Trace Probability Calculation on Probabilistic Process Trees

Adam T. Burke[1][0000−0003−4407−2199], Sander J.J. Leemans[2][0000−0002−5201−7125], and Moe T. Wynn[1][0000−0002−7205−8821]

[1] Queensland University of Technology, Australia {at.burke,m.wynn}@qut.edu.au
[2] RWTH Aachen, Germany s.leemans@bpm.rwth-aachen.de

Abstract. An important calculation for many problems in stochastic process mining is determining the probability of a trace in a given process model. We show two closed form solutions for this problem, within a parameterised precision bound, on two data structures: a type of weighted automata, and a stochastic extension of process trees.

Keywords: stochastic process mining · trace probability · process trees

1 Introduction

The promise of stochastic process models is that they can be used for detailed and precise descriptions of probability, including the probability that a trace will occur under a given model, or, the Trace-Prob problem [9].

This paper provides solutions to Trace-Prob on stochastic versions of two data structures, given an approximation bound. The first is a form of stochastic finite automata [11]. The second is a type of stochastic process tree [2]. Though solutions exist for Trace-Prob on other types of process model, process trees are a heavily used data structure in process mining, most famously in the Inductive Miner [8]. Precise approximation bounds and calculation reuse are both useful properties for calculating entire stochastic languages, as in stochastic process quality metrics, and potentially within discovery algorithms themselves. Other solutions to Trace-Prob [9, 12, 3] rely on other process representations such as Petri nets. Another recent solution on process trees [5] chooses a representation that limits straightforward translation to stochastic Petri nets; this paper retains translation at the cost of a more complicated calculation.

Below, we introduce formal preliminaries in Section 2. We define Weighted Stochastic Finite Automata (WSFA) and use them as a formal grounding for Probabilistic Process Trees (PPTs) in Section 3. Sections 4 and 5 show trace probability calculations on WSFAs and PPTs respectively. Section 6 discusses the result and related work. Section 7 concludes. An extended treatment of this paper, with more properties and special cases, is also available [1, Ch. 5].

2 Preliminaries

In this paper, quantifier variables are separated from their predicates with ▪, e.g., $\forall x \in \mathbb{N} \blacksquare x \geqslant 1$. A finite sequence over set X of length n is a mapping $\sigma \in \{1..n\} \to X$ and denoted by $\sigma = \langle a_1, a_2, ..., a_n \rangle \in X^*$ where $\forall i \blacksquare a_i = \sigma[i]$. Concatenation operator $+$ appends one sequence to another such that $\langle a_1, ..., a_n \rangle + \langle b_1, ..., b_m \rangle = \langle a_1, ...a_n, b_1, ..., b_m \rangle$. We also use slice ($[...‡...]$) and length $|\sigma|$ operations. $\sigma[i‡j]$ is a slice from indexes i to j inclusive, e.g., $\langle a, b, c, d \rangle [2‡3] = \langle b, c \rangle$.

Following process mining conventions, we consider a universe of *activities* that can be performed, the set $\mathcal{A}$. Given a set of activities for some process $A \subseteq \mathcal{A}$, *traces* are sequences of which activities are performed for a particular case, $\sigma \in A^*$. Structural elements of a process model which do not correspond to a recorded activity are *silent*, and represented by τ.

3 Process Trees as Weighted Automata

3.1 Weighted Automata

Stochastic Finite Automata [11], also called Probabilistic Finite Automata [11, 4], are a well-studied formalism consisting of states, and transitions between them, governed by a probability function with static odds. We make use of a particular class of SFAs where probabilities are derived from weights on arcs, and there is a single terminal state. Our definition builds on the literature on common semantics for stochastic models [4].

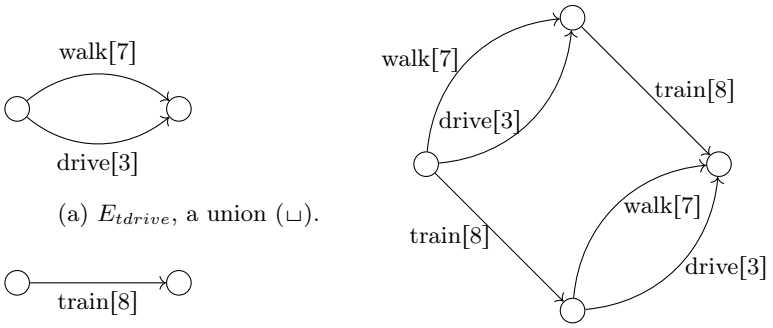
Definition 1 (Weighted Stochastic Finite Automata). *A* Weighted Stochastic Finite Automaton *(WSFA) is a five-tuple* $(S, Act, \hookrightarrow, s_0, s_\omega)$*, where*

- S *is a non-empty, finite set of states,*
- Act *is the activities performed by the automata,* $Act \subseteq \mathcal{A} \cup \{\tau\}$
- $\hookrightarrow \subset S \times Act \times S \to \mathbb{R}^+$ *is a weight function,*
- $GS \subset S \times Act \times S \to [0, 1]$ *is a transition function, from one state to another, generating a symbol, at a given probability,*
- GS *fully connects the states in* S*,*
- $GS(s_1, x, s_2) = \frac{w}{W_T} \equiv \hookrightarrow (s_1, x, s_2) = w$
  where $W_T = \sum_{s_3 \in S, y \in Act} \hookrightarrow (s_1, y, s_3)$ *where* $\hookrightarrow$ *is defined,*
- $s_0 \in S$ *is the initial state, and has no incoming arcs*
- $s_\omega \in S$ *is the final state, and a deadlock state, with no outgoing arcs.*

For notational convenience, $\hookrightarrow (s_1, x, s_2) = w$ is written as $s_1 \overset{x[w]}{\hookrightarrow} s_2$. The set of all WSFAs is $\mathcal{WS}$. The sequence of symbols generated by proceeding from the initial state form a path. Traces generated by the model are finite sequences of activities on paths which include the final state.

$s_1 \overset{x[w]}{\hookrightarrow} s_2$ can also be used to indicate an instance member of the transition set. Figure 1a shows a simple example WSFA, E_{tdrive}, with a choice between activities *walk* and *drive*. $\hookrightarrow = \{s_0 \overset{walk[7]}{\hookrightarrow} s_\omega, s_0 \overset{drive[3]}{\hookrightarrow} s_\omega\}$, meaning there is a $\frac{7}{10}$ chance of walking and a $\frac{3}{10}$ chance of driving.

3.2 Operations on Weighted Automata



(a) E_{tdrive}, a union ($\sqcup$).

(b) E_{ttrain}.

(c) Two trips executed in a race, $E_{tdrive} \,||\, E_{ttrain}$.

Fig. 1: WSFA travel examples.

We define concatenation, union and race operators on weighted automata. We follow the convention that states in operand automata are disjoint [10], as it is straightforward to construct a copy function to ensure it.

Concatenation ($+\!\!+$) chains two automata together.

Definition 2 (WSFA concatenation). *Two automatons execute serially. Let* $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ *and* $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$ *be WSFAs. Further let* E_1 *and* E_2 *be WSFA already made disjoint through copying if necessary. Then* $E_1 +\!\!+ E_2 = (S^{+\!\!+}, A^{+\!\!+}, \hookrightarrow^{+\!\!+}, s_0^{+\!\!+}, s_\omega^{+\!\!+})$*, with:*

$$
\begin{aligned}
S^{+\!\!+} &= S_1 \cup S_2 \cup \{(s_{1.\omega}, s_{2.0})\} \backslash \{s_{1.\omega}, s_{2.0}\} \\
A^{+\!\!+} &= A_1 \cup A_2 \quad \textit{and} \quad s_0^{+\!\!+} = s_{1.0} \quad \textit{and} \quad s_\omega^{+\!\!+} = s_{2.\omega} \\
\hookrightarrow^{+\!\!+} &= \hookrightarrow_1 \cup \hookrightarrow_2 \cup \{s_1 \overset{x[w]}{\hookrightarrow} (s_{1.\omega}, s_{2.0}) \mid s_1 \overset{x[w]}{\hookrightarrow}_1 s_{1.\omega}\} \\
&\quad \cup \{(s_{1.\omega}, s_{2.0}) \overset{x[w]}{\hookrightarrow} s_2 \mid s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s_2\} \backslash \{s \overset{x[w]}{\hookrightarrow} s' \mid s = s_{2.0} \vee s' = s_{1.\omega}\}
\end{aligned}
$$

The disjointness condition ensures that no states are shared between operands in operations like $E +\!\!+ E$. A new shared state, $(s_{1.\omega}, s_{2.0})$, replaces the final state of the left operand and the initial state of the right operand. WSFA concatenation is associative, allowing extension to a multi-child operator by composition.

A weighted disjoint union operator $\sqcup$ allows exactly one of its operands to completely execute. It constructs a new automaton that may behave as either of its component WSFAs, with a probability determined by the relative weights of the existing arcs.

Definition 3 (WSFA union). *Let* $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ *and* $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$ *be WSFAs already made disjoint through copying. Then*

$E_1 \sqcup E_2 = (S^{\sqcup}, A^{\sqcup}, \hookrightarrow^{\sqcup}, s_0^{\sqcup}, s_\omega^{\sqcup})$, *with:*

$$
\begin{aligned}
S^{\sqcup} &= S_1 \cup S_2 \cup \{(s_{1.0}, s_{2.0}), (s_{1.\omega}, s_{2.\omega})\} \backslash \{s_{1.0}, s_{2.0}, s_{1.\omega}, s_{2.\omega}\} \\
A^{\sqcup} &= A_1 \cup A_2 \quad \textit{and} \quad s_0^{\sqcup} = (s_{1.0}, s_{2.0}) \quad \textit{and} \quad s_\omega^{\sqcup} = (s_{1.\omega}, s_{2.\omega}) \\
\hookrightarrow^{\sqcup} &= \hookrightarrow_1 \cup \hookrightarrow_2 \cup \{(s_{1.0}, s_{2.0}) \overset{x[w]}{\hookrightarrow} s \mid s_{1.0} \overset{x[w]}{\hookrightarrow}_1 s \vee s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s\} \\
&\quad \cup \{s \overset{x[w]}{\hookrightarrow} (s_{1.\omega}, s_{2.\omega}) \mid s \overset{x[w]}{\hookrightarrow}_1 s_{1.\omega} \vee s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s_{2.\omega}\} \\
&\quad \backslash \{s \overset{x[w]}{\hookrightarrow} s' \mid s \in \{s_{1.0}, s_{2.0}\} \vee s' \in \{s_{1.\omega}, s_{2.\omega}\}\}
\end{aligned}
$$

As a gloss, the operator merges the initial and final states of the original automata, which establishes a weighted choice over which automata to execute.

In a weighted race, represented by $||$, two automatons progress in parallel. At each step, the child automaton to progress next is determined by a weighted choice between arcs. The result is a Cartesian product of possible states.

Definition 4 (WSFA race). *Let* $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ *and* $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$ *be WSFAs already made disjoint through copying. Then* $E_1 \,||\, E_2 = (S^{||}, A^{||}, \hookrightarrow^{||}, s_0^{||}, s_\omega^{||})$, *with:*

$$
\begin{aligned}
S^{||} &= \{(s_1, s_2) \mid s_1 \in S_1 \wedge s_2 \in S_2\} \\
A^{||} &= A_1 \cup A_2 \quad \textit{and} \quad s_0^{||} = (s_{1.0}, s_{2.0}) \quad \textit{and} \quad s_\omega^{||} = (s_{1.\omega}, s_{2.\omega}) \\
\hookrightarrow^{||} &= \{(s_{1.y}, s_{2.y}) \overset{x[w]}{\hookrightarrow} (s_{1.z}, s_{2.y}) \mid s_{1.y} \overset{x[w]}{\hookrightarrow}_1 s_{1.z}\} \\
&\quad \cup \{(s_{1.y}, s_{2.y}) \overset{x[w]}{\hookrightarrow} (s_{1.y}, s_{2.z}) \mid s_{2.y} \overset{x[w]}{\hookrightarrow}_2 s_{2.z}\}
\end{aligned}
$$

In a race between WSFAs, there is no "communication" between states in the sense used by process algebras (synchronous shared events). WSFA concatenation, union and race relations are associative, allowing extension to multi-child operators by composition. Union and race are also commutative.

Figure 1c shows the automata for two trips conducted in a race (say between two travellers). A second traveller takes the train:

$$E_{ttrain} = (\{s_0, s_\omega, \{train\}, \{s_0 \overset{train[8]}{\hookrightarrow} s_\omega\}, s_0, s_\omega)$$

The resulting WSFA $E_{tdrive} \,||\, E_{ttrain}$ simulates both automata executing at once, with each transition progressing one automaton or the other.

3.3 Probabilistic Process Trees

Probabilistic Process Trees [2] can be defined using stochastic finite automata, where probabilities are derived from weights on each node.

Definition 5 (Probabilistic Process Trees (PPTs)). *Let* $x{:}\,w$ *be a node, where* $w \in \mathbb{R}^+$ *is a weight and* x *the remainder. The universe of PPTs is* $\mathcal{PPT}$, *recursively and exhaustively defined as:*

1. *A single activity. For* $a \in \mathcal{A}$, $a\colon w \in \mathcal{PPT}$.
2. *A silent activity, represented by the constant* τ. *Note* $\tau \notin \mathcal{A} \wedge \tau\colon w \in \mathcal{PPT}$.
3. *A unary operator* $\oplus_1$. *Given* $u \in \mathcal{PPT}$, *then* $\oplus_1(u)\colon w \in \mathcal{PPT}$.
4. *An n-ary operator* $\oplus_n$ *over one or more child trees. Given* $m \geqslant 1$, $u_1, ..., u_m \in \mathcal{PPT}$, *then* $\oplus_n(u_1, ..., u_m)\colon w \in \mathcal{PPT}$.

The PPT operators are $\bigoplus = \{\rightarrow, \wedge, \times, \circlearrowleft_n, \circlearrowleft_p\}$: sequence, concurrency, choice, a fixed loop, and a probabilistic loop. These are related to control-flow process tree operators [8], but include weight semantics. These semantics are described by an equivalent WSFA for each node, which recursively defines the semantics function $\mathbf{wa} : \mathcal{PPT} \rightarrow \mathcal{WS}$.

$$
\begin{aligned}
\text{Leaf nodes. } & \mathbf{wa}(x\colon w) = (\{s_0, s_\omega\}, \{x\}, \{s_0 \overset{x[w]}{\hookrightarrow} s_\omega\}, s_0, s_\omega) \text{ for } x \in \mathcal{A} \cup \{\tau\} \\
\text{Sequence. } & \mathbf{wa}(\rightarrow(x_1\colon w, x_2\colon w)\colon w) = \mathbf{wa}(x_1\colon w_1) +\!\!+ \mathbf{wa}(x_2\colon w_2) \\
& \forall \rightarrow(x_1\colon w_1, x_2\colon w_2)\colon w \blacksquare w_1 = w_2 = w \\
\text{Choice. } & \mathbf{wa}(\times(x_1\colon w_1, x_2\colon w_2)\colon w) = \mathbf{wa}(x_1\colon w_1) \sqcup \mathbf{wa}(x_2\colon w_2) \\
& \forall \times(x_1\colon w_1, x_2\colon w_2)\colon w \blacksquare w = \sum_{i=1}^{i=m} w_i \\
\text{Concurrency. } & \mathbf{wa}(\wedge(x_1\colon w_1, ..., x_m\colon w_m)\colon w) = (S^\wedge, A^\wedge, \hookrightarrow^\wedge, s_0, s_\omega) \\
\text{where } & (S_r, A_r, \hookrightarrow_r, s_{r.0}, s_{r.\omega}) = x_1\colon w_1 \,||\, ... \,||\, x_m\colon w_m \\
& S^\wedge = S_r \cup \{s_0, s_\omega\} \quad \text{and} \quad A^\wedge = A_r \cup \{\tau\} \\
& \hookrightarrow^\wedge \;=\; \hookrightarrow_r \cup \{s_0 \overset{\tau[w]}{\hookrightarrow} s_{r.0}, s_{r.\omega} \overset{\tau[w]}{\hookrightarrow} s_\omega\} \\
& \forall \wedge(x_1\colon w_1, ..., x_m\colon w_m)\colon \blacksquare w = \sum_{i=1}^{i=m} w_i \\
\text{Fixed loops. } & \circlearrowleft_n^m(u)\colon w \equiv \rightarrow(u, \ ... \ m \text{ times } ...)\colon w \\
& \forall \circlearrowleft_n^m(x\colon w_1)\colon w_2 \blacksquare w_1 = w_2 \\
\text{Probabilistic loops. } & \mathbf{wa}(\circlearrowleft_p^\rho (x\colon w)\colon w) = (S_L, A_L, \hookrightarrow_L, s_0, s_\omega) \\
\text{where } & \mathbf{wa}(x\colon w) = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega}) \\
& S_L = S_1 \cup \{s_0, s_\omega\} \backslash \{s_{1.\omega}\} \quad \text{and} \quad A_L = A_1 \cup \{\tau\} \\
& \hookrightarrow_L \;= \{s_0 \overset{\tau[w]}{\hookrightarrow} s_{1.0}\} \cup \{s_{1.0} \overset{\tau[\frac{w}{\rho}]}{\hookrightarrow} s_\omega\} \\
& \quad \cup \{s_{1.1} \overset{x[v]}{\hookrightarrow} s_{1.2} \mid s_{1.1} \overset{x[v']}{\hookrightarrow}_1 s_{1.2} \wedge s_{1.2} \neq s_{1.\omega} \wedge v = \frac{v' \cdot (\rho - 1)}{\rho}\} \\
& \quad \cup \{s_{1.1} \overset{x[v]}{\hookrightarrow} s_{1.0} \mid s_{1.1} \overset{x[v']}{\hookrightarrow}_1 s_{1.\omega} \wedge v = \frac{v' \cdot (\rho - 1)}{\rho}\}
\end{aligned}
$$

This concludes Definition 5, formally describing PPTs. Figure 2 shows a PPT modelling a realistic example, and its WSFA equivalent, which describes both variations and probabilities in a simple insurance claim process.

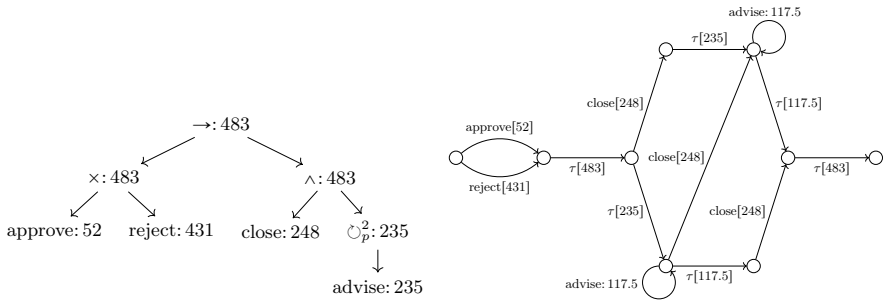


Fig. 2: Detail of a claims process as a Probabilistic Process Tree (PPT) and the equivalent WSFA.

4 Trace Probability For Weighted Stochastic Finite Automata

Weighted Stochastic Finite Automata (WSFAs) are built around probabilities for transitioning between states, so have a trace probability calculation implicit in their design. However, WSFA, like many automata, have no special treatment for silence. τ labels are treated as just another symbol. The presence of silent loops, as in the example in Figure 3, make the number of paths potentially infinite. This is similar to the problem of establishing the stochastic language for SLPNs with silent transitions and loops [7]. A naive solution ignoring silent loops is straightforward [1]; below is a solution which accommodates silent loops within an approximation bound.

To handle silent loops we introduce $\pi_{st\epsilon}$, a total function which returns the trace probability for a given state accurate to an approximation bound.

$$\pi_{st\epsilon}\colon A^* \times \mathcal{WS} \times S \times (0,1] \times [0,1] \times \mathbb{P}(S) \to [0,1]$$

where S, A are sets of states and activities for the input WSFA

Let $E = (S, A \cup \{\tau\}, \hookrightarrow, s_0, s_\omega)$

Inputs: trace σ, WSFA E, state s, approximation bound ϵ, cumulative probability p, and visited states *seen*

$$\pi_{st\epsilon}(\langle a \rangle + \sigma, E, s, \epsilon, p, seen) = \frac{1}{w_T} \sum_{T_a} w \cdot \pi act + \frac{1}{w_T} \sum_{T_\tau} w \cdot \pi\tau \text{ , if } p > \epsilon$$

$$\pi_{st\epsilon}(\langle a \rangle + \sigma, E, s, \epsilon, p, seen) = \frac{1}{w_T} \sum_{T_a} w \cdot \pi act \text{ , if } p \leqslant \epsilon$$

$$\pi_{st\epsilon}(\langle\rangle, E, s_\omega, \epsilon, p, seen) = 1 \text{ ; expected termination}$$

$$\pi_{st\epsilon}(\langle\rangle, E, s, \epsilon, p, seen) = \frac{1}{w_T} \sum_{T_\tau} w \cdot \pi_{st\epsilon}(\langle\rangle, E, s', \epsilon', \frac{pw}{w_T}, seen \cup \{s\})$$

if $p > \epsilon$ and $s \neq s_\omega$

$$\pi_{st\epsilon}(\langle\rangle, E, s, \epsilon, p, seen) = 0 \text{ , if } p \leqslant \epsilon \text{ and } s \neq s_\omega \text{ ; terminate repeated path}$$

where $T_a = \{s \overset{a[w]}{\hookrightarrow} s'\}$ are activity arcs, $T_\tau = \{s \overset{\tau[w]}{\hookrightarrow} s'\}$ are silent arcs

$$w_T = \sum_{s \overset{x[w]}{\hookrightarrow} s'} w$$

$$\pi act = \pi_{st\epsilon}(\sigma, E, s', \epsilon', \frac{pw}{w_T}, seen \cup \{s\})$$

$$\pi\tau = \pi_{st\epsilon}(\langle a \rangle + \sigma, E, s', \epsilon', \frac{pw}{w_T}, seen \cup \{s\})$$

$$\epsilon' = \begin{cases} \epsilon & \text{where } s \in seen \text{ ; retain threshold for repeated state} \\ \frac{w \cdot \epsilon}{w_T} & \text{otherwise; scale down for new state} \end{cases}$$

Given an input state, $\pi_{st\epsilon}$ looks at all the transitions from that state. For those that match the head of the input trace, it takes the chance of that transition multiplied by the probability of the suffix subtrace. For those that are silent, it takes the chance of those multiplied by the entire input trace. For the empty trace, if the WSFA is at the terminal state, it's a match. If it is not, only silent transitions can still result in a match, and any other symbol terminates the recursion. Improbable paths involving silence are treated as probability zero. To achieve this, the function keeps track of the visited states in the parameter *seen*, and the cumulative trace probability. If visiting a new state, the suppression threshold, ϵ, is scaled down, so that the overall probability error will be within the original threshold. If visiting a previously seen state, the suppression threshold is unchanged. So in a loop involving silence, the current probability will monotonically reduce, while the suppression threshold will not, eventually terminating the calculation. Overall trace probability function $\pi_{w\epsilon}$ is then:

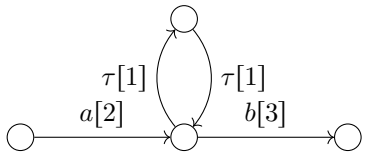


Fig. 3: Example WSFA with silent loop, $E_{\tau loop}$.

$$\pi_{w\epsilon}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega), \epsilon) = \pi_{st\epsilon}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega), s_0, \epsilon, 1, \varnothing)$$

An example probability calculation uses the WSFA $E_{\tau loop}$ in Figure 3.

$$\pi_{w\epsilon}(\langle a, b \rangle, \pi_{w\epsilon}, \frac{1}{10}) = \overbrace{\frac{2}{2} \cdot \frac{2}{3}}^{\text{path } \langle a,b \rangle} + \underbrace{\frac{2}{2} \cdot \frac{1}{3} \cdot \frac{2}{3}}_{\text{path } \langle a,\tau,\tau,b \rangle} + \overbrace{\frac{2}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3}}^{\text{path } \langle a,\tau,\tau,\tau,\tau,b \rangle}$$
$$+ \underbrace{\frac{2}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 0 \cdot \frac{2}{3}}_{\text{path } \langle a,\tau,\tau,\tau,\tau,\tau,b \rangle \text{ and thereafter}}$$
$$= \frac{2}{3} + \frac{2}{9} + \frac{2}{27} = \frac{26}{27}$$

As $E_{\tau loop}$ supports only one valid trace, we can see that the cumulative sum should be equal to 1, but is instead approximated by $\frac{26}{27}$, with an error of $\frac{1}{27}$, within the bound of $\frac{1}{10}$.

5 Trace Probability on Probabilistic Process Trees

Using the example model in Figure 2, u_{claim}, we might ask the probability that claimants have to be advised of a successful claim outcome exactly once. Using the techniques in this section it can be shown this is:

$$\pi_\epsilon(\langle \text{approve}, \text{close}, \text{advise} \rangle, u_{claim}, 0.01) = 0.021$$

This section presents a bounded approximation for TRACE-PROB on PPTs using terminating recursive function π_ϵ.

Definition 6 (Trace Probability and Bound Approximation for PPTs). *Let $A \subseteq \mathcal{A}$ be the activities for a given PPT.*

$$\pi_\epsilon \colon A^* \times \mathcal{PPT} \times (0,1] \to [0,1]$$

$\pi_\epsilon(\sigma, u, \epsilon)$ gives the trace probability for trace σ on PPT u within error range ϵ, defined by cases.

Leaf Nodes Activity and silent nodes have a trace probability of zero or one.

$$\pi_\epsilon(\langle a \rangle, a \colon w, \epsilon) = 1 \qquad \pi_\epsilon(\langle\rangle, \tau : w, \epsilon) = 1$$

$$\text{otherwise, } \pi_\epsilon(\sigma, x \colon w, \epsilon) = 0 \text{ where } x \in A \cup \{\tau\}$$

Choice The choice operator $\times$ reflects a weighted sum of subtree probabilities.

$$\pi_\epsilon(\sigma, \times(x_1 \colon w_1, ..., x_m \colon w_m) \colon w, \epsilon) = \frac{1}{w_T} \sum_i^m w_i \cdot \pi_\epsilon(\sigma, x_i \colon w_i, \frac{w_i \cdot \epsilon}{w_T})$$

$$\text{where } w_T = \sum_i^m w_i$$

Concurrency The state space explosion described by concurrency $\wedge$ requires calculation at the level of weighted automata.

$$\pi_\epsilon(\sigma, \wedge(u_1, u_2, ...) \colon w, \epsilon) = \pi_{w\epsilon}(\sigma, \mathbf{wa}(\wedge(u_1, u_2, ...) \colon w, \epsilon)$$

Sequences Trace probability for sequences $\to$ first evaluates the empty trace probability. Function π_S then considers each possible non-empty two-way split of input trace σ. The function takes a trace, a sequence index, and a PPT model as parameters. This also applies to fixed loops $\circlearrowleft_n$.

$$\pi_S \colon A^* \times \mathbb{N} \times \mathcal{PPT} \times (0,1] \to [0,1]$$

Function π_S splits at index n and recurses

$$\begin{aligned}
\pi_S(\sigma, n, \to(u_1, u_2, ...) \colon w) &= \pi_\epsilon(\sigma[1\ddagger n], u_1), \epsilon) \cdot \pi_\epsilon(\sigma[n+1\ddagger|\sigma|], \to(u_2, u_3, ...) \colon w, \epsilon) \\
&\quad + \pi_S(\sigma, n+1, \to(u_1, u_2, ...) \colon w) \\
&\quad \text{where } n < |\sigma| \\
\pi_S(\sigma, |\sigma|, \to(u_1, u_2, ...) \colon w) &= \pi_\epsilon(\sigma, u_1) \cdot \pi_\epsilon(\langle\rangle, \to(u_2, u_3, ...) \colon w, \epsilon) \text{ , terminal case} \\
\text{otherwise, } \pi_S(\sigma, u) &= 0 \\
\pi_\epsilon(\sigma, \to(u_1, u_2, ...) \colon w) &= \pi_\epsilon(\langle\rangle, u_1) \cdot \pi_\epsilon(\sigma, \to(u_2, ...) \colon w) \\
&\quad + \pi_S(\sigma, 1, \to(u_1, u_2, ...) \colon w)
\end{aligned}$$

Fixed Loops Fixed loops are treated as sequences.

$$\pi_\epsilon(\sigma, \circlearrowleft_n^m(u)\colon w, \epsilon) = \pi_\epsilon(\sigma, \rightarrow(u, ...m \text{ times...})\colon w)$$

Approximating Loops Including Silence In a probabilistic loop including arbitrary silent activities, the trace probability can be approximated within a probability bound ϵ. Loosely speaking, the loop is "unrolled" until the probability of any further execution is non-zero but very small. Let $\pi_{ppt}(\sigma, u)$ give the (possibly intractable) trace probability for trace σ on PPT u.

Execution of a loop i times is the definition of a fixed loop, equivalent to a sequence $\rightarrow$ of length i.

$$\pi_{ppt}(\sigma, \circlearrowleft_p^\rho(u)\colon w) = \sum_{i=0}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w)$$

Choose k such that $\frac{(\rho-1)^{k+1}}{(\rho)^k} \leqslant \epsilon$. Let $q(i) = \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i$.

$$\pi_{ppt}(\sigma, \circlearrowleft_p^\rho(u)\colon w) = \sum_{i=0}^{i=k} q(i) \cdot \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w)$$
$$+ \sum_{i=k+1}^{i=\infty} q(i) \cdot \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \text{ partition at } k \text{ iterations}$$

$$\sum_{i=k+1}^{i=\infty} q(i) = \sum_{i=0}^{i=\infty} q(i) - \sum_{i=0}^{i=k} q(i) \text{ expand term for iterations } > k$$

$$\text{Let } r = \frac{\rho-1}{\rho} \text{ and note by definition } \rho > 0$$

$$\sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i = r\frac{1}{1-r} - r\frac{1-r^{k+1}}{1-r} \text{ by sum of geometric series}$$

$$= \frac{r^{k+1}}{1-r} = \frac{(\rho-1)^{k+1}}{(\rho)^k}$$

$$\sum_{i=k+1}^{i=\infty} q(i) \leqslant \epsilon \text{ by the definition of } k \text{ and } \epsilon$$

$$\text{Since probability bounds apply, } 0 \leqslant \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \leqslant 1$$

$$\text{Replace } \sum_{i=k+1}^{i=\infty} q(i) \text{ with } \epsilon \text{ to give the inequality}$$

$$\sum_{i=0}^{i=k} q(i) \cdot \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \leqslant \pi_{ppt}(\sigma, \circlearrowleft_p^\rho(u)\colon w) \leqslant \sum_{i=0}^{i=k} q(i) \cdot \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) + \epsilon$$

This final inequality puts upper and lower bounds on the π_{ppt} function, with a width of ϵ. Accordingly, an approximation function is

$$\pi_\epsilon(\sigma, \circlearrowleft_p^\rho(u)\colon w, \epsilon) = \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_\epsilon(\sigma, \circlearrowleft_n^i(u)\colon w, \epsilon)$$

The contribution of a subtree approximation range to a trace probability approximation never increases with inclusion in a larger tree. Consider PPT u with child u_s, and trace σ with subtrace σ_s. The trace probability for σ and u is conditional on the behaviour of u_s, or $\pi(\sigma, u) = Pr(\sigma \text{ matches } u | \sigma_s \text{ matches } u_s)$. As conditional probability multiplies by a value less than one, the probability π_ϵ and the approximation ϵ are not increased, so the result remains bounded by ϵ.

Exact solutions for empty traces on loops, and for loops of a single leaf activity, also exist, using properties of the sum of a geometric progression [1].

5.1 Example Trace Probability Calculation

In Figure 4 we have an example PPT with sequence, choice, a probabilistic loop, and a silent activity. The PPT contains a probabilistic loop including silence, so the probability must be approximated within a bound.

$$\pi_\epsilon(\langle a, b\rangle, u_{loop}, \frac{1}{10}) = 1 \cdot \pi_\epsilon(\langle b\rangle, \circlearrowleft_p^2 (\times(b{:}\,8, \tau{:}\,2){:}\,10){:}\,10)$$

From parameters, ρ =2 and $\epsilon = \frac{1}{10}$

$$\text{So } \frac{1^{k+1}}{2^k} \leqslant \frac{1}{10} \text{ and hence } k = 3$$

$$\pi_\epsilon(\langle a, b\rangle, u_{loop}, \frac{1}{10}) = \sum_{i=0}^{i=3} \frac{1}{2} \cdot \frac{1}{2^i} \pi_\epsilon(\langle b\rangle, \circlearrowleft_n^i (\times(b{:}\,8, \tau{:}\,2){:}\,10), \frac{1}{10})$$

$$\begin{aligned} &= 0 \text{ for } i = 0 \\ &\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \text{ for } i = 1 \\ &\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \cdot \frac{4}{5} \text{ for } i = 2 \\ &\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \cdot \frac{4}{5} \cdot \frac{4}{5} \\ &\qquad \text{for } i = 3 \\ &= 0.312 \end{aligned}$$

The probability calculation for the concurrency operator is per the WSFA. Using the example model in Figure 2, consider the probability that an approved claim is closed before the client is advised, as in the trace $\sigma_{e1} = \langle \text{approve}, \text{close}, \text{advise}\rangle$. Using the the formulae in this section, and an epsilon of 0.001, probability $\pi_\epsilon(\sigma_{e1}, 0.001) = 0.021$, rounded to three places. Including cases where the claimant is advised up to five times, the probability is 0.040, or 4%.

5.2 Complexity

In evaluating computational complexity, we can recognise that the choices taken by the trace probability calculation form a tree of possible paths we will call a *path tree*. This is similar to a prefix tree [6, p492]. In the worst case, where all of the model is within a concurrent tree, the computational complexity of path tree navigation is exponential, from the number of combinations of ordered strings. The time bound is $O(n \cdot k \cdot 2^m)$ where n is the number of nodes in the original PPT, k is the loop unrolling approximation bound, and m is the largest concurrent subtree.

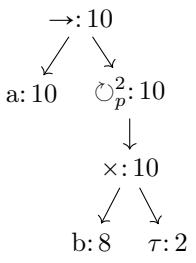


Fig. 4: Example PPT u_{loop} with sequence, choice, probabilistic loop, and silence.

Let the size of the path tree model be n_t. The sequence and loop calculations consider $|\sigma|$ splits of input σ. The worst case complexity bound for the probability calculation is then $O(|\sigma|^2 \cdot n_t)$. The path tree can be conceptual, or realised as a concrete data structure that can be reused on subsequent calculations, if the approximation bound is held constant. In this latter case, the memory complexity matches the time complexity for the worst case.

Though the worst case bound is poor, the trace probability calculation is sensitive to model quality. Simple models reduce cost by reducing n_t. High fitness and low-precision models typically rely on loops for their generality, such as the Petri net flower model. Precise models may have many process tree nodes, n, where complexity scales linearly, but the worst case stems from large concurrent regions, which are also often imprecise. The expansive state space will result in a large k and therefore n_t, where a more precise model will not.

6 Discussion and Related Work

Probabilistic Process Trees were introduced for stochastic process model discovery [2] as an extension of process trees [8] allowing translation to weighted stochastic Petri nets. These have also been used with genetic miners in a laboratory setting [3]. Though Petri nets are not the focus of this paper, their broad use in process mining motivated the use of a translatable representation.

The trace probability problem, Trace-Prob, was identified in defining a stochastic process quality measure based on the Earth movers' distance [7]. Other solutions to trace probability use linear optimisation [9], sampling on probabilistic grammars [12], and Petri net playout [3]. A stochastic language is a mapping from traces to trace probabilities, and is potentially infinite for models with loops. Many existing stochastic quality metrics rely on the stochastic language of a process model as an input (per a recent survey [3]). Practical use of stochastic languages in calculating metrics then requires substituting some kind of finite representations. The solutions presented here, by using a fixed approximation bound, give a clear quantitative criteria for excluding traces from these languages, e.g., when below a probability threshold.

A recent and alternative stochastic extension of process trees [5] associates nodes with fixed probabilities rather than weights. This deliberately eschews a direct translation to stochastic forms of Petri nets, while still allowing calculation of stochastic languages, and discovery of estimated values for the stochastic attributes of the tree. An approximate trace probability calculation is also included, which depends on a fixed maximum loop length, though the possibility of using a probability bound, as in Sections 4 and 5, is anticipated. The non-local

probability impacts caused by enabled transitions under concurrency, noted in Section 5, are also critiqued in [5] as lacking "structural causality", that is, probability relationships are not always reflected in model artifacts such as graph edges. They also note process tree representations can reduce the number of stochastic parameters to optimise, under an assumption Petri net weights are treated as a separate parameter. Trace probability on concurrenct subtrees appears to have exponential complexity in both [5] and the current paper.

7 Conclusion

We have presented closed form solutions for trace probability on Weighted Stochastic Finite Automata and on Probabilistic Process Trees that work within a parameterised approximation bound. Extensions of this work can include empirical evaluations of these techniques, both independently and integrated into algorithms for stochastic discovery and quality metrics.

References

1. Burke, A.: Process mining with labelled stochastic nets. Ph.D. thesis, Queensland University of Technology (February 2024)
2. Burke, A., Leemans, S.J.J., Wynn, M.T.: Discovering Stochastic Process Models By Reduction and Abstraction. In: Application and Theory of Petri Nets and Concurrency. pp. 312–336. Lecture Notes in Computer Science, Springer (2021)
3. Burke, A.T., Leemans, S.J.J., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M.: A chance for models to show their quality: Stochastic process model-log dimensions. Information Systems **124**, 102382 (Sep 2024)
4. Eisentraut, C., Hermanns, H., Zhang, L.: Concurrency and composition in a stochastic world. In: CONCUR 2010-Concurrency Theory: 21th International Conference. Proceedings 21. pp. 21–39 (2010)
5. Horváth, A., Ballarini, P., Cry, P.: Probabilistic Process Discovery with Stochastic Process Trees. In: EAI VALUETOOLS 2024. Milan (2024), arXiv:2504.05765
6. Knuth, D.: The Art of Computer Programming: Volume 3: Sorting and Searching. Pearson Education (1998)
7. Leemans, S.J.J., van der Aalst, W.M.P., Brockhoff, T., Polyvyanyy, A.: Stochastic process mining: Earth movers' stochastic conformance. Information Systems **102**, 101724 (Feb 2021)
8. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Software & Systems Modeling **17**(2), 599–631 (2018)
9. Leemans, S.J.J., Maggi, F.M., Montali, M.: Reasoning on Labelled Petri Nets and Their Dynamics in a Stochastic Setting. In: Business Process Management. pp. 324–342. Lecture Notes in Computer Science (2022)
10. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing **2**(2), 250–273 (1995)
11. Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., Carrasco, R.C.: Probabilistic finite-state machines - part I. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(7), 1013–1025 (Jul 2005)
12. Watanabe, A., Takahashi, Y., Ikeuchi, H., Matsuda, K.: Grammar-Based Process Model Representation for Probabilistic Conformance Checking. In: 2022 4th International Conference on Process Mining (ICPM). pp. 88–95. IEEE (2022)