Reasoning on Labelled Petri Nets and their Dynamics in a Stochastic Setting

Sander J.J. Leemans[1][0000−0002−5201−7125], Fabrizio Maria Maggi[2][0000−0002−9089−6896], and Marco Montali[2][0000−0002−8021−3430]

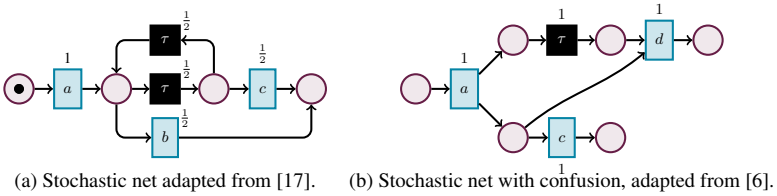
[1] RWTH Aachen, Germany
[2] Free University of Bozen-Bolzano, Italy
{s.leemans}@qut.edu.au, {maggi, montali}@inf.unibz.it

Abstract. Interest in stochastic models for business processes has been revived in a recent series of studies on uncertainty in process models and event logs, with corresponding process mining techniques. In this context, variants of stochastic labelled Petri nets, that is with duplicate labels and silent transitions, have been employed as a reference model. Reasoning on the stochastic, finite-length behaviours induced by such nets is consequently central to solve a variety of model-driven and data-driven analysis tasks, but this is challenging due to the interplay of uncertainty and the potentially infinitely traces (including silent transitions) induced by the net. This explains why reasoning has been conducted in an approximated way, or by imposing restrictions on the model. The goal of this paper is to provide a deeper understanding of such nets, showing how reasoning can be properly conducted by leveraging solid techniques from qualitative model checking of Markov chains, paired with automata-based techniques to suitably handle silent transitions. We exploit this connection to solve three central problems: computing the probability of reaching a particular final marking; computing the probability of a trace or that a temporal property, specified as a finite-state automaton, is satisfied by the net; checking whether the net stochastically conforms to a probabilistic Declare model. The different techniques have all been implemented in a proof-of-concept prototype.

Keywords: Stochastic Petri nets · stochastic process mining · qualitative verification · Markov chains

1 Introduction

In process mining, recorded organisational process data is leveraged to gain insights into business processes by means of analysis techniques. Process mining techniques have traditionally taken the frequency and timing of observed behaviour into account implicitly: depending on the type of analysis performed, the most-occurring happy paths vs. little-occurring deviations, as well as quick vs. slow performing activities, might be of interest and is essential for quantifiable insights. For instance, a quality control process with 30% failed checks is a considerably different process than a process with 2% failed checks, even though the control flow would be equivalent. Explicitly modelling the stochastic perspective of process models – how likely each behaviour is – may assist in obtaining quantifiable insights, and in ensuring the quality of simulation, prediction



(a) Stochastic net adapted from [17]. (b) Stochastic net with confusion, adapted from [6].

Fig. 1: Two examples of labelled stochastic Petri nets.

and recommendation. Recent work includes discovery techniques to automatically discover stochastic process models [22,5].

Stochastic process models (such as generalised stochastic Petri nets [19]) provide this information explicitly by indicating the likelihood and timing of steps in the process and, indirectly, of process behaviour (traces). Not surprisingly, interest in these stochastic models has been revived in the context of process mining, with a series of recent studies focused on: (1) discovery of stochastic process models that indicate the likelihood of behaviour [5,22]; (2) repair of process models [21]; (3) conformance checking, either comparing the stochastic behaviour of a log with that expected by a stochastic process models to gain insights from their differences [15,16], or using the likelihood of model traces when aligning observed traces with a reference model [2].

When attacking these problems, it becomes essential to reason on the stochastic behaviour captured by the process, for example to determine the likelihood of model traces. Traditional techniques relying on the connection between stochastic Petri nets and Markov chains [20,19] cannot be readily applied to this setting, due to key conceptual mismatches related to the usage of stochastic Petri nets to represent business processes. First, transitions in the net must be labelled with corresponding (names of) activities in the processes, possibly using the same label for multiple transitions. Second, silent transitions should be supported, to represent control-flow structures in the process (such as gateways) that do not correspond to any visible activity. Third, when analysing the dynamics of the net the focus is not on infinite, recurring behaviour, but on finite traces representing the possible executions of process instances, moving a case object from the initial to a final state without considering which silent steps have been taken in between.

Supporting all these modelling requirements makes it difficult to actually reason on the traces supported by these nets and their probabilities. To see this, consider the labelled stochastic Petri net shown in Figure 1(a) (we will introduce these nets formally in Section 3). This model has two traces, however computing the likelihood of the traces may be counterintuitive: the likelihood of the trace of a followed by b is $\frac{2}{3}$ [17]. The challenge here stems from the loop of silent transitions, which "favours" b over c.

Another example of potentially counterintuitive likelihoods of traces is shown in 1(b). In this net, the likelihood of a followed by c is $\frac{3}{4}$. The challenge in this example is again the silent transition, which is used here in a semi-concurrent context: the transition c is mutually exclusive with transition d, but as c is part of two runs (that is, executed before or after the silent transition), its probability is higher than one might expect [6, confusion]. In Section 2, we describe how existing techniques address or circumvent these challenges.

The main contribution of the paper is to take stochastic process mining a step further by providing analytic methods to solve the following related problems:

(Outcome probability) Given a stochastic Petri net and a set of final markings, what is the likelihood of a trace of the net ending in one of the markings?

(Verification) Given a temporal property captured by a finite-state automaton (e.g., an LTL_f formula or a Declare model), what is the probability that the net generates a trace satisfying the property? A special case of this problem is calculating the probability of a trace.

(Stochastic model conformance) Given a set of probabilistic temporal constraints [18] and a labelled stochastic Petri net, does the net conform to the set by comparing their stochastic behaviours?

We address these problems by transforming them into problems that can be solved using well-established techniques. In particular, we build on the connection [19] between stochastic Petri nets and Markov chains [11,12], paired with automata-based techniques to handle their qualitative verification against temporal properties [1, Ch.10]. We use the former to compute the probability of reaching a target marking, and the latter to handle silent transitions, and to isolate the behaviour induced by the net that satisfy a property of interest.

The methods have been implemented as part of the ProM framework [10].

2 Related Work

Stochastic process-based models have been studied extensively in literature. In the context of this work, we are interested in formal, Petri net-based stochastic models that are at the basis of the recent series of approaches in stochastic process discovery [5,22] and conformance checking [15,16,3]. Such approaches all refer to the model of (generalised) stochastic Petri nets, or fragments thereof. A first version of this model was proposed in [20], extending Petri nets by assigning exponentially distributed firing *rates* to transitions. This was extended in [19] by distinguishing timed (as in [20]) and immediate transitions. Immediate transitions have priority over timed ones, and have *weights* to define their relative likelihood. As these two types of transitions, abstracting from time, behave homogeneously, we may capture the stochastic behaviour of the net through a discrete-time Markov chain [19].

Several variants of stochastic Petri nets have been investigated starting from the seminal work in [19]. These variants differ from each other depending on the features they support (e.g., arbiters to resolve non-determinism, immediate vs timed transitions) and the way they express probabilities. Such nets may aid modellers in expressing certain constructs. An orthogonal, important dimension is to ensure that probabilities and concurrency interact properly. This can be achieved through good modelling principles [19,6] or automated techniques [4].

Contrasting these formal models with recent works in stochastic process mining, key differences exist. Traditional stochastic nets do not support transition labels nor silent transitions, and put emphasis on recurring, infinite executions and the so-called steady-state analysis, focused on calculating the probability that an execution is currently placed in a given state. This is done by constructing a discrete-time Markov chain that characterises the stochastic behaviour of the net [20,19]. Finding the probability of a finite-length trace in such nets is trivial, as every trace corresponds to a single path. However, no transition labels or silent steps are supported, which limits their usefulness for process mining due to the omnipresence of such transitions in process models. On

the other hand, when these features are incorporated in stochastic Petri nets, which is precisely what we target in this paper, computing the probability of a trace cannot be approached directly anymore, as infinitely many paths would in principle need to be inspected. At the same time, in business processes we are interested in behaviour at the trace level rather than at the process level – that is, traces have a finite length and are in principle independent –, thus the large body of work on steady-state-based analyses does not apply for our purposes. This explains why reasoning on the stochastic behaviour of such extended nets has been conducted in an approximated way [15,16], or by imposing restrictions on the model [3].

To bridge this gap, in this paper we take the most basic stochastic Petri nets: we do not consider time or priority, but we add (duplicate) labels and silent transitions. Importantly, *our results seamlessly carry over bounded, generalised stochastic Petri nets*, thanks to the fact that incorporating priorities in bounded nets is harmless, and that timed and immediate transitions are homogeneous from the stochastic point of view. To the best of our knowledge, outside of recent work using stochastic Petri nets with silent transitions [15,17,3], such nets have not been defined or studied before.

While intuitively stochastic conformance checking techniques need to obtain the probability of a given trace in a stochastic process model (for instance, [17] explicitly obtains this probability to compute a distance measure between a log and a stochastic process model), some stochastic conformance checking techniques avoid computing the probability for a single trace, for instance by playing out the model to obtain a sample of executions [15], or by assuming that the model is deterministic [16]. The results presented in this paper therefore enable the practical application of [17], and may enable further stochastic conformance checking techniques and, consequently, new types of analysis.

Silent steps have been studied in the context of automata. For instance, in [13] an ad-hoc method is described to iteratively remove all silent steps from a stochastic automaton. Due to concurrency and confusion (see for instance Figure 1(b)), such techniques are not directly applicable to stochastic Petri nets. A result of this paper is that silent steps can be handled directly, without the need for ad-hoc techniques.

3 Stochastic Petri Net-Based Processes

We first provide some brief preliminaries on multisets. A multiset a over a set U (which defines the support of the multiset) is a function $a : U \to \mathbb{N}$, where for $u \in U$, $a(u)$ indicates the multiplicity (i.e., the number of occurrences) of u. Given two multisets a and b over U, we write:

- $a + b$ for the *union* of a and b, defined as the multiset that assigns to each $u \in U$ multiplicity $a(u) + b(u)$;
- $a \leq b$ if for every $u \in U$, we have $a(u) \leq b(u)$;
- assuming $a \leq b$, $b - a$ for the *difference* of b and a, defined as the multiset that assigns to each $u \in U$ multiplicity $b(u) - a(u)$.

The set of all multisets over U is defined as $\mathbb{M}(U)$. Multiset a is explicitly represented placing inside squared brackets $[\ldots]$ each element u with non-zero multiplicity, using notation $u^{a(u)}$.

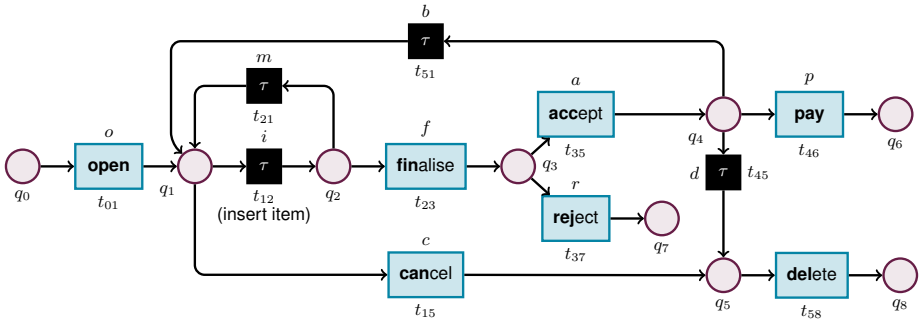


Fig. 2: Stochastic net of an order-to-cash process. Weights are presented symbolically. Transition t_{12} captures a task that cannot be logged, and so is modelled as silent.

3.1 Labelled Petri Nets

As underlying control-flow structure to specify work processes, we consider **Petri nets** that are **labelled** with (atomic) tasks. As customary in Business Process Management, the same task may be used to **label multiple** transitions in the process, and among all labels, we include a special label to indicate **silent steps**, which are internal execution steps of the process that are not explicitly exposed to the external environment (and are thus not recorded in event logs). To capture such labels, we assume a finite set Σ to denote task (names), and a special label $\tau \notin \Sigma$ to indicate a silent step. We also use $\overline{\Sigma} = \Sigma \cup \{\tau\}$ to denote the extended set containing task names and the silent label.

Definition 1 (Labelled Petri net). *A* labelled Petri net N *is a tuple* $\langle Q, T, F, \ell \rangle$*, where: (i)* Q *is a finite set of* places*; (ii)* T *is a finite set of* transitions*, disjoint from* Q *(i.e.,* $Q \cap T = \emptyset$*); (iii)* $F \subseteq (Q \times T) \cup (T \times Q)$ *is a* flow relation *connecting places to transitions and transitions to places; (iv)* $\ell : T \to \overline{\Sigma}$ *is a* labelling function *mapping each transition* $t \in T$ *to a corresponding label* $\ell(t)$ *that is either a task name from* Σ *or the silent label* τ*.* ◁

In the paper, we adopt a dot notation to extract the component of interest from a net, that is, given a net N, its places are denoted by $N.Q$, etc. We will adopt the same notational convention for the other definitions as well. Given a net N and an element $x \in N.Q \cup N.T$, the *preset* and *post-set* of x are respectively defined by ${}^\bullet x = \{y \mid \langle y, x \rangle \in F\}$ and $x^\bullet = \{y \mid \langle x, y \rangle \in F\}$. If x is a transition, then its pre- and post-set respectively denote its input and output places.

Figure 2 shows a labelled Petri net where silent transitions are either used to capture control-flow structures (t_{41} for looping, and t_{45} for rerouting), or tasks that cannot be logged (t_{12}, which represents a non-loggable task for inserting an item). Silent transitions may result from modelling skips, loopbacks, or to start and join concurrent branches, however can also be used to represent processes with non-loggable tasks (Figure 2 even contains a loop of silent transitions). Also, Petri net discovery algorithms may produce nets containing silent loops. This motivates why we study such nets.

An execution state of a net is described by a marking, which is a multiset of places. A transition is enabled in a marking if its input places contain at least one token each.

Firing an enabled transition produces a new marking where one token per input place is consumed, and each output place gets one token more.

Definition 2 (Marking). *A* marking m *of a net* N *is a multiset over the places of* N*, mapping each place* $q \in N.Q$ *to the number* $m(q)$ *of tokens on* q*. Given a marking* m *of* N*, and a transition* $t \in N.T$*, we say that:*
- t *is* enabled *in* m*, written* $m[t\rangle_N$*, if* ${}^\bullet t \leq m$*;*
- $E_N(m)$ *is the set of enabled transitions in a marking* m*.*
- *assuming* $m[t\rangle_N$*,* t fires *in* m *for* N *producing a new marking* m' *of* N*, written* $m[t\rangle_N m'$*, if* $m' = (m - {}^\bullet t) + t^\bullet$*;* ◁

The next definitions are essential to capture that we are interested in finite-trace executions over nets. Specifically, as customary in BPM, each execution represents the evolution of a process instance from the initial state to a final state.

Definition 3 (Execution). *An* execution *of a net* N *from a marking* m_s *to a marking* m_f *of* N *is a (possibly empty) finite sequence* $t_0, \ldots, t_n$ *of transitions in* $N.T$ *such that there exist markings* $m_0, \ldots, m_{n+1}$ *of* N *with (i)* $m_0 = m_s$*, (ii)* $m_{n+1} = m_f$*, (iii) for every* $i \in \{0, \ldots, n\}$ *we have* $m_i[t_i\rangle_N m_{i+1}$*.* ◁

Definition 4 (Deadlock, livelock). *A* marking m *of a net* N *is a:*
- deadlock *if there is no transition enabled:* $E_N(m) = \emptyset$*;*
- livelock *if there is no execution of* N *from* m *to a deadlock marking.* ◁

Definition 5 (Petri net-based process, runs). *A* Petri net-based process (PNP) *is a triple* $\langle N, m_0, M_f\rangle$*, where: (i)* N *is a net; (ii)* m_0 *is a marking of* N *denoting the* initial state*; (iii)* M_f *is a finite set of deadlock markings of* N *denoting its possible* final states*. An execution (Definition 3) starting in* m_0 *and ending in an* $m' \in M_f$ *is a* run*. A PNP* $\mathcal{N}$ *is:*
- deadlock-free *if the only reachable deadlock markings are from* $\mathcal{N}.M_f$*;*
- livelock-free *if* $RG(\mathcal{N})$ *does not contain any livelock marking.* ◁

A single transition in a PNP without an input place makes the net unable to reach a final marking, and thus the PNP has no runs. This is a special case of an unavoidable livelock.

By fixing an initial marking and a set of final markings, we define a process:

Restricting final states to deadlocks markings is without loss of generality: one can take a non-deadlock marking and turn it into a deadlock one by introducing a new silent transition pointing to a dedicated, exclusive "final" deadlock place.

Remark 1. There are two types of execution of a PNP $\mathcal{N}$ that, starting from its initial state, cannot be extended into proper runs:
- Executions ending in a deadlock marking not being a final marking in $\mathcal{N}$;
- Executions in a livelock from which no deadlock marking can be reached. ◁

A *trace* is a sequence $\sigma = e_0, \ldots, e_n \in \Sigma^*$ of *events* over Σ, where, for simplicity, each event e_i indicates the execution of a task by means of the firing of a transition. A trace is a *model trace* for a PNP if it is produced by one of its runs, considering only the visible labels of the transitions contained in the run.

Definition 6 (Model trace). *A trace σ is a* model trace *of PNP $\mathcal{N}$ if there exists a run $\eta = t_0, \ldots, t_m$ of $\mathcal{N}.N$ whose corresponding sequence of labels $\mathcal{N}.N.\ell(t_0), \ldots, \mathcal{N}.N.\ell(t_m)$ coincides with σ once all τ elements are removed. In this case, we say that η* induces *σ.* ◁

A model trace σ of PNP $\mathcal{N}$ may be induced by multiple, possibly infinitely many, runs. The set of runs of $\mathcal{N}$ inducing σ is denoted by $runs_{\mathcal{N}}(\sigma)$.

The execution semantics of a PNP can be described through a reachability graph, namely a (possibly infinite-state) labelled transition system whose states correspond to reachable markings, and whose transitions match transition firings of the PNP.

Definition 7 (Labelled transition system). *A* labelled transition system *is a tuple $\langle S, s_0, S_f, \varrho \rangle$ where: (i) S is a (possibly infinite) set of* states*; (ii) $s_0 \in S$ is the* initial state*; (iii) $S_f \subseteq S$ is the set of* accepting states*; (iv) $\varrho \subseteq S \times \overline{\Sigma} \times S$ is a $\overline{\Sigma}$-labelled* transition relation*. A* run *is a finite sequence of transitions leading from s_0 to one of the states in S_f in agreement with ϱ.* ◁

Due to our requirement that all final markings are deadlock markings, accepting states have no outgoing transitions either.

Definition 8 (Reachability graph). *The* reachability graph *$RG(\mathcal{N})$ of a PNP $\mathcal{N}$ is a labelled transition system $\langle S, s_0, S_f, \varrho \rangle$ whose components are defined by mutual induction as the minimal sets satisfying the following conditions:*

1. *$s_0 = m_0 \in S$;*
2. *for every state $m \in S$, every transition $t \in T$, and every marking $m' \in \mathbb{M}(Q)$, if $m[t\rangle_N m'$ we have that (a) $m' \in S$; (b) if $m' \in \mathcal{N}.M_f$, then $m' \in S_f$; (c) $\langle m, \ell(t), m' \rangle \in \varrho$.* ◁

The runs of $RG(\mathcal{N})$ capture all and only the runs of $\mathcal{N}$. It will be useful later to refer to outgoing transitions from a given state s. We do so with notation $succ_{RG(\mathcal{N})}(s)$.

We close this part by defining some key, standard properties of PNPs. In particular, we fix the last control-flow feature of our model, namely the fact that we focus on **bounded** processes.

Definition 9 (Bounded PNP). *A PNP $\mathcal{N}$ is* bounded *if there exists a number k such that, for every reachable marking $m \in RG(\mathcal{N}).S$ and every place $q \in \mathcal{N}.N.Q$, we have $m(p) \leq k$.* ◁

A key property of bounded PNPs is that they induce a reachability graph that has finitely many states. Boundedness is a standard property assumed when capturing business processes; verifying boundedness is decidable [9] and well-known techniques exist. In the remainder of this paper, we assume bounded PNPs.

3.2 Stochastic Behaviour

We now extend PNPs with stochastic behaviour, by incorporating **stochastic decision making to determine which enabled transition to fire**. Technically, this is done by adding a weight to each transition in a PNP [19]. The probability of firing an enabled transition is the fraction of the weight of the transition compared to the sum of the weights of all enabled transitions.

A *stochastic PNP* is then a PNP of which the transitions similarly have a weight.

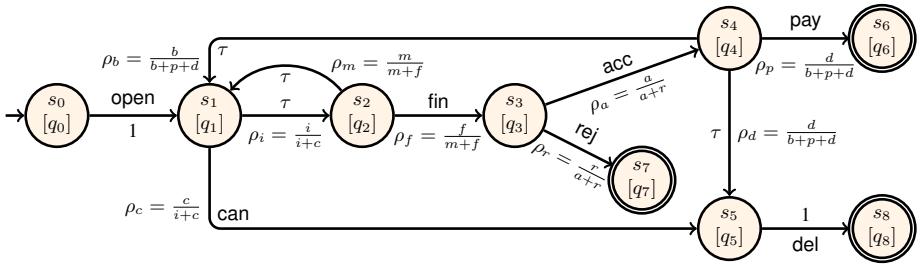


Fig. 3: Stochastic reachability graph of the order-to-cash bounded stochastic PNP. States are named. The initial state is shown with a small incoming edge. Final states have a double contour.

Definition 10 (Stochastic Petri net). *A* stochastic Petri net N *is a tuple* $\langle Q, T, F, \ell, w \rangle$*, where* $\langle Q, T, F, \ell \rangle$ *is a labelled Petri net, and* $W \colon N.T \to \mathbb{R}^+$ *is a* weight function *assigning a positive weight to each transition in* N*. Given a marking* m *of* N*, and an enabled transition* $t \in E_N(m)$*, the* firing probability *of* t *in* m*,* $\mathbb{P}_{m,N}(t)$*, is* $\frac{N.w(t)}{\sum_{t' \in E_N(m)} N.w(t')}$*. [19]* ◁

Please note that stochastic Petri nets inherit the concurrency properties of Petri nets: execution is atomic, and "true" concurrency needs to be added on top. The stochastic perspective may still matter in concurrency, for instance in resource-constrained settings. It is easy to see that the firing probability defines a discrete probability distribution over $E_N(m)$, as $\sum_{t \in E_N(m)} \mathbb{P}_{m,N}(t) = 1$. Then, we can define the semantics of stochastic PNPs using a stochastic transition system and a stochastic reachability graph.

Definition 11 (Stochastic transition system). *A* stochastic transition system *is a tuple* $\langle S, s_0, S_f, \varrho, p \rangle$ *where* $\langle S, s_0, S_f, \varrho \rangle$ *is a transition system, while* p *is a* transition probability function *mapping each transition in* ϱ *to a corresponding probability value in* $[0, 1]$*, such that for every state* $s \in S$*,* $\sum_{\xi \in succ_{\mathcal{N}}(s)} p(\xi) = 1$*.* ◁

The reachability graph $RG(\mathcal{N})$ of a stochastic PNP $\mathcal{N}$ is hence defined as a stochastic transition system obtained as in Definition 8, defining the transition probability function as follows: for every transition $\langle m, \ell(t), m' \rangle$, its probability is set to $\mathbb{P}_{m,\mathcal{N}.N}(t)$.

Example 1. Figure 2 shows an stochastic PNP ($\mathcal{N}_{\text{order}}$) capturing an order-to-cash process. The reachability graph of $\mathcal{N}_{\text{order}}$ is shown in Figure 3, where transition probabilities are calculated using the weights of the stochastic net. If, for example, we fix the weight a of transition t_{35} to 80, and the weight r of transition t_{37} to 20, we get that in marking $[q_3]$ (corresponding to the state where the order has been finalised), there is 0.2 chance that the order is rejected, and 0.8 chance that the order is accepted. ◁

Remark 2. The probability of firing an enabled transition only depends on the current marking, thus stochastic Petri nets and stochastic PNPs are Markovian. ◁

Consequently, to calculate the probability of a run, we may consider each choice therein as independent. In a stochastic PNP $\mathcal{N} = \langle N, m_0, M_f \rangle$, we denote the probability of a run $\eta = t_0, \ldots, t_n$ with $\mathbb{P}_{\mathcal{N}}(\eta)$. Let $m_0, \ldots, m_{n+1}$ be the markings corresponding to η; then $\mathbb{P}_{\mathcal{N}}(\eta) = \prod_{i \in \{1,\ldots,n\}} \mathbb{P}_{m_{i-1},N}(t_i)$. The probability $\mathbb{P}_{\mathcal{N}}(\eta)$ of a

trace σ of $\mathcal{N}$ is in turn obtained by summing up the probabilities of all runs of $\mathcal{N}$ that induce σ: $\mathbb{P}_{\mathcal{N}}(\sigma) = \sum_{\eta \in runs_{\mathcal{N}}(\sigma)} \mathbb{P}_{\mathcal{N}}(\eta)$. Notice that this may be an infinite sum.

Remark 3. Our approach directly lifts to full, bounded generalised stochastic Petri nets [19] as follows: *(i)* transitions are partitioned into *immediate* and *timed* (for the latter, interpreting weights as *rates* of an exponential distribution); *(ii)* a timed transition is enabled if it is so in the usual sense, and *there is no enabled immediate transition*. ◁

4 Outcome Probability

In this section, we tackle a first, fundamental problem: *computing outcome probabilities*, that is, computing what the probability is that a process instance of the bounded stochastic PNP of interest evolves from the initial to one among a desired set of final states (representing the desired outcomes). For example, we may be interested in knowing the probability that the bounded stochastic PNP $\mathcal{N}_{\text{order}}$ of our running example (Figure 2) evolves an order from opening to payment.

Technically, given a bounded stochastic PNP $\mathcal{N}$, we borrow the standard notion of conditional probability and indicate the probability that $\mathcal{N}$ evolves marking m into some marking from a set M as $\mathbb{P}_{\mathcal{N}}(M|m_1)$. Formally, this corresponds to the sum of the probabilities of all executions of $\mathcal{N}$ from m_1 to some marking in M (in the sense of Definition 3). This leads us to the formulation of the OUTCOME-PROB$(\mathcal{N}, F)$ problem:
Input: Bounded stochastic PNP $\mathcal{N}$, set $F \subseteq \mathcal{N}.M_f$ of desired final states;
Output: Probability value $\mathbb{P}_{\mathcal{N}}(F|\mathcal{N}.m_0) = \sum_{\eta \text{ run of } \mathcal{N} \text{ ending in } m \in F} \mathbb{P}_{\mathcal{N}}(\eta)$.
Notice that the same problem can also get, as input, a stochastic transition system in place of a bounded stochastic PNP.

OUTCOME-PROB cannot be solved exactly through an enumeration of runs, as there may be infinitely many. It can be approximated by fixing a maximum threshold either on the length of runs [15], or on their minimum probability [2]. To obtain an exact answer, we build on the connection between bounded stochastic PNPs and discrete-time Markov chains [11], lifting [19] to our setting.[3]

Remark 4. The reachability graph $RG(\mathcal{N}) = \langle S, s_0, S_f, \varrho, p \rangle$ of a bounded stochastic PNP $\mathcal{N}$ can be seen as a discrete-time Markov chain C where: *(i)* S is the finite set of states of C, with s_0 the initial state; *(ii)* S_f are the absorption/exit states of C; *(iii)* ϱ and p define the transition matrix of C, where the entry for a pair $s_1, s_2 \in S$ gets value $p(s, l, s')$ for some label $l \in \overline{\Sigma}$ if $\langle s, l, s' \rangle \in$, 0 otherwise. ◁

We exploit this, noticing that the OUTCOME-PROB problem corresponds to the problem of calculating *exit distributions* in a discrete-time Markov chain [11] (also called the problem of calculating *absorption/hit probabilities* [12]). To analytically solve the problem, we take OUTCOME-PROB$(\mathcal{N}, F)$ and create a system of equations, starting from the reachability graph $RG(\mathcal{N})$. Specifically, each state s of $RG(\mathcal{N}).S$ corresponds to a state variable x_{s_i} denoting the probability $\mathbb{P}_{\mathcal{N}}(F|s)$ of reaching one of the states in F from s; hence $x_{RG(\mathcal{N}).s_0}$ represents the solution of the problem. Then, each equation defines the value of one of the state variables x_s as follows:

[3] In case of generalised stochastic Petri nets, the resulting discrete-time Markov chain is the so-called *embedded/jump* chain obtained from the continuous-time Markov chain capturing the execution semantics of the net [20,19].

Base case if s has no successor states (i.e., is a deadlock marking), then $x_{s_i} = 1$ if s corresponds to a final marking, otherwise $x_{s_i} = 0$;

Inductive case if s has successors, its variable is equal to sum of the state variables of its successor states, weighted by the transition probability to move to that successor.

Formally, OUTCOME-PROB$(\mathcal{N}, F)$ with $RG(\mathcal{N}) = \langle S, s_0, S_f, \varrho, p \rangle$ gets encoded into the following linear optimisation problem $\mathcal{E}^F_{\mathcal{N}}$:

Return x_{s_0} from the minimal non-negative solution of

$$x_{s_i} = 1 \qquad \text{for each } s_i \in F \tag{1}$$

$$x_{s_j} = 0 \qquad \text{for each } s_j \in S \setminus F \text{ s.t. } |succ_{RG(\mathcal{N})}(s_j)| = 0 \tag{2}$$

$$x_{s_k} = \sum_{\langle s_k, l, s'_k \rangle \in succ_{RG(\mathcal{N})}(s_k)} p(\langle s_k, l, s'_k \rangle) \cdot x_{s'_k} \qquad \text{for each } s_k \in S \text{ s.t. } |succ_{RG(\mathcal{N})}(s_k)| > 0 \tag{3}$$

By recalling that states of $RG(\mathcal{N})$ are markings of $\mathcal{N}$, the schema (1) of equations deals with final (deadlock) states, that in (1) with non-final deadlock states, and that in (1) with non-final, non-deadlock states.

$\mathcal{E}^F_{\mathcal{N}}$ has always at least a solution. However, it may be indeterminate and thus admit infinitely many ones, requiring in that case to pick the least committing (i.e., minimal non-negative) solution. The latter case happens when $\mathcal{N}$ contains livelock markings. This is illustrated in the following examples.

Example 2. Consider bounded stochastic PNP $\mathcal{N}_{\text{order}}$ (Figure 2). We want to solve the problem OUTCOME-PROB$(\mathcal{N}_{\text{order}}, [q_6])$, to compute the probability that a created order eventually completes the process by being paid. To do so, we solve $\mathcal{E}^{[q_6]}_{\mathcal{N}_{\text{order}}}$ by encoding the reachability graph of Figure 3 into:

$$x_{s_8} = 0 \qquad x_{s_5} = x_{s_8} \qquad x_{s_2} = \rho_m x_{s_1} + \rho_f x_{s_3}$$
$$x_{s_7} = 0 \qquad x_{s_4} = \rho_b x_{s_1} + \rho_d x_{s_5} + \rho_p x_{s_6} \qquad x_{s_1} = \rho_i x_{s_2} + \rho_c x_{s_5}$$
$$x_{s_6} = 1 \qquad x_{s_3} = \rho_a x_{s_4} + \rho_r x_{s_7} \qquad x_{s_0} = x_{s_1}$$

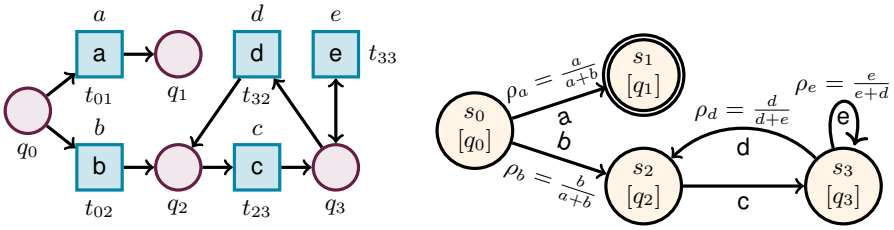
This yields $x_{s_0} = \frac{\rho_i \rho_f \rho_a \rho_p x_{s_6} + \rho_i \rho_f \rho_r x_{s_7} + (\rho_i \rho_f \rho_a \rho_d + \rho_c) x_{s_8}}{1 - \rho_i \rho_m - \rho_i \rho_f \rho_a \rho_b} = \frac{\rho_i \rho_f \rho_a \rho_p}{1 - \rho_i \rho_m - \rho_i \rho_f \rho_a \rho_b}$, which is the only solution. If we assume that the weights of $\mathcal{N}_{\text{order}}$ are all equal, the probability distributions for choosing the next transition are all uniform, leading to $\rho_i = \rho_f = \rho_m = \rho_a = \frac{1}{2}$ and $\rho_p = \rho_b = \frac{1}{3}$, and, in turn, that the probability of completing the process by paying the order is $x_{s_0} = \frac{1}{17} \sim 0.06$.

With an analogous approach, we can prove that the probability that an order gets deleted is $\frac{13}{17}$, and the one that an order gets rejected is $\frac{3}{17}$. Notice that the sum of all such probabilities is, as expected, 1, that is, every order gets paid, deleted or rejected. ◁

Example 3. Consider the bounded stochastic PNP $\mathcal{N}_{\text{live}}$ in Figure 4. To compute the outcome probability of its single final state, we solve $\mathcal{E}^{[q_1]}_{\mathcal{N}_{\text{live}}}$ by encoding the reachability graph of Figure 4(b) into:

$$x_{s_0} = \rho_a x_{s_1} + \rho_b x_{s_2} \qquad x_{s_1} = 1 \qquad x_{s_2} = x_{s_3} \qquad x_{s_3} = \rho_d x_{s_2} + \rho_e x_{s_3}$$

We get $x_{s_3} = \rho_d x_{s_3} + \rho_e x_{s_3} = (\rho_d + \rho_e) x_{s_3} = x_{s_3}$, making the system indeterminate. Its minimal non-negative solution is then the one where $x_{s_3} = 0$, and in turn $x_{s_0} = \rho_a$.◁



(a) Stochastic net.

(b) Reachability graph.

Fig. 4: Reachability graph (b) of a bounded stochastic PNP with net shown in (a), initial marking $[q_0]$ and final marking $[q_1]$. States s_2 and s_3 are livelock markings.

Example 3 illustrates how the technique implicitly gets rid of livelock markings, associating to them a 0 probability. This captures the essential fact that, by definition, a livelock marking can never reach any final marking. More in general, we can in fact solve OUTCOME-PROB$(\mathcal{N}, F)$ by turning the linear optimisation problem $\mathcal{E}^F_{\mathcal{N}}$ into the following system of equalities, which is guaranteed to have exactly one solution:

$$x_{s_i} = 1 \qquad \text{for each deadlock marking } s_i \in F \tag{4}$$

$$x_{s_j} = 0 \qquad \text{for each deadlock marking } s_j \in S \setminus F \tag{5}$$

$$x_{s_k} = 0 \qquad \text{for each livelock marking } s_k \in S \tag{6}$$

$$x_{s_h} = \sum_{\langle s_h, l, s'_h \rangle \in succ_{RG(\mathcal{N})}(s_h)} p(\langle s_h, l, s'_h \rangle) \cdot x_{s'_h} \qquad \text{for each remaining marking } s_h \in S \tag{7}$$

Recall that checking whether a marking s is livelock can be done over $RG(\mathcal{N})$ by checking (non-)reachability of some deadlock marking in $RG(\mathcal{N})$ from s. This check does not involve probabilities at all, but extends to probabilistic settings as per Definition 10, all transitions have a non-zero weight.

5 Qualitative Verification and Trace Probability

We now further leverage the connection between bounded stochastic PNPs and discrete-time Markov chains (cf. Remark 4), to deal with the verification of (qualitative, i.e., non-probabilistic) temporal/dynamic properties over bounded stochastic PNPs. This amounts to compute the probability that a run of the PNP indeed satisfies the property of interest. We rely on [1, Ch. 10] and employ automata-theoretic techniques coupled with the computation of outcome probabilities to solve the problem. We then show how this technique also solves another, related problem: that of computing trace probabilities.

5.1 Verification of Temporal Properties

Properties of interest intensionally describe a (possibly infinite) set of desired finite-length traces that may be induced by runs of the stochastic PNP under scrutiny. Such traces are defined over the task names in Σ (*without* τ). We opt for a very general formalism to describe such properties: (deterministic) finite-state automata.

Definition 12 (DFA, acceptance, language). *A* deterministic finite-state automaton *(DFA) over L is a tuple $A = \langle L, S, s_0, S_f, \delta \rangle$, where: (i) L is a finite* alphabet *of*

symbols; (ii) S is a finite set of states*, with $s_0 \in S$ the* initial state *and $S_f \subseteq S$ the set of* final states*; (iii) $\delta : S \times L \to S$ is a transition transition function that, given a state $s \in S$ and a label $l \in L$, returns the successor state $\delta(s, l)$. A accepts a trace $\sigma = l_0, \ldots, l_n$ over $L^\star$ if there exists a sequence of states $s_0, \ldots, s_{n+1}$ starting from the initial state and such that: (i) $s_{n+1} \in S_f$, and (ii) for every $i \in \{0, \ldots, n\}$, we have $s_{i+1} = \delta(s_i, l_i)$. The* language $\mathcal{L}(A)$ *of A is the set of all traces accepted by A.* ◁

This accounts for non-deterministic automata (NFAs), as each NFA can be encoded into a corresponding DFA. Also, it makes our approach directly operational for other property specification languages, as long as they can get encoded into DFAs. This holds, e.g., when for regular expressions, LTL$_f$/LDL$_f$ temporal formulae over finite traces [8], and Declare possibly extended with meta-constraints [7].

In this setting, verification takes as input a bounded stochastic PNP $\mathcal{N}$ and an automaton A whose transitions are labelled by task names, and returns the probability that $\mathcal{N}$ generates a model trace that belongs to the language of A. Technically, we define the VERIFY-PROB$(\mathcal{N}, A)$ problem as follows:

Input: Bounded stochastic PNP $\mathcal{N}$, DFA A over Σ;

Output: Probability value equal to $\sum_{\sigma \text{ model trace of } \mathcal{N} s.t.\ \sigma \in \mathcal{L}(A)} \mathbb{P}_{\mathcal{N}}(\sigma)$.

To solve the problem, we need to account for three different aspects:

1. deal with the mismatch between runs over $\mathcal{N}$ and traces of A;
2. single out all and only those model traces of $\mathcal{N}$ that are also traces of A;
3. compute the collective probability of all such traces.

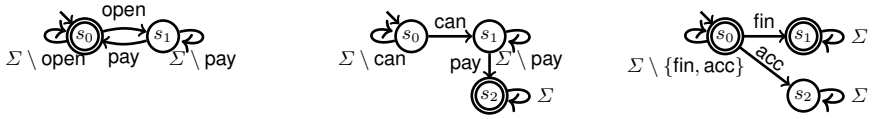
We tackle these three aspects with corresponding three steps.

Automaton with silent transitions. Definition 6 indicates that the set of runs inducing a trace consists of all those runs that insert an arbitrary number of τs before and after each event in the trace. For a trace $\sigma = \mathsf{a}_0, \ldots, \mathsf{a}_n$, this set corresponds to the language of the regular expression $\tau^*; \mathsf{a}_0; \tau^*; \ldots; \tau^*; \mathsf{a}_n; \tau^*$. We then take the input automaton A over Σ and turn it into a corresponding automaton $\bar{A}$ over $\overline{\Sigma}$ whose language $\mathcal{L}(\bar{A})$ corresponds to all and only the possible runs that induce the traces of $\mathcal{L}(A)$. This is done by simply expanding it with τ-labelled self-loops connecting every state to itself.

Definition 13 (Run DFA). *Given a DFA $A = \langle \Sigma, S, s_0, S_f, \delta \rangle$ over Σ, its* run DFA $\bar{A}$ *is a DFA over $\overline{\Sigma}$ defined as $\langle \overline{\Sigma}, S, s_0, S_f, \delta' \rangle$ with identical states (including the initial and final ones), and where $\delta' = \delta \cup \{\langle s, \tau \rangle \to s \mid s \in S\}$.* ◁

Product stochastic transition system. We now consider $RG(\mathcal{N})$ and $\bar{A}$. Since they are both run-generating devices, we can obtain an intensional representation of all the runs of $\mathcal{N}$ by constructing a *product* stochastic transition system generates all and only runs that are common to $\mathcal{N}$ and $\bar{A}$, which in turn are the runs of $\mathcal{N}$ that induce traces of A. This can be done by the usual product automaton construction, with the only difference that we need to retain the stochastic information coming from $\mathcal{N}$. This is straightforward, as $\bar{A}$ is qualitative, i.e., does not contain probabilities.

Definition 14 (Product system). *Let $\mathcal{N}$ by a bounded stochastic PNP with $RG(\mathcal{N}) = \langle S_1, s_0^1, S_f^1, \varrho_1, p_1 \rangle$, and $\bar{A} = \langle \overline{\Sigma}, S_2, s_0^2, S_f^2, \delta_2 \rangle$ a DFA over $\overline{\Sigma}$. The* product system $\Upsilon_{\mathcal{N}}^{\bar{A}}$ *of $\mathcal{N}$ and $\bar{A}$ is a stochastic transition system $\langle S, s_0, S_f, \varrho, p \rangle$ whose states are pairs of*



(a) Every open is followed by pay (b) At some point can and then pay (c) Acc only possible after fin

Fig. 5: DFAs of three properties for the order-to-cash example. A single edge labelled by a set L of task names describes a set of edges, each labelled by a task name from L.

states from $S_1 \times S_2$, and whose components are defined by mutual induction as the sets satisfying the following conditions:

1. $s_0 = \langle s_0^1, s_0^2 \rangle \in S$;
2. *for every state $\langle s_1, s_2 \rangle \in S$ and every label $l \in \overline{\Sigma}$ such that (i) $\langle s_1, l, s_1' \rangle \in \varrho_1$ for some $s_1' \in S^1$, and (ii) $\delta_2(s_2, l) = s_2'$ for some $s_2' \in S^2$, by fixing $s' = \langle s_1', s_2' \rangle$ we have: (a) $s' \in S$, (b) $\langle s, l, s' \rangle \in \varrho$, (c) $p(\langle s, l, s' \rangle) = p_1(s_1')$, (d) if $s_1' \in S_f^2$ and $s_2' \in S_f^2$, then $s' \in S_f$.* ◁

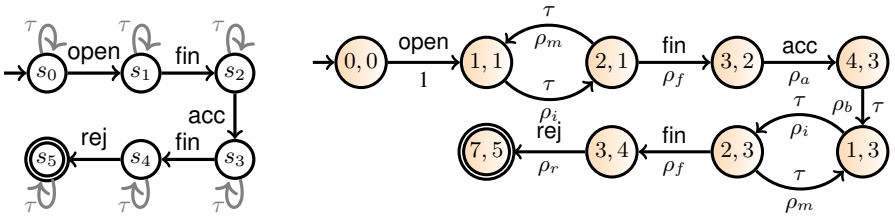
The so-defined product system is not a complete stochastic transition system: there may be states whose successor probabilities do not add up to one. It can be made complete by adding a fresh non-final sink state and transitions pointing from such incomplete states to the fresh one, each decorated with the probability value needed to reach 1, and labelled with whatever label from $\overline{\Sigma}$. This completion is not essential for the consequent computation (as the state variable for such a sink state would be equal to 0).

Verification as outcome probability computation. We are now ready to bring everything together, exploiting the notions of run DFA and product system to show how the VERIFY-PROB problem can be reduced to the OUTCOME-PROB, invoked on $\Upsilon_{\mathcal{N}}^{\bar{A}}$ considering all its final states.

Theorem 1. *For every bounded stochastic PNP $\mathcal{N}$ and DFA A, we have that* VERIFY-PROB$(\mathcal{N}, A) =$ OUTCOME-PROB$(\Upsilon_{\mathcal{N}}^{\bar{A}}, \Upsilon_{\mathcal{N}}^{\bar{A}}.S_f)$. ◁

Proof. Considering that the definition of VERIFY-PROB, and that the probability of a model trace of $\mathcal{N}$ is the sum of the probabilities of the runs of $\mathcal{N}$ inducing that trace, we have VERIFY-PROB$(\mathcal{N}, A) = \sum_{\sigma \text{ model trace of} \mathcal{N} s.t.\ \sigma \in \mathcal{L}(A)} \mathbb{P}_{\mathcal{N}}(\sigma) = \sum_{\eta \text{ run of } \mathcal{N} \text{ inducing } \sigma s.t.\ \sigma \in \mathcal{L}(A)} \mathbb{P}_{\mathcal{N}}(\eta)$. By Definitions 13 and 14, we have that the set of runs of $\mathcal{N}$ inducing traces in $\mathcal{L}(A)$ coincides with the set of runs of $\Upsilon_{\mathcal{N}}^{\bar{A}}$. This, together with the definition of OUTCOME-PROB, yields: $\sum_{\eta \text{ run of } \mathcal{N} \text{ inducing } \sigma s.t.\ \sigma \in \mathcal{L}(A)} \mathbb{P}_{\mathcal{N}}(\eta) = \sum_{\eta \text{ run of } \Upsilon_{\mathcal{N}}^{\bar{A}}} =$ OUTCOME-PROB$(\Upsilon_{\mathcal{N}}^{\bar{A}}, \Upsilon_{\mathcal{N}}^{\bar{A}}.S_f)$. ⊣

Example 4. Figure 5 shows three properties of interest for $\mathcal{N}_{\text{order}}$. Solving VERIFY-PROB for them gives: (1) The probability that $\mathcal{N}_{\text{order}}$ verifies the property of Figure 5(a) coincides with the solution of OUTCOME-PROB when asking the probability that an order gets paid (cf. Example 2). (2) The probability that $\mathcal{N}_{\text{order}}$ verifies the property of Figure 5(b) is 0, as there is no run of $\mathcal{N}_{\text{order}}$ where an order is first cancelled and then paid. (3) The probability that $\mathcal{N}_{\text{order}}$ verifies the property of Figure 5(c) coincides with that of a run of $\mathcal{N}_{\text{order}}$ reaching completion, as each run either does not



(a) DFAs A_σ and $\bar{A}_\sigma$. (b) Product system between $\bar{A}_\sigma$ and $RG(\mathcal{N}_{\text{order}})$.

Fig. 6: DFAs for a trace and product system with the reachability graph of Figure 3.

finalise the order, or does so before possibly accepting it. This probability is actually 1 (as the process does not contain livelocks nor non-final deadlocks). ◁

5.2 Computing Trace Probabilities

A key problem in stochastic conformance checking [15,2] is that of computing the probability of a trace in a stochastic Petri net. We cast this problem in our setting as the TRACE-PROB problem, where TRACE-PROB$(\mathcal{N}, \sigma)$ is defined as:
Input: Bounded stochastic PNP $\mathcal{N}$, trace σ over Σ;
Output: Probability $\mathbb{P}_{\mathcal{N}}(\sigma)$.
This problem is clearly subsumed by the VERIFY-PROB problem described before. In fact, we can simply solve it by constructing a so-called trace automaton that trivially encodes σ as a DFA that only accepts that trace, then invoking VERIFY-PROB on it.

Definition 15 (Trace DFA). *Given a trace $\sigma = a_0, \ldots, a_n$ over Σ, its* trace DFA A_σ *is the DFA $\langle \Sigma, S, s_0, S_f, \delta \rangle$ over Σ such that: (a) $S = \{s_0, \ldots, s_{n+1}\}$ contains $n+1$ states; (b) $S_f = \{s_{n+1}\}$; (c) for every $i \in \{0, \ldots, n\}$, $\delta(s_i, a_i) = s_{i+1}$ (and nothing else is in δ).*

Theorem 2. *For every bounded stochastic PNP $\mathcal{N}$ and every trace σ over Σ^*, we have that* TRACE-PROB$(\mathcal{N}, \sigma)$ = OUTCOME-PROB$(\mathcal{N}, A_\sigma)$. ◁

Proof. Direct from the definition of the problems, noticing that $\mathcal{L}(A_\sigma) = \{\sigma\}$. ⊣

Example 5. We compute the probability that $\mathcal{N}_{\text{order}}$ generates trace $\sigma = \mathsf{open}, \mathsf{fin}, \mathsf{acc}, \mathsf{fin}, \mathsf{rej}$, where an order is filled, finalised, accepted, then modified, finalised again, and this second time rejected. Following the described technique, we first transform σ into its trace DFA A_σ, and then further into its run DFA $\bar{A}_\sigma$. This is shown in Figure 6(a). We then compute the product system $\Upsilon^{\bar{A}}_{RG(\mathcal{N}_{\text{order}})}$ of $\bar{A}_\sigma$ and $RG(\mathcal{N}_{\text{order}})$ (shown in Figure 3), obtaining Figure 6(b) (notice how silent transitions unfold in this transition system). Finally, we construct $\mathcal{E}^{\langle 7,5 \rangle}_{\Upsilon^{\bar{A}}_{RG(\mathcal{N}_{\text{order}})}}$ getting $x_{00} = \frac{\rho_i \rho_f \rho_a \rho_b \rho_i \rho_f \rho_r}{(1-\rho_i \rho_m)^2}$, which yields the solution to the TRACE-PROB$(\mathcal{N}_{\text{order}}, \sigma)$ problem. ◁

6 Stochastic Conformance with Probabilistic Declare

We now employ the verification machinery from Section 5 to check how the probabilistic behaviour encoded in a stochastic PNP relates to that declaratively specified using ProbDeclare [18]. We start with a gentle introduction to ProbDeclare, then showing how we can check whether a bounded stochastic PNP conforms to a ProbDeclare model.

6.1 Probabilistic Declare

Declare is a constraint-based process modelling language based on LTL$_f$. A model comes with a set of LTL$_f$ constraint, and their conjunction must be respected by the process. This imposes a *crisp* interpretation of constraints: a trace satisfies a Declare model if it satisfies *every* constraints contained therein. This crisp semantics was relaxed in [18]: there, each constraint comes with a probability condition indicating the allowed probabilities for which a satisfying trace should be generated by the process. The semantics is formally defined using stochastic languages, and is therefore compatible with that of stochastic PNPs. We recall the necessary definitions.

Definition 16 (Probabilistic constraint). *A* probabilistic constraint *is a triple* $\langle \varphi, \bowtie, p \rangle$*, where: (i)* φ *is an LTL$_f$ formula over* Σ *representing the* constraint formula*; (ii)* $\bowtie \in \{=, \neq, \leq, \geq, <, >\}$ *is the* constraint probability operator*; (iii)* p *is a rational value in* $[0, 1]$ *representing the* constraint probability. ◁

Definition 17 (ProbDeclare). *A* ProbDeclare model *is a triple* $\langle \Sigma, \mathcal{C}, \mathcal{P} \rangle$*, where* $\mathcal{C}$ *is a finite set of LTL$_f$ formulae called* crisp constraints*, while* $\mathcal{P}$ *is a set of (genuinely) probabilistic constraints.* ◁

Since each constraint in $\mathcal{P}$ can be satisfied or violated, a ProbDeclare model induces $2^{|\mathcal{P}|}$ *scenarios*, each associated to a corresponding LTL$_f$ formula.

Definition 18 (Scenario). *A* scenario *for a ProbDeclare model* $\mathcal{D} = \langle \Sigma, \mathcal{C}, \mathcal{P} \rangle$ *is a total boolean function* $\mathcal{S} : \mathcal{P} \to \{0, 1\}$ *indicating which probabilistic constraints are satisfied, and which violated. The set of scenarios is denoted by* $\mathbb{S}_{\mathcal{D}}$*.* ◁

Scenarios come with two dimensions, induced by the crisp and probabilistic constraints: a temporal dimension indicating which traces belong to which scenarios, and a probabilistic dimension indicating how likely it is that a trace belongs to a scenario.

Definition 19 (Characteristic formula). *The characteristic formula* $\Phi_{\mathcal{S}}$ *of a scenario* $\mathcal{S}$ *for ProbDeclare model* $\langle \Sigma, \mathcal{C}, \mathcal{P} \rangle$ *is the LTL$_f$ formula* $\bigwedge_{\varphi_i \in \mathcal{C}} \varphi_i \wedge \bigwedge_{\langle \varphi_j, \bowtie_j, p_j \rangle \in \mathcal{P},\ \mathcal{S}(\langle \varphi_j, \bowtie_j, p_j \rangle)=1} \varphi_j \wedge \bigwedge_{\langle \varphi_k, \bowtie_k, p_k \rangle \in \mathcal{P},\ \mathcal{S}(\langle \varphi_k, \bowtie_k, p_k \rangle)=0} \neg \varphi_k$*. Scenario* $\mathcal{S}$ *is consistent if* $\Phi_{\mathcal{S}}$ *is satisfiable (i.e., has at least one satisfying trace).* ◁

Definition 20 (Valid scenario distribution). *A* valid scenario distribution *over scenarios* $\mathbb{S}_{\mathcal{D}}$ *of a ProbDeclare model* $\mathcal{D} = \langle \Sigma, \mathcal{C}, \mathcal{P} \rangle$ *is a probability distribution* $\mathbb{P}_{\mathcal{D}} : \mathbb{S}_{\mathcal{D}} \to [0, 1]$ *such that: (a) for every scenario* $\mathcal{S} \in \mathbb{S}_{\mathcal{D}}$*, if* $\mathcal{S}$ *is not consistent then* $\mathbb{P}_{\mathcal{D}}(\mathcal{S}) = 0$*; (b) For every probabilistic constraint* $\langle \varphi, \bowtie, p \rangle \in \mathcal{P}$*, we have* $\sum_{\mathcal{S} \in \mathbb{S}_{\mathcal{D}} \text{ s.t. } \mathcal{S}(\langle \varphi, \bowtie, p \rangle)=1} \mathbb{P}_{\mathcal{D}}(\mathcal{S}) \bowtie p$*.* ◁

Example 6. Consider the order-to-cash ProbDeclare model $\mathcal{D}_{\text{order}}$, with one crisp *not coexistence* constraint C_1 indicating that pay and rej cannot be both in a trace, and two *response* probabilistic constraints C_2 and C_3, indicating that open must be eventually followed by pay with a probability of $\geq \frac{1}{20}$, and that open must be eventually followed by rej with a probability of $\leq \frac{1}{4}$. Of the four scenarios over C_2 and C_3, one is that both C_2 and C_3 are satisfied and is inconsistent as it clashes with the crisp constraint C_1. ◁

6.2 Checking Stochastic Conformance

In [18], it is shown how a system of inequalities can be constructed so as to compute possible valid scenario distributions in accordance with Definition 20. Here we are just interested in using that system to *check* whether a probability distribution over scenarios is indeed valid, and thus we can keep this system of inequalities as a black box.

In a non-stochastic setting, one can check whether a bounded Petri net satisfies a Declare model by verifying that every trace it generates belongs to the language accepted by the model. In our stochastic setting, we define a stochastic variant of this problem. We start by showing that a bounded stochastic PNP induces a probability distribution over scenarios of a ProbDeclare model, obtained by collecting, scenario by scenario, the probability of all PNP traces that satisfy the characteristic formula of that scenario.

Definition 21 (Induced scenario distribution). *Let $\mathcal{N}$ be a bounded stochastic PNP, and $\mathcal{D}$ a ProbDeclare model. The* scenario distribution $\mathbb{P}^{\mathcal{N}}_{\mathcal{D}}$ *induced by $\mathcal{N}$ over scenarios $\mathbb{S}_{\mathcal{D}}$ is the probability distribution defined as follows: for every $\mathcal{S} \in \mathbb{S}_{\mathcal{D}}$, we have that* $\mathbb{P}^{\mathcal{N}}_{\mathcal{D}}(\mathcal{S}) = \sum_{\sigma \text{ trace of } \mathcal{N} s.t. \sigma \text{ satisfies } \Phi_{\mathcal{S}}} \mathbb{P}_{\mathcal{N}}(\sigma)$. ◁

We then define the stochastic conformance problem S-CONFORM$(\mathcal{N}, \mathcal{D})$ as:
Input: bounded stochastic PNP $\mathcal{N}$, ProbDeclare model $\mathcal{D}$;
Output: Whether $\mathbb{P}^{\mathcal{N}}_{\mathcal{D}}$ is valid (in the sense of Definition 20).

We address the problem through iterated invocations of the VERIFY-PROB problem, one per scenario. Specifically, for each scenario $\mathcal{S} \in \mathbb{S}_{\mathcal{D}}$: (1) Construct DFA $A_{\mathcal{S}}$ for characteristic formula $\Phi_{\mathcal{S}}$ with standard techniques [18,7]; (2) Get the probability value $p =$ VERIFY-PROB$(\mathcal{N}, A_{\mathcal{S}})$; (3) Check whether p is valid for $\mathcal{S}$ using the system of inequalities for Definition 20; (4) If this is the case, proceed with the next scenario, otherwise return No; and (5) If all scenarios have been checked, return Yes.

Example 7. Consider the bounded stochastic PNP $\mathcal{N}_{\text{order}}$ (assuming equal weights for all transitions) and the ProbDeclare model $\mathcal{D}_{\text{order}}$. The only scenario where C_2 holds is the one where C_2 is satisfied while C_3 is violated. The probability induced by $\mathcal{N}_{\text{order}}$ for this scenario corresponds to the outcome probability for $\mathcal{N}_{\text{order}}$ to finish with a payment. As discussed in Example 2, this is $\frac{1}{17}$, which is indeed $\geq \frac{1}{20}$. The only scenario where C_3 holds is the one where C_2 is violated while C_2 is satisfied. The probability induced by $\mathcal{N}_{\text{order}}$ for this scenario corresponds to the outcome probability for $\mathcal{N}_{\text{order}}$ to finish with a rejection. As discussed in Example 2, this is $\frac{3}{17}$, which is indeed $\leq \frac{1}{4}$. This witnesses that $\mathcal{N}_{\text{order}}$ stochastically conforms to $\mathcal{D}_{\text{order}}$. ◁

In case S-CONFORM is negative, the standard Earth Mover's Distance can be used to measure the deviation between the scenario distribution induced by $\mathcal{N}$ and the closed valid scenario distribution for $\mathcal{D}$. This realises a form of *stochastic delta analysis*.

7 Conclusion

We have provided formal methods and algorithmic techniques solving three key problems concerning reasoning on labelled Petri nets and their executions in a stochastic setting: outcome probability, verification, and stochastic model conformance. All techniques are implemented in the *StochasticLabelledPetriNets* plug-in of ProM. For solving systems of inequalities, we use a Java LP solver (LPSolve). Our approach lazily

handles silent transitions when combining the reachability graph of the net with the automaton of a temporal property of interest.

A natural extension of this work is to incorporate our techniques into stochastic process mining pipelines, validating the resulting framework experimentally either using our own implementation or by invoking probabilistic model checkers [14]. We also want to study if and how our results transfer to richer settings, such as stochastic nets that map to Markov decision processes, as well as non-Markovian nets.

References

1. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
2. Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic trace alignment. In: ICPM. pp. 9–16. IEEE (2021)
3. Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: A tool for probabilistic trace alignments. In: CAiSE Forum. Springer (2021)
4. Bruni, R., Melgratti, H.C., Montanari, U.: Concurrency and probability: Removing confusion, compositionally. Log. Methods Comput. Sci. **15**(4) (2019)
5. Burke, A., Leemans, S., Wynn, M.: Stochastic process discovery by weight estimation. In: PQMI (10 2020)
6. Chiola, G., Marsan, M.A., Balbo, G., Conte, G.: Generalized stochastic Petri nets: A definition at the net level and its implications. IEEE Trans. on soft. eng. **19**(2), 89–107 (1993)
7. De Giacomo, G., De Masellis, R., Maggi, F.M., Montali, M.: Monitoring constraints and metaconstraints with temporal logics on finite traces. ACM TOSEM (2022)
8. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Proceedings IJCAI. AAAI Press (2013)
9. Desel, J., Reisig, W.: Place/transition Petri nets. In: Lectures on Petri Nets I: Basic Models: Advances in Petri Nets. pp. 122–173. Springer, Springer (1998)
10. van Dongen, B.F., et al.: The ProM framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets. vol. 3536, pp. 444–454 (2005)
11. Durrett, R.: Essentials of Stochastic Processes - 2nd Edition. Springer (2012)
12. Fewster, R.: Stochastic Processes. Course Notes Stas 325, University of Auckland (2008)
13. Hanneforth, T., De La Higuera, C.: Epsilon-removal by loop reduction for finite-state automata over complete semirings. Studia Grammatica **72**, 297–312 (2010)
14. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. LNCS, vol. 6806, pp. 585–591. Springer (2011)
15. Leemans, S.J.J., van der Aalst, W.M.P., Brockhoff, T., Polyvyanyy, A.: Stochastic process mining: Earth movers' stochastic conformance. Inf. Syst. **102**, 101724 (2021)
16. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware conformance checking: An entropy-based approach. In: Advanced Information Systems Engineering. pp. 217–233. Springer (2020)
17. Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Proceedings of the BPM Forum. vol. 360, pp. 127–143 (2019)
18. Maggi, F.M., Montali, M., Peñaloza, R., Alman, A.: Extending temporal business constraints with uncertainty. In: Proceedings of BPM. LNCS, vol. 12168, pp. 35–54. Springer (2020)
19. Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM TOCS **2**(2), 93–122 (1984)
20. Molloy, M.K.: Performance analysis using stochastic petri nets. IEEE Transactions on Computers **31**, 913–917 (1982)
21. Rogge-Solti, A., Mans, R., Aalst, W., Weske, M.: Repairing Event Logs Using Timed Process Models. In: OTM 2013 Workshops. LNCS, vol. 8186, pp. 705–708. Springer (2013)
22. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In: BPMW13. pp. 15–27 (2013)