Stochastic Decision Petri Nets

Florian Wittbold[1], Rebecca Bernemann[1], Reiko Heckel[2], Tobias Heindel[3], and Barbara König[1]

[1] Universität Duisburg-Essen, Duisburg, Germany
[2] University of Leicester, UK
[3] Heliax Technologies GmbH

Abstract. We introduce stochastic decision Petri nets (SDPNs), which are a form of stochastic Petri nets equipped with rewards and a control mechanism via the deactivation of controllable transitions. Such nets can be translated into Markov decision processes (MDPs), potentially leading to a combinatorial explosion in the number of states due to concurrency. Hence we restrict ourselves to instances where nets are either safe, free-choice and acyclic nets (SAFC nets) or even occurrence nets and policies are defined by a constant deactivation pattern. We obtain complexity-theoretic results for such cases via a close connection to Bayesian networks, in particular we show that for SAFC nets the question whether there is a policy guaranteeing a reward above a certain threshold is $\mathsf{NP}^{\mathsf{PP}}$-complete. We also introduce a partial-order procedure which uses an SMT solver to address this problem.

1 Introduction

State-based probabilistic systems are typically modelled as Markov chains [28], i.e., transition systems where transitions are annotated with probabilities. This admits an intuitive graphical visualization and efficient analysis techniques [17]. By introducing additional non-determinism, one can model a system where a player can make decisions, enriched with randomized choices. This leads to the well-studied model of Markov decision processes (MDPs) [6,15] and the challenge is to synthesize strategies that maximize the reward of the player.

In this paper we study stochastic systems enriched with a mechanism for decision making in the setting of concurrent systems. Whenever a system exhibits a substantial amount of concurrency, i.e., events that may potentially happen in parallel, compiling it down to a state-based system – such as an MDP – can result in a combinatorial state explosion and a loss in efficiency of MDP-based methods. We base our models on stochastic Petri nets [21], where Petri nets are a standard formalism for modelling concurrent systems, especially such systems where resources are generated and consumed. When considering the discrete-time semantics of such stochastic nets, it is conceptually easy to transform them into Markov chains, but this typically leads to a state space explosion.

There exist successful partial order methods for analyzing concurrent systems that avoid explicit interleavings and the enumeration of all reachable states.

Instead, they work with partial orders – instead of total orders – of events. While such techniques are well understood in the absence of random choices, leading for instance to methods such as unfoldings [14], there are considerable difficulties to reconcile probability and partial order. Progress has been made for instance by the introduction of the concept of branching cells [1] that encapsulate independent choices, but to our knowledge there is no encompassing theory that provides off-the-shelf partial order methods for computing the probability of reaching a certain goal (e.g. marking a certain place) in a stochastic net.

The contributions of this paper are the introduction of a new model: stochastic decision Petri nets (SDPNs) and its connection to Markov decision processes (MDPs). The transformation of SDPNs into MDPs is relatively straightforward, but may lead to state space explosion, i.e., exponentially many markings, due to the concurrency inherent in the Petri net. This can make the computation of the optimal policy infeasible. We restrict ourselves to a subclass of nets which are safe, acyclic and free-choice (SAFC) and to constant policies and study the finding of a policy that guarantees a payoff above some bound. Our finding is that the problem SAFC-POL of finding such a policy, despite the restrictions, is still $\mathsf{NP}^{\mathsf{PP}}$-complete. We reduce from the D-MAP problem for Bayesian networks [24] (in fact the two problems are interreducible under mild restrictions) and show the close connection of reasoning about stochastic Petri nets and Bayesian networks. Furthermore, for SAFC nets, there is a partial-order solution procedure via an SMT solver, for which we obtain encouraging runtime results. For the simpler free-choice occurrence nets, we obtain an NP-completeness result.

Note that the main body of the paper contains some proof sketches, while full proofs and an additional example can be found in [29].

B: What about applications? → security attacks? crisis communication?

2 Preliminaries

By $\mathbb{N}$ we denote the natural numbers without 0, while $\mathbb{N}_0$ includes 0.

Given two sets X, Y we denote by $(X \to Y)$ the set of all functions from X to Y. Given a function $f\colon X \to \mathbb{N}_0$ or $f\colon X \to \mathbb{R}$ with X finite, we define $\|f\|_\infty = \max_{x \in X} f(x)$ and $\mathrm{supp}(f) = \{x \in X \mid f(x) \neq 0\}$.

Complexity Classes: In addition to well-known complexity classes such as P and NP, our results also refer to PP (see [23]). This class is based on the notion of a probabilistic Turing machine, i.e., a non-deterministic Turing machine whose transition function is enriched with probabilities, which means that the acceptance function becomes a random variable. A language L lies in PP if there exists a probabilistic Turing machine M with polynomial runtime on all inputs such that a word $w \in L$ iff it is accepted with probability strictly greater than $1/2$. As probabilities we only allow numbers ρ that are efficiently computable, meaning that the i-th bit of ρ is computable in a time polynomial in i. (See [2] for a discussion on why such probabilistic Turing machines have equal expressivity with those based on fair coins, which is not the case if we allow arbitrary numbers.)

Given two complexity classes A, B and their corresponding machine models, by A^B we denote the class of languages that are solved by a machine of class

A, which is allowed to use an oracle answering yes/no-questions for a language $L \in B$ at no extra cost in terms of time or space complexity. In particular $\mathsf{NP}^{\mathsf{PP}}$ denotes the class of languages that can be accepted by a non-deterministic Turing machine running in polynomial time that can query a black box oracle solving a problem in PP.

By Toda's theorem [27], a polynomial time Turing machine with a PP oracle (P^{PP}) can solve all problems in the polynomial hierarchy.

In order to prove hardness results we use the standard polynomial-time many-one reductions, denoted by $A \leq_p B$ for problems A, B (see [16]).

Stochastic Petri Nets: A stochastic Petri net [21] is given by a tuple $N = (P, T, {}^\bullet(\,), (\,)^\bullet, \Lambda, m_0)$ where P and T are finite sets of places and transitions, ${}^\bullet(\,), (\,)^\bullet : T \to (P \to \mathbb{N}_0)$ determine for each transition its pre-set and post-set including multiplicities, $\Lambda \colon T \to \mathbb{R}_{>0}$ defines the firing rates and $m_0 \colon P \to \mathbb{N}_0$ is the initial marking. By $\mathcal{M}(N)$ we denote the set of all markings of N, i.e., $\mathcal{M}(N) = (P \to \mathbb{N}_0)$.

We will only consider the discrete-time semantics of such nets. The firing rates determine stochastically which transition is fired in a marking where multiple transitions are enabled: When transitions $t_1, \dots, t_n \in T$ are enabled in a marking $m \in \mathcal{M}(N)$ (i.e., ${}^\bullet t_i \leq m$ pointwise), then transition t_i fires with probability $\Lambda(t_i) / \sum_{j=1}^{n} \Lambda(t_j)$, resulting in a discrete step $m \to_{t_i} m' \coloneqq m - {}^\bullet t_i + t_i{}^\bullet$. In particular, the firing rates have no influence on the reachability set $\mathcal{R}(N) \coloneqq \{m \in \mathcal{M}(N) \mid m_0 \to^* m\}$ but only define the probability of reaching certain places or markings. Defining "empty" transitions $m \to_\varepsilon m$ for markings $m \in \mathcal{R}(N)$ where no transition is enabled, such a stochastic Petri net can be interpreted as a Markov chain on the set of markings $\mathcal{M}(N)$.

This Markov chain thus generates a (continuous) probability space over sequences $(m_0, m_1, \dots) \in \mathcal{M}(N)^\omega$ where a sequence is called valid if m_0 is the initial marking of the Petri net and all cones $(m_0, \dots, m_n) = \{(m'_0, m'_1, \dots) \in \mathcal{M}(N)^\omega \mid \forall k = 0, \dots, n : m'_k = m_k\}$ have non-zero probability. We write $\mathcal{FS}(N) \coloneqq \{\mu \in \mathcal{M}(N)^\omega \mid \mu \text{ is valid}\}$ to denote the set of valid sequences. A valid sequence corresponds to a firing sequence $\mu : m_0 \to_{t_1} m_1 \to_{t_2} \dots$. We assume that no two transitions have the same pre- and postconditions to have a one-to-one-correspondence.

For a firing sequence μ, we write $\mu^k : m_0 \to_{t_1} m_1 \to_{t_2} \cdots \to_{t_k} m_k$ to denote the finite subsequence of the first k steps, $\mathrm{len}(\mu) \coloneqq \min\{k \in \mathbb{N} \mid t_k = \varepsilon\} - 1$, for its length (finite sequences, i.e., sequences with $\mathrm{len}(\mu) < \infty$, always have non-zero probability), as well as

$$pl(\mu) \coloneqq \bigcup_{n=0}^{\infty} \mathrm{supp}(m_n) \qquad\qquad tr(\mu) \coloneqq \{t_n \mid n \in \mathbb{N}\} \setminus \{\varepsilon\}$$

to denote the set of places reached in μ (or, analogously, μ^k), and the set of fired transitions in μ (independent of their firing order), respectively.

We are, furthermore, interested in the following properties of Petri nets: A Petri net N as above is called

- *ordinary* iff all transitions require and produce at most one token in each place ($\| {}^\bullet t\|_\infty, \|t^\bullet\|_\infty \leq 1$ for all $t \in T$);
- *safe* iff it is ordinary and all reachable markings also only have at most one token in each place ($\|m\|_\infty \leq 1$ for all $m \in \mathcal{R}(N)$);
- *acyclic* iff the transitive closure $\prec_N^+$ of the causal relation $\prec_N$ (with $p \prec_N t$ if ${}^\bullet t(p) > 0$ and $t \prec_N p$ if $t^\bullet(p) > 0$) is irreflexive;
- an *occurrence net* iff it is safe, acyclic, free of backward conflicts (all places have at most one predecessor transition, i.e., $|\{t \mid t^\bullet(p) > 0| \leq 1$ for all $p \in P$) and self-conflicts (for $x \in P \cup T$, there exist no two distinct conflicting transitions $t, t' \in T$, i.e., transitions sharing preconditions, on which x is causally dependent, i.e., $t, t' \prec_N^+ x$), and the initial marking has no causal predecessors (for all $p \in P$ with $m_0(p) = 1$, we have $t^\bullet(p) = 0$ for all $t \in T$);
- *free-choice* [13] iff it is ordinary and all transitions $t, t' \in T$ are either both enabled or disabled in all markings (i.e., ${}^\bullet t = {}^\bullet t'$ or $\mathrm{supp}({}^\bullet t) \cap \mathrm{supp}({}^\bullet t') = \emptyset$);
- *φ-bounded* (for $\varphi\colon \mathbb{N}_0 \to \mathbb{N}_0$) iff all its runs, starting from m_0, have at most length $\varphi(|P| + |T|)$, i.e., iff $\mathrm{len}(\mu) \leq \varphi(|P| + |T|)$ for all firing sequences $\mu \in \mathcal{FS}(N)$.

We will abbreviate the class of free-choice occurrence Petri nets as FCON, safe and acyclic free-choice nets as SAFC nets, and the class of φ-bounded Petri nets as $[\varphi]$BPN. Note that FCON $\subseteq$ SAFC and also SAFC $\subseteq [id]$BPN for the identity id. [4]

We also introduce some notation specifically for SAFC nets: As common in the analysis of safe Petri nets, we will interpret markings as well as pre- and postconditions of transitions as subsets of the set P of places rather than functions $P \to \{0, 1\} \subseteq \mathbb{N}_0$.

The set of maximal configurations will be denoted by $\mathcal{C}^\omega(N) \coloneqq \{tr(\mu) \mid \mu \in \mathcal{FS}(N)\}$ and configurations by $\mathcal{C}(N) \coloneqq \{tr(\mu^k) \mid \mu \in \mathcal{FS}(N), k \in \mathbb{N}_0\}$.

An important notion in the analysis of a (free-choice) net are branching cells $\mathbb{C}$ (see also [8,1]). We will define a cell to be a subset of transitions $\mathbb{C} \subseteq T$ where all transitions $t \in \mathbb{C}$ share their preconditions and all $t' \in T \setminus \mathbb{C}$ share no preconditions with $t \in \mathbb{C}$. In other words, $\mathbb{C}$ is an equivalence class of a relation $\leftrightarrow$ on T defined by

$$\forall t, t' \in T : t \leftrightarrow t' \iff {}^\bullet t = {}^\bullet t'.$$

We will write $\mathbb{C}_t \coloneqq [t]^\leftrightarrow$ to denote the equivalence class of transition $t \in T$ and ${}^\bullet\mathbb{C} \coloneqq \bigcup_{t \in \mathbb{C}} {}^\bullet t$ as well as $\mathbb{C}^\bullet \coloneqq \bigcup_{t \in \mathbb{C}} t^\bullet$ to denote the sets of pre- and postplaces of $\mathbb{C}$, respectively. The set of all cells of a net N is denoted by $BC(N)$.

Markov decision processes: A Markov decision process (MDP) is a tuple (S, A, δ, r, s_0) consisting of finite sets S, A of states and actions, a function $\delta\colon S \times A \to \mathcal{D}(S)$ of probabilistic transitions (where $\mathcal{D}(S)$ is the set of probability distributions on S), a reward function $r\colon S \times A \times S \to \mathbb{R}$ of rewards and an initial state $s_0 \in S$ (see also [6,15]).

[4] Indeed, $[id]$BPN contains any safe and acyclic Petri net, omitting the free-choice constraint.

A policy (or strategy) for an MDP is some function $\pi\colon S \to A$. It has been shown that such stationary deterministic policies can act optimally in an MDP setting (see also [**?**]). A policy gives rise to a Markov chain on the set of states with transitions $s \mapsto \delta(s, \pi(s)) \in \mathcal{D}(S)$. The associated probability space is $s_0 S^\omega$, the set of all infinite paths on S starting with s_0, which – due to its uncountable nature – has to be dealt with using measure-theoretic concepts. We equip the probability space with a σ-algebra generated by all cones, i.e., all sets of words sharing a common prefix.

The value (or payoff) of a policy π is then given as the expectation of the (undiscounted) total reward (where $\mathbf{s}_i$, $i \in \mathbb{N}_0$ are random variables, mapping an infinite path to i-th state, i.e., they represent the corresponding Markov chain):

$$\mathbb{E}\left[\sum_{n\in\mathbb{N}_0} r(\mathbf{s}_n, \pi(\mathbf{s}_n), \mathbf{s}_{n+1})\right].$$

To avoid infinite values, we have to assume that the sum is bounded.

The problem of finding an optimal policy $\pi\colon S \to A$ for a given MDP (S, A, δ, r, s_0) with finite state and action space is known to be solvable in polynomial time using linear programming[15,19].

Bayesian Networks: Bayesian networks are graphical models that give compact representations of discrete probability distributions, exploiting the (conditional) independence of random variables.

A (finite) probability space $(\Omega, \mathbb{P})$ consists of a finite set Ω and a probability function $\mathbb{P}\colon \Omega \to [0,1]$ such that $\sum_{\omega\in\Omega} \mathbb{P}(\omega) = 1$. A Bayesian network [25] is a tuple (X, Δ, P) where

- $X = (X_i)_{i=1,\dots,n}$ is a (finite) family of random variables $X_i\colon \Omega \to V_i$, where V_i is finite.
- $\Delta \subseteq \{1,\dots,n\} \times \{1,\dots,n\}$ is an acyclic relation that describes dependencies between the variables, i.e., its transitive closure Δ^+ is irreflexive. By $\Delta^i = \{j \mid (j,i) \in \Delta\}$ we denote the parents of node i according to Δ.
- $P = (P_i)_{i=1,\dots,n}$ is a family of probability matrices $P_i\colon \prod_{j\in\Delta^i} V_j \to \mathcal{D}(V_i)$, whose entries are given by $P_i(v_i \mid (v_j)_{j\in\Delta^i})$.

A probability function $\mathbb{P}$ is consistent with such a Bayesian network whenever for $v = (v_i)_{i=1,\dots,n} \in \prod_{i=1}^n V_i$ we have

$$\mathbb{P}(X = v) = \prod_{i=1}^{n} P_i(v_i \mid (v_j)_{j\in\Delta^i}).$$

The size of a Bayesian network is not just the size of the graph, but the sum of the size of all its matrices (where the size of an $m \times n$-matrix is $m \cdot n$). In particular, note that a node with k parents in a binary Bayesian network (i.e., with $|V_i| = 2$ for all i) is associated with a 2×2^k probability matrix.

Example 2.1. *An example Bayesian network is given in Figure 1. There are four random variables (a, b, c, d) with codomain $\{0, 1\}$. The tables in the figure denote the conditional probabilities, for instance $P_d(0 \mid 01) = \mathbb{P}(X_d = 0 \mid X_a = 0, X_b = 1) = 1/6$, i.e., one records the probability that a random variable has a certain value, dependent on the value of its parents in the graph. The probability $\mathbb{P}(X = 0100) = \mathbb{P}(X_a = 0, X_b = 1, X_c = 0, X_d = 0)$ is obtained by multiplying $P_a(0) \cdot P_b(1) \cdot P_c(0 \mid 0) \cdot P_d(0 \mid 01) = 1/3 \cdot 1/2 \cdot 2/3 \cdot 1/6 = 1/54$.*

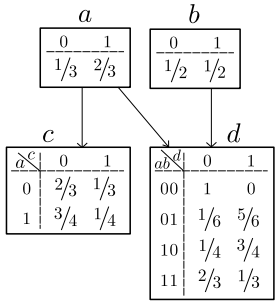

Fig. 1. A Bayesian Network

We are interested in the following two problems for Bayesian networks (see also [24]):

- D-PR: Given the Bayesian network (X, Δ, P) and $E = \{X_{i_1}, \ldots, X_{i_\ell}\} \subseteq X$, $e \in V_E := \prod_{j=1}^{\ell} V_{i_j}$ (the evidence) and a rational $p > 0$, does it hold that $\mathbb{P}(E = e) > p$? This problem is known to be PP-complete [20].
- D-MAP: Given a Bayesian network (X, Δ, P), a rational number $p > 0$, disjoint subsets $E, F \subseteq X$,[5] and evidence $e \in V_E$, does there exist $f \in V_F$ such that $\mathbb{P}(F = f, E = e) > p$, or, if $\mathbb{P}(E = e) \neq \emptyset$, equivalently, $\mathbb{P}(F = f \mid E = e) > p$ (by adapting the bound p). It is known that this problem, also known as maximum a-posteriori problem, is $\mathsf{NP}^{\mathsf{PP}}$-complete (see [20,11]).

The corresponding proof in [24] also shows that the D-MAP problem remains $\mathsf{NP}^{\mathsf{PP}}$-complete if F only contains uniformly distributed 'input' nodes, i.e., nodes X_i with $\Delta^i = \emptyset$ and $P_i(x_i) = 1/|V_i|$, as well as $V_i = \{0, 1\}$ for all $i = 1, \ldots, n$.

In particular, the following problem (where E, F are switched!) is still $\mathsf{NP}^{\mathsf{PP}}$-complete: Given a binary Bayesian network (X, Δ, P) (i.e., $V_i = \{0, 1\}$ for all i), a rational $p > 0$, disjoint subsets $E, F \subseteq X$ where F only contains uniformly distributed input nodes, as well as evidence $e \in V_E$, does there exist $f \in V_F$ such that $\mathbb{P}(E = e \mid F = f) > p$ (as $\mathbb{P}(F = f) = 1/2^{|F|}$ is independent of f and known due to uniformity). We will, in the rest of this paper, refer to this modified problem as D-MAP instead of the original problem above.

Example 2.2 (D-MAP)**.** *Given the Bayesian Network in Figure 1 with $F = \{X_a\}$ (MAP variable), $E = \{X_c, X_d\}$, $e = (0, 1) \in V_c \times V_d$ (evidence) and $p = 1/3$, we ask whether $\exists f \in \{0, 1\}$: $\mathbb{P}(X_c = 0, X_d = 1 \mid X_a = f) > 1/3$. When choosing $f = 1 \in V_a$, the probability $\mathbb{P}(X_c = 0, X_d = 1 \mid X_a = 1) = 3/4 \cdot (1/2 \cdot 3/4 + 1/2 \cdot 1/3) = 13/32 > 1/3$ exceeds the bound. Note that to compute the value in this way, one has to sum up over all possible valuations of those variables that are neither evidence nor MAP variables, indicating that this is not a trivial task.*

[5] The variables contained in F are called MAP variables.

3 Stochastic decision Petri nets

We will enrich the definition of stochastic Petri nets to allow for interactivity, similar to how MDPs [6] extend the definition of Markov chains.

Definition 3.1. *A stochastic decision Petri net (SDPN) is a tuple* $(P, T, {}^\bullet(), ()^\bullet, \Lambda, m_0, C, R)$ *where* $(P, T, {}^\bullet(), ()^\bullet, \Lambda, m_0)$ *is a stochastic Petri net;* $C \subseteq T$ *is a set of controllable transitions;* $R\colon \mathcal{P}(P) \to \mathbb{R}$ *is a reward function.*

Here we describe the semantics of such SDPNs in a semi-formal way. The precise semantics is obtained by the encoding of SDPNs into MDPs in Section 4.

Given an SDPN, an external agent may in each step choose to manually deactivate any subset $D \subseteq C$ of controllable transitions (regardless of whether their preconditions are fulfilled or not). As such, if transitions $D \subseteq C$ are deactivated in marking $m \in \mathcal{M}(N)$, the SDPN executes a step according to the semantics of the stochastic Petri net $N_D = (P, T \setminus D, {}^\bullet(), ()^\bullet, \Lambda_D, m_0)$ where the pre- and post-set functions and Λ_D are restricted accordingly.

For all rewarded sets $Q \in \operatorname{supp}(R)$, the agent receives an "immediate" reward $R(Q)$ once all the places $p \in Q$ are reached at one point in the execution of the Petri net (although not necessarily simultaneously). In particular, any reward is only received once. Note that this differs from the usual definition of rewards as in MDPs, where a reward is received each time certain actions is taken in given states. However, representing logical formulae over reached places (such as "places p_1 and p_2 are reached without reaching place q") are much more natural to be represented by such one-time rewards instead of cumulative rewards[6]. The framework can be extended to reward markings instead of places but at the cost of an exponential explosion since, to be able to compute the one-time step-wise rewards, not only already reached places but already reached markings would have to be memorized. Note that a reward need not be positive.

More formally, given a firing sequence $\mu : m_0 \to_{t_1} m_1 \to_{t_2} \ldots$, the agent receives a value or payoff of $V(pl(\mu))$ where $V(M) \coloneqq \sum_{Q \subseteq M} R(Q)$.

Example 3.2. *As an example consider the SDPN in Figure 2. The objective is to mark both places coloured in yellow at some point in time (not necessarily at the same time). This can be described by a reward function* R *which assigns* 1 *to the set* $\{p_4, p_5\}$ *containing both yellow places and* 0 *to all other sets.*

The transitions with double borders (t_1, t_2) *are controllable and it turns out that the optimal strategy is to deactive both* t_1 *and* t_2 *first, in order to let* t_5 *or* t_6 *mark either of the two goal places before reaching the marking* $(1, 1, 0, 0, 0)$ *from which no information can be gained which of the two goal places have been marked. An optimal strategy thus has to have knowledge of already achieved subgoals in terms of visited places. In this case, the strategy can deactivate one of the transitions* (t_1, t_2) *leading to the place already visited.*

[6] Firings of transitions can also easily be rewarded by adding an additional place.

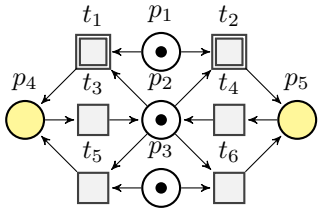

Fig. 2. Example SDPN

Policies may be dependent on the current marking and the places accumulated so far. Now, for a given policy $\pi : \mathcal{M}(N) \times \mathcal{P}(P) \to \mathcal{P}(C)$, determining the set $\pi(m, Q) \subseteq C$ of deactivated transitions in marking m for the set Q of places seen so far, we consider the (continuous) probability space $m_0\mathcal{M}(N)^\omega$, describing the infinite sequence $m_0 \to_{t_1} m_1 \to_{t_2} \dots$ of markings generated by the Petri net under the policy π (i.e., if in step n the transitions $D_{n+1} \coloneqq \pi(m_n, \bigcup_{k=0}^{n-1} \operatorname{supp}(m_k))$ are deactivated).

Then we can consider the expectation of the random variable $V \circ pl$, i.e.,

$$\mathbb{V}^\pi \coloneqq \mathbb{E}^\pi \left[V \circ pl \right],$$

over the probability space $m_0\mathcal{M}(N)^\omega$. We will call this the value of π and, if $\pi \equiv D \subseteq C$ is constant, simply write $\mathbb{V}^D$ which we will call the value of D.

For the complexity analyses we assume that R is only stored on its support, e.g., as a set $R \subseteq \mathcal{P}(P) \times \mathbb{R}$ which we will interpret as a dictionary with entries $[Q : R(Q)]$ for some $Q \subseteq P$, as for many problems of interest the size of the support of the reward function can be assumed to be polynomially bounded w.r.t. to the set of places and transitions.

We consider the following problems for stochastic Petri nets, where we parameterize over a class $\mathcal{N}$ of SDPNs and (for the second problem) over a class $\Psi \subseteq (\mathcal{M}(N) \times \mathcal{P}(P) \to \mathcal{P}(C))$ of policies:

- $\mathcal{N}$-VAL: Given a rational $p > 0$, a net $N \in \mathcal{N}$ and a policy π for N, decide whether $\mathbb{V}^\pi > p$.
- $\mathcal{N}$-POL: Given a rational $p > 0$ and a net $N \in \mathcal{N}$, decide whether there exist a policy $\pi \in \Psi$ such that $\mathbb{V}^\pi > p$.
  Although paramterized over sets of policies, we will omit Ψ if is clear from the context (in fact we will now restrict to constant policies from Section 5 onwards).

4 Stochastic decision Petri nets as Markov decision processes

We now describe how to transform an SDPN into an MDP, thus fixing the semantics of such nets. For unbounded Petri nets, the resulting MDP has an infinite state space, but we will restrict to the finite case later.

Definition 4.1. *Given an SDPN $N = (P, T, F, \Lambda, C, R, m_0)$ where m_0 is not the constant zero function, the MDP for N is defined as the tuple (S, A, δ, r, s_0) where*

- $S = \mathcal{R}(N) \times \mathcal{P}(P)$ *(product of reachable markings and places collected),*
- $A = \mathcal{P}(C)$ *(sets of deactivated transition as actions),*

– $\delta\colon (\mathcal{R}(N) \times \mathcal{P}(P)) \times \mathcal{P}(C) \to \mathcal{D}(\mathcal{R}(N) \times \mathcal{P}(P))$, *with*

$$\delta((m,Q),D)((m',Q')) \coloneqq \begin{cases} p(m' \mid m, D) & \text{if } Q' = Q \cup \operatorname{supp}(m), \\ 0 & \text{otherwise,} \end{cases}$$

where

$$p(m' \mid m, D) = \frac{\sum_{t \in En(m,D), m \to_t m'} \Lambda(t)}{\sum_{t \in En(m,D)} \Lambda(t)}$$

whenever $En(m,D) \coloneqq \{t \in T \backslash D \mid {}^\bullet t \leq m\} \neq \emptyset$. *If* $En(m,D) = \emptyset$, *we set* $p(m' \mid m, D) = 1$ *if* $m = m'$ *and* 0 *if* $m \neq m'$. *That is,* $p(m' \mid m, D)$ *is the probability of reaching* m' *from* m *when transitions* D *are deactivated.*

– $r\colon S \times A \times S \to \mathbb{R}$ *(reward function) with*

$$r((m,Q),D,(m',Q')) \coloneqq \begin{cases} \sum_{Q \subseteq Y \subseteq Q'} R(Y) & \text{if } Q = \emptyset, \\ \sum_{Q \subsetneq Y \subseteq Q'} R(Y) & \text{if } Q \neq \emptyset. \end{cases}$$

– $s_0 = (m_0, \emptyset)$

The transition probabilities are determined as for regular stochastic Petri nets where we consider only the rates of those transitions that have not been deactivated and that can be fired for the given marking. If no transition is enabled, we stay at the current marking with probability 1.

Note that the reward for the places reached in a marking m is only collected when we fire a transition leaving m. This is necessary as in the very first step we also obtain the reward for the empty set, which might be non-zero, and due to the fact that the initial marking is assumed to be non-empty, this reward for the empty set is only collected once.

The following result shows that the values of policies $\pi : S \to A$ (note that these are exactly the policies for the underlying SDPN) over the MDP are equal to the ones over the corresponding SDPN.

Proposition 4.2. *Let* $N = (P, T, F, \Lambda, C, R, m_0)$ *be an SDPN and* $M = (S, A, \delta, r, s_0)$ *the corresponding MDP. For any policy* $\pi : S \to A$, *we have*

$$(\mathbb{V}^\pi =)\mathbb{E}^\pi\left[V \circ pl\right] = \mathbb{E}^\pi\left[\sum_{n \in \mathbb{N}_0} r(\mathbf{s}_n, \pi(\mathbf{s}_n), \mathbf{s}_{n+1})\right]$$

where $(\mathbf{s}_n)_n$ *is the Markov chain resulting from following policy* π *in* M.

This provides an exact semantic for SDPNs via MDPs. Note, however, that for analysis purposes, even for safe Petri nets, the reachability set $\mathcal{R}(N)$ (as a subset of $\mathcal{P}(P)$) is generally of exponential size whence the transformation into an MDP can at best generally only yield algorithms of exponential worst-case-time. Hence, we will now restrict to specific subproblems and it will turn out that even with fairly severe restrictions to the type of net and the policies allowed, we obtain completeness results for complexity classes high in the polynomial hierarchy.

5 Complexity analysis for specific classes of Petri nets

For the remainder of this paper, we will consider the problem of finding optimal *constant* policies for certain classes of nets. In other words, the agent chooses *before* the execution of the Petri net which transitions to deactivate for its *entire* execution. For a net N, the policy space is thus given by

$$\Psi(N) = \{\pi : \mathcal{M}(N) \to \mathcal{P}(C) \mid \pi \equiv D \subseteq C\} \mathrel{\hat{=}} \mathcal{P}(C).$$

Since one can non-deterministically guess the maximizing policy (there are only exponentially many) and compute its value, it is clear that the complexity of the policy optimization problem $\mathcal{N}$-POL is bounded by the complexity of the corresponding value problem $\mathcal{N}$-VAL as follows: If, for a given class $\mathcal{N}$ of Petri nets, $\mathcal{N}$-VAL lies in the complexity class C, then $\mathcal{N}$-POL lies in NP^{C}.

We will now show the complexity of these problems for the three Petri net classes FCON, SAFC, and $[\varphi]$BPN and work out the connection to Bayesian networks. In the following we will assume that all probabilities are efficiently computable, allowing us to simulate all probabilistic choices with fair coins.

5.1 Complexity of safe and acyclic free-choice decision nets

We will first consider the case of Petri nets where the length of runs is bounded.

Proposition 5.1. *For any polynomial φ, the problem* $[\varphi]$BPN-VAL *is in* PP. *In particular,* $[\varphi]$BPN-POL *is in* $\mathsf{NP}^{\mathsf{PP}}$.

Proof (sketch). Given a Petri net N, a policy π and a bound p, a PP-algorithm for $[\varphi]$BPN-VAL can simulate the execution of the Petri net and calculate the resulting value, checking whether the expected value for π is greater than the pre-defined bound p. For this, we have to suitably adapt the threshold so that the probabilistic Turing machine accepts with probability greater than $1/2$ iff the reward for the given policy is strictly greater than p.

As the execution of the Petri net takes only polynomial time in the size of the Petri net (φ), this can be performed by a probabilistic Turing machine in polynomial time whence $[\varphi]$BPN-VAL lies in PP.

Since a policy can be guessed in polynomial time, we can also infer that $[\varphi]$BPN-POL is in $\mathsf{NP}^{\mathsf{PP}}$. □

This easily gives us the following corollary for SAFC nets.

Corollary 5.2. *The problem* SAFC-VAL *is in* PP *and* SAFC-POL *in* $\mathsf{NP}^{\mathsf{PP}}$.

Proof. This follows directly from Proposition 5.1 and the fact that SAFC $\subseteq$ $[id]$BPN. □

Proposition 5.3. *The problem* SAFC-POL *is* $\mathsf{NP}^{\mathsf{PP}}$*-hard and, therefore, also* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof (sketch). This can be proven via a reduction D-MAP $\leq_p$ SAFC-POL, i.e., representing the modified D-MAP problem for Bayesian networks as a decision problem in safe and acyclic free-choice nets. $\mathsf{NP}^{\mathsf{PP}}$-completeness then follows together with Corollary 5.2. Note that we are using the restricted version of the D-MAP problem as explained in Section 2 (uniformly distributed input nodes, binary values).

We sketch the reduction via an example: we take the Bayesian network in Figure 1 and consider a D-MAP instance where $E = \{c, d\}$ (evidence, where we fix the values of c, d to be $0, 1$), $F = \{a\}$ (MAP variables) and p is a threshold. That is, the question being asked for the Bayesian network is whether there exists a value x such that $\mathbb{P}(X_a = x \mid X_c = 0, X_d = 1) > p$.

This Bayesian network is encoded into the SAFC net in Figure 3, where transitions with double borders are controllable and the yellow places give a reward of 1 when both are reached (not necessarily at the same time). Transitions either have an already indicated rate of 1 or the rate can be looked up in the corresponding matrix of the BN. The rate of a transition $t^i_{x_1x_2\to x_3}$ is the probability value $P_i(x_3 \mid x_1x_2)$, where P_i is the probability matrix for $i \in \{a, b, c, d\}$.

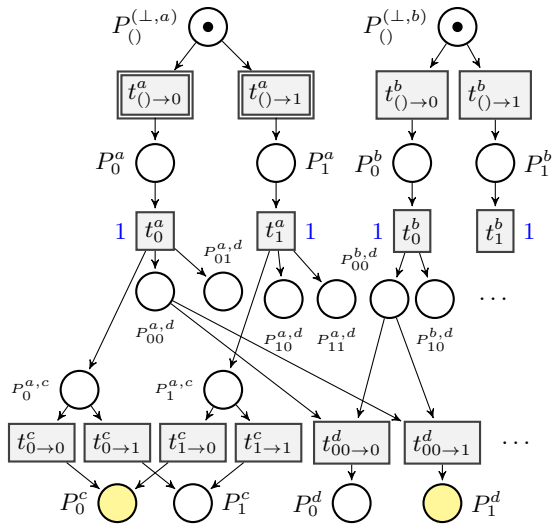

Fig. 3. SAFC net corresponding to BN in Figure 1

Intuitively the first level of transitions simulates the probability tables of P^a, P^b, the nodes without predecessors in the Bayesian network, where for instance the question of whether P^a_0 or P^a_1 are marked corresponds to the value of the random variable X_a associated with node a. Since X_a is a MAP variable, its two transitions are controllable. Note that enabling both transitions will never give a higher reward than enabling only one of them. (This is due to the fact that $\max\{x, y\} \geq p_1 \cdot x + p_2 \cdot y$ for $p_1, p_2 \geq 0$ with $p_1 + p_2 = 1$.)

The second level of transitions (each with rate 1) is inserted only to obtain a free-choice net by creating sufficiently many copies of the places in order to make all conflicts free-choice.

The third level of transitions simulates the probability tables of P^c, P^d, only to ensure the net being free-choice we need several copies. For instance, transition $t^c_{0\to 0}$ consumes a token from place $P^{a,c}_0$, a place specifically created for the entry $P^c(c = 0 \mid a = 0)$ in the probability table of node c.

In the end the aim is to mark the places P^c_0 and P^d_1, and we can find a policy (deactivating either $t^a_{()\to 0}$ or $t^a_{()\to 0}$) such that the probability of reaching both places exceeds p if and only if the D-MAP instance specified above has a solution.

This proof idea can be extended to more complex Bayesian networks, for a more formal proof see [29]. □

In fact, a reduction in the opposite direction (from Petri nets to Bayesian networks) is possible as well under mild restrictions, which shows that these problems are closely related.

Proposition 5.4. *For two given constants k, ℓ, consider the following problem: let N be a SAFC decision Petri net, where for each branching cell the number of controllable transitions is bounded by some constant k. Furthermore, given its reward function R, we assume that $|\cup_{Q \in \mathrm{supp}(R)} Q| \leq \ell$. Given a rational number p, does there exist a constant policy π such that $\mathbb{V}^{\pi} > p$?*

This problem can be polynomially reduced to D-MAP.

Proof (sketch). We sketch the reduction, which is inspired by [8], via an example: consider the SAFC net in Figure 5, where the problem is to find a deactivation pattern such that the payoff exceeds p. We encode the net into a Bayesian network (Figure 4), resulting in an instance of the D-MAP problem.

We have four types of random variables: place variables (X_p, $p \in P$), which record which place is marked; transition variables ($X_{t_1}, X_{t_5}, X_{t_6}$), one for each controllable transition, which are the MAP variables; cell variables ($X_{\mathbb{C}_i}$ for $\mathbb{C}_1 = \{t_1, t_2\}$, $\mathbb{C}_2 = \{t_3, t_4\}$, $\mathbb{C}_3 = \{t_5, t_6\}$) which are non-binary and which record which transition in the cell was fired or whether no transition was fired (ε): reward variable (X_{rew}) such that $\mathbb{P}(X_{rew} = 1)$ equals the function ψ applied to the payoff. Note that we use the affine function ψ from the proof of Proposition 5.1 to represent rewards as probabilities in the interval $[0, 1]$. Furthermore, the threshold for the D-MAP instance is $\psi(p)$.

Dependencies are based on the structure of the given SAFC net. For instance, $X_{\mathbb{C}_3}$ is dependent on X_{p_3}, X_{p_4} (since ${}^\bullet\mathbb{C}_3 = \{p_3, p_4\}$) and X_{t_5}, X_{t_6} (since t_5, t_6 are the controllable transitions in $\mathbb{C}_3$).

Both the matrices of cell and place variables could become exponentially large, however this problem can be resolved easily by dividing the matrices into smaller ones and cascading them. Since the number of controllable transitions is bounded by k and the number of rewarded places by ℓ, they will not cause an exponential blowup of the corresponding matrix. □

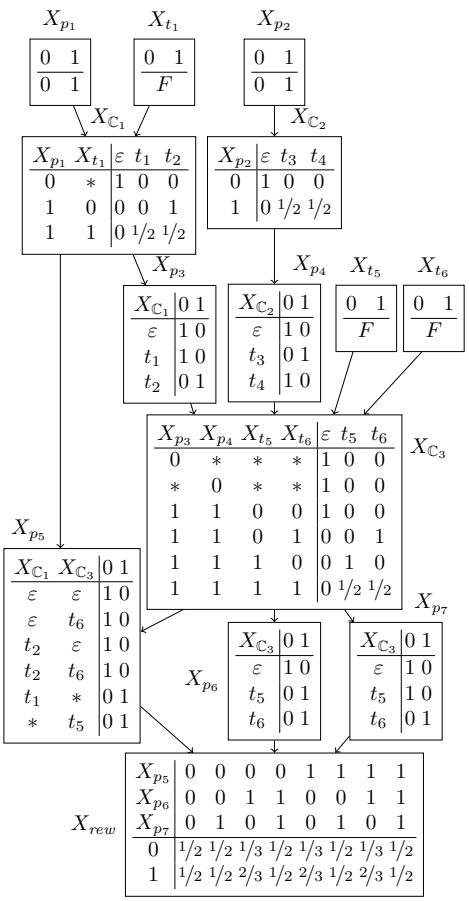

Fig. 4. Bayesian network obtained from the SAFC net in Figure 5 below. Entries $*$ are 'don't-care' values.

Corollary 5.5. *The problem* SAFC-VAL *is* PP*-hard and, therefore, also* PP*-complete.*

Proof. We note that using the construction in the proof of Proposition 5.3 with the set F of MAP variables being empty, we can reduce the D-PR problem for Bayesian networks to the SAFC-VAL problem, showing that SAFC-VAL is PP-hard. Using Corollary 5.2, this yields that SAFC-VAL is PP-complete. □

Corollary 5.6. *For any polynomial* $\varphi : \mathbb{N}_0 \to \mathbb{N}_0$ *fulfilling* $\varphi(n) \geq n$ *for all* $n \in \mathbb{N}_0$*, the problemm*$[\varphi]$BPN-VAL *is* PP*-complete and* $[\varphi]$BPN-POL *is* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof. As any safe and acyclic free-choice net is an *id*-bounded net, it is, in particular, a φ-bounded net with φ as above, and we have SAFC-VAL $\leq_p$ $[\varphi]$BPN-VAL and SAFC-POL $\leq_p$ $[\varphi]$BPN-POL. Propositions 5.1 and 5.3 as well as Corollary 5.5, therefore, show that $[\varphi]$BPN-VAL is PP-complete and $[\varphi]$BPN-POL is $\mathsf{NP}^{\mathsf{PP}}$-complete. □

5.2 Complexity of free-choice occurrence decision nets

Now we further restrict SAFC nets to occurrence nets, which leads to a substantial simplification. The main reason for this is the absence of backwards-conflicts, which means that each place is uniquely generated, making it easier to trace causality, i.e., there is a unique minimal configuration that generates each place.

Proposition 5.7. *The problem* FCON-VAL *is in* P*. In particular,* FCON-POL *is in* NP*.*

Proof (sketch). Determining the probability of reaching a set of places Q in an occurrence net amounts to multiplying the probabilities of the transitions on which the places in Q are causally dependent. This can be done for every set Q in the support of the reward function R, which enables us to determine the expected value in polynomial time, implying that FCON-VAL lies in P. By guessing a policy for an occurrence net with controllable transitions, we obtain that FCON-POL lies in NP. □

Proposition 5.8. *The problem* FCON-POL *is* NP*-hard and, therefore, also* NP*-complete.*

Proof (sketch). To show NP-hardness we reduce 3-SAT (the problem of deciding the satisfiability of a propositional formula in conjunctive normal form with at most three literals per clause) to FCON-POL. Given a formula ψ, this is done by constructing a simple occurrence net with parallel controllable transitions, one for each atomic proposition ℓ in ψ. Then we define a reward function with polynomial support in such a way that the expected reward for the constructed net is larger or equal than the number of clauses iff the formula has a model. The correspondence between the model and the policy is such that transitions whose atomic propositions are evaluated as true are deactivated. □

6 An algorithm for SAFC decision nets

Here we present a partial-order algorithm for solving the policy problem for SAFC (decision) nets. It takes such a net and converts it into a formula for an SMT solver. We will assume the following, which is also a requirement for occurrence nets:

Assumption 6.1. *For all places $p \in m_0$: ${}^\bullet p \coloneqq \{t \in T \mid p \in {}^\bullet t\} = \emptyset$.*

This is a mild assumption since any transition $t \in {}^\bullet p$ for a place $p \in m_0$ in a safe and acyclic net has to be dead as all places can only be marked once.

We are now using the notion of (branching) cells, introduced in Section 2: The fact that the SDPN is safe, acyclic and free-choice ensures that choices in different cells are taken independently from another, so that the probability of a configuration $\tau \in \mathcal{C}(N)$ under a specific deactivation pattern $D \subseteq C$ is given by

$$\mathbb{P}^D(tr \supseteq \tau) = \prod_{t \in \tau} \frac{\chi_{T \setminus D}(t)\Lambda(t)}{\sum_{t \in \mathbb{C}_t \setminus D} \Lambda(t)} = \begin{cases} 0 & \text{if } \tau \cap D \neq \emptyset \\ \prod_{t \in \tau} \frac{\Lambda(t)}{\sum_{t' \in \mathbb{C}_t \setminus D} \Lambda(t')} & \text{otherwise} \end{cases}$$

where $\chi_{T \setminus D}$ is the characteristic function of $T \setminus D$ and $0/0$ is defined to yield 0.

The general idea of the algorithm is to rewrite the reward function $R : \mathcal{P}(P) \to \mathbb{R}$ on sets of places to a reward function on sets of transitions that yields a compact formula for computing the value $\mathbb{V}^D$ for specific sets D (i.e., solving SAFC-VAL), that we can also use to solve the policy problem SAFC-POL via an SMT solver.

We first need some definitions:

Definition 6.2. *For a maximal configuration $\tau \in \mathcal{C}^\omega(N_D)$ for a given deactivation pattern $D \subseteq C$, we define its set of prefixes in $\mathcal{C}(N_D)$ to be*

$$\mathrm{pre}^D(\tau) \coloneqq \{\tau' \in \mathcal{C}(N_D) \mid \tau' \subseteq \tau\}$$

which corresponds to all configurations that can lead to the configuration τ. We also define the set of extensions of a configuration $\tau \in \mathcal{C}(N_D)$ in $\mathcal{C}^\omega(N_D)$, which corresponds to all maximal configurations that τ can lead to, as

$$\mathrm{ext}^D(\tau) \coloneqq \{\tau' \in \mathcal{C}^\omega(N_D) \mid \tau \subseteq \tau'\}.$$

Definition 6.3. *Let N be a Petri net with a reward function $R\colon \mathcal{P}(P) \to \mathbb{R}$ on places and a deactivation pattern D. A reward function $[R]\colon \mathcal{P}(T) \to \mathbb{R}$ on transitions is called* consistent *with R if for each firing sequence $\mu \in \mathcal{FS}(N_D)$:*

$$V(pl(\mu)) = \sum_{Q \subseteq pl(\mu)} R(Q) = \sum_{\tau \in \mathrm{pre}^D(tr(\mu))} [R](\tau).$$

This gives us the following alternative method to determine the expected value for a net (with given policy D):

Lemma 6.4. *Using the setting of Definition 6.3, whenever $[R]$ is consistent with the reward function R and $[R](\tau) = 0$ for all $\tau \notin \mathcal{C}(N)$, the expected value for the net N under the constant policy D is:*

$$\mathbb{V}^D = \sum_{\tau \subseteq T} \mathbb{P}^D(tr \supseteq \tau) \cdot [R](\tau).$$

Note that $[R](tr(\mu)) \coloneqq V(pl(\mu))$ for $\mu \in \mathcal{FS}(N)$ fulfills these properties trivially. However, rewarding only maximal configurations can lead, already in occurrence nets with some concurrency, to an exponential support (w.r.t. the size of the net and its reward function). The goal of our algorithm is to instead make use of the sum over the configurations by rewarding reached places immediately in the corresponding configuration, generating a function $[R]$ that fulfills the properties above and remains of polynomial size in occurrence nets. Hence, we have some form of partial-order technique, in particular concurrent transitions receive the reward independently of each other (if the reward is not dependent on firing both of them).

The rewriting process is performed by iteratively 'removing maximal cells' and resembles a form of backward-search algorithm. First of all, $\preceq_N^*$ (the reflexive and transitive closure of causality $\prec_N$) induces a partial order $\sqsubseteq$ on the set $BC(N)$ of cells via

$$\forall \mathbb{C}, \mathbb{C}' \in BC(N) : \mathbb{C} \sqsubseteq \mathbb{C}' \iff \exists t \in \mathbb{C}, t' \in \mathbb{C}' : t \preceq_N^* t'.$$

Let all cells $(\mathbb{C}_1, \ldots, \mathbb{C}_m)$ with $m = |BC(N)|$ be ordered conforming to $\sqsubseteq$, then we let N_k denote the Petri net consisting of places $P_k \coloneqq P \setminus (\bigcup_{l>k} \mathbb{C}_l{}^\bullet) \cup (\bigcup_{l \leq k} \mathbb{C}_l{}^\bullet)$ (where the union with the post-sets is only necessary if backward-conflicts exist) and transitions $T_k \coloneqq \bigcup_{l \leq k} \mathbb{C}_l$, the remaining components being accordingly restricted (note that the initial marking m_0 is still contained in P_k by Assumption 6.1). In particular, it holds that $N = N_m$ as well as $T_0 = \emptyset$ and $P_0 = \{p \in P \mid \forall t \in T : p \notin t^\bullet\}$.

Let N be a Petri net with deactivation pattern D, $\mu \in \mathcal{FS}(N_D)$ be a firing sequence and $k \in \{1, \ldots, |BC(N)|\}$. We write $tr_{\leq k}(\mu) \coloneqq tr(\mu) \cap T_k$ for the transitions in the first k cells and $tr_{>k}(\mu) \coloneqq tr(\mu) \setminus T_k$ for the transitions in the cells after the k-th cell as well as $pl_{\leq k}(\mu) \coloneqq m_0 \cup (\bigcup_{t \in tr_{\leq k}(\mu)} t^\bullet)$ for the places reached after all transitions in the first k cells were fired.

We will now construct auxiliary reward functions $R[k]$ that take pairs of a set of places ($U \subseteq P_k$) and of transitions ($V \subseteq T \setminus T_k$) as input and return a reward. Intuitively, $R[k](U, V)$ corresponds to the reward for reaching all places in U and then firing all transitions in V afterwards where reaching U ensures that all transitions in V can fire.

Starting with the reward function $R[m] : \mathcal{P}(P) \times \{\emptyset\} \to \mathbb{R}, (M, \emptyset) \mapsto R(M)$, we iteratively compute reward functions $R[k] : \mathcal{P}(P_k) \times \mathcal{P}(T \setminus T_k) \to \mathbb{R}$ for $k \geq 0$:

$$R[k](U,V) \coloneqq \begin{cases} R[k+1](U,V) & \text{if } \mathbb{C}_{k+1} \cap V = \emptyset \\ \sum\limits_{\substack{U' \cap t^\bullet \neq \emptyset \\ U = U' \setminus t^\bullet \cup {}^\bullet t}} R[k+1](U', V \setminus \{t\}) & \text{if } \mathbb{C}_{k+1} \cap V = \{t\} \\ 0 & \text{otherwise} \end{cases}$$

The first case thus describes a scenario where no transition from the $(k+1)$-th cell is involved while the second case sums up all rewards that are reached when some transition t in the cell has to be fired (that is, all rewards that are given when some of the places in $t^\bullet$ are reached). We give non-zero values only to sets V that contain at most one transition of each cell and U has to contain the full pre-set of t of the transition t removed from V. This is done in order to ensure that in subsequent steps those transitions that generate ${}^\bullet t$ are in the set to which we assign the reward. This ensures that V is always a configuration of N for the initial marking U while $R[k](U,V)$ is zero if the transitions in V cannot be fired after U. In this way, rewards are ultimately only given to configurations and to no other sets of transitions, enabling us later to compute the probabilities of those configurations.

And if N is an occurrence net, every entry in $R[k+1]$ produces at most one entry in $R[k]$, meaning that $\operatorname{supp}(R[k]) \leq \operatorname{supp}(R[k+1])$.

Now we can prove that the value of a firing sequence is invariant when rewriting the auxiliary reward functions as described above.

Proposition 6.5. *The auxiliary reward functions satisfy*

$$\sum_{V \subseteq tr_{>k}(\mu)} \sum_{U \subseteq pl_{\leq k}(\mu)} R[k](U,V) = \sum_{V \subseteq tr_{>k+1}(\mu)} \sum_{U \subseteq pl_{\leq k+1}(\mu)} R[k+1](U,V),$$

for $k \in \{0, \ldots, |BC(N)| - 1\}$.

Hence, for every $\mu \in \mathcal{FS}(N)$

$$V(pl(\mu)) = \sum_{U \subseteq pl(\mu)} R[|BC(N)|](U,\emptyset) = \sum_{V \subseteq tr_{>k}(\mu)} \sum_{U \subseteq pl_{\leq k}(\mu)} R[k](U,V),$$

which means that we obtain a reward function on transitions consistent with R *by defining* $[R] : \mathcal{P}(T) \to \mathbb{R}$ *as*

$$[R](V) \coloneqq \sum_{U \subseteq m_0} R[0](U,V).$$

This leads to the following corollary:

Corollary 6.6. *Given a net* N *and a deactivation pattern* D*, we can calculate the expected value*

$$\mathbb{V}^D = \mathbb{E}[V \circ pl] = \sum_{\tau \subseteq T} \prod_{t \in \tau} \frac{\chi_{T \setminus D}(t) \cdot \Lambda(t)}{\sum_{t' \in \mathbb{C}_t \setminus D} \Lambda(t')} [R](\tau).$$

Checking whether some deactivation pattern D exists such that this term is greater than some bound p can be checked by an SMT solver.

Note that, in contrast to the naive definition of $[R]$ only on maximal configurations, this algorithm constructs a reward function on configurations that, for occurrence nets, has a support with at most $\mathrm{supp}(R)$ elements. For arbitrary SAFC nets, the support of $[R]$ might be of exponential size.

Example 6.7. *We take the Petri net from Figure 5 as an example (where all transitions have firing rate 1). The reward function R is given in the table below. By using the inclusion-exclusion principle we ensure that one obtains reward 1 if one or both of the yellow places are marked at some point without ever marking the red place.*
The optimal strategy is obviously to only deactivate the one transition (t_6) which would mark the red place.
The net has three cells $\mathbb{C}_1 = \{t_1, t_2\}, \mathbb{C}_2 = \{t_3, t_4\}$, and $\mathbb{C}_3 = \{t_5, t_6\}$ where $\mathbb{C}_1, \mathbb{C}_2 \sqsubseteq \mathbb{C}_3$. As such, $R[3] \mathrel{\hat{=}} R$ with R above and obtain $R[2]$ (due to $P_2 = \{p_1, p_2, p_3, p_4, p_5\}$). In the next step, we get (by removing t_3 and t_4) $R[1]$ and finally $R[0]$, from which we can derive $[R]$, the reward function on transitions, as described above.

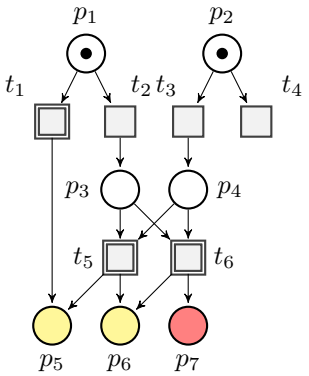

Fig. 5. A SAFC decision net. The goal is to mark one or both of the yellow places at some point without ever marking the red place.

This allows us to write the value for a set D of deactivated transitions as follows (where if both $t_5, t_6 \in D$, we assume the last quotient to be zero)

$$\mathbb{V}^D = \frac{\chi_{T\setminus D}(t_1)}{\chi_{T\setminus D}(t_1)+1} + \frac{1}{\chi_{T\setminus D}(t_1)+1}\frac{1}{2}\frac{\chi_{T\setminus D}(t_5)}{\chi_{T\setminus D}(t_5)+\chi_{T\setminus D}(t_6)}$$

$$\begin{aligned}
R &= [\{p_5\}:1, \{p_6\}:1, \{p_5,p_6\}:-1, \{p_5,p_7\}:-1, \{p_6,p_7\}:-1, \{p_5,p_6,p_7\}:1]\\
R[2] &= [(\{p_5\},\emptyset):1, (\{p_3,p_4\},\{t_5\}):1, (\{p_3,p_4,p_5\},\{t_6\}):-1]\\
R[1] &= [(\{p_5\},\emptyset):1, (\{p_2,p_3\},\{t_3,t_5\}):1, (\{p_2,p_3,p_5\},\{t_3,t_6\}):-1]\\
R[0] &= [(\{p_1\},\{t_1\}):1, (\{p_1,p_2\},\{t_2,t_3,t_5\}):1]\\
[R] &= [\{t_1\}:1, \{t_2,t_3,t_5\}:1]
\end{aligned}$$

Writing $x_i \coloneqq \chi_{T\setminus D}(t_i) \in \{0,1\}, i = 1,5,6$, the resulting inequality

$$\frac{x_1}{x_1+1} + \frac{1}{2}\frac{1}{x_1+1}\frac{x_5}{x_5+x_6} > p$$

can now be solved by an SMT solver with Boolean variables x_1, x_5, and x_6 (i.e., $x_1, x_5, x_6 \in \{0,1\}$).

Runtime results: To test the performance of our algorithm, we performed runtime tests on specific families of simple stochastic decision Petri nets, focussing on the impact of concurrency and backward-conflicts on its runtime. All families are based on a series of simple branching cells each containing two transitions, one controllable and one non-controllable, reliant on one place as a precondition. Each non-controllable transition marks a place to which we randomly assigned a reward according to a normal distribution (in particular, it can be negative). The families differ in how these cells are connected, testing performance with concurrency, backward-conflicts, and sequential problems (for a detailed overview of the experiments see [29]), respectively.

Rewriting the reward function (and, thus, solving the value problem) produced expected results: Runtimes on nets with many backward-conflicts are exponential while the rewriting of reward functions of occurrence nets exhibits a much better performance, reflecting its polynomial complexity.

To solve the policy problem based on the rewritten reward function, we compared the performances of naively calculating the values of each possible deactivation pattern with using an SMT solver (Microsoft's z3, see also [12]). Tests showed a clear impact on the representation of the control variables (describing the deactivation set D) as booleans or as integers bounded by 0 and 1 with the latter showing a better performance. Furthermore, the runtime of solving the rewritten formula with an SMT solver showed a high variance on random reward values. Nonetheless, the results show the clear benefit of using the SMT solver on the rewritten formula in scenarios with a high amount of concurrency, with much faster runtimes than the brute force approach. In scenarios without concurrency, this benefit vanishes, and in scenarios with many backward-conflicts, the brute force approach is considerably faster than solving the rewritten function with an SMT solver. The latter effect can be explained by the rewritten reward function $[R]$ having an exponential support in this scenario.

All in all, the runtime results reflect the well-known drawbacks and benefits of most partial-order techniques, excelling in scenarios with high concurrency while having a reduced performance if there are backward- and self-conflicts.

7 Conclusion

We have introduced the formalism of stochastic decision Petri nets and defined its semantics via an encoding into Markov decision processes. It turns out that finding optimal policies for a model that incorporates concurrency, probability and decisions, is a non-trivial task. It is computationally hard even for restricted classes of nets and constant policies. However, we remark that workflow nets are often SAFC nets and a constant deactivation policy is not unreasonable, given that one cannot monitor and control a system all the time. We have also presented an algorithm for the studied subproblem, which we view as a step towards efficient partial-order techniques for stochastic (decision) Petri nets.

Related Work: Petri nets [26] are a well-known and widely studied model of concurrent systems based on consumption and generation of resources. Several

subclasses of Petri nets have received attention, among them free-choice nets [13] and occurrence nets, where the latter are obtained by unfolding Petri nets for verification purposes [14].

Our notion of stochastic decision Petri nets is an extension to the well-known model of stochastic Petri nets [21]. This model and a variety of generalizations are used for the quantitative analyses of concurrent systems. Stochastic Petri nets come in a continuous-time and in a discrete-time variant, as treated in this paper. That is, using the terminology of [28], we consider the corresponding Markov chain of jumps, while in the continuous-time case, firing rates determine not only the probability which transition fires next, but also how fast a transition will fire dependent on the marking. These firing times are exponentially distributed, a distribution that is memoryless, meaning that the probability of a transition firing is independent on its waiting time.

B: remove this sentence if we need space.

Our approach was motivated by extending the probabilistic model of stochastic Petri nets by a mechanism for decision making, as in the extension of Markov chains [28] to Markov decision processes (MDPs) [6]. Since the size of a stochastic Petri net might be exponentially smaller than the Markov chain that it generates, the challenge is to provide efficient methods for determining optimal strategies, preferably partial order methods that avoid the explicit representation of concurrent events in an interleaving semantics. Our complexity results show that the quest for such methods is non-trivial, but some results can be achieved by suitably restricting the considered Petri nets.

R2: If possible, discuss how the models compare to each other

A different approach to include decision-making in Petri nets was described by Beccuti et al. as Markov decision Petri nets [5,4]. Their approach, based on a notion of well-formed Petri nets, distinguishes explicitly between a probabilistic part and a non-deterministic part of the Petri net as well as a set of components that control the transitions. They use such nets to model concurrent systems and obtain experimental results. In a similar vein, graph transformation systems – another model of concurrent systems into which Petri nets can be encoded – have been extended to probabilistic graph transformation systems, including decisions in the MDP sense [18]. The decision is to choose a set of rules with the same left-hand side graph and a match, then a randomized choice is made among these rules. Again, the focus is on modelling and to our knowledge neither of these approaches provides complexity results.

Another problem related to the ones considered in this paper is the computation of the expected execution time of a timed probabilistic Petri net as described in [22]. The authors treated timed probabilistic workflow nets (TPWNs) which assumes that every transition requires a fixed duration to fire, separate from the firing probability. They showed that approximating the expected time of a sound SAFC TPWN is #P-hard which is the functional complexity class corresponding to PP. While the problems studied in their paper and in our paper are different, the fact that both papers consider SAFC nets and obtain a #P- respectively PP-hardness result seems interesting and deserves further study.

Our complexity results are closely connected with the analysis of Bayesian networks [25], which are a well-known graphical formalism to represent condi-

tional dependencies among random variables and can be employed to reason about and compactly represent probability distributions. The close relation between Bayesian networks and occurrence nets was observed in [8], which gives a Bayesian network semantics for occurrence nets, based on the notion of branching cells from [1] that were introduced in order to reconcile partial order methods – such as unfoldings – and probability theory. We took inspiration from this reduction in Proposition 3 and another of our reductions (Proposition 5.3) – encoding Petri nets as Bayesian networks – is a transformation going into the other direction, from Bayesian networks to SAFC nets.

In our own work [9,7] we considered a technique for uncertainty reasoning, combining both Petri nets and Bayesian networks, albeit in a rather different setting. There we considered Petri nets with uncertainty, where one has only probabilistic knowledge about the current marking of the net. In this setting Bayesian networks are used to compactly store this probabilistic knowledge and the main challenge is to update respectively rewrite Bayesian networks representing such knowledge whenever the Petri net fires.

Future Work: As future work we plan to consider more general classes of Petri nets, lifting some of the restrictions imposed in this paper. In particular, it would be interesting to extend the method from Section 6 to nets that allow infinite runs. Furthermore, dropping the free-choice requirement is desirable, but problematic. While the notion of branching cells does exist for stochastic nets (see [1,8]), it does not accurately reflect the semantics of stochastic nets (see e.g. the discussion on confusion in the introduction of [8]).

As already detailed in the introduction, partial-order methods for analyzing probabilistic systems, modelled for instance by stochastic Petri nets, are in general poorly understood. Hence, it would already be a major result to obtain scalable methods for computing payoffs values for a stochastic net without decisions, but with a high degree of concurrency.

In addition we plan to use the encoding of Petri nets into Bayesian networks from [8] (on which we based the proof of Proposition 5.4) and exploit it to analyze such nets by using dedicated methods for reasoning on Bayesian networks.

Naturally, it would be interesting to extend analysis techniques in such a way that they can deal with uncertainty and derive policies when we have only partial knowledge, as in partially observable Markov decision process (POMDPs), first studied in [3]. However, this seems complex, given the fact that determining the best strategy for POMDPs is a non-trivial problem in itself [10].

Similarly, it is interesting to introduce a notion of time as in continuous-time Markov chains [28], enabling us to compute expected execution times as in [22].

Last but not least, our complexity analysis and algorithm focus on finding optimal constant policies. A natural step would be to instead consider the problem of finding optimal positional strategies as defined in Chapter 3, which is the standard problem in the analysis of Markov decision processes (see for example [**?**]).

References

1. Samy Abbes and Albert Benveniste. True-concurrency probabilistic models: branching cells and distributed probabilities for event structures. *Information and Computation*, 204(2):231–274, 2006.
2. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
3. K. J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.
4. Marco Beccuti, Elvio Gilberto Amparore, Susanna Donatelli, Dimitri Scheftelowitsch, Peter Buchholz, and Giuliana Franceschinis. Markov decision Petri nets with uncertainty. In *Prof. of EPEW '15*, volume 9272 of *LNCS*, pages 177–192. Springer, 2015.
5. Marco Beccuti, Giuliana Franceschinis, and Serge Haddad. Markov decision Petri net and Markov decision well-formed net formalisms. In *Petri Nets and Other Models of Concurrency*, volume 4546 of *LNCS*, pages 43–62. Springer, 2007.
6. Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.
7. Rebecca Bernemann, Benjamin Cabrera, Reiko Heckel, and Barbara König. Uncertainty reasoning for probabilistic Petri nets via Bayesian networks. In *Proc. of FSTTCS '20*, volume 182 of *LIPIcs*, pages 38:1–38:17. Schloss Dagstuhl – Leibniz Center for Informatics, 2020.
8. Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Bayesian network semantics for Petri nets. *Theor. Comput. Sci.*, 807:95–113, 2020.
9. Benjamin Cabrera, Tobias Heindel, Reiko Heckel, and Barbara König. Updating probabilistic knowledge on Condition/Event nets using Bayesian networks. In *Proc. of CONCUR '18*, volume 118 of *LIPIcs*, pages 27:1–27:17. Schloss Dagstuhl – Leibniz Center for Informatics, 2018.
10. Anthony R. Cassandra. *Exact and Approximate Algorithms for Markov Decision Processes*. PhD thesis, Brown University, USA, 1998.
11. Cassio P. de Campos. New complexity results for MAP in Bayesian networks. In *Proc. of IJCAI '11*, pages 2100–2106. IJCAI/AAAI, 2011.
12. Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Proc. of TACAS '08*, pages 337–340. Springer, 2008.
13. Jörg Desel and Javier Esparza. *Free choice Petri nets*. Number 40 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1995.
14. Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial Order Approach to Model Checking*. Springer, 2008.
15. E.A. Feinberg and A. Shwartz, editors. *Handbook of Markov Decision Processes*. Kluwer, Boston, MA, 2002.
16. M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, 1979.
17. Charles Grinstead and Laurie Snell. Markov chains. In *Introduction to Probability*, chapter 11, pages 405–470. American Mathematical Society, second edition, 1997.
18. Christian Krause and Holger Giese. Probabilistic graph transformation systems. In *Proc. of ICGT '12*, volume 7562 of *LNCS*, pages 311–325. Springer, 2012.
19. Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proc. of UAI '95*, pages 394–402. Morgan Kaufmann, 1995.
20. Michael L. Littman, Stephen M. Majercik, and Toniann Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.

21. Marco Ajmone Marsan. Stochastic Petri nets: an elementary introduction. In *Advances in Petri Nets 1989*, volume 424 of *LNCS*, pages 1–29. Springer, 1989.
22. Philipp J. Meyer, Javier Esparza, and Philip Offtermatt. Computing the expected execution time of probabilistic workflow nets. In *Proc. of TACAS '19*, volume 11428 of *LNCS*, pages 154–171. Springer, 2019.
23. Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
24. James D. Park and Adnan Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.
25. Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.
26. Wolfgang Reisig. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1985.
27. Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
28. Anders Tolver. An introduction to Markov chains. *Department of Mathematical Sciences, University of Copenhagen*, 2016.
29. Florian Wittbold, Rebecca Bernemann, Reiko Heckel, Tobias Heindel, and Barbara König. Stochastic Decision Petri Nets, 2023. arXiv:????.????