Stochastic Conformance Checking Based on Expected Subtrace Frequency

Eduardo Goulart Rocha
Celonis Labs GmbH & RWTH Aachen
Munich, Germany
e.goulartrocha@celonis.com

Sander J.J. Leemans
RWTH Aachen University & Fraunhofer
Aachen, Germany
s.leemans@bpm.rwth-aachen.de

Wil M.P. van der Aalst
RWTH Aachen University & Celonis
Aachen, Germany
wvdaalst@pads.rwth-aachen.de

***Abstract*—Conformance checking focuses on quantifying behavioral differences between desired and observed process behavior. Stochastic conformance checking considers not only the desired control flow of a process but also the relative frequency of each sequence. State-of-the-art stochastic conformance measures either cannot gracefully handle partially matching traces or are prohibitively expensive to compute. This paper bridges this gap by introducing the stochastic Markovian abstraction. The abstraction is defined as the relative occurrences of sub-traces in a stochastic language. Two stochastic languages can be compared via their Markovian abstractions using existing language comparison techniques. We show how to compute this abstraction for bounded, livelock-free stochastic labeled Petri nets. One of its derived measures is qualitatively and quantitatively evaluated on a series of artificial and real-world datasets. The experiments show that the abstraction can be efficiently computed and is successful in handling partially mismatching traces.**

***Index Terms*—Process Mining, Stochastic Conformance Checking, Stochastic Petri nets**

I. Introduction

Conformance checking measures quantify the extent of agreement between the observed data, in the form of an event log, and a de-jure model, specified as a process model. Traditional conformance checking techniques have mostly ignored the stochastic perspective of the process model, leading to one-sided analyses that only consider the frequency information from the event log. In recent years, stochastic conformance checking has gained traction. Its goals are the same as with traditional conformance checking, but considering the probability of traces in stochastic models, e.g. stochastic Petri nets.

A particular challenge for stochastic conformance measures is that in business processes, traces may only partially match one another. For instance, if a model trace and a log trace disagree in only one activity, we do not want to consider both traces as completely distinct. Therefore, ideally, a stochastic conformance measure applies partial trace matching. Furthermore, a stochastic conformance measure should be efficient to compute, and take into account the frequency information of both the model and the log. This is a difficult goal to achieve and none of the existing state-of-the-art stochastic conformance measures satisfies all these requirements.

We propose the stochastic Markovian abstraction to mitigate runtime and robustness issues in stochastic conformance checking techniques. As shown in Figure 1, the idea is to compare two stochastic languages based on their abstraction. The abstraction essentially encodes the expected number of occurrences of each subtrace in the language. By working with subtraces, the abstraction is naturally better suited to handle partial matches between traces. At the same time, the measure can be computed exactly for bounded, livelock-free stochastic labeled Petri nets, a widely adopted class of stochastic process models in process mining. We evaluate one of its derived metrics on synthetic and real-world datasets and show that using the abstraction helps mitigating these issues.

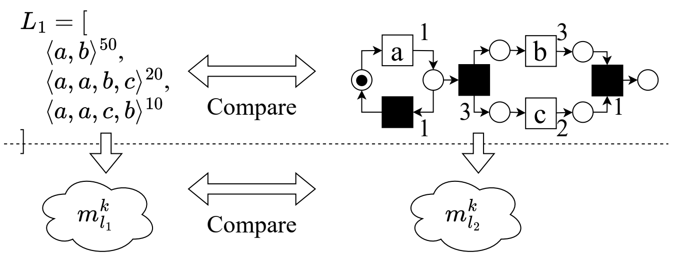

$L_1 = [$
$\langle a,b\rangle^{50}$,
$\langle a,a,b,c\rangle^{20}$,
$\langle a,a,c,b\rangle^{10}$
$]$
Compare
a
b
c
1
1
3
3
2
1
Compare
$m^k_{l_1}$
$m^k_{l_2}$

Fig. 1: Comparing the event log L_0 and the stochastic labeled Petri net SN_0 directly and via their Markovian abstraction.

The remainder of this paper is organized as follows: Section II introduces some basic concepts, Section III presents the stochastic Markovian abstraction, Section IV evaluates the abstraction on artificial and real-world datasets, Section V presents related work. Finally, Section VI concludes the paper with directions for future work.

II. Preliminaries

This section presents some basic definitions required to understand the main methods of the paper. $\mathbb{N}_0$ are the natural numbers including 0. We denote the indicator function as $\mathbf{1}_A(x)$ equals to 1 if $x \in A$, and 0 otherwise. Given an alphabet of activities Σ, we define $\tau \notin \Sigma$ as the *invisible label* and write $\Sigma_\tau = \Sigma \cup \{\tau\}$. $\Sigma^{\leq k}$ denotes the set of all traces over Σ of length less than or equal k, and Σ^* denotes the set of all finite traces over Σ. Given a trace $\sigma = \langle l_1, l_2, \cdots l_n\rangle \in \Sigma^*$, $\sigma^{i \to j} = \langle l_i, l_{i+1}, \cdots l_j\rangle, 1 \leq i \leq j \leq n$ 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 for every $s \in S$, $B(s)$ returns the frequency of s in B. We denote a multiset B as $B = [s_1^{B(s_1)}, s_2^{B(s_2)}, \cdots]$. If $B(s_i) = 1$, we omit the superscript and if $B(s_i) = 0$ we do not write s_i, e.g. for $S = \{x, y, z\}$, $[x^2, y]$ is the multiset containing 2 copies of x, one copy of y, and no copy of z. We use $\uplus$ and $\setminus$ to denote the union and difference of two multisets, i.e. $(B_1 \uplus B_2)(s) = B_1(s) + B_2(s)$ and $(B_1 \setminus B_2)(s) = B_1(s) - B_2(s)$ (this requires $B_1(s) \geq B_2(s)\ \forall s \in S$). We define $supp\, B = \{s \in S \mid B(s) > 0\}$ as the multiset's support and $|B| = \sum_{s \in S} B(s)$ as its cardinality. Furthermore, $\mathcal{B}(S)$ denotes the set of all multisets over S. Last, we abuse notation and write $x < \infty$ and $x = \infty$ to denote that an expression/function x is bounded/defined, respectively unbounded/undefined. We start with a general definition of a stochastic language:

Definition 1. (Stochastic Language) Let Σ be an alphabet, a stochastic language l is a function $l : \Sigma^* \rightarrow [0, 1]$ such that $\sum_{\sigma \in \Sigma^*} l(\sigma) = 1$.

An event log is a collection of events describing the system's observed behavior. It implicitly defines a stochastic language.

Definition 2. (Event Log and its Induced Stochastic Language) Let Σ be an alphabet. An event log L is a multiset of traces $L \in \mathcal{B}(\Sigma^*)$. An event log L defines a stochastic language $l_L : \Sigma^* \rightarrow [0, 1]$ where $l_L(\sigma) = \frac{L(\sigma)}{|L|}$.

Figure 1 shows an example log L_0. Its stochastic language is $l_{L_0} = [\langle a, b\rangle^{\frac{5}{8}}, \langle a, a, b, c\rangle^{\frac{2}{8}}, \langle a, a, c, b\rangle^{\frac{1}{8}}]$. In contrast to event logs, stochastic process models often have an unbounded number of traces and are usually specified using formalisms such as stochastic labeled Petri nets (SLPN).

Definition 3. (Stochastic Labeled Petri Net (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. $\tau \notin \Sigma$, $\lambda : T \rightarrow \Sigma_\tau$ is a transition labeling function, $M_0 \in \mathcal{B}(P)$ is the initial marking, and $w : T \rightarrow \mathbb{R}^+$ is a weight function.

SLPNs extend traditional labeled Petri nets with the stochastic perspective. The state of a SLPN is represented as a multiset of places $M \subseteq \mathcal{B}(P)$ called a *marking*. For each transition $t \in T$, we define $\bullet t = \{p \in P | (p, t) \in F\}$ and $t\bullet = \{p \in P | (t, p) \in F\}$ as the transition's pre-/post sets. A transition $t \in T$ is *enabled* at marking M (written $M[t\rangle$) if $\bullet t \leq M$. Firing an enabled transition t changes the state of the Petri net from M to $M' = (M \setminus \bullet t) \uplus t\bullet$ (we write $M[t\rangle M'$). Let $T' \subseteq T$ be the set of enabled transitions in M. A transition $t' \in T'$ fires with probability $w(t')/\sum_{t'' \in T'} w(t'')$.

A *run* is a sequence of transitions $\sigma = \langle t_1, t_2 \cdots t_n\rangle \in T^*$ such that $M_0[t_1\rangle M_1[t_2\rangle M_2 \cdots M_{n-1}[t_n\rangle M_n$. A marking M is reachable if there exists a run $\sigma \in T^*$ such that $M_0[\sigma\rangle M$. A marking is a *deadlock marking* if $\nexists t \in T \mid \bullet t \leq M$, i.e. no transition is enabled at M. For this work, we consider all deadlock markings as the net's accepting states. If M_n is an accepting state, then σ is an *accepting run*. Given a SLPN SN, $\mathcal{R}(SN) = \{M \in \mathcal{B}(P) | M \text{ is reachable}\}$ defines its set of reachable markings. A *livelock* is a marking from which no accepting state can be reached. A SLPN is *livelock-free* if it contains no reachable livelock and it is *bounded* if it has finitely many reachable states. Bounded livelock-free SLPNs generate a stochastic language and have a close link to Stochastic Non-Deterministic Finite Automata (SNFA):

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. $\tau \notin \Sigma$, $\delta : Q \times \Sigma_\tau \times Q \rightarrow [0, 1]$ is the transition probability function, q_0 is the initial state and $p_f : Q \rightarrow [0, 1]$ is the function defining the final probability of each state, where $\forall q \in Q : \sum_{(l,q') \in (\Sigma_\tau \times Q)} \delta(q, l, q') + p_f(q) = 1$.

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

A state in N is *useful* if it appears in at least one path of N with non-zero probability and *accessible* if it can be reached from the starting state with non-zero probability. N is consistent if all of its accessible states are useful [1]. Furthermore, N is *trimmed* if all of its states are accessible and useful (a SNFA can be trimmed in linear time).

If N is consistent, then $\mathbb{P}_N : \Sigma^* \rightarrow [0, 1]$ defines a stochastic language [1]. The class of *stochastic regular languages* is the class of stochastic languages that can be generated by a consistent SNFA [1]. In this paper, we will only consider trimmed, consistent SNFAs. 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 $N = (\mathcal{R}(SN), \Sigma, \delta, M_0, p_f)$ such that:

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

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 2 shows the embedded SNFA from the SLPN

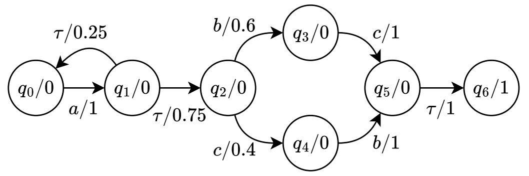


Fig. 2: SN_0's embedded SNFA N_0

of Figure 1. 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 [2].

A. The Earth Mover's Stochastic Conformance Measure

This work is heavily based on the Earth Mover's Stochastic Conformance (EMSC) measure [3]. Before we can define that, we must first define the notion of a *reallocation function*.

Definition 6. (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:

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

In other words, a reallocation function defines how to move the probability mass from one trace distribution into another. The EMSC measure is based on the optimal reallocation strategy with respect to some cost function.

Definition 7. (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:

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

Two cost functions are commonly used in process mining. The normalized string edit (Levenshtein) distance [3] and the unit distance between traces $u : \Sigma^* \times \Sigma^* \to [0,1]$ defined as $u(\sigma, \sigma') = \mathbf{1}_{\{\sigma\}}(\sigma')$, defining the EMSC and uEMSC stochastic conformance measures respectively.

Both measures have significant weaknesses. The EMSC is arguably considered as the ground truth for measuring language similarity. However, it 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 reasonable probability mass from the model, which is in itself a challenging task for models with a high degree of parallelism. Even if the sampling succeeds, solving the associated minimization problem requires solving an optimal transportation problem [3], which is computationally expensive. In contrast, the uEMSC can be computed in an exact manner [4] and is efficient in practice, however, the unit cost function severely penalizes partial trace mismatches. Hence, existing techniques require the user to choose between scalability, robustness to partial mismatches, and exact measure computation.

III. The Stochastic Markovian Abstraction

This section presents the stochastic Markovian abstraction, a natural extension of the Markovian abstraction from [5] to stochastic languages. We first define the abstraction and discuss under which conditions it exists. Then, we show how to compute it for stochastic regular languages. Finally, we present a specific conformance measure based on this abstraction. We start by defining the multiset of k-trimmed subtraces:

Definition 8. (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:

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

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 9. (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, 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 9 is extended to stochastic languages by considering trace probabilities as follows:

Definition 10. (K-th Order Stochastic Markovian Abstraction) Let $l : \Sigma^* \to [0,1]$ be a stochastic language over alphabet Σ and $k \geq 2$. Define function $f^k_l : \Sigma^*_\pm \to \mathbb{R}$ as:

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

The k-th order stochastic Markovian abstraction of l is the function $m^k_l : \Sigma^*_\pm \to [0,1]$ defined as:

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

Function f_l^k 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 is the normalization of f_l^k. This is a natural generalization of the non-stochastic version from [6].

The use of prefix/suffix markers $+/-$ is due to two reasons. First, being able to distinguish between prefixes, suffixes, and infixes of the language makes it easier to define operations on this set, as done in [6]. Second, it helps distinguishing between "scaled" languages. For example, given languages $l_1 = [\langle a,a\rangle^{1.0}]$ and $l_2 = [\langle a,a,a\rangle^{1.0}]$, then $m^2_{l_1} = [\langle +,a\rangle^{\frac{1}{3}}, \langle a,a\rangle^{\frac{1}{3}}, \langle a,-\rangle^{\frac{1}{3}}]$ and $m^2_{l_2} = [\langle +,a\rangle^{0.25}, \langle a,a\rangle^{0.5}, \langle a,-\rangle^{0.25}]$. If no $+/-$ markers were used, then $m^2_{l_1} = m^2_{l_2} = [\langle a,a\rangle^{1.0}]$.

For the example event log L_0 from Figure 1, we obtain that $f^2_{l_{L_0}}(\langle +,a\rangle) = \frac{5}{8}*1+\frac{2}{8}*1+\frac{1}{8}*1 = 1$, and (in general), $f^2_{l_{L_0}} = [\langle +,a\rangle^1, \langle a,b\rangle^{\frac{7}{8}}, \langle b,-\rangle^{\frac{6}{8}}, \langle a,a\rangle^{\frac{3}{8}}, \langle b,c\rangle^{\frac{2}{8}}, \langle c,-\rangle^{\frac{2}{8}}, \langle a,c\rangle^{\frac{1}{8}}, \langle c,b\rangle^{\frac{1}{8}}]$. And so $m^2_{l_{L_0}} = [\langle +,a\rangle^{\frac{8}{30}}, \langle a,b\rangle^{\frac{7}{30}}, \langle b,-\rangle^{\frac{6}{30}}, \langle a,a\rangle^{\frac{3}{30}}, \langle b,c\rangle^{\frac{2}{30}}, \langle c,-\rangle^{\frac{2}{30}}, \langle a,c\rangle^{\frac{1}{30}}, \langle c,b\rangle^{\frac{1}{30}}]$.

Clearly, m^k is well-defined for stochastic languages with finite support, such as event logs. For arbitrary stochastic languages, this might not be the case. Consider $l : \{a\}^* \to [0,1]$ defined as $l(\sigma) = \begin{cases} \frac{1}{2^{n+1}} & \sigma = \langle\underbrace{a,a,\cdots a}_{2^n+1 \text{ times}}\rangle, n \in \mathbb{N}_0 \\ 0 & \text{otherwise} \end{cases}$. Then 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 lemma below gives a necessary and sufficient condition for $f_l^k < \infty$.

Lemma 11. (A Necessary and Sufficient Condition for Well-Defined f_l^k) Given a stochastic language $l : \Sigma^* \to [0,1]$. Then for any $k \geq 2$

$$f_l^k(\gamma) < \infty \;\forall\gamma \in \Sigma_\pm^{\leq k} \iff \sum_{\sigma\in\Sigma^*} l(\sigma)|\sigma| < \infty \tag{5}$$

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

($\Rightarrow$) $f_l^k(\gamma) < \infty \;\forall\gamma$ implies $\sum_{\gamma\in\Sigma_\pm^{\leq k}} f_l^k(\gamma) < \infty$, then:

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

And since $|M_\sigma^k| = |S^k_{+\sigma-}| \geq \frac{|\sigma|}{k}$, then:

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

($\Leftarrow$) Observe that $|S_\sigma^k| \leq |\sigma| + 1$ (can be easily proven by induction), therefore $|M_\sigma^k| = |S^k_{+\sigma-}| \leq |\sigma| + 3$. 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 $\sum_{\sigma\in\Sigma^*} l(\sigma)|\sigma| < \infty$. □

It follows directly from $supp(f_l^k) \subseteq \Sigma_\pm^{\leq k}$ that $f_l^k < \infty \Rightarrow m_l^k < \infty$. The condition from Lemma 11 above holds for stochastic regular languages. We show that by presenting a procedure to compute it for a τ-free SNFA. But before, we must introduce patched SNFAs:

Definition 12. (Patched SNFA) Let $N = (Q, \Sigma, \delta, q_0, p_f)$ be a 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

$$\hat{\delta}(q,l,q') = \begin{cases} p_f(q) & q' = q_- \text{ and } l = - \\ 1 & q = q_+ \text{ and } l = + \\ \delta(q,l,q') & \text{otherwise} \end{cases}$$

$\hat{N}$ adds a source state q_+ and a sink state q_- to N. It accepts the stochastic language $\hat{l} : \Sigma_\pm^* \to \mathbb{R}$ equal to language l concatenated with suffix $+$ and prefix $-$, i.e. $l(t) = \hat{l}(+t-)$.

Lemma 13. (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:

$$f_l^k(\gamma) = \sum_{q\in\hat{Q}} x_q \cdot \hat{\phi}(\gamma|q) \tag{6}$$

Where $\mathbf{x} = [x_{q_+}\; x_{q_0} \cdots\; x_{q_-}]^\mathsf{T}$ is the solution to the equation $(I - \hat{\Delta})\mathbf{x} = [1\;0\;\cdots\;0]^\mathsf{T}$ ($\hat{\Delta}$ is $\hat{N}$'s transition matrix) and $\hat{\phi}(\gamma|q) = \sum_{\rho\in\Theta_{\hat{N}}(q),\lambda(\rho_\gamma)=\gamma}\hat{\delta}(\rho)$.

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

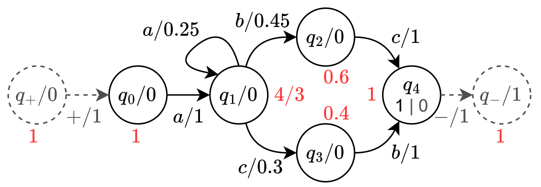
For $|\gamma| = k$, we define the boundary-affixes set $\mathcal{B}_\sigma(\gamma) = \{(\alpha,\beta) \mid \alpha,\beta \in \Sigma_\pm^*, +\sigma- = \alpha\gamma\beta\}$. Then $M_\sigma^k(\gamma) = |\mathcal{B}_\sigma(\gamma)|$. Therefore, $f_l^k(\gamma) = \sum_{\sigma\in\Sigma^*} l(\sigma)M_\sigma^k(\gamma) = \sum_{\sigma\in\Sigma^*}\sum_{\alpha,\beta\in\mathcal{B}_\sigma(\gamma)}\mathbb{P}_{\hat{N}}(+\sigma-) = \sum_{\alpha,\beta\in\Sigma_\pm^*}\mathbb{P}_{\hat{N}}(\alpha\gamma\beta)$. (To check the last step, expand $\mathcal{B}_\sigma(\gamma)$ 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:

$$\sum_{\alpha,\beta\in\Sigma_\pm^*}\mathbb{P}_{\hat{N}}(\alpha\gamma\beta) = \sum_{\substack{\rho_\alpha,\rho_\gamma,\rho_\beta\in\hat{E}^* \\ \rho_\alpha\rho_\gamma\rho_\beta\in\Theta_{\hat{N}}(q_+) \\ \lambda(\rho_\gamma)=\gamma}} P_{\hat{N}}(\rho_\alpha\rho_\gamma\rho_\beta) \tag{7}$$

The right side of Equation 7 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:

$$= \sum_{\substack{\rho_\alpha,\rho_\gamma\in\hat{E}^* \\ \rho_\alpha\rho_\gamma\in\Theta_{\hat{N}}(q_+) \\ \lambda(\rho_\gamma)=\gamma}}\Big(\hat{\delta}(\rho_\alpha\rho_\gamma)\sum_{\rho_\beta\in\Theta_{\hat{N}}(tgt(\rho_\gamma))} P_{\hat{N}}(\rho_\beta)\Big) \tag{8}$$



(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. 3: N_0 from Figure 2 after τ-removal (3a black) and patching $\hat{N}_0$ (3a gray), and its transition matrix $\hat{\Delta}$ (3b)

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

$$= \sum_{\substack{\rho_\alpha,\rho_\gamma \in \hat{E}^* \\ \rho_\alpha\rho_\gamma \in \Theta_{\hat{N}}(q_+) \\ \lambda(\rho_\gamma)=\gamma}} \hat{\delta}(\rho_\alpha\rho_\gamma) = \sum_{\rho_\alpha \in \Theta_{\hat{N}}(q_+)} \Big(\hat{\delta}(\rho_\alpha) \cdot \sum_{\substack{\rho_\gamma \in \Theta_{\hat{N}}(tgt(\rho_\alpha)) \\ \lambda(\rho_\gamma)=\gamma}} \hat{\delta}(\rho_\gamma) \Big) \quad (9)$$

$$= \sum_{q \in \hat{Q}} \sum_{\rho_\alpha \in \Theta_{\hat{N}}(q_+,q)} \Big(\hat{\delta}(\rho_\alpha) \cdot \hat{\phi}(\gamma|q) \Big) \quad (10)$$

Now define $\psi(q_i, q_j, n) = \sum_{\rho \in \Theta_{\hat{N}}(q_i,q_j,n)} \hat{\delta}(\rho)$. Then $\sum_{\rho_\alpha \in \Theta_{\hat{N}}(q_+,q)} \hat{\delta}(\rho_\alpha) = \sum_{n=0}^{\infty} \psi(q_+, q, n)$. 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, $\sum_{n=0}^{\infty} \psi(q+, q, n) = (I + \hat{\Delta} + \hat{\Delta}^2 + \cdots)_{q_+,q} = (I - \hat{\Delta})^{-1}_{q_+,q} = x_q$ ([7]). □

Lemma 13 directly yields a method to compute m_l^k given a SNFA N accepting l. First, we transform N into a τ-free SNFA N' with a τ-removal step [2] and patch N' by adding the $+/-$ markers, obtaining $\hat{N}$. Then $\mathbf{x}$ can be computed by solving the system of linear equations $(I - \hat{\Delta})\mathbf{x} = [1\ 0 \cdots 0]^\mathsf{T}$, and $\hat{\phi}(\gamma|q)$ can be computed by replaying γ from q.

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

1) Extending to Other Modeling formalisms: The section above shows that it is possible to compute m_l^k of a bounded, livelock-free SLPN via its embedded SNFA. Nevertheless, there also exists unbounded livelock-free SLPNs for which m_l^k is well-defined. For example, the net from Figure 4, accepting the language $l(\sigma) = \begin{cases} \frac{1}{2^{n+1}} & \sigma = \langle \underbrace{a, \cdots a}_{n \text{ times}}, \underbrace{b, \cdots b}_{n \text{ times}} \rangle,\ n \in \mathbb{N}_0 \\ 0 & \text{otherwise} \end{cases}$

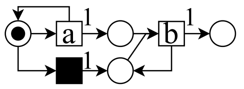


Fig. 4: Unbounded, livelock-free SLPN with well-defined m_l^k.

Observe that $\sum_{\sigma \in \Sigma^*} |\sigma| l(\sigma) = \sum_{n=0}^{\infty} \frac{2n}{2^{n+1}} < \infty$. Therefore (Lemma 11), m_l^k is well-defined. The question arises whether m_l^k is well-defined for all livelock-free SLPNs. We could neither prove nor disprove this statement.

2) Runtime: m^k of an event log can be computed with a linear pass. Let SN be a SLPN. Let $N = (Q, \Sigma, \delta, q_0, p_f)$ be its embedded SNFA 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. Then $|Q| = |\mathcal{R}(SN)|$, where $|\mathcal{R}(SN)|$ is exponential with the number of places in SN. $|Q'| \le |Q|$ [2], but N' has higher edge density than N.

The computation of Lemma 13 occurs in two steps: First, solve the linear system $(I - \hat{\Delta})\mathbf{x} = [1\ 0 \cdots 0]^T$. Then, compute $\hat{\phi}(\gamma|q)$ for each $(\gamma, q) \in \Sigma_\pm^{\le k} \times \hat{Q}$. 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 verion m^k [6]. Despite the relatively high exponents, Section IV shows that the computation is still feasible.

3) The Stochastic Markovian-Based Conformance Measures: Definition 10 defines a stochastic language. Hence, given two stochastic languages l_1 and l_2, it is possible to compare $m_{l_1}^k, m_{l_2}^k$ using any existing stochastic conformance measure. In this paper, we propose to use the uEMSC measure to ensure the scalability of the approach. The m^k-uEMSC measure can be efficiently computed using the formula $m^k\text{-}uEMSC(l_1, l_2) = 1 - \sum_{\gamma \in \Sigma_\pm^{\le k}} max(m_{l_1}^k(\gamma) - m_{l_2}^k(\gamma), 0)$ [3]. By using the stochastic Markovian abstraction, m^k-uEMSC compensates for the rigidity of the uEMSC measure regarding partially matching traces. Intuitively, $m^k\text{-}uEMSC$ compares the expected frequency of subtraces in both languages. Notice that m_l^k approaches $+l-$ as $k \to \infty$. Therefore, for large ks, $m^k\text{-}uEMSC(l_1, l_2)$ approaches $uEMSC(+l_1-, +l_2-) = uEMSC(l_1, l_2)$.

IV. Experimental Evaluation

This section evaluates the m^k-uEMSC measure qualitatively and quantitatively. The approach has been implemented in Python 3.11. We use optimized native libraries for the τ-removal step [8] (which implements an approximated variant) and for solving linear systems. We choose the truncated Levenshtein-based earth mover's stochastic conformance (tEMSC) [3] measure and the unit-cost earth mover's stochastic conformance (uEMSC) [4] measure as baseline. The first

TABLE I: Measure value (model ranking) for log L_1.

	tEMSC	uEMSC	m^2	m^3	m^4
SN_1	0.973 (2)	0.953 (2)	0.937 (2)	0.924 (2)	0.897 (2)
SN_2	0.579 (5)	0.368 (3)	0.685 (3)	0.575 (3)	0.536 (3)
SN_3	0.727 (4)	0.000 (8)	0.644 (4)	0.349 (4)	0.270 (4)
SN_4	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)
SN_5	0.744 (3)	0.000 (8)	0.626 (5)	0.310 (5)	0.224 (5)
SN_6	0.423 (6)	0.017 (4)	0.325 (6)	0.090 (6)	0.021 (6)
SN_7	0.285 (8)	0.001 (5)	0.221 (8)	0.034 (8)	0.005 (8)
SN_8	0.371 (7)	0.000 (8)	0.222 (7)	0.052 (7)	0.010 (7)

is chosen as it is arguably considered to provide the ground truth in terms of model rankings. The later is chosen to illustrate the issue with partial mismatches. Both measures are implemented in JAVA as part of the ProM framework [9]. For m^k-uEMSC, we vary k from 2 to 4 (m^2, m^3, and m^4). For all measures, we set a total timeout of one hour and 16 gigabytes of memory limit. All experiments are run single-threaded on an Intel(R) Xeon(R) E-2276M CPU running Ubuntu 22.04. As a termination criterion for the sampling step of the $tEMSC$ measure, we set a minimum sampling mass of 95% of model behavior, 50 thousand Petri net runs, or twenty minutes runtime (whichever comes first). [1]

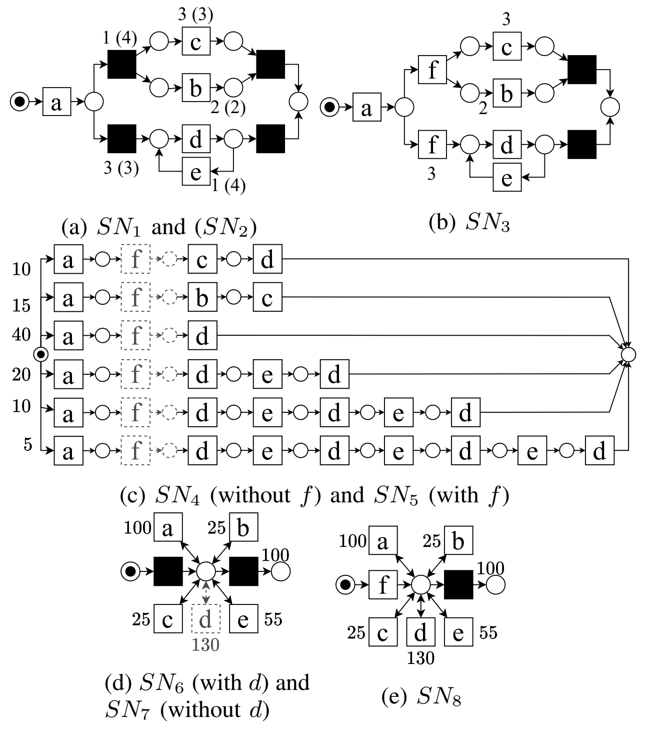
A. Robustness to Partial Mismatches and Induced Ranking

We evaluate the robustness of the m^k-uEMSC measure with respect to partial trace mismatches and compare its induced ranking with other state-of-the-art measures. We consider the rank induced by the tEMSC as the ground-truth. For that, we consider a manually designed dataset consisting of the event log $L_1 = [\langle a,c,b\rangle^{10}, \langle a,b,c\rangle^{15}, \langle a,d\rangle^{40}, \langle a,d,e,d\rangle^{20}, \langle a,d,e,d,e,d\rangle^{10}, \langle a,d,e,d,e,d,e,d\rangle^{5}]$ and the process models from Figure 5. Model SN_1 is manually designed to best represent L_1. Models SN_2, SN_3 are modifications of SN_1 with changed transition weights and an added transition f respectively. SN_4 is the *trace model* accepting the same stochastic language as L_1 and SN_5 is the trace model with an activity f added at the beginning of each trace. SN_6 is the flower model with transition weights derived from the relative frequency of each activity in the log. SN_7 and SN_8 are variations of SN_6 obtained by removing/adding transitions d and f respectively. The conformance of the log with respect to each model is evaluated. The results are shown in Table I.

The first thing to notice is that, as expected, the trace model SN_4 is assigned a conformance of 1.0 by all measures since it perfectly models the event log's distribution. Model SN_1 is ranked as the second-best model by all measures. This is in line with expectations since it best approximates the probabilities of each trace (after SN_4).

When considering models SN_3, SN_5, and SN_8, we see that the $uEMSC$ measure assigns them a conformance of 0. This is 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

[1]The experiment artifacts can be found at zenodo.org/records/11531791



(a) SN_1 and (SN_2)
(b) SN_3
(c) SN_4 (without f) and SN_5 (with f)
(d) SN_6 (with d) and SN_7 (without d)
(e) SN_8

Fig. 5: 8 synthetic stochastic models. Unless marked, transition weights = 1. This dataset has been adapted from [10]

example of model SN_5. While $tEMSC$ ranks this model as the third best model, $uEMSC$ assigns it a conformance of 0.

Considering the stochastic Markovian measure m^k, we see that the induced model ranks agree with those from $tEMSC$ (except for models SN_2 and SN_5 that have their orders swapped). As k increases, the ranks induced by m^k remain stable, but their score values decrease. This is expected since the measure m^k approaches $uEMSC$ as k increases. Finally, when considering relative values, we see that $tEMSC$ is too generous and assigns a high score for the flower model.

B. Runtime and Induced Rankings for Real-World Datasets

This section evaluates the scalability of the approach on real-world datasets and compares its induced rankings with other state-of-the-art measures. We choose the Italian road fines (RF) [11], the BPI challenge 2013 [12], and the sepsis (SEPSIS) [13] event logs. The BPI Challenge 2013 is split into three logs: "open problems", "closed problems" and "incidents" (BPI13-OP, BPI13-CP, and BPI13-I respectively). For each event log, a process model is discovered using the Inductive Miner infrequent variant [14] with a frequency threshold of 0.2 (IMf02), the Directly Follows Miner with a frequency threshold of 0.8 (DFM) and the Flower Miner (FM). Each control flow model is then enriched using a weight estimator [15]. We used the frequency-based estimator (FBE), the Bill Clinton estimator (BCE) (fork distribution estimator in [15]), and the alignment-based estimator (ABE). For the directly-follows miner and the flower miner, both FBE and BCE return models with equal weights (this is intrinsic to how

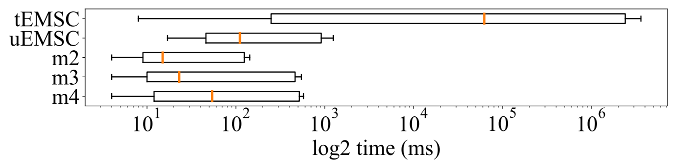


Fig. 6: Runtime distribution for each measure. Notice the log scale on the x-axis. Computations that went out of memory were discarded.

both estimators work and we refer to [15] for further details). Because of that, we discard models DFM-BCE and FM-BCE.

For a fixed control-flow model, one expects the alignment-based estimator to be the most accurate of all three, followed by the Bill-Clinton estimator (if available) and the frequency-based estimator as the least accurate. When using the same weight estimator, one expects the flower model to be the least accurate as it allows for too much extra behavior. It is not possible to define a preference between the inductive miner and the directly-follows miner since their performances depend on the input log as well as the chosen noise thresholds. When enough behavior can be sampled from the model, the tEMSC measure provides what we consider to be the ground-truth value for the induced ranks. The value and runtime results are shown in Table II and Figure 6.

TABLE II: Comparison of proposed m^k-uEMSC measure with state-of-the-art measures. For the tEMSC measure, scenarios where less than 50%/10% of model probability mass is sampled are colored orange/red.

DS	measure	IMf02			DFM		FM	
		FBE	BCE	ABE	FBE	ABE	FBE	ABE
RF	tEMSC	0.529	0.523	0.761	0.877	0.949	0.422	0.503
	uEMSC	0.014	0.010	0.294	0.684	0.820	0.024	0.019
	m^2	0.526	0.534	0.780	0.866	0.900	0.286	0.265
	m^3	0.375	0.360	0.635	0.793	0.860	0.079	0.078
	m^4	0.241	0.247	0.471	0.757	0.806	0.026	0.020
BPI13-OP	tEMSC	0.417	0.421	0.740	0.690	0.925	0.703	0.730
	uEMSC	4e-4	3e-4	0.416	0.219	0.750	0.339	0.344
	m^2	0.496	0.474	0.598	0.611	0.931	0.609	0.624
	m^3	0.278	0.262	0.420	0.451	0.837	0.433	0.471
	m^4	0.103	0.111	0.290	0.321	0.722	0.348	0.387
BPI13-CP	tEMSC	0.459	0.461	0.766	0.941	0.915	0.492	0.624
	uEMSC	1e-4	1e-4	0.427	2e-4	0.657	0.052	0.049
	m^1	0.436	0.465	0.667	0.545	0.928	0.460	0.492
	m^2	0.167	0.251	0.440	0.330	0.808	0.275	0.344
	m^3	0.069	0.129	0.238	0.205	0.679	0.173	0.254
BPI13-I	tEMSC	!oom	!oom	!to	!to	!to	0.336	!to
	uEMSC	0.000	0.000	0.030	0.044	0.150	0.005	0.005
	m^2	0.251	0.090	0.341	0.687	0.956	0.372	0.517
	m^3	0.099	0.001	0.131	0.468	0.756	0.189	0.292
	m^4	0.059	0.000	0.063	0.331	0.636	0.092	0.186
SEPSIS	tEMSC	!to	0.545	0.684	!to	0.765	!to	!to
	uEMSC	0.000	4e-6	3e-4	5e-5	0.046	5e-5	3e-5
	m^2	0.377	0.464	0.479	0.606	0.973	0.292	0.469
	m^3	0.222	0.270	0.295	0.452	0.814	0.131	0.300
	m^4	0.139	0.168	0.186	0.296	0.633	0.048	0.146

The first thing to notice is that tEMSC times out or goes out of memory in multiple scenarios, while none of the other measures suffers from these problems. Even in the situations where the tEMSC computation succeeds, it is by far the most expensive method, being up to four orders of magnitude slower than the second slowest method. Curiously, tEMSC also achieves one of the smallest runtimes of all in one situation (the BPI13-OP-DFM-ABE model). The algorithm samples a sufficiently large probability mass by sampling only a few variants, hence the subsequent optimization step is efficient. In contrast, the uEMSC measure can be fairly efficiently computed, but it frequently returns very small (or even 0) conformance values. This exposes the trade-off between runtime and robustness offered by both measures.

In general, the rankings induced by all measures are similar. For m^k, some model ranks might flip as k increases. This happens especially for models that have very similar scores, e.g. RF-IMf02-FBE and RF-IMf02-BCE.

For all miners, all measures agree on ABE as being the best estimator. The exceptions are the uEMSC and m^k measures for the RF-FM models and the tEMSC measure for the BPIC13-CP log with the DFM miner. In the latter case, the tEMSC measure assigns a higher conformance value to the FBE estimators than to the ABE estimator, which is counterintuitive. However, in both situations, less than 0.1% of model behavior is sampled. Hence, any conclusion taken from this measure is based on little evidence and should be disregarded.

For the RF and BPI13-CP logs, m^k assigns relatively similar scores for FM-FBE and FM-ABE models, despite tEMSC reporting a large difference. This is due to the fact that for the flower model, both estimators assign the same weight to all visible transitions. The models only differ in their τ transitions. FBE assigns a low weight to τ exiting the loop, causing longer traces to be more likely, which increases its total cost. In comparison, the m^k measure is less sensitive to that as it computes the relative subtrace frequency.

For the BPI13-I log, almost all tEMSC computations time out or go out of memory. At the same time, the uEMSC assigns very low scores for all models, which indicates a high degree of partial trace mismatches. Something similar happens for the SEPSIS log, where almost all tEMSC computations time out, and those that do not time out sample too little model behavior to obtain reliable conclusions. Similarly, most uEMSC values are too low, possibly indicating a high amount of partial trace mismatches. Therefore, m^k is arguably the only measure capable of processing these logs. For these two logs, the results produced by m^k are similar to the ones obtained for the other logs and in line with our initial expectations.

Finally, when comparing the runtimes for the uEMSC and the m^k-uEMSC measures, one notices that the m^k-uEMSC has lower median runtime, in the range of double-digit milliseconds. For small ks $(= 2, 3)$, the computation is dominated by the linear pass over the event log. Furthermore, it is important to notice the difference in the chosen implementation languages. The m^k measures are implemented in Python whereas the uEMSC is implemented in JAVA. We believe that implementing the m^k measures in a compiled language would widen the performance gap between both approaches.

In summary, the real-world experiments show that

m^k-$uEMSC$ induces a ranking in line with our intuition and with state-of-the-art techniques, while mitigating issues of sampling enough behavior for the tEMSC and partial mismatches for the uEMSC.

V. Related Work

In process mining, a wide range of non-stochastic conformance measures has been introduced [5], [16], [17]. This work builds upon on the Markovian-based abstraction first presented in [5] and later redefined in [6]. The original motivation for extending it to the stochastic perspective was to address the issue of vanishing precision observed in [6] as k increases. As shown in Section IV, this might still happen, albeit in a more attenuated form, and is now linked to partial mismatches in the subtraces.

The stochastic markovian metric was informally introduced in [18] to compare the behavior of simulation models. Our work differs in which 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.

The literature on stochastic conformance checking is relatively scarce. As far as we are aware, the EMSC measure [3] 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 [4]. In [19], the EMSC measure is extended to consider the time perspective in the task of concept-drift detection. In [10], a measure based on gain entropy is introduced. However, this measure does not properly consider the stochastic information of both artifacts. If the log and model share the same support language, then [10] returns fitness and precision of 1, even if their distributions differ.

Last, probabilistic trace alignments have been proposed in [20] 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. However, the approach focuses on generating diagnostics and no conformance measure is proposed based on probabilistic alignments. Finally, there exists methods to compute sub-string probabilities given a stochastic regular grammar [21]. In comparison, we provide a method for computing sub-string probabilities from a SNFA.

VI. Conclusion

We introduced the Markovian abstraction for stochastic languages and evaluated its application to conformance checking. The abstraction defines a stochastic language, such that it can be combined with any established stochastic conformance checking techniques. The experiments indicate that using the abstraction helps to mitigate weaknesses of existing measures in terms of scalability and handling of partial mismatches.

As future work, we want to investigate how to more efficiently compute this abstraction for certain classes of models, as done in [6]), and evaluate the combination of m^k with other conformance measures. Furthermore, we believe that a direct computation that bypasses the τ-removal step is possible for k-bounded workflow nets [20]. Finally, the method can be extended to consider time aspects [19].

Acknowledgment

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

[1] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco, "Probabilistic finite-state machines - part i," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, 2005.

[2] M. Mohri, "Generic ϵ-removal and input ϵ-normalization algorithms for weighted transducers," *International Journal of Foundations of Computer Science*, vol. 13, 11 2011.

[3] S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, and A. Polyvyanyy, "Stochastic process mining: Earth movers' stochastic conformance," *Information Systems*, vol. 102, p. 101724, 2021.

[4] S. J. Leemans, F. M. Maggi, and M. Montali, "Enjoy the silence: Analysis of stochastic petri nets with silent transitions," *Information Systems*, vol. 124, p. 102383, 2024.

[5] A. Augusto, A. Armas-Cervantes, R. Conforti, M. Dumas, and M. La Rosa, "Measuring fitness and precision of automatically discovered process models: A principled and scalable approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 4, 2022.

[6] E. Goulart Rocha and W. M. P. van der Aalst, "Polynomial-time conformance checking for process trees," in *International Conference on Business Process Management*, 2023, pp. 109–125.

[7] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. AMS, 2003.

[8] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *Implementation and Application of Automata*, 2007, pp. 11–23.

[9] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The prom framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005*, 2005, pp. 444–454.

[10] S. J. J. Leemans and A. Polyvyanyy, "Stochastic-aware conformance checking: An entropy-based approach," in *Advanced Information Systems Engineering*, 2020, pp. 217–233.

[11] M. de Leoni and F. Mannhardt, "Road traffic fine management process," 2 2015, doi: 10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[12] W. Steeman, "Bpi challenge 2013, incidents," 2013, doi: 10.1109/JPROC.2010.2070470.

[13] F. Mannhardt, "Sepsis cases - event log," 2016, doi: 10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460.

[14] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *Application and Theory of Petri Nets and Concurrency*, 2014, pp. 91–110.

[15] A. Burke, S. J. J. Leemans, and M. T. Wynn, "Stochastic process discovery by weight estimation," in *Process Mining Workshops*, 2021, pp. 260–272.

[16] A. Adriansyah, B. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *15th International Enterprise Distributed Object Computing Conference*, 2011, pp. 55–64.

[17] S. J. J. Leemans, D. Fahland, and W. van der Aalst, "Scalable process discovery and conformance checking," *Software and Systems Modeling*, vol. 17, pp. 599 – 631, 2016.

[18] D. Chapela-Campa, I. Benchekroun, O. Baron, M. Dumas, D. Krass, and A. Senderovich, "Can i trust my simulation model? measuring the quality of business process simulation models," in *International Conference on Business Process Management*, 2023, pp. 20–37.

[19] T. Brockhoff, M. S. Uysal, and W. M. P. van der Aalst, "Time-aware concept drift detection using the earth mover's distance," in *2020 2nd International Conference on Process Mining (ICPM)*, 2020, pp. 33–40.

[20] G. Bergami, F. M. Maggi, M. Montali, and R. Peñaloza, "Probabilistic trace alignment," in *2021 3rd International Conference on Process Mining (ICPM)*, 2021, pp. 9–16.

[21] A. L. N. Fred, "Computation of substring probabilities in stochastic grammars," in *Grammatical Inference: Algorithms and Applications, 5th International Colloquium, ICGI 2000*, vol. 1891, 2000, pp. 103–114.