Interventions and Counterfactuals in Tractable Probabilistic Models

Ioannis Papantonis

University of Edinburgh
i.papantonis@sms.ed.ac.uk

Vaishak Belle

University of Edinburgh &
Alan Turing Institute
vaishak@ed.ac.uk

Abstract

In recent years, there has been an increasing interest in studying causality-related properties in machine learning models generally, and in generative models in particular. While that is well motivated, it inherits the fundamental computational hardness of probabilistic inference, making exact reasoning intractable. Probabilistic tractable models have also recently emerged, which guarantee that conditional marginals can be computed in time linear in the size of the model, where the model is usually learned from data. Although initially limited to low tree-width models, recent tractable models such as sum product networks (SPNs) and probabilistic sentential decision diagrams (PSDDs) exploit efficient function representations and also capture high tree-width models.

In this paper, we ask the following technical question: can we use the distributions represented or learned by these models to perform causal queries, such as reasoning about interventions and counterfactuals? We answer in the negative. We show that when transforming SPNs to a causal graph interventional reasoning reduces to computing marginal distributions; in other words, only trivial causal reasoning is possible. For PSDDs the situation is only slightly better. We first provide an algorithm for constructing a causal graph from a PSDD, which introduces augmented variables. Intervening on the original variables, once again, reduces to marginal distributions, but when intervening on the augmented variables, a deterministic but nonetheless "causal-semantics" can be provided for PSDDs.

1 Introduction

In recent years, there has been an increasing interest in studying causality-related properties in machine learning models. For example, [16] argue for the ability to assess past observations and explain away alternative causes in deep reinforcement learning methods. In [4], the question of what units are responsible for controlling and manipulating certain features within an image is considered. In [5], strategies to give a causal interpretation to the intrinsic structure of deep learning models is investigated. Broadly speaking [24], the motivation stems from extending the query and reasoning capabilities over probabilistic domains. That is, in standard probabilistic models, one is simply interested in *conditioning on observations* $\Pr(y \mid x)$: e.g., what is the likelihood of lung inflammations given that the patient smokes? Causal reasoning allows us to reason about *interventions* $\Pr(y \mid do(x))$: e.g., how are lung inflammations affected when the patient reduces the amount of

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

tobacco smoked in a day? *Counterfactual queries* allow us to directly reason about alternate worlds $\Pr(y \mid x^*)$: e.g., what state would the patient's lung inflammations be in had he not smoked in the previous year? Thus, causal reasoning allows us to inspect our domain model much more comprehensively than possible by observational conditioning alone.

A fundamental challenge underlying stochastic models, however, is the intractability of inference [2]. This has led to the paradigm of *tractable probabilistic models*, where conditional or marginal distributions can be computed in time linear in the size of the model. Although initially limited to low tree-width models [3], recent tractable models such as sum product networks (SPNs) [28, 13] and probabilistic sentential decision diagrams (PSDDs) [17] are derived from arithmetic circuits (ACs) and knowledge compilation approaches, more generally [10, 6], which exploit efficient function representations and also capture high tree-width models. These models can also be learnt from data, and owing to the efficiency of inference, learning too can be made efficient or given a closed-form expression [25, 17]. Consider that in classical structure learning approaches for graphical models, once learned, inference would have to be approximated, owing to its intractability. In that regard, such models offer a robust and tractable framework for learning and inferring from data.

Naturally, owing to such attractive properties, the theoretical underpinnings of such models have received considerable attention. One the one hand, when viewed from a knowledge representation angle, they are related to tractable representations of Boolean functions, including BDDs [1] and d-DNNFs [9, 17]. On the other hand, from a probabilistic modeling perspective, they can be derived as instances of ACs, providing a tractable representation of probabilistic reasoning, owing to the fact that ACs can compactly represent and compute the network polynomial of a Bayesian network (BN) [8]. In the presence of latent variables they can also be seen as a deep architecture with probabilistic semantics [28], leading to numerous extensions, e.g., for mixed discrete-continuous domains [21], and applications, including preference ranking [7], classification [20] and computer vision [28, 29]. Owing to its clear probabilistic semantics, in [12], the expressive power of such deep models is studied, and in [30], the relationship between SPNs and BNs has been further analyzed.

In this paper we push the envelope on this inquiry towards the following objective: can tractable models offer not only a computationally attractive but also compelling alternative to arbitrary graphical models, especially when reasoning about causality? We answer in the negative for SPNs and PSDDs.

For SPNs, using the transformation from [30] we are going to show that the resulting graph is not sufficient for studying causal relationships between the variables. Roughly, the problem is that this class of models allows for a lot of expressive freedom, and, because of that, all the correlations between the variables are attributed to external latent factors. Next, for PSDDs, we first provide an algorithm for constructing a causal graph that also needs to introduce some augmented variables, which conforms to PSDD on all probabilistic queries. On the one hand, intervening on the original variables in the resulting causal graph is also uninteresting, and reduces to computing marginal distributions, like in SPNs. However we can perform non-trivial counterfactuals on the augmented variables. This is possible because, in contrast to SPNs, PSDDs impose more restrictions on the structure of the resulting model, specifically in terms of its equivalence to a propositional formula, which we then can use to recover a structural equation model (SEM) [22]. We note that this structure is of a somewhat "deterministic" nature, and so, in a sense, the result is also negative. Nonetheless, we can provide a "causal semantics" for PSDDs in the process.

We reiterate that our focus is purely on the distributions represented or learnt using tractable probabilistic models, and specifically SPNs and PSDDs. These models do not come with any guarantees that the dependencies learnt actually capture the underlying causal process of the domain (in contrast to approaches such as [14]). Throughout the rest of our analysis, we suppose that the causal graph is not known beforehand and our aim is to examine what kind of information can be recovered using the trained SPNs or PSDDs. Our results demonstrate that regardless of whether these models do capture causal information, performing causal reasoning on them is not immediately attractive.

We are organized as follows. We first consider the SPN case, and then move on to the PSDD case. For reasons of space, a (brief) discussion on SPNs, PSDDs and SEMs can be found in the appendix. We refer the reader to [17, 28, 15] for more extensive discussions. In the context of our results, our notation will be as follows: An uppercase letter X denotes a Boolean random variable. In the context of a probabilistic statement, we will use a lowercase letter x to denote an assignment to X; for example, $\Pr(X = x)$ where $x \in \{0, 1\}$ denotes the probability of the event where X is assigned the

value x. In the context of a logical formula, X and $\neg X$ respectively assign *true* ($\top$) and *false* ($\bot$) to variable X. Sets of variables $\mathbf{X}$ and joint assignments $\mathbf{x}$ are denoted in **bold**.

2 Background

We will briefly review PSDDs, SPNs and SEMs.

Our notation will be as follows: An uppercase letter X denotes a Boolean random variable. In the context of a probabilistic statement, we will use a lowercase letter x to denote an assignment to X; for example, $\Pr(X = x)$ where $x \in \{0, 1\}$ denotes the probability of the event where X is assigned the value x. In the context of a logical formula, X and $\neg X$ respectively assign *true* ($\top$) and *false* ($\bot$) to variable X. Sets of variables $\mathbf{X}$ and joint assignments $\mathbf{x}$ are denoted in **bold**.

PSDDs. The idea behind PSDDs is to use Sentential Decision Diagrams (SDDs) [11] to represent a propositional logic theory, and then recursively define a probability distribution over it. Terminal nodes can be either a literal, $\top$, or $\bot$, while decision (intermediate) nodes are of the form $(p_1 \wedge s_1) \vee \cdots \vee (p_k \wedge s_k)$, where the $p_i's$ are called primes and the $s_i's$ subs. The primes form a partition, meaning they are mutually exclusive and their disjunction is valid. Each prime p_i in a decision node is assigned a non-negative parameter θ_i such that $\sum_{i=1}^{k} \theta_i = 1$ and $\theta_i = 0$ if and only if $s_i = \bot$. Additionally each terminal node corresponding to $\top$ has a parameter θ such that $0 < \theta < 1$. Using this notation, a PSDD node n defines a distribution over the variables of the *vtree node* u that it is normalized for, as follows. (The notion of a vtree, defined in [27], is needed to fully define an SDD; they can be obtained directly from data or by compiling domain constaints [18].)

- If n is terminal node, and u has variable X, then

n	$Pr_n(X)$	$Pr_n(\neg X)$
$\top$	θ	1 - θ
$\bot$	0	0
X	1	0
$\neg X$	0	1

- If n is a decision node and u has left children $\mathbf{X}$ and right children $\mathbf{Y}$, then $Pr_n(\mathbf{x}, \mathbf{y}) = Pr_{p_i}(\mathbf{x}) \cdot Pr_{s_i}(\mathbf{y}) \cdot \theta_i$, for i where $\mathbf{x} \models p_i$, and where $Pr_{p_i}(\cdot), Pr_{s_i}(\cdot)$ denote the distribution of the PSDD nodes corresponding to p_i, s_i, respectively.

SPNs. SPNs are rooted directed graphical models that provide for an efficient way of representing the network polynomial [8] of a BN [28], as a multilinear function $\sum_{\mathbf{x}} f(\mathbf{x}) \prod_{n=1}^{N} \mathbb{1}_{x_n}$. Here $f(\cdot)$ is the (possibly unormalized) probability distribution of the BN, $\mathbf{x}$ is a vector containing all the variables of the model, i.e., $x_1, \cdots, x_N$, the summation is over all possible states, and $\mathbb{1}_{x_n}$ is the indicator function. An SPN $\mathcal{S}$ over Boolean variables $x_1, \cdots, x_N$ has leaves corresponding to indicators $\mathbb{1}_{x_1}, \cdots, \mathbb{1}_{x_n}$ and $\mathbb{1}_{\bar{x}_1}, \cdots, \mathbb{1}_{\bar{x}_n}$ and whose internal nodes are sums and products.

Any edge exiting a sum node has a non-negative weight assigned to it. The value of a product node is the product of its children, while the value of a sum node is a weighted sum of its children, $\sum_{u_j \in Ch(u_i)} w_{ij} \mathcal{S}_{|}(\mathbf{x})$, where $Ch(u_i)$ is the set containing the children of node u_i, and $\mathcal{S}_{|}$ is the sub-SPN rooted at node u_j. SPNs can represent a wide class of models, including weighted mixtures of univariate distributions; see [28] for discussions.

Causality. We base our causal analysis on SEMs [22], which provide an effective way to encode dependencies between variables, as well as allow for queries regarding interventions and counterfactuals. In this setting we can represent a set of probabilistic dependencies through a BN, as usual, but on top of that we can also encode the specific mechanism that determines the value of each variable. In this sense, it is more general than just having a BN, since we not only possess a distribution over the variables, but also a (either stochastic or deterministic) set of equations. In what follows we denote by $\mathbf{V}$ the set of variables that are internal to the model, and by $\mathbf{U}$ the exogenous or external variables (that act as random, latent, factors). We use $\mathcal{R}$ to denote the set containing the plausible values of each variable. Every endogenous (internal) variable is assigned an equation determining its value as a function of both its endogenous and exogenous parents in the BN, called structural equation. Finally, in what follows, we make the standard assumption that these BNs do not contain any directed cycle, so they are equivalently referred to as directed acyclic graphs (DAGs).

Definition 1: A causal model $\mathcal{M}$ is a pair $(\mathcal{S}, \mathcal{F})$ where $\mathcal{S}$ is a signature $(\mathbf{U}, \mathbf{V}, \mathcal{R})$ and $\mathcal{F}$ is a set of structural equations $\{\mathcal{F}_{\mathcal{V}} : \mathcal{V} \in \mathbf{V}\}$.

One of the advantages of employing graphical models is that by just utilizing the topology of the graph we can answer probabilistic queries, such as whether two sets of variables are dependent.

Definition 2: A directed path is d-separated (blocked) by a set of nodes, $\mathbf{Z}$, iff one of these hold:

1. It contains a triple $X \rightarrow Z \rightarrow Y$ or $X \leftarrow Z \rightarrow Y$, such that $Z \in \mathbf{Z}$.
2. It contains a triple $X \rightarrow Z \leftarrow Y$, such that neither Z nor any of its descendants are in $\mathbf{Z}$.

Two sets $\mathbf{X}$,$\mathbf{Y}$ are d-separated by $\mathbf{Z}$ if and only if every path between any two nodes $X \in \mathbf{X}$, $Y \in \mathbf{Y}$ is blocked by $\mathbf{Z}$. It is a well established result that if two nodes are d-separated by a set $\mathbf{Z}$, then they are conditionally independent (where $\mathbf{Z}$ is the conditioning set). As we mentioned earlier, SEMs allow for studying interventional distributions, meaning the distribution of a set of variables, after we force a second set of variables to attain certain values. We denote the distribution of Y after the intervention $X = x$, by $\Pr(Y = y|do(X = x))$ or $\Pr(y|do(x))$. In order to study such probabilistic statements we transform the original DAG corresponding to our model, by deleting all the edges pointing towards X, set X to x, and then proceed with the analysis. What follows is an essential graphical tool for deciding under what conditions we can reduce interventional queries to conditional ones. Here, $G_{\overline{x}}$ denotes the graph obtained after deleting all the edges pointing to X, $G_{\underline{x}}$ the one resulting after deleting all the edges emerging from X, and $G_{\overline{x}\underline{z}}$ for deleting both kinds of edges from X.

Definition 3: **(Rules of do-Calculus)** Let $\mathcal{G}$ be a DAG corresponding to a SEM and $\Pr(\cdot)$ the probability measure induced by it. If $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W}$ are disjoint sets, then the following hold:

Rule 1: $\Pr(y|do(x), z, w) = \Pr(y|do(x), w)$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X}, \mathbf{W})_{G_{\overline{X}}}$.

Rule 2: $\Pr(y|do(x), do(z), w) = \Pr(y|do(x), z, w)$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X}, \mathbf{W})_{G_{\overline{X}\underline{Z}}}$.

Rule 3: $\Pr(y|do(x), do(z), w) = \Pr(y|do(x), w)$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X}, \mathbf{W})_{G_{\overline{X}\overline{Z(W)}}}$, where $\mathbf{Z}(\mathbf{W})$ is the set of $\mathbf{Z}$-nodes that are not ancestors of any $\mathbf{W}$-node in $G_{\overline{X}}$.

3 Main Results

3.1 The SPN Case

As discussed, SPNs are an elegant formalism for capturing weighted mixtures of distribution, and so the expressive power of SPNs and standard BNs has been of considerable interest. The question of how to transform SPNs to BNs and the recoverability of the SPN from the transformation was studied in [30]. For space reasons, we cannot provide too many details on this transformation, but the key idea is to compactly represent the local conditional probability distribution in the corresponding BN by exploiting context specific independence. Intuitively, we create a node for every observable variable, a latent variable for every sub-SPN, and then draw an arrow from each latent variable to the observable variables corresponding to the scope of the sub-SPN. This procedure yields a bipartite graph with arrows stemming only from latent to observable variables.

To our knowledge this is the only way proposed so far to turn an SPN to a BN, and many subsequent papers on SPNs' theoretical properties [26] are similar in thrust. In fact, the authors of [30] also state that:

> *The structure of the resulting BNs can be used to study probabilistic dependencies and causal relationships between the variables of the original SPNs.*

So we will base our analysis on this approach. We first make the following technical observation about graphs having this topology.

Theorem 4: *Let $\mathcal{G}$ be the DAG associated with a causal model $\mathcal{M} = (\mathcal{S}, \mathcal{F})$. For any set $\mathbf{X} \subseteq \mathbf{V}$ such that no node in $\mathbf{X}$ has an edge coming out of it, the interventional distribution of the remaining variables equals their joint distribution, i.e. $\Pr(u, v_{-x}|do(x)) = \Pr(u, v_{-x})$, where $u \subseteq \mathbf{U}$ and $v_{-x} \subseteq$*

$\mathbf{V} \setminus \mathbf{X}$. *More specifically, we have that the rest of the remaining variables are unaffected by the intervention, i.e.* $\Pr(v_{-x}|do(x)) = \Pr(v_{-x})$.

Unfortunately, since the BN stemming from the algorithm in [30] has no edge coming out of an observable variable, we get the following:

Theorem 5: *The BN, $\mathcal{G}$, that results after transforming an SPN using the procedure described in [30] satisfies the property* $\Pr(v_{-x}|do(x)) = \Pr(v_{-x})$*, for any* $\mathbf{X} \subseteq \mathbf{V}$.

So this result answers that the method proposed in [30] for producing a BN is not useful for causal inference tasks, since we cannot really study interventional distributions utilizing it.

What are the reasons for this limitation? As has been noted in previous work [26, 28], sum nodes in SPNs can be interpreted as marginalized, latent, variables, whose values correspond to the children of the sum node. Thus, when an SPN is turned into a BN all of the variables within the scope of a sum node are treated as children of a latent variable. This leads to every probabilistic dependency being attributed to an unobserved confounder, and there is no edge between the SPN variables. Thus, it is reasonable that any intervention on a subset of the observable variables would not affect the rest, because the mechanism encoded in the graph tells that no variable has any causal effect on the others.

A special class of SPNs, referred to as selective SPNs were introduced recently [25]. They impose determinism in that only one of the children of a sum node can be true for any given variable assignment. Interestingly, even this stipulation does not remedy the problem, since the discussion in [26] makes clear the resulting BN would still have no edges between the SPN variables. Consequently, we get:

Theorem 6: *The BN, $\mathcal{G}$, that results after transforming a selective SPN using the procedure described in [26] satisfies the property* $\Pr(v_{-x}|do(x)) = \Pr(v_{-x})$*, for any* $\mathbf{X} \subseteq \mathbf{V}$.

We suspect that to get rid of this limitation SPNs should be augmented in a way that captures the functional dependency between the variables in the scope of a sum node. Another strategy perhaps is to enable a more expressive way to represent and reason about probabilistic dependencies between the variables, although it is not immediate how this could be made possible.

3.2 The PSDD Case

Interestingly, the situation turns out to be slightly better for PSDDs. The intuitive reason for that is because of the dependency that can be established between a node in the PSDD and its children. More precisely, consider that each node n in a PSDD [17] has a support – the set of assignments it assigns a positive probability – which is related to the support of its children. This set is called the *base* of n and is denoted by $[n]$. It can also be defined as a logical formula: if n is a decision node $(p_1 \wedge s_1) \vee \cdots \vee (p_k \wedge s_k)$, then $[n] = \bigvee_{i=1,\ldots,k} [p_i] \wedge [s_i]$. Since p_i's form a partition, their corresponding bases are disjoint, as well, so a decision node can be seen as deciding between different possible worlds, based on which prime base was satisfied. Since the prime bases of a node form a partition we can apply the law of total probability and Proposition 1 from [19] to get that $Pr_n(\mathbf{x}, \mathbf{y}) = \sum_{i=1,\ldots,k} Pr_n(\mathbf{x}, \mathbf{y}, [p_i]) = \sum_{i=1,\ldots,k} Pr_n(\mathbf{x}, \mathbf{y}|[p_i]) \cdot Pr_n([p_i]) = \sum_{i=1,\ldots,k} Pr_n(\mathbf{x}|[p_i]) \cdot Pr_n(\mathbf{y}|[p_i]) \cdot Pr_n([p_i])$. Combining this expression with the semantics provided in Proposition 1 in [19] and Theorem 2 in [17], as well as the fact that under any given assignment the only non-zero term of the form $Pr_n(\mathbf{x}|[p_i])$ is the one for which $\mathbf{x} \models p_i$, we see that the probability of a node is not a mixture over its children. Indeed, the distribution of decision node is understood very differently. In fact, we can also see that PSDD nodes do not condition on a latent variable, but on their prime bases instead, which do not depend on unobserved quantities.

Our work builds on this observation and the fact that by construction PSDDs are probabilistic extensions of SDDs, which, in turn, denote a propositional formula. Basically, we use that formula to create an augmented set of variables, not just the original ones the PSDD used for training, in such a way so the PSDD distribution and the BN one agree on the original variables. It is worth noting that the resulting BN is also equipped with a set of equations that determine the value of the children as functions of their parents, so we end up having a SEM. Below we present the procedure to construct this SEM, where the input propositional formula is the one represented by the trained PSDD.

Algorithm 1: Expression to SEM

Input: A formula $\phi = c_1 \vee \cdots \vee c_n$, where $c_i = p_i \wedge s_i$, over propositional variables $x_1, \ldots x_k$
Output: A SEM model with the augmented variables

1 Create a variable corresponding to the whole expression, $v_0 = \phi$;
2 Create a variable, v_i, for each c_i;
3 Create an arrow from v_i to v_0, $i = 1, \ldots, n$;
4 For each $c_i = p_i \wedge s_i$ create a variable v_i^p for p_i and a variable v_i^s for s_i;
5 Create an arrow from v_i^j to v_i, for $j \in \{p, s\}$;
6 Repeat this process recursively, until the original variables are reached;
7 When this procedure is over, create a hidden variable and connect it with each one of the original variables with an arrow pointing at them.

About the hidden variable: The latent variable, $\mathbf{H}$, is the only component of the graph that is purely stochastic, and we motivate its need here. Note that any instantiation of it is enough to determine all the other variables in the model. Conversely, each probabilistic query about any of the rest of the variables can be reduced into another query relying solely on $\mathbf{H}$ (since there is no other source of randomness in the model). Its dimension is equal to the number of the original variables in the PSDD and its distribution is equal to the PSDD distribution of the original variables. Denoting by $\Pr_{SEM}(\cdot)$ the probability measure over the DAG's variables and by $\Pr_{PSDD}(\cdot)$ the PSDD distribution over the original variables, we set these two measures to satisfy the following condition $\Pr_{SEM}(H_1, H_2, \ldots, H_n) = \Pr_{PSDD}(X_1, X_2, \ldots, X_n)$. Suppose the PSDD is comprised of n variables, $X_1, X_2, \ldots, X_n$, then $\mathbf{H} = (H_1, H_2, \ldots, H_n)$. The structural equations connecting them are: $X_1 = H_1, X_2 = H_2, \ldots, X_n = H_n$. Looking at these equations we see that: $\Pr_{SEM}(X_1, X_2, \ldots, X_n) = \Pr_{SEM}(H_1, H_2, \ldots, H_n) = \Pr_{PSDD}(X_1, X_2, \ldots, X_n)$. This remark assures us about the consistency between the PSDD and the SEM distribution of the original variables. We would also like to note that although $\mathbf{H}$ is introduced as a vector, it could be rewritten as a simple categorical variable with an exponential number of states, each one corresponding to a different configuration of the original variables. We present this result in a more formal way, using the vectorized version of $\mathbf{H}$.

Theorem 7: *Let $\boldsymbol{P}$ be a PSDD over variables $X_1, X_2, \ldots, X_n$ and let $\mathcal{G}$ be the DAG resulting from Algorithm 1. The distribution of $X_1, X_2, \ldots, X_n$, induced by $\mathcal{G}$ is equal to their PSDD distribution, meaning that* $\Pr_{SEM}(X_1, X_2 \ldots, X_n) = \Pr_{PSDD}(X_1, X_2, \ldots, X_n)$.

Interestingly, the SEM obtained from a PSDD in this manner has the same limitations as identified for SPNs when intervening on the original variables:

Theorem 8: *The SEM, $\mathcal{S}$, that results after applying Algorithm 1 to a PSDD compiled formula satisfies the property* $\Pr(v_{-x}|do(x)) = \Pr(v_{-x})$*, where X is any subset of the original variables, and v_{-x} denotes the rest of the original variables.*

However, when intervening on the augmented variables, we are able to enable non-trivial (but also non-standard) causal reasoning, a point we return to shortly.

Moreover $\mathbf{H}$ serves another purpose as we will shortly discuss. Using the BN from Algorithm 1 without including the hidden variable, it is not difficult to see that the original PSDD variables are independent, since all the paths connecting them are blocked by *v-structures*, meaning that (2) in Definition 2 is satisfied, with $\mathbf{Z} = \emptyset$. On the other hand, it is not necessarily the case that the PSDD distribution encodes such properties about the variables, so there is a chance that the BN distribution enforces independences that do not agree with the PSDD one, rendering the DAG unfaithful [23]. By including the hidden variable we eliminate this behaviour, but we introduce a new property, the other extreme, that all of the variables are dependent. This might also not be the actual case either, but we think that it is safer to assume dependency among the variables, rather than independency, which is a fairly strong assumption. A better way to address this behaviour would be to utilize the PSDD distribution and some independency tests in order to decide the subsets of dependent variables, and then use as many hidden variables as the dependent subsets, so we explicitly encode only the dependencies that are implied by the PSDD distribution. (Incidentally, such tests are used

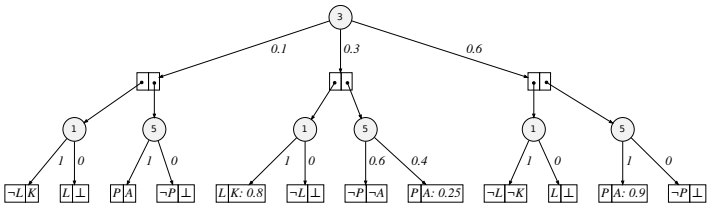

Figure 1: A PSDD over variables A, L, K, P

when learning SPNs [13].) In this work we are mostly interested in introducing the connection between BNs and PSDDs, so we leave this for future research.

We should also note that for any node X in the graph resulting from Algorithm 1, denoting the set of its parents as $\mathbf{PA}_X$, we have:

$$\Pr(X = 1|\mathbf{PA}_X) = \begin{cases} 1 & \text{if assignments in } \mathbf{PA}_X \text{ render } X = 1 \\ 0 & \text{otherwise} \end{cases}$$

Building on top of this remark, the distribution of X given the specification of any partial subset of its parents $\mathbf{V} \subset \mathbf{PA}_X$ is as follows:

$$\Pr(X = 1|\mathbf{V}) = \begin{cases} 1 & \text{if assignments in } \mathbf{V} \text{ render } X = 1 \\ 0 & \text{if assignments in } \mathbf{V} \text{ render } X = 0 \\ \Pr(X|_{\mathbf{V}} = 1) & \text{otherwise} \end{cases}$$

where $X|_{\mathbf{V}}$ denotes the formula that results from X after substituting the assignments from $\mathbf{V}$ in it. Finally, the marginal distribution of X, for example, can be computed by using the PSDD.

Example: We will give an example of how to construct a SEM model using a PSDD. We start by studying the PSDD in Figure 1. This is the PSDD corresponding to a problem considered in [17]. The setting is there is a department having four courses: Logic (L), Knowledge Representation (K), Probability (P), and Artificial Intelligence (A). Students must enroll to them, but at the same time they have to obey the following constraints: $P \vee L$, $A \Rightarrow P$, $K \Rightarrow (A \vee L)$ (where implication means if they enroll in LHS, then they must enroll in RHS). The objective is to learn the joint distribution of L, K, P, A using a dataset of student enrollments and the above constraints. The authors utilize PSDDs to perform this task and the resulting model can be seen in Figure 1.

Starting from the bottom of Figure 1 and moving towards the root, we see that it corresponds to the following propositional formula:

$$\begin{aligned} &(((\neg L \wedge K) \vee (L \wedge \bot))) \wedge ((P \wedge A) \vee (\neg P \wedge \bot))) \\ &\vee (((L \wedge K) \vee (\neg L \wedge \top)) \wedge ((\neg P \wedge \neg A) \vee (P \wedge A))) \\ &\vee (((\neg L \wedge \neg K) \vee (L \wedge \bot)) \wedge ((P \wedge A) \vee (\neg P \wedge \bot))) \end{aligned}$$

This is the raw form of the formula, so some terms are tautologically false. Rewriting the above expression after eliminating inconsistencies yields the following:

$$((\neg L \wedge K) \wedge (P \wedge A)) \vee ((L \wedge K) \wedge ((\neg P \wedge \neg A) \vee (P \wedge A))) \vee ((\neg L \wedge \neg K) \wedge (P \wedge A)) \qquad (\star)$$

Algorithm 1 takes $(\star)$ as input and constructs a SEM model, as follows: The first thing is to create a node corresponding to the whole expression. Then, since $(\star)$ is composed of three disjunctions, we make three new variables, one for each of them, and draw arrows from them pointing to the first variable. We continue this procedure recursively; so for example, the term $(\neg L \wedge K) \wedge (P \wedge A)$ is made from two formulas that are connected with a conjunction, so we create two new nodes, one for $\neg L \wedge K$ and one for $P \wedge A$, and draw arrows from them towards the node representing their conjunction. Now we have reached the point where the formulas under consideration are just conjunctions of literals, so if we look at $\neg L \wedge K$, we make a node for the variable L (although it is $\neg L$ that is part of the formula) and one for K. We repeat the above procedure until we go through all the formulas and in the end we create an additional latent variable that is a parent of all the original PSDD variables, here A, L, K, P. The resulting BN can be seen in Figure 2 (Left). It is also worth noting that since

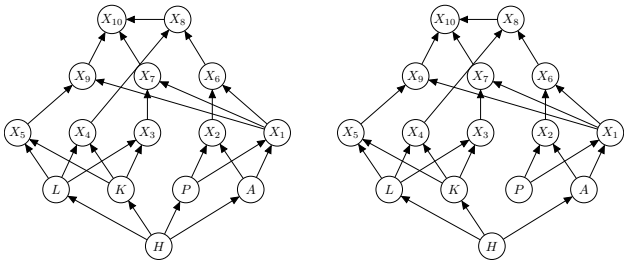

Figure 2: Left: The augmented BN model corresponding to the PSDD of figure 1. Right: The same model, after intervening on P.

L	K	P	A	PSDD distribution
0	0	1	0	6.0%
0	0	1	1	54.0%
0	1	1	1	10.0%
1	0	0	0	3.6%
1	0	1	0	1.8%
1	0	1	1	0.6%
1	0	0	0	3.6%
1	1	0	0	14.4%
1	1	1	0	7.2%
1	1	1	1	2.4%

Table 1: Distribution of L, K, P, A from [17]

we create at most two new nodes for any disjunction or conjunction, the size of the BN is linear in the their number. Furthermore, looking at the procedure described above, we see that Algorithm 1 is not directly applicable to SPNs, since sums and products are between distributions, while in PSDDs, conjunctions are disjunctions are between variables, which is exactly what Algorithm 1 exploits in order to construct the resulting SEM.

We have kept the names of the original variables the same and have named the rest as $X_1, \ldots, X_{10}$. In addition, the latent variable is a vector $(H_1, \ldots, H_4)$ of 4 random variables, since the PSDD was over four variables. By construction, it is now apparent that the structural equations of this BN are:

$$A = H_1, L = H_2, K = H_3, P = H_4, X_1 = P \wedge A, X_2 = \neg P \wedge \neg A, X_3 = \neg L \wedge K, X_4 = L \wedge K,$$
$$X_5 = \neg L \wedge \neg K, X_6 = X_1 \vee X_2, X_7 = X_1 \wedge X_3, X_8 = X_4 \wedge X_6, X_9 = X_1 \wedge X_5, X_{10} = X_7 \vee X_8 \vee X_9$$

We can immediately see that an intervention on one of the augmented variables will result in a non-trivial interventional distribution. The next section demonstrates this for the case of counterfactuals.

3.3 Counterfactuals

In this section we will examine if it is possible to use the BN from Algorithm 1 in order to compute counterfactual quantities. We will mostly investigate counterfactuals conditioned on some evidence, which is equivalent to computing probabilistic statements of the form $\Pr(Y = y|do(X = x), E = e)$. These statements can be handled using the following major result [23]:

Theorem 9: *[23] Let $\mathcal{M}$ be a causal model and $\Pr(\cdot)$ a probability measure over the variables in $\mathbf{U}$. The counterfactual probability $\Pr(Y = y|do(X = x), E = e)$, meaning "Had X been x then Y would have been y, given evidence e", can be computed as follows:*

- ***Abduction:*** *Update the distribution $\Pr(\mathbf{U})$ by incorporating the evidence, to obtain $\Pr(\mathbf{U}|e)$.*
- ***Action:*** *Construct the graph that results from the intervention $do(X = x)$.*
- ***Prediction:*** *Use the probability measure and the graph, from the previous steps, to compute the probability of $Y = y$.*

Since our model is deterministic, we do not have to do a lot of probabilistic calculations, but mostly we are going to manipulate logical expressions. We will go on with our working example to demonstrate how we could study counterfactuals and their properties. The question of interest is the following: Supposing we have observed that $X_1 = 0$, what is the probability it would have been equal

to 1, had P been equal to 1? At this point we would like to emphasize that an intervention on one of the augmented variables corresponds to multiple interventions on the original ones. For example, suppose that later on we decide to intervene on X_1 and force it to become equal to zero. In turn, this would mean that we force $A \wedge P$ to become zero. We notice that this outcome can be achieved by several assignments on these two variables, namely $A = P = 0$, $A = 0$ and $P = 1$, and $A = 1$ and $P = 0$. This means that a single intervention on X_1 induced three interventions on A and P simultaneously. We would also like to emphasize that although X_1 belongs to the augmented set of variables, it still has an interpretation relating it to the original variables, as it is the case with any of the augmented variables. In this case, X_1 just represents the event of taking both courses, A and P.

Formally, we ask for the probability of the following expression $\Pr(X_1 = 1|do(P = 1), X_1 = 0)$. The first step is to update the distribution of our exogenous variables (in our case, this is $\mathbf{H}$) conditioning on the evidence $X_1 = 0$. As we have already discussed, $X_1 = 0 \Leftrightarrow (P \wedge A) = 0$, so P or A is equal to zero. This means the updated distribution should assign zero probability to the case of P and A being true at the same time. Since this is the only fact we can recover from the conditioning observation, the posterior and the prior distributions should agree on all other cases. Thus, we end up with $\Pr(H_1, H_2, H_3, H_4|X_1 = 0)$ being obtained as:

$$\begin{cases} \frac{\Pr(H_1,H_2,H_3,H_4,X_1=0)}{\Pr(X_1=0)} = \frac{\Pr(H_1,H_2,H_3,H_4)}{\Pr(X_1=0)} & \text{if } H_1 = 0 \text{ or } H_4 = 0 \\ 0 & \text{otherwise} \end{cases}$$

The upper branch equality follows from Bayes' Theorem and the fact that $(H_1 = 0 \vee H_4 = 0) \Leftrightarrow X_1 = 0$. Next, we construct the graph corresponding to the world where we intervene on P and force it to be true, which is shown in Figure 2 (Right). Now we update the structural equations, by substituting $P = 1$ to all the equations. Since we are not going to make use of all of them in this particular example, we will write down only the first few.

$$A = H_1, L = H_2, K = H_3, P = 1, X_1 = A, X_2 = 0$$

Now we are ready to perform all the desired calculations,in our case the probability of $X_1 = 1$ in the causal graph of Figure 2 (Right). We proceed as follows:

$$\begin{aligned} \Pr(X_1 = 1) = \Pr(A = 1) = \Pr(H_1 = 1) = \Pr(H_1 = 1, H_4 = 0) \\ = \sum_{H_2} \sum_{H_3} \frac{\Pr(H_1 = 1, H_2, H_3, H_4 = 0)}{\Pr(X_1 = 0)} \end{aligned}$$

We immediately see that all of the needed probabilistic quantities can be calculated right away using the PSDD and the correspondence between H_1, H_2, H_3, H_4 and A, P, K, L.

We could also ask more complex counterfactual queries as well. This time we will include the actual numeric values, so that we can compare the various distribution of the variable of interest. The data is taken from [17] is shown here, but the full calculations can be found in the supplementary material. The question this time is supposing we have witnessed that $X_9 = P \wedge A \wedge \neg L \wedge \neg K = 0$, meaning there is a student not satisfying the property "he/she has taken both A and P, while not taking neither L or K": what is the probability of him/her satisfying this property, had A been equal to 1. We repeat the same steps as before, to obtain the probability $\Pr(X_9 = 1|do(A = 1), X_9 = 0)$, which turns out to be equal to $\Pr(H_1 = 0, H_2 = 0, H_3 = 0, H_4 = 1)/0.46 = 0.06/0.46 \approx 0.13$. We compare the resulting counterfactual distribution to the conditional $\Pr(X_9|A = 1)$ and the plain marginal $\Pr(X_9)$. The results can be seen in Figure 3. It is evident that the counterfactual distribution is vastly different from the others, expanding the semantics of PSDDs in a non-trivial way.

4 Discussion and Conclusions

Tractable models are attractive in offering polynomial time inference capabilities, and hence are gaining in popularity. The theoretical properties of such models have received considerable attention recently. The question of whether these models can also be useful for causal reasoning was studied in this work, and we showed that the results are mostly of a negative nature. For SPNs, we showed that we cannot really study interventional distributions. For PSDDs, we motivated a way to construct a SEM from a trained PSDD. We showed that when intervening on the original variables, the situation is once again uninteresting, but when non-trivial properties emerge when augmented variables are considered. While this does provide a causal semantics for PSDDs, we observe the causal graph

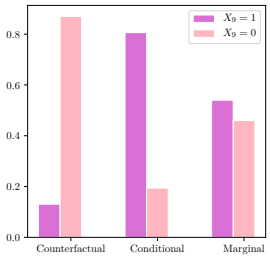

Figure 3: Comparison of distributions

is very unusual in lacking noise. So, the overall usefulness of these class of tractable models is questionable for causal reasoning. We would like to reiterate that the thrust of this contribution assumes that the only information we have is the probabilistic circuit. Clearly if we had the original BN in hand, we would perform causal reasoning directly on that BN. However, starting from the circuit, we show that going to the BN loses information about the underline mechanisms that the variables interact with each other, as it is evident when using SPNs. For PSDDs, although the outcome of the analysis is about the same, the problem is of a different nature, and it is mostly due to absence of latent factors. In many cases, in causal modeling, we tend to include some latent variables, in order to account for unobserved background factors, but it is unclear why one should do this for PSDDs. From a causal viewpoint, SPNs and PSDDs also seem to be on the opposite sides of the spectrum, one former attributing everything to latent factors, while the latter attributing nothing to them.

We think there are many interesting directions for the future. For example, given our last observation about latent factors, are there tractable models that enable causal graphs with lie somewhere on the middle ground wrt causal graphs? Current structure learning algorithms for tractable models also do not attempt to capture the underlying causal process. In that regard, the quality of the causal graph obtained in Algorithm 1 is only going to be as good as the quality of the PSDD. We think there are three ways to improve this situation, both under the assumption that we are in possession of prior knowledge in terms of certain dependencies and independencies. Firstly, a brute force (and very likely inefficient) structure learner would first build a PSDD, use Algorithm 1 to recover all the dependencies and interactions between variables and test whether our prior knowledge is in agreement with what the PSDD has learned. An insufficient model would then be discarded by means of a suitable evaluation metric. Secondly, it is shown in [19] that the training of PSDDs can be subjected to logical prior knowledge. It may be possible to extend that approach, in that we learn PSDDs that are also subjected to independency constraints expressed as probabilistic prior knowledge. Thirdly, and perhaps most significantly, investigating whether ideas from the existing literature on learning causal relations (e.g.,[14]) can be imported to tractable learners is a worthwhile question. Of course, such an endeavor would be most useful if we discover ways to augment SPNs and PSDDs in some (clever) way that goes beyond trivial and/or deterministic reasoning. That would be perhaps the main open challenge resulting from our work.

Acknowledgements
The authors wish to thank several anonymous reviewers for their helpful feedback. Vaishak Belle was supported by a Royal Society University Research Fellowship.

References

[1] S. B. Akers. Binary decision diagrams. *IEEE Trans. Comput.*, 27(6):509–516, June 1978.

[2] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving sat and bayesian inference with backtracking search. *J. Artif. Intell. Res. (JAIR)*, 34:391–442, 01 2009.

[3] Francis R Bach and Michael I Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2002.

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. GAN dissection: Visualizing and understanding generative adversarial networks. *CoRR*, abs/1811.10597, 2018.

[5] Michel Besserve, Rémy Sun, and Bernhard Schölkopf. Counterfactuals uncover the modular structure of deep generative models. *CoRR*, abs/1812.03253, 2018.

[6] Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. *arXiv preprint arXiv:1708.06846*, 2017.

[7] Arthur Choi, Guy Van den Broeck, and Adnan Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *IJCAI*, 2015.

[8] Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50:280–305, 2000.

[9] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11, 04 2000.

[10] Adnan Darwiche. A logical approach to factoring belief networks. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning*, pages 409–420, 2002.

[11] Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. pages 819–826, 01 2011.

[12] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 666–674. Curran Associates, Inc., 2011.

[13] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, 2013.

[14] AmirEmad Ghassami, Saber Salehkaleybar, Negar Kiyavash, and Elias Bareinboim. Budgeted experiment design for causal structure learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1719–1728, 2018.

[15] Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part ii: Explanations. *The British Journal for the Philosophy of Science*, 56(4):889–911, 2005.

[16] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1809–1818. JMLR. org, 2017.

[17] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, pages 558–567, 2014.

[18] Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *UAI*, 2017.

[19] Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[20] Yitao Liang and Guy Van den Broeck. Learning logistic circuits. *ArXiv*, abs/1902.10798, 2018.

[21] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, AU , Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. 01 2018.

[22] Judea Pearl. Causal inference in statistics: An overview. 2009.

[23] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[24] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

[25] Robert Peharz, Rüdiger Gens, and Pedro M. Domingos. Learning selective sum-product networks. 2014.

[26] Robert Peharz, Rüdiger Gens, Franz Pernkopf, and Pedro M. Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2030–2044, 2016.

[27] Thammanit Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. volume 1, pages 517–522, 01 2008.

[28] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690, Nov 2011.

[29] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. 11 2017.

[30] Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 116–124. JMLR.org, 2015.

A Proofs

Proof for Theorem 4: Using the 3rd rule of Pearl's do-calculus, it suffices to show that $(\mathbf{X} \perp\!\!\!\perp \mathbf{U} \cup (\mathbf{V} \setminus \mathbf{X}))_{G_{\overline{X}}}$. By assumption, no edges emanate from nodes in $\mathbf{X}$, which implies that each of them will be isolated in $G_{\overline{X}}$, so the desired independence holds, meaning that $\Pr(u, v_{-x}|do(x)) = \Pr(u, v_{-x})$. In addition, we have that $\Pr(v_{-x}|do(x)) = \sum_U \Pr(u, v_{-x}|do(x)) = \sum_U \Pr(u, v_{-x}) = \Pr(v_{-x})$. ∎

Proof for Theorem 5: As we have already discussed, $\mathcal{G}$ has no edges stemming from observable variables, so the result follows by applying the above proposition. ∎

Proof for Theorem 7: We are going to use the 3rd rule of Pearl's do-calculus, so it is enough to show that for any path bewtween the original variables is blocked. Let X be the variable we intervene on and let Y be any of the rest of the original variables. We have to show that $(X \perp\!\!\!\perp Y)_{G_{\overline{X}}}$. By construction, since the BN is created using Algorithm 1, there are no edges between the original variables. Furthermore, no original variable is a descendant of another one, since the only parent of an original variable is the latent variable. This means that, in $G_{\overline{X}}$, all the paths connecting X and Y contain v-structures, so they are blocked and the 3rd rule is satisfied. Since Y was chosen at random, we can generalize this result for arbitrary subsets of the original variables, concluding the proof. ∎

B Example

Here we give details about the calculations performed in the second part of our example. The steps are completely analogous to the ones outlined for the first part of the same example. Following exactly the same reasoning, the first step is to update the distribution of the hidden variables. The evidence is that $X_9 = A \wedge P \wedge \neg L \wedge \neg K = 0$, so the only non valid configuration is the one rendering $x_9 = 1$, meaning that $\Pr(H_1, H_2, H_3, H_4|X_9 = 0) = 0$if $H_1 = 1$, $H_2 = 0$, $H_3 = 0$ and $H_4 = 1$, while in all other cases:

$$\Pr(H_1, H_2, H_3, H_4|X_9 = 0) = \frac{\Pr(H_1, H_2, H_3, H_4, X_1 = 0)}{\Pr(X_9 = 0)} =$$
$$\frac{\Pr(H_1, H_2, H_3, H_4)}{\Pr(X_9 = 0)} = \frac{\Pr(H_1, H_2, H_3, H_4)}{1 - \Pr(X_9 = 1)} =$$
$$\frac{\Pr(H_1, H_2, H_3, H_4)}{0.46}$$

where we calculated the $\Pr(X_9 = 1)$, based on the numbers of Table 1. Next, we update all the structural equations according to the intervention $do(A = 1)$. We are interested in the updated equation of x_9, which is $P \wedge \neg L \wedge \neg K$. The last step is to compute the probability of $x_9 = 1$ using the the newly acquired distribution and equation.

$$Pr(x_9 = 1) = Pr(P \wedge \neg L \wedge \neg K = 1) =$$
$$Pr(H_2 = 0, H_3 = 0, H_4 = 1) =$$
$$Pr(H_1 = 0, H_2 = 0, H_3 = 0, H_4 = 1) =$$
$$\frac{Pr(H_1 = 0, H_2 = 0, H_3 = 0, H_4 = 1)}{0.46} = \frac{0.06}{0.46} \approx 0.13$$

In the above expression we abused notation slightly, since we used $Pr(\cdot)$ to denote both the counterfactual and the observational distributions, but we hope it is clear that the first three terms involve the counterfactual distribution, while the rest is referred to the observational one. The actual numbers can be found in Table 1. The computations for the conditional as well as the marginal distribution are trivial and can be performed using, again, table.