de Mori R, Satta G (1992) Computation of upper-bounds for stochastic context-free languages. In: Swartout WR (ed) Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992. AAAI Press / The MIT Press, pp 344–349.

Corazzat A, Mori RD, Gretter R et al (1990) Computation of probabilities for island-driven parsers. Proc. ICSLP 1990:897–900

de Leoni M, Mannhardt F (2015). Road traffic fine management process. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

Dupont P, Denis F, Esposito Y (2005) Links between probabilistic automata and hidden markov models: Probability distributions, learning models and induction algorithms. Pattern Recognit 38:1349–1371. https://doi.org/10.1016/j.patcog.2004.03.020

Fred ALN (2000) Computation of substring probabilities in stochastic grammars. In: Grammatical Inference: Algorithms and Applications, 5th International Colloquium, ICGI 2000, pp 103–114. https://doi.org/10.1007/978-3-540-45257-7_9

García P, Vidal E (1990) Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Trans Pattern Anal Mach Intell 12(9):920–925. https://doi.org/10.1109/34.57687
Goulart Rocha E, van der Aalst WMP (2023) Polynomial-time conformance checking for process trees. In: Di Francescomarino C, Burattin A, Janiesch C and Sadiq S (eds) Business Process Management - 21st International Conference, (BPM) 2023, Utrecht, The Netherlands, September 11-15, 2023, Proceedings. Lecture notes in computer science, vol 14159, Springer, Utrecht, pp 109–125. https://doi.org/10.1007/978-3-031-41620-0_7
Grinstead CM, Snell JL (2003) Introduction to Probability. AMS
Hanneforth T, de la Higuera C (2010) Epsilon-removal by loop reduction for finite-state automata over complete semirings. Studia Grammatica. 72:297–312. https://api.semanticscholar.org/CorpusID:124584692
Khayatbashi S, Hartig O, Jalali A (2023) Bpi challenge 2017 (ocel). https://doi.org/10.4121/6889ca3f-97cf-459a-b630-3b0b0d8664b5.v1
Leemans SJJ, Fahland D, van der Aalst WMP (2014) Discovering block-structured process models from incomplete event logs. In: Ciardo G and Kindler E (eds) Application and Theory of Petri Nets and Concurrency - 35th International Conference, {PETRI} {NETS} 2014, Tunis, Tunisia, June 23-27, 2014. Lecture notes in computer science, vol 8489. Springer, Tunis, pp 91–110. https://doi.org/10.1007/978-3-319-07734-5_6
Leemans SJJ, Fahland D, van der Aalst W (2016) Scalable process discovery and conformance checking. Softw Syst Model 17:599–631
Leemans SJ, Li T, van Detten JN (2024a) Ebi-a stochastic process mining framework. In: ICPM 2024 Doctoral Consortium and Demo Track. CEUR Workshop Proceedings, vol-3783, CEUR-WS.org
Leemans SJ, Maggi FM, Montali M (2024) Enjoy the silence: Analysis of stochastic petri nets with silent transitions. Inf Syst 124:102383. https://doi.org/10.1016/j.is.2024.102383
Leemans SJJ, Polyvyanyy A (2020) Stochastic-aware conformance checking: An entropy-based approach. In: Dustdar S, Yu E, Salinesi C, Rieu D and Pant V (eds) Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. Lecture Notes in Computer Science, vol 12127. Springer, Grenoble, pp 217–233. https://doi.org/10.1007/978-3-030-49435-3_14
Leemans SJJ, Polyvyanyy A (2023) Stochastic-aware precision and recall measures for conformance checking in process mining. Inf Syst 115:102197. https://doi.org/10.1016/J.IS.2023.102197
Leemans SJJ, Poppe E, Wynn MT (2019) Directly follows-based process mining: Exploration & a case study. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019. IEEE, pp 25–32. https://doi.org/10.1109/ICPM.2019.00015
Leemans SJJ, van der Aalst WMP, Brockhoff T et al (2021) Stochastic process mining: Earth movers' stochastic conformance. Inf Syst 102:101724
Li T, Leemans SJ, Polyvyanyy A (2025) The jensen-shannon distance metric for stochastic conformance checking. In: ICPM 2024 International Workshops, Lyngby, Denmark, October 14–18, 2024, Revised Selected Papers, Springer Nature Switzerland, Cham, pp70–83. ISBN978-3-031-82225-4
Mannhardt F (2016). Sepsis cases - event log. https://doi.org/10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460
Mannhardt F (2017). Hospital billing - event log. https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
Marsan MA, Balbo G, Conte G, et al (1995) Modelling with generalized stochastic petri nets. SIGMETRICS Perform Eval Rev 26:2.
Mohri M (2011) Generic ϵ-removal and input ϵ-normalization algorithms for weighted transducers. Int J Found Comput Sci 13. https://doi.org/10.1142/S0129054102000996
Ney H, Essen U, Kneser R (1994) On structuring probabilistic dependences in stochastic language modelling. Comput Speech Lang 8(1):1–38. https://doi.org/10.1006/CSLA.1994.1001
Rocha EG, Leemans SJJ, van der Aalst WMP (2024) Stochastic conformance checking based on expected subtrace frequency. In: 6th International Conference on Process Mining, ICPM 2024, Kgs. Lyngby, Denmark, October 14-18, 2024. IEEE, pp 73–80. https://doi.org/10.1109/ICPM63005.2024.10680673
Steeman W (2013) Bpi challenge 2013, incidents. https://doi.org/10.1109/JPROC.2010.2070470
van Dongen BB (2014) Bpi challenge 2014. https://doi.org/10.4121/UUID:C3E5D162-0CFD-4BB0-BD82-AF5268819C35
van Dongen BB (2015) Bpi challenge 2015. https://doi.org/10.4121/UUID:31A308EF-C844-48DA-948C-305D167A0EC1
van Dongen B (2020) Bpi challenge 2020: Request for payment. https://doi.org/10.4121/uuid:895b26fb-6f25-46eb-9e48-0dca26fcd030
Vidal E, Thollard F, de la Higuera C, et al (2005a) Probabilistic finite-state machines - part i. IEEE Trans Pattern Anal Mach Intell 27(7). https://doi.org/10.1109/TPAMI.2005.147
Vidal E, Thollard F, de la Higuera C et al (2005) Probabilistic finite-state machines-part II. IEEE Trans Pattern Anal Mach Intell 27(7):1026–1039. https://doi.org/10.1109/TPAMI.2005.148

Publisher's Note