Uncertainty Reasoning for Probabilistic Petri Nets via Bayesian Networks

Rebecca Bernemann
University of Duisburg-Essen, Germany
rebecca.bernemann@uni-due.de

Benjamin Cabrera
University of Duisburg-Essen, Germany
benjamin.cabrera@uni-due.de

Reiko Heckel
University of Leicester, UK
rh122@leicester.ac.uk

Barbara König
University of Duisburg-Essen, Germany
barbara_koenig@uni-due.de

Abstract

This paper exploits extended Bayesian networks for uncertainty reasoning on Petri nets, where firing of transitions is probabilistic. In particular, Bayesian networks are used as symbolic representations of probability distributions, modelling the observer's knowledge about the tokens in the net. The observer can study the net by monitoring successful and failed steps.

An update mechanism for Bayesian nets is enabled by relaxing some of their restrictions, leading to modular Bayesian nets that can conveniently be represented and modified. As for every symbolic representation, the question is how to derive information – in this case marginal probability distributions – from a modular Bayesian net. We show how to do this by generalizing the known method of variable elimination. The approach is illustrated by examples about the spreading of diseases (SIR model) and information diffusion in social networks. We have implemented our approach and provide runtime results.

2012 ACM Subject Classification Mathematics of computing → Bayesian networks; Software and its engineering → Petri nets

Keywords and phrases uncertainty reasoning, probabilistic knowledge, Petri nets, Bayesian networks

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.38

Related Version A full version of the paper is available as [1], `https://arxiv.org/abs/2009.14817`.

Funding This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant GRK 2167, Research Training Group "User-Centred Social Media".

1 Introduction

Today's software systems and the real-world processes they support are often distributed, with agents acting independently based on their own local state but without complete knowledge of the global state. E.g., a social network may expose a partial history of its users' interactions while hiding their internal states. An application tracing the spread of a virus can record test results but not the true infection state of its subjects. Still, in both cases, we would like to derive knowledge under uncertainty to allow us, for example, to predict the spread of news in the social network or trace the outbreak of a virus.

Using Petri nets as a basis for modelling concurrent systems, our aim is to perform uncertainty reasoning on Petri nets, employing Bayesian networks as compact representations of probability distributions. Assume that we are observing a discrete-time concurrent system modelled by a Petri net. The net's structure is known, but its initial state is uncertain, given only as an a-priori probability distribution on markings. The net is probabilistic: Transitions are chosen at random, either from the set of enabled transitions or independently, based on probabilities that are known but may change between steps. We cannot observe which transition actually fires, but only if firing was successful or failed. Failures occur if the chosen transition is not enabled under the current marking (in the case where we choose transitions independently), if no transition can fire, or if a special fail transition is chosen. After observing the system for a number of steps, recording a sequence of "success" and "failure" events, we then determine a marginal distribution on the markings (e.g., compute the probability that a given place is marked), taking into account all observations.

First, we set up a framework for uncertainty reasoning based on time-inhomogeneous Markov chains that formally describes this scenario, parameterized over the specific semantics of the probabilistic net. This encompasses the well-known stochastic Petri nets [29], as well as a semantics where the choice of the marking and the transition is independent (Sct. 2 and 3). Using basic Bayesian reasoning (reminiscent of methods used for hidden Markov models [32]), it is conceptually relatively straightforward to update the probability distribution based on the acquired knowledge. However, the probability space is exponential in the number of places of the net and hence direct computations become infeasible relatively quickly.

Following [5], our solution is to use (modular) Bayesian networks [36, 13, 31] as compact symbolic representations of probability distributions. Updates to the probability distribution can be performed very efficiently on this data structure, simply by adding additional nodes. By analyzing the structure of the Petri net we ensure that this node has a minimal number of connections to already existing nodes (Sct. 4 and 5).

As for every symbolic representation, the question is how to derive information, in this case marginal probability distributions. We solve this question by generalizing the known method of variable elimination [14, 13] to modular Bayesian networks. This method is known to work efficiently for networks of small treewidth, a fact that we experimentally verify in our implementation (Sct. 6 and 7).

We consider some small application examples modelling gossip and infection spreading.

Summarized, our contributions are:

- We propose a framework for uncertainty reasoning based on time-inhomogeneous Markov chains, parameterized over different types of probabilistic Petri nets (Sct. 2 and 3).
- We use modular Bayesian networks to symbolically represent and update probability distributions (Sct. 4 and 5).
- We extend the variable elimination method to modular Bayesian networks and show how it can be efficiently employed in order to compute marginal distributions (Sct. 6). This is corroborated by our implementation and runtime results (Sct. 7).

All proofs and further material can be found in the full version [1].

2 Markov Chains and Probabilistic Condition/Event Nets

2.1 Markov Chains

Markov chains [18, 35] are a stochastic state-based model, in which the probability of a transition depends only on the state of origin. Here we restrict to a finite state space.

▶ **Definition 1** (Markov chain). *Let Q be a finite state space. A* (discrete-time) Markov chain *is a sequence $(X_n)_{n\in\mathbb{N}_0}$ of random variables such that for $q, q_0, \ldots, q_n \in Q$:*

$$P(X_{n+1} = q \mid X_n = q_n) = P(X_{n+1} = q \mid X_n = q_n, \ldots, X_0 = q_0).$$

Assume that $|Q| = k$. Then, the probability distribution over Q at time n can be represented as a k-dimensional vector p^n, indexed over Q. We abbreviate $p^n(q) = P(X_n = q)$. We define $k \times k$-transition matrices P^n, indexed over Q, with entries[1] for the entry of matrix M at row q' and column q: $P^n(q' \mid q) = P(X_{n+1} = q' \mid X_n = q)$. Note that $p^{n+1} = P^n \cdot p^n$. We do not restrict to time-homogeneous Markov chains where it is required that $P^n = P^{n+1}$ for all $n \in \mathbb{N}_0$. Instead, the probability distribution on the transitions might vary over time.

2.2 Probabilistic Condition/Event Nets

As a basis for probabilistic Petri nets we use the following variant of condition/event nets [33]. Deviating from [33], we omit the initial marking and furthermore the fact that the post-condition is marked is not inhibiting the firing of a transition. That is, we omit the so-called contact condition, which makes it easier to model examples from application scenarios where the contact condition would be unnatural. Note however that we could easily accommodate the theory to include this condition, as we did in the predecessor paper [5].

▶ **Definition 2** (condition/event net). *A* condition/event net *(C/E net or simply Petri net) $N = (S, T, {}^\bullet(), ()^\bullet)$ is a four-tuple consisting of a finite set of* places *S, a finite set of* transitions *T with* pre-conditions *${}^\bullet() : T \to \mathcal{P}(S)$ and* post-conditions *$()^\bullet : T \to \mathcal{P}(S)$. A* marking *is any subset of places $m \subseteq S$ and will also be represented by a bit string $m \in \{0,1\}^{|S|}$ (assuming an ordering on the places).*

A transition t can fire *for a marking $m \subseteq S$ if ${}^\bullet t \subseteq m$. Then marking m is transformed into $m' = (m \setminus {}^\bullet t) \cup t^\bullet$, written $m \overset{t}{\Rightarrow} m'$. We write $m \overset{t}{\Rightarrow}$ to indicate that there exists some m' with $m \overset{t}{\Rightarrow} m'$ and $m \overset{t}{\not\Rightarrow}$ if this is not the case. We denote the* set of all markings *by $\mathcal{M} = \mathcal{P}(S)$.*

In order to obtain a Markov chain from a C/E net, we need the following data: given a marking m and a transition t, we denote by $r_n(m,t)$ the probability of firing t in marking m (at step n), and by $r_n(m, \mathrm{fail})$ the probability of going directly to a fail state $*$.

▶ **Definition 3.** *Let $N = (S, T, {}^\bullet(), ()^\bullet)$ be a condition/event net and let $T_f = T \cup \{\mathrm{fail}\}$ (the set of transitions enriched with a fail transition). Furthermore let $r_n \colon \mathcal{M} \times T_f \to [0,1]$, $n \in \mathbb{N}_0$ be a family of functions (the transition distributions at step n), such that for each $n \in \mathbb{N}_0$, $m \in \mathcal{M}$: $\sum_{t\in T_f} r_n(m,t) = 1$.*

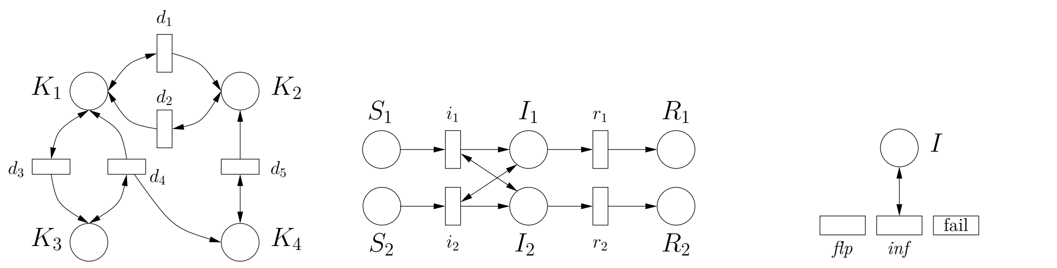
The Markov chain generated from *N, r_n has states $Q = \mathcal{M} \cup \{*\}$ and for $m, m' \in \mathcal{M}$:*

$$P(X_{n+1} = m' \mid X_n = m) = \textstyle\sum_{t\in T, m \overset{t}{\Rightarrow} m'} r_n(m,t) \qquad P(X_{n+1} = m' \mid X_n = *) = 0$$
$$P(X_{n+1} = * \mid X_n = m) = \textstyle\sum_{t\in T_f, m \overset{t}{\not\Rightarrow}} r_n(m,t) \qquad P(X_{n+1} = * \mid X_n = *) = 1$$

where we assume that $m \overset{\mathrm{fail}}{\not\Rightarrow}$ for every $m \in \mathcal{M}$.

Note that we can make a transition from m to the fail state $*$ either when there is a non-zero probability for performing such a transition directly or when we pick a transition that

[1] We are using the notation $M(q' \mid q)$, resembling conditional probability,



(a) A Petri net modelling gossip diffusion in a social network (K_i: i knows information).

(b) A Petri net modelling spread of a disease (S: susceptible, I: infected, R: removed).

(c) A Petri net modelling a test with false positives and negatives (I: infected).

Figure 1 Example Petri nets.

cannot be fired in m. Requiring that $m \overset{\text{fail}}{\not\Rightarrow}$ for every m is for notational convenience, since we have to sum up all probabilities leading to the fail state $*$ to compute $P(X_{n+1} = * \mid X_n = m)$. In this way the symbol $\not\Rightarrow$ always signifies a transition to $*$.

By parametrising over r_n we obtain different semantics for condition/even nets. In particular, we consider the following two probabilistic semantics, both based on probability distributions $p_T^n \colon T \to [0,1]$, $n \in \mathbb{N}_0$ on transitions. We work under the assumption that this information is given or can be gained from extra knowledge that we have about our environment.

Independent case: Here we assume that the marking and the transition are drawn independently, where markings are distributed according to p^n and transitions according to p_T^n. It may happen that the transition and the marking do not "match" and the transition cannot fire. Formally, $r_n(m,t) = p_T^n(t)$, $r_n(m, \text{fail}) = 0$ (where $m \in \mathcal{M}$, $t \in T$). This extends to the case where fail has non-zero probability, with probability distribution $p_T^n \colon T_f \to [0,1]$.

Stochastic net case: We consider stochastic Petri nets [29] which are often provided with a semantics based on continuous-time Markov chains [35]. Here, however we do not consider continuous time, but instead model the embedded discrete-time Markov chain of jumps that abstracts from the timing. The firing rate of a transition t is proportional to $p_T^n(t)$.

Intuitively, we first sample a marking m (according to p^n) and then sample a transition, restricting to those that are enabled in m. Formally, for every $t \in T_f$, $r_n(m,t) = 0$, $r_n(m, \text{fail}) = 1$ if no transition can fire in m and $r_n(m,t) = p_T^n(t) / \sum_{m \overset{t'}{\Rightarrow}} p_T^n(t')$, $r_n(m, \text{fail}) = 0$ otherwise.

Other semantics might make sense, for instance the probability of firing a transition could depend on a place not contained in its pre-condition. Furthermore, it is possible to mix the two semantics and do one step in the independent and the next in the stochastic semantics.

▶ **Example 1.** The following nets illustrate the two semantics. The first net (Fig. 1a) explains the diffusion of gossip in a social network: There are four users and each place K_i represents the knowledge of user i. To convey the fact that user i knows some secret, place K_i contains a token. The diffusion of information is represented by transitions d_j. E.g., if 1 knows the secret he will tell it to either 2 or 3 and if 3 knows a secret she will broadcast it to both 1 and 4. Note that a person will share the secret even if the recipient already knows, and she will retain this knowledge (see the double arrows in the net).[2]

[2] Hence, in the Petri net semantics, we allow a transition to fire although the post-conditions is marked.

Here we use the stochastic semantics: only transitions that are enabled will be chosen (unless the marking is empty and no transition can fire). We assume that $p_T(d_2) = 1/3$ and $p_T(d_1) = p_T(d_3) = p_T(d_4) = p_T(d_5) = 1/6$, i.e., user 2 is more talkative than the others.

One of the states of the Markov chain is the marking $m = 1100$ (K_1, K_2 are marked – users 1 and 2 know the secret – and K_3, K_4 are unmarked – users 3 and 4 do not). In this situation transitions d_1, d_2, d_3 are enabled. We normalize the probabilities and obtain that d_2 fires with probability $1/2$ and the other two with probability $1/4$. By firing d_1 or d_2 we stay in state 1100, i.e., the corresponding Markov chain has a loop with probability $3/4$. Firing d_3 gives us a transition to state 1110 (user 3 now knows the secret too) with probability $1/4$.

The second net (Fig. 1b) models the classical SIR infection model [24] for two persons. A person is *susceptible* (represented by a token in place S_i) if he or she has not yet been infected. If the other person is infected (i.e. place I_1 or I_2 is marked), then he or she might also get *infected* with the disease. Finally, people recover (or die), which means that they are *removed* (places R_i). Again we use the stochastic semantics.

The third net (Fig. 1c) models a test (for instance for an infection) that may have false positives and false negatives. A token in place I means that the corresponding person is infected. Apart from I there is another random variable R (for result) that tells whether the test is positive or negative. In order to faithfully model the test, we assign the following probabilities to the transitions: $p_T(\mathit{flp}) = P(R \mid \bar{I})$ (false or lucky positive: this transition can fire regardless of whether I is marked, in which case the test went wrong and is only accidentally positive), $p_T(\mathit{inf}) = P(R \mid I) - P(R \mid \bar{I})$ (the remaining probability,[3] such that the probabilities of *flp* and *inf* add up to the true positive) and $p_T(\text{fail}) = P(\bar{R} \mid I)$ (false negative). Here we use the independent semantics, assuming that we have a random test where the ground truth (infected or not infected) is independent of the firing probabilities of the transitions.

3 Uncertainty Reasoning for Condition/Event Nets

We now introduce the following scenario for uncertainty reasoning: assume that we are given an initial probability distribution p^0_* on the markings of the Petri net. We stipulate that the fail state $*$ cannot occur, assuming that the state of the net is always some (potentially unknown) well-defined marking. If this fail state would be reached in the Markov model, we assume that the marking of the Petri net does not change, i.e., we perform a "reset" to the previous marking.

Furthermore, we are aware of all firing probabilities of the various transitions, given by the functions $(r_n)_{n\in\mathbb{N}_0}$ and hence all transition matrices P^n that specify the transition probabilities at step n.

Then we observe the system and obtain a sequence of *success* and *failure* occurrences. We are not told which exact transition fires, but only if the firing is successful or fails (since the pre-condition of the transition is not covered by the marking). Note that according to our model, transitions *can* be chosen to fire, although they are not activated. This could happen if either a user or the environment tries to fire such a transition, unaware of the status of its pre-condition. Failure corresponds to entering state $*$ and in this case we assume the marking does not change. That is, we keep the previous marking, but acquire additional knowledge – namely that firing fails – which is used to update the probability distribution according to Prop. 4 (by performing the corresponding matrix multiplications, including normalization).

[3] Here we require that $P(R \mid \bar{I}) \leq P(R \mid I)$.

We use the following notation: let M be a matrix indexed over $\mathcal{M} \cup \{*\}$. Then we denote by M_* the matrix obtained by deleting the $*$-indexed row and column from M. Analogously for a vector p. Note that $(M \cdot p)_* = M_* \cdot p_*$. Furthermore if p_* is a sub-probability vector, indexed over $\mathcal{M}$, $\mathrm{norm}(p_*)$ stands for the corresponding normalized vector, where the m-entry is $p_*(m)/\big(\sum_{m' \in \mathcal{M}} p_*(m')\big)$.

▶ **Proposition 4.** *Let $r_n \colon \mathcal{M} \times T_f \to [0,1]$ and $p^n \colon \mathcal{M} \cup \{*\} \to [0,1]$ be given as above. Let N be a C/E net and let $(X_n)_{n \in \mathbb{N}_0}$ be the Markov chain generated from N, r_n. Then*

- $P(X_{n+1} = m' \mid X_{n+1} \neq *, X_n \neq *) = P(X_{n+1} = m' \mid X_{n+1} \neq *) = \mathrm{norm}(P^n_* \cdot p^n_*)(m')$
- $P(X_n = m \mid X_{n+1} = *, X_n \neq *) = \mathrm{norm}(F^n_* \cdot p^n_*)(m)$

where $p^n(m) = P(X_n = m)$, $p^n(*) = P(X_n = *)$ and F^n is a diagonal matrix with $F^n(\bar{m} \mid \bar{m}) := P^n(* \mid \bar{m})$, $\bar{m} \in \mathcal{M}$, and $F^n(* \mid *) := P^n(* \mid *) = 1$, all other entries are 0.

Hence, in case we observe a success we update the probability distribution to $\bar{p}_{n+1}$ by computing $P^n_* \cdot \bar{p}^n$ (and normalizing). Instead, in the case of a failure we assume that the marking stays unchanged, but by observing the failure we have gathered additional knowledge, which means that we can replace $\bar{p}_{n+1}$ by $F^n_* \cdot \bar{p}_n$ (after normalization).

P^n_* and F^n_* are typically not stochastic, but only sub-stochastic. For a (sub-)probability matrix M_* and a (sub-)probability vector p_* it is easy to see that $\mathrm{norm}(M_* \cdot p_*) = \mathrm{norm}(M_* \cdot \mathrm{norm}(p_*))$. Hence another option is to omit the normalization steps and to normalize at the very end of the sequence of observations. Normalization may be undefined (in the case of the 0-vector), which signifies that we assumed an a priori probability distribution that is inconsistent with reality.

▶ **Example 2.** We get back to Ex. 1 and discuss uncertainty reasoning. Assume that in the net in Fig. 1b person 1 is susceptible (S_1 is marked), person 2 is infected (I_2 is marked) and the i_j-transitions have a higher rate (higher probability of firing) than the r_j-transitions. Then, in the next step the probability that both are infected is higher than the probability that 1 is still susceptible and 2 has recovered.

Regarding the net in Fig. 1c we can show that in the next step, in the case of success, the probability distribution is updated in such a way that place I is marked with probability $P(I \mid R)$ and unmarked with probability $P(\bar{I} \mid R)$ ($P(I \mid \bar{R})$, $P(\bar{I} \mid \bar{R})$ in the case of failure), exactly as required. For more details see [1].

4 Modular Bayesian Networks

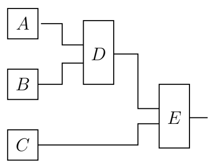


Figure 2 An example Bayesian network.

In order to implement the updates to the probability distributions described above in an efficient way, we will now represent probability distributions over markings symbolically as Bayesian networks [31, 8]. Bayesian networks (BNs) model certain probabilistic dependencies of random variables through conditional probability tables and a graphical representation.

Consider for instance the Bayesian network in Fig. 2. Each node (A, B, C, D, E) represents a binary random variable, where a node without predecessors (e.g., A) is associated with the probabilities $P(A)$ and $P(\bar{A})$. Edges denote dependencies: for instance D is dependent on A, B, which means that D is associated with a conditional probability table (matrix) with entries $P(D \mid A, B)$, similar for E (entries of the form $P(E \mid D, C)$). In both cases, the matrix contains $2 \cdot 4 = 8$ entries.

We will later describe how to derive probability distributions and marginal probabilities (for instance $P(E)$) from a Bayesian network.

We deviate from the literature on Bayesian networks in three respects: first, since we will update and transform those networks, we need a structure where we can easily express compositionality via sequential and parallel composition. To this end we use the representation of Bayesian networks via PROPs as in [16, 22]. Second, we permit sub-stochastic matrices. Third, we allow a node to have several outgoing wires, whereas in classical Bayesian networks a node is always associated to the distribution of a single random variable. This is needed since we need to add nodes to a network that represent stochastic matrices of arbitrary dimensions (basically the matrices P^n and F^n of Proposition 4). We rely on the notation introduced in [5], but extend it by taking the last item above into account.

4.1 Causality Graphs

The syntax of Bayesian networks is provided by *causality graphs* [5]. For this we fix a set of node labels G, also called *generators*, where every $g \in G$ is associated with a type $n_g \to m_g$, where $n_g, m_g \in \mathbb{N}_0$.

▶ **Definition 5** (Causality Graph (CG)). *A* causality graph (CG) *of type $n \to m$, $n, m \in \mathbb{N}_0$, is a tuple $B = (V, \ell, s, \mathrm{out})$ where*

- *V is a set of* nodes
- *$\ell \colon V \to G$ is a* labelling function *that assigns a generator $\ell(v) \in G$ to each node $v \in V$.*
- *$s \colon V \to W_B^*$ is the* source function *that maps a node to a sequence of input wires, where $|s(v)| = n_{\ell(v)}$ and $W_B = \{(v, p) \mid v \in V, p \in \{1, \dots, m_{\ell(v)}\}\} \cup \{i_1, \dots, i_n\}$ is the* wire set.
- *$\mathrm{out} \colon \{o_1, \dots, o_m\} \to W_B$ is the* output function *that assigns each output port to a wire.*

Moreover, the corresponding directed graph (defined by s) has to be acyclic.

We also define the target function *$t \colon V \to W_B^*$ with $t(v) = (v, 1) \dots (v, m_{\ell(v)})$ and the set of* internal wires *$IW_B = W_B \backslash \{i_1, \dots, i_n, \mathrm{out}(o_1), \dots, \mathrm{out}(o_m)\}$.*

We visualize such causality graphs by drawing the n input wires on the left and the m outputs on the right. Each node v is drawn as a box, with n_v ingoing wires and m_v outgoing wires, ordered from top to bottom. Connections induced by the source and by the output function are drawn as undirected edges (see Fig. 2).

We define two operations on causality graphs: sequential composition and tensor. Given B of type $n \to k$ and B' of type $k \to m$, the sequential composition is obtained via concatenation, by identifying the output wires of B with the input wires of B', resulting in $B; B'$ of type $n \to m$. The tensor takes two causality graphs B_i of type $n_i \to m_i$, $i \in \{1, 2\}$ and takes their disjoint union, concatenating the sequences of input and output wires, resulting in $B_1 \otimes B_2$ of type $n_1 + n_2 \to m_1 + m_2$. For formal definitions see [5, 4].

4.2 (Sub-)Stochastic Matrices

The semantics of modular Bayesian networks is given by *(sub-)stochastic matrices*, i.e., matrices with entries from $[0, 1]$, where column sums will be at most 1. If the sum equals exactly 1 we obtain stochastic matrices.

We consider only matrices whose dimensions are a power of two. Analogously to causality graphs, we type matrices, and say that a matrix has type $n \to m$ whenever it is of dimension $2^m \times 2^n$. We again use a sequential composition operator ; that corresponds to *matrix* *multiplication* $(P;Q = Q \cdot P)$ and the *Kronecker product* $\otimes$ as the tensor. More concretely, given $P\colon n_1 \to m_1$, $Q\colon n_2 \to m_2$ we define $P \otimes Q\colon n_1 + n_2 \to m_1 + m_2$ as $(P \otimes Q)(\mathbf{x}_1\mathbf{x}_2 \mid \mathbf{y}_1\mathbf{y}_2) = P(\mathbf{x}_1 \mid \mathbf{y}_1) \cdot Q(\mathbf{x}_2 \mid \mathbf{y}_2)$ where $\mathbf{x}_i \in \{0,1\}^{m_i}$, $\mathbf{y}_i \in \{0,1\}^{n_i}$.

4.3 Modular Bayesian Networks

Finally, *modular Bayesian networks*, adapted from [5], are causality graphs, where each generator $g \in G$ is associated with a (sub-)stochastic matrix of suitable type.

▶ **Definition 6** (Modular Bayesian network (MBN)). *An MBN is a tuple (B, ev) where B is a causality graph and ev an* evaluation function *that assigns to every generator $g \in G$ of type $n \to m$ a $2^m \times 2^n$ sub-stochastic matrix $ev(g)$. An MBN (B, ev) is called an* ordinary Bayesian network (OBN) *whenever B has no inputs (i.e. it has type $0 \to m$), each generator is of type $n \to 1$,* out *is a bijection and every node is associated with a stochastic matrix.*

We now describe how to evaluate an MBN to obtain a (sub-)stochastic matrix. For OBNs – which are exactly the Bayesian networks considered in [17] – this coincides with the standard interpretation and yields a probability vector of dimension m.

▶ **Definition 7** (MBN evaluation). *Let (B, ev) be an MBN where B is of type $n \to m$. Then $M_{ev}(B)$ is a $2^m \times 2^n$-matrix, which is defined as follows:*

$$M_{ev}(B)(\mathrm{x}_1 \ldots \mathrm{x}_m \mid \mathrm{y}_1 \ldots \mathrm{y}_n) = \sum_{b \in \mathcal{B}} \prod_{v \in V} ev(l(v))\,(b(t(v)) \mid b(s(v)))$$

with $\mathrm{x}_1, \ldots, \mathrm{x}_m, \mathrm{y}_1, \ldots, \mathrm{y}_n \in \{0,1\}$. $\mathcal{B}$ is the set of all functions $b : W_B \to \{0,1\}$ such that $b(i_j) = \mathrm{y}_j$, $b(out(o_k)) = \mathrm{x}_k$, where $k \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$. The functions b are applied pointwise to sequences of wires.

Calculating the underlying probability distribution of an MBN can also be done on a graphical level by treating every occurring wire as a boolean variable that can be assigned either 0 or 1. Function $b \in \mathcal{B}$ assigns the wires, ensuring consistency with the input/output values. After the wire assignment, the corresponding entries of each matrix $ev(l(v))$ are multiplied. After iterating over every possible wire assignment, the products are summed up.

Note that M_{ev} is compositional, it preserves sequential composition and tensor. More formally, it is a functor between symmetric monoidal categories, or – more specifically – between CC-structured PROPs (More details on PROPs are given in the full version [1].).

▶ **Example 3.** We illustrate Def. 7 by evaluating the Bayesian network (B', ev) in Fig. 2. This results in a 2×1-matrix $M_{ev}(B')$, assigning (sub-)probabilities to the only output wire in the diagram being 1 or 0, respectively. More concretely, we assign values to the four inner wires to obtain:

$$M_{ev}(B')(\mathrm{e}) = \sum_{\mathrm{a} \in \{0,1\}} \sum_{\mathrm{b} \in \{0,1\}} \sum_{\mathrm{c} \in \{0,1\}} \sum_{\mathrm{d} \in \{0,1\}} \big(A(\mathrm{a}) \cdot B(\mathrm{b}) \cdot C(\mathrm{c}) \cdot D(\mathrm{d} \mid \mathrm{ab}) \cdot E(\mathrm{e} \mid \mathrm{cd})\big),$$

where $\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}$ correspond to the output wire of the corresponding matrix (A, B, C, D, E).

5 Updating Bayesian Networks

An MBN B of type $0 \to k$, as defined above, symbolically represents a probability distribution on $\{0,1\}^k$, that is, a probability distribution on markings of a net with $|S| = k$ places.

Under uncertainty reasoning (cf. Section 3), the probability distribution in the next step p^{n+1} is obtained by multiplying p^n with a matrix M (either P_*^n in the successful case or F_*^n in the case of failure). Hence, a simple way to update B would be to create an MBN B_M with a single node v (labelled by a generator g with $ev(g) = M$), connected to k inputs and k outputs. Then the updated B' is simply $B; B_M$ (remember that sequential composition corresponds to matrix multiplication). However, at dimension $2^k \times 2^k$ the matrix M is huge and we would sacrifice the desirable compact symbolic representation. Hence the aim is to decompose $M = M' \otimes \mathrm{Id}$ where Id is an identity matrix of suitable dimension. Due to the functoriality of MBN evaluation this means composing with a smaller matrix and a number of identity wires (see e.g. Fig. 3b).

This decomposition arises naturally from the structure of the Petri net N, in particular if there are only relatively few transitions that may fire in a step. In this case we intuitively have to attach a stochastic matrix only to the wires representing the places connected to those transitions, while the other wires can be left unchanged. If there are several updates, we of course have to attach several matrices, but each of them might be of a relatively modest size.

In order to have a uniform treatment of the various semantics, we assume that for each step n there is a set $\bar{S} \subseteq S$ of places[4] and a set $\bar{T} \subseteq T_f$ of transitions such that: (i) $r_n(m, t) = 0$ whenever $t \not\in \bar{T}$; (ii) $r_n(m_1 m_2, t) = \bar{r}(m_1, t)$ for some function $\bar{r}$ (where m_1 is a marking of length $\ell = |\bar{S}|$, corresponding to the places of $\bar{S}$); (iii) $\bar{S}$ contains at least ${}^\bullet t, t^\bullet$ for all $t \in \bar{T}$. Intuitively, $\bar{S}$, $\bar{T}$ specify the relevant places and transitions.

For the two Petri net semantics studied earlier, these conditions are satisfied if we take as $\bar{T}$ the support of p_T^n and as $\bar{S}$ the union of all pre- and post-sets of $\bar{T}$. The function r_n can in both cases be defined in terms of $\bar{r}$: in the independent case this is obvious, whereas in the stochastic net case we observe that $r_n(m, t)$ is only dependent on p_T^n and on the set of transitions that is enabled in m and this can be derived from m_1.

Now, under these assumptions, we can prove that we obtain the decomposition mentioned above.

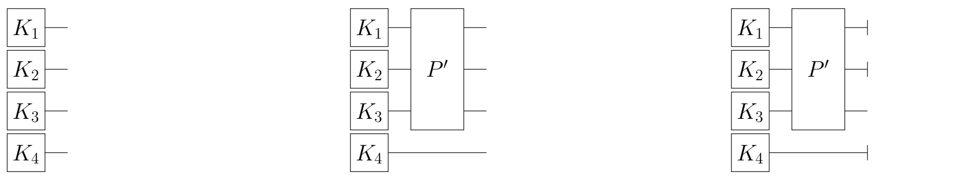
▶ **Proposition 8.** *Assume that N is a condition/even-net together with a function r_n. Assume that we have $\bar{S} \subseteq S$, $\bar{T} \subseteq T_f$ satisfying the conditions above. Then*
- $P_*^n = P' \otimes \mathrm{Id}_{2^{k-\ell}}$ *where* $P'(m_1' \mid m_1) = \sum_{t \in \bar{T}, m_1 \overset{t}{\Rightarrow} m_1'} \bar{r}(m_1, t)$.
- $F_*^n = F' \otimes \mathrm{Id}_{2^{k-\ell}}$ *where* $F'(m_1' \mid m_1) = \sum_{t \in \bar{T}, m_1 \overset{t}{\not\Rightarrow}} \bar{r}(m_1, t)$ *if* $m_1 = m_1'$ *and* 0 *otherwise.*

Here P', F' are $2^\ell \times 2^\ell$-matrices and $m_1, m_1' \subseteq \bar{S}$. Note also that we implicitly restricted the firing relation to the markings on $\bar{S}$.

▶ **Example 4.** In order to illustrate this, we go back to gossip diffusion (Fig. 1a, Ex. 1). Our input is the following: an initial probability distribution, describing the a priori knowledge, given by an MBN. Here we have no information about who knows or does not know the secret and hence we assume a uniform probability distribution over all markings. This is represented by the Bayesian network in Fig. 3a where each node is associated with a 2×1-matrix (vector) K_i where both entries are $1/2$.

Also part of the input is the family of transition distributions $(r_n)_{n \in \mathbb{N}_0}$. Here we assume that the firing probabilities of transitions are as in Example 1, but not all users are active at the same time. We have information that in the first step only users 1 and 2 are active, hence by normalization we obtain probabilities $1/4$, $1/2$, $1/4$ for transitions d_1, d_2, d_3 (the other transitions are deactivated).

[4] Without loss of generality we assume that the outputs have been permuted such that places in $\bar{S}$ occur first in the sequence of places.



(a) An MBN modelling a uniform probability distribution.

(b) An MBN after performing an update (observation of a successful step).

(c) Computing a marginal probability distribution from an MBN.

Figure 3 Example: transformation of modular Bayesian networks.

Now we observe a success step. According to Sct. 3 we can make an update with P_* where P is the transition matrix of the Markov chain. Since none of the transitions is attached to place K_4 the optimizations of this section allow us to represent P_* as $P' \otimes \mathrm{Id}_2$ where P' is an 8×8-matrix. E.g., as discussed in Ex. 1, we have $P'(110 \mid 110) = 3/4$, $P'(111 \mid 110) = 1/4$. This matrix is simply attached to the modular Bayesian network (see Fig. 3b).

Now assume that it is our task to compute the probability that place K_3 is marked. For this, we compute the corresponding marginal probabilities by terminating each output wire (apart from the third one) (see Fig. 3c). "Terminating a wire" means to remove it from the output wires. This results in summing up over all possible values assigned to each wire, where we can completely omit the last component, which is the unit of the Kronecker product. Note that the resulting vector is sub-stochastic and still has to be normalized. The normalization factor can be obtained by terminating also the remaining third wire, which gives us the probability mass of the sub-probability distribution. Our implementation will now tell us that place K_3 is marked with probability $5/8$.

6 Variable Elimination and Tree Decompositions

6.1 Motivation

Given a modular Bayesian network, it is inefficient to obtain the full distribution, not just from the point of view of the computation, but also since its direct representation is of exponential size. However what we often need is to compute a marginal distribution (e.g., the probability that a certain place is marked) or a normalization factor for a sub-stochastic probability distribution (cf. Ex. 4). Another application would be to transform an MBN into an OBN, by isolating that part of the network that does not conform to the properties of an OBN, evaluating it and replacing it by an equivalent OBN.

Def. 7 gives a recipe for the evaluation, which is however quite inefficient. Hence we will now explain and adapt the well-known concept of variable elimination [14, 13]. Let us study the problem with a concrete example. Consider the Bayesian network B' in Fig. 2 and its evaluation described in Ex. 3. If we perform this computation one has to enumerate $2^4 = 16$ bit vectors of length 4. Furthermore, after eliminating d we have to represent a matrix (also called *factor* in the literature on Bayesian networks) that is dependent on four random variables $(\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{e})$, hence we say that it has width 4 ($2^4 = 16$ entries).

However, it is not difficult to see that we can – via the distributive law – reorder the products and sums to obtain a more efficient way of computing the values:

$$M_{ev}(B')(\mathrm{e}) = \sum_{\mathrm{d}\in\{0,1\}} \Big(\sum_{\mathrm{c}\in\{0,1\}} \Big(\sum_{\mathrm{b}\in\{0,1\}} \Big(\sum_{\mathrm{a}\in\{0,1\}} \big(A(\mathrm{a}) \cdot D(\mathrm{d} \mid \mathrm{ab})\big) \cdot B(\mathrm{b})\Big) \cdot C(\mathrm{c})\Big) \cdot E(\mathrm{e} \mid \mathrm{cd})\Big).$$

In this way we obtain smaller matrices, the largest matrix (or factor) that occurs is D (width 3). Choosing a different elimination order might have been worse. For instance, if we had eliminated d first, we would have to deal with a matrix dependent on $\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{e}$ (width 4).

6.2 Variable elimination

The literature of Bayesian networks [14, 13] extensively studies the best variable elimination order and discusses the relation to treewidth. For our setting we have to extend the results in the literature, since we also allow generators with more than one output.

▶ **Definition 9** (Elimination order). *Let $B = (V, \ell, s, \mathrm{out})$ be the causality graph of a modular Bayesian network of type $n \to m$. As in Def. 5 let W_B be the set of wires.*

We define an undirected graph U_0 that has as vertices[5] *the wires W_B and two wires w_1, w_2 are connected by an edge whenever they are connected to the same node. More precisely, they are connected whenever they are input or output wires for the same node (i.e. w_1, w_2 are both in $s(v)t(v)$ for a node $v \in V$).*

Now let $w_1, \dots, w_k$ (where $k = |IW_B|$) be an ordering of the internal wires, a so-called elimination ordering*. We update the graph U_{i-1} to U_i by removing the next wire w_i and connecting all of its neighbours by edges (so-called* fill in*). External wires are never eliminated. The* width *of the elimination ordering is the size of the largest clique that occurs in some graph U_i. The* elimination width *of B is the least width taken over all orderings.*

In the case of Bayesian networks, the set of wires of an OBN corresponds to the set of random variables. In the literature, the graph U_0 is called the moralisation of the Bayesian network, it is obtained by taking the Bayesian network (an acyclic graph), forgetting about the direction of the edges, and connecting all the parents (i.e., the predecessors) of a random variable, i.e. making them form a clique. This results in the same graph as the construction described above.

To introduce the algorithm, we need the notion of a *factor*, already hinted at earlier.

▶ **Definition 10** (Factor). *Let (B, ev) be a modular Bayesian network with a set of wires W_B. A* factor *$(f, \tilde{w})$ of size s consists of a map $f \colon \{0,1\}^s \to [0,1]$ together with a sequence of wires $\tilde{w} \in W_B^*$. We require that $\tilde{w}$ is of length s ($|\tilde{w}| = s$) and does not contain duplicates.*

Given a wire $w \in W_B$ and a multiset $\mathcal{F}$ of factors, we denote by $C_w(\mathcal{F})$ all those factors $(f, \tilde{w}) \in \mathcal{F}$ where $\tilde{w}$ contains w. By $X_w(\mathcal{F})$ we denote the set of all wires that occur in the factors in $C_w(\mathcal{F})$, apart from w.

We now consider an algorithm that computes the probability distribution represented by a modular Bayesian network of type $n \to m$. We assume that an evaluation map ev, mapping generators to their corresponding matrices, and an elimination order $w_1, \dots, w_k$ of internal wires is given. Furthermore, given a sequence of wires $\tilde{w} = w'_1 \dots w'_s$ and a bitstring $\mathbf{x} = \mathrm{x}_1 \dots \mathrm{x}_s$, we define the substitution function $b_{\tilde{w},\mathbf{x}}$ from wires to bits as $b_{\tilde{w},\mathbf{x}}(w'_j) = \mathrm{x}_j$.

▶ **Algorithm 11** (Variable elimination).
Input: *An MBN (B, ev) of type $n \to m$*

- *Let $\mathcal{F}_0$ be the initial multiset of factors. For each node v of type $n_{\ell(v)} \to m_{\ell(v)}$, it contains the matrix $ev(v)$, represented as a factor f, together with the sequence $s(v)t(v)$. That is $f(\mathbf{xy}) = ev(v)(\mathbf{y} \mid \mathbf{x})$ where $\mathbf{x} \in \{0,1\}^{n_{\ell(v)}}$, $\mathbf{y} \in \{0,1\}^{m_{\ell(v)}}$.*

[5] We talk about the *nodes* of an MBN B and the *vertices* of an undirected graph U_i.

- *Now assume that we have a set $\mathcal{F}_{i-1}$ of factors and take the next wire w_i in the elimination order. We choose all those factors that contain w_i and compute a new factor $(f, \tilde{w})$. Let $\tilde{w}$ be a sequence that contains all wires of $X_w(\mathcal{F}_{i-1})$ (in arbitrary order, but without duplicates). Let $s = |\tilde{w}|$. Then f is a function of type $f\colon \{0,1\}^s \to [0,1]$, defined as:*

$$f(\mathbf{y}) = \sum_{\mathbf{z}\in\{0,1\}} \prod_{(g,\tilde{w}^g)\in C_{w_i}(\mathcal{F}_{i-1})} g(b_{\tilde{w}w_i,\mathbf{yz}}(\tilde{w}^g)).$$

We set $\mathcal{F}_i = \mathcal{F}_{i-1}\backslash C_{w_i}(\mathcal{F}_{i-1}) \cup \{(f,\tilde{w})\}$.
- *After the elimination of all wires we obtain a multiset of factors $\mathcal{F}_k$, whose sequences contain only input and output wires. The resulting probability distribution is $p\colon \{0,1\}^{n+m} \to [0,1]$, where $\mathbf{x} \in \{0,1\}^n$, $\mathbf{y} \in \{0,1\}^m$, $\tilde{\imath} = i_1 \dots i_n$, $\tilde{o} = \mathrm{out}(o_1)\dots\mathrm{out}(o_m)$:*

$$p(\mathbf{xy}) = \prod_{(f,\tilde{w}^f)\in\mathcal{F}_k} f(b_{\tilde{\imath}\tilde{o},\mathbf{xy}}(\tilde{w}^f))$$

That is, given the next wire w_i we choose all factors that contain this wire, remove them from $\mathcal{F}_{i-1}$ and multiply them, while eliminating the wire. The next set is obtained by adding the new factor. Finally, we have factors that contain only input and output wires and we obtain the final probability distribution by multiplying them.

▶ **Proposition 12.** *Given a modular Bayesian network (B, ev) where B is of type $n \to m$, Algorithm 11 computes its corresponding (sub-)stochastic matrix $M_{ev}(B)$, that is*

$$M_{ev}(B)(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{xy}) \qquad \text{for } \mathbf{x} \in \{0,1\}^n,\ \mathbf{y} \in \{0,1\}^m.$$

Furthermore, the size of the largest factor in any multiset $\mathcal{F}_i$ is bounded by the width of the elimination ordering.

6.3 Comparison to Treewidth

We conclude this section by investigating the relation between elimination width and the well-known notion of treewidth [2].

▶ **Definition 13** (Treewidth of a causality graph). *Let $B = (V, \ell, s, \mathrm{out})$ be a causality graph of type $n \to m$. A* tree decomposition *for B is an undirected tree $T = (V_T, E_T)$ such that*
- *every node $t \in V_T$ is associated with a bag $X_t \subseteq W_B$,*
- *every wire $w \in W_B$ in contained in at least one bag X_t,*
- *for every node $v \in V$ there exists a bag X_t such that all input and output wires of v are contained in X_t (i.e., all wires in $s(v)$ and $t(v)$ are in X_t)* and
- *for every wire $w \in W_B$, the tree nodes $\{t \in V_T \mid w \in X_t\}$ form a subtree of T.*

The width of a tree decomposition is given by $\max_{t\in V_T} |X_t| - 1$.

The treewidth of B is the minimal width, taken over all tree decompositions.

Note that the treewidth of a causality graph corresponds to the treewidth of the graph U_0 from Def. 9. Now we are ready to compare elimination width and treewidth.

▶ **Proposition 14.** *Elimination width is always an upper bound for treewidth and they coincide when B is a causality graph of type $0 \to 0$. For a network of type $0 \to m$ the treewidth may be strictly smaller than the elimination width.*

The treewidth might be strictly smaller since we are now allowed to eliminate output wires. However, it is easy to see that the treewidth plus the number of output wires always provides an upper bound for the elimination width.

The paper [2] also discusses heuristics for computing good elimination orderings, an opimization problem that is NP-hard. Hence the treewidth of a causality graph gives us an upper bound for the most costly step in computing its corresponding probability distribution. [26] shows that a small treewidth is actually a necessary condition for obtaining efficient inference algorithms.

We can also compare elimination width to the related notion of term width, more details can be found in [1].

7 Implementation and Runtime Results

We extended the implementation presented in the predecessor paper [5] by incorporating probabilistic Petri nets and elimination orderings, in order to evaluate the performance of the proposed concepts. The implementation is open source and freely available from GitHub.[6]

Runtime results were obtained by randomly generating Petri nets with different parameters, e.g. number of places, transitions and tokens, initial marking. The maximal number of places in pre- and post-conditions is restricted to three and at most five transitions are enabled in each step. With these parameters, the worst case scenario is the creation of a matrix of type $30 \to 30$. After the initialization of a Petri net, which can be interpreted with either semantics (independent/stochastic), transitions and their probabilities are picked at random. Then we observe either success or failure and update the probability distribution accordingly.

We select the elimination order via a heuristics by preferring wires with minimal degree in the graph U_i (cf. Def 9). Furthermore we apply a few optimizations: Nodes with no output wires will be evaluated first, nodes without inputs second. The observation of a failure will generate a diagonal matrix, which enables an optimized evaluation, as its input and output wires have to carry the same value (otherwise we obtain a factor 0). In addition, we use optimizations whenever we have definitive knowledge about the marking of a particular place (of a pre-condition), by drawing conclusions about the ability to fire certain transitions.

The plot on the left of Fig. 4 compares runtimes when incorporating ten success/failure observations directly on the joint distribution (i.e. the naive representation of a probability distribution) versus our MBN implementation. We initially assume a uniform distribution of tokens and calculate the probability that the first place is marked after the observations. Both approaches evaluate the same Petri net and therefore calculate the same results. The data is for the independent semantics, but it is very similar for the stochastic semantics.

While the runtime increases exponentially when using joint distributions, our MBN implementation stays relatively constant (see Fig. 4, left). Due to memory issues, handling Petri nets with more than 30 places is not anymore feasible for the direct computation of joint distributions. We use the median for comparison (see Fig. 4, left), but if an MBN consists of very large matrices, the evaluation time will be rather high. The right plot of Fig. 4 shows this correlation, where colours denote the runtime and the y-axis represents the number of wires attached to the largest matrix. (Here we actually count equivalence classes by grouping those wires that have to carry the same value, due to their attachment to a diagonal matrix, see also the optimization explained above.)

[6] `https://github.com/RebeccaBe/Bayesian-II`

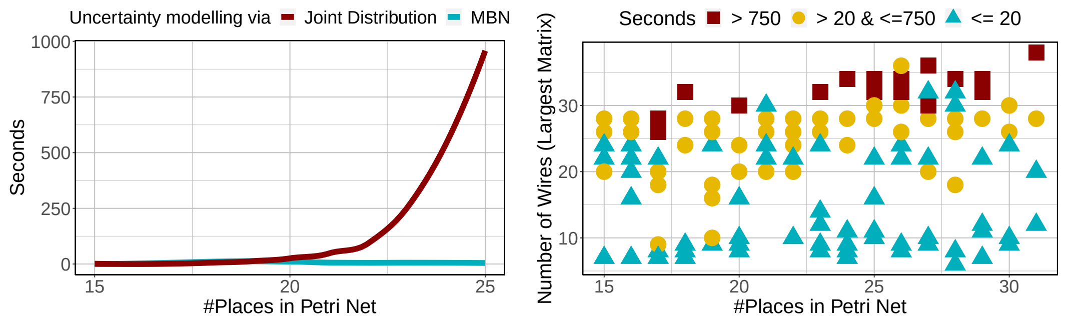


Figure 4 Left: Median of runtimes performing after 10 transitions on a Petri net. Right: Effect of large matrices on the runtimes of the MBN implementation.

The advantage of our approach decreases when we have substantially more places in the pre- and post-set, more transitions that may fire and a larger number of steps, since then the Bayesian network is more densely connected and contains larger matrices. Furthermore, one might generally expect the state (containing tokens or not) of places of the Petri net to become more and more coupled over time, as more transitions have fired, decreasing the performance improvement we gain from using MBNs. However, recall that the transitions that can fire at any time are explicitly controlled by the input p_T^n. This allows our model to capture situations where different parts of the network stay uncoupled over time and where using MBNs is an advantage. Furthermore the observation of a failure allows an optimizated variable elimination, as explained above.

8 Conclusion

We propose a framework for uncertainty reasoning for probabilistic Petri nets that represents probability distributions compactly via Bayesian networks. In particular we describe how to efficiently update and evaluate Bayesian networks.
Related work: Naturally, uncertainty reasoning has been considered in many different scenarios (for an overview see [19]). Here we review only those approaches that are closest to our work.

In [5] we studied a simpler scenario for nets whose transitions do not fire probabilistically, but are picked by the observer, resulting in a restricted set of update operations. Rather than computing marginal distributions directly via variable elimination as in this paper, our aim there was to transform the resulting modular Bayesian network into an ordinary one. Since the updates to the net were of a simpler nature, we were able to perform this conversion. Here we are dealing with more complex updates where this can not be done efficiently. Instead we are concentrating on extracting information, such as marginal distributions, from a Bayesian network.

Furthermore, uncertainty reasoning as described in Sct. 3 is related to the methods used for hidden Markov models [32], where the observations refer to the states, whereas we (partially) observe the transitions.

There are several proposals which enrich Petri nets with a notion of uncertainty: possibilistic Petri nets [27], plausible Petri nets [9] that combine discrete and continuous processes or fuzzy Petri nets [6, 34] where firing of transitions is governed by the truth values of statements. Uncertainty in connection with Petri nets is also treated in [25, 23], but without introducing a formal model. As far as we know neither approach considers symbolic representation of probability distributions via Bayesian networks.

In [3] the authors exploit the fact that Petri nets also have a monoidal structure and describe how to convert an occurrence (Petri) net with a truly concurrent semantics into a Bayesian network, allowing to derive probabilistic information, for instance on whether a place will eventually be marked. This is different from our task, but it will be interesting to compare further by unfolding our nets and equipping them with a truly concurrent semantics, based on the probabilistic information from the time-inhomogeneous Markov chain.

We instead propose to use Bayesian networks as symbolic representations of probability distributions. An alternative would be to employ multi-valued (or multi-terminal) binary decision diagrams (BDDs) as in [20]. An exact comparison of both methods is left for future work. We believe that multi-valued BDDs will fare better if there are only few different numerical values in the distribution, otherwise Bayesian networks should have an advantage.

As mentioned earlier, representing Bayesian networks by PROPs or string diagrams is a well-known concept, see for instance [16, 22]. The paper [21] describes another transformation of Bayesian networks by string diagram surgery that models the effect of an intervention.

In addition there is a notion of dynamic Bayesian networks [30], where a random variable has a separate instance for each time slice. We instead keep only one instance of every random variable, but update the Bayesian network itself.

In addition to variable elimination, a popular method to compute marginals of a probability distribution is based on belief propagation and junction trees [28]. In order to assess the potential efficiency gain, this approach has to be adapted for modular Bayesian networks. However due to the dense interconnection and large matrices of MBNs, an improvement in runtime is unclear and deserves future investigation.

Future work: One interesting avenue of future work is to enrich our model with timing information by considering continuous-time Markov chains [35], where firing delays are sampled from an exponential distribution. Instead of asking about the probability distribution after n steps we could instead ask about the probability distribution at time t.

We would also like to add mechanisms for controlling the system, such as transitions that are under the control of the observer and can be fired whenever enabled. Then the task of the observer would be to control the system and guide it into a desirable state. In this vein we are also interested in studying stochastic games [11] with uncertainty.

The interaction between the structure of the Petri net and the efficiency of the analysis method also deserves further study. For instance, are free-choice nets [15] – with restricted conflicts of transitions – more amenable to this type of analysis than arbitrary nets?

Recently there has been a lot of interest in modelling compositional systems via string diagrams, in the categorical setting of symmetric monoidal categories or PROPs [10]. In this context it would be interesting to see how the established notion of treewidth [2] and its algebraic characterizations [12] translates into a notion of width for string diagrams. We started to study this for the notion of term width, but we are not aware of other approaches, apart from [7] which considers monoidal width.

References

1 Rebecca Bernemann, Benjamin Cabrera, Reiko Heckel, and Barbara König. Uncertainty reasoning for probabilistic petri nets via Bayesian networks, 2020. arXiv:2009.14817. URL: `https://arxiv.org/abs/2009.14817`.

2 H.L. Bodlaender and A.M.C.A. Koster. Treewidth computations I. Upper bounds. Technical Report UU-CS-2008-032, Department of Information and Computing Sciences, Utrecht University, September 2008.

3 R. Bruni, H. C. Melgratti, and U. Montanari. Bayesian network semantics for Petri nets. *Theoretical Computer Science*, 807:95–113, 2020.

4 B. Cabrera. *Analyzing and Modeling Complex Networks – Patterns, Paths and Probabilities*. PhD thesis, Universität Duisburg-Essen, 2019.

5 B. Cabrera, T. Heindel, R. Heckel, and B. 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. URL: `http://drops.dagstuhl.de/opus/frontdoor.php?source_opus=9565`.

6 J. Cardoso, R. Valette, and D. Dubois. Fuzzy Petri nets: An overview. In *Proc. of 13th Triennal World Congress*, 1996.

7 A. Chantawibul and P. Sobociński. Towards compositional graph theory. In *Proc. of MFPS XXXI*. Elsevier, 2015. ENTCS 319.

8 E. Charniak. Bayesian networks without tears. *AI magazine*, 12(4):50–50, 1991.

9 M. Chiachio, J. Chiachio, D. Prescott, and J.D. Andrews. A new paradigm for uncertain knowledge representation by plausible Petri nets. *Information Sciences*, 453:323–345, 2018.

10 B. Coecke and A. Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

11 A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

12 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic, A Language-Theoretic Approach*. Cambridge University Press, June 2012.

13 A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2011.

14 R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

15 J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

16 B. Fong. Causal theories: A categorical perspective on Bayesian networks. Master's thesis, University of Oxford, 2012. arXiv:1301.6201.

17 N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

18 C.M. Grinstead and J.L. Snell. *Introduction to probability*. American Mathematical Soc., 2012.

19 J.Y. Halpern. *Reasoning about Uncertainty*. MIT Press, second edition edition, 2017.

20 H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi-terminal binary decision diagrams to represent and analyse continuous-time markov chains. In *Proc. of NSMC '99 (International Workshop on the Numerical Solution of Markov Chains)*, pages 188–207, 1999.

21 B. Jacobs, A. Kissinger, and F. Zanasi. Causal inference by string diagram surgery. In *Proc. of FOSSACS '19*, pages 313–329. Springer, 2019. LNCS 11425.

22 B. Jacobs and F. Zanasi. A formal semantics of influence in Bayesian reasoning. In *Proc. of MFCS*, volume 83 of *LIPIcs*, pages 21:1–21:14, 2017.

23 I. Jarkass and M. Rombaut. Dealing with uncertainty on the initial state of a Petri net. In *Proc. of UAI '98 (Uncertainty in Artificial Intelligence)*, pages 289–295, 1998.

24 M.J. Keeling and K.T.D. Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

25 M. Kuchárik and Z. Balogh. Modeling of uncertainty with Petri nets. In *Proc. of ACIIDS '19 (Asian Conference on Intelligent Information and Database Systems)*, pages 499–509. Springer, 2019. LNAI 11431.

26 J.H.P. Kwisthout, H.L. Bodlaender, and L.C. Van Der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *Proc. of ECAI '10 (European Conference on Artificial Intelligence)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 237–242. IOS Press, 2010.

27 J. Lee, K.F.R. Liu, and W. Chiang. Modeling uncertainty reasoning with possibilistic Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(2):214–224, 2003.

28 V. Lepar and P.P. Shenoy. A comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer architectures for computing marginals of probability distributions. In G.F. Cooper and S. Moral, editors, *Proc. of UAI '98 (Uncertainty in Artificial Intelligence)*, pages 328–337, 1998. URL: `http://arxiv.org/abs/1301.7394`.

29 M. Ajmone Marsan. Stochastic Petri nets: an elementary introduction. In *Proc. of the European Workshop on Applications and Theory in Petri Nets*, volume 424 of *Lecture Notes in Computer Science*, pages 1–29. Springer, 1990.

30 K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, Computer Science Division, 2002.

31 J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proc. of the 7th Conference of the Cognitive Science Society*, pages 329–334, 1985. UCLA Technical Report CSD-850017.

32 L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

33 W. Reisig. *Petri Nets: An Introduction.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1985.

34 Z. Suraj. Generalised fuzzy Petri nets for approximate reasoning in decision support systems. In *Proc. of CS&P '12 (International Workshop on Concurrency, Specification and Programming)*, volume 928 of *CEUR Workshop Proceedings*, pages 370–381. CEUR-WS.org, 2012.

35 A. Tolver. An introduction to Markov chains. Department of Mathematical Sciences, University of Copenhagen, November 2016.

36 W. Wiegerinck, W. Burgers, and B. Kappen. Bayesian networks, introduction and practical applications. In *Handbook on Neural Information Processing*, pages 401–431. Springer, 2013.