Compositional Probabilistic and Causal Inference using Tractable Circuit Models

Benjie Wang
Department of Computer Science
University of Oxford

Marta Kwiatkowska
Department of Computer Science
University of Oxford

Abstract

Probabilistic circuits (PCs) are a class of tractable probabilistic models, which admit efficient inference routines depending on their structural properties. In this paper, we introduce *md-vtrees*, a novel structural formulation of (marginal) determinism in structured decomposable PCs, which generalizes previously proposed classes such as probabilistic sentential decision diagrams. Crucially, we show how md-vtrees can be used to derive tractability conditions and efficient algorithms for advanced inference queries expressed as *arbitrary compositions* of basic probabilistic operations, such as marginalization, multiplication and reciprocals, in a sound and generalizable manner. In particular, we derive the first polytime algorithms for causal inference queries such as backdoor adjustment on PCs. As a practical instantiation of the framework, we propose MDNets, a novel PC architecture using md-vtrees, and empirically demonstrate their application to causal inference.

1 INTRODUCTION

Probabilistic circuits (PC) (Choi et al., 2020) are a broad family of tractable probabilistic models that are known for their ability to perform *exact* and *efficient* probabilistic inference. For example, in contrast to neural probabilistic models such as variational autoencoders (VAE) (Kingma and Welling, 2013), generative adversarial networks (GAN) (Goodfellow et al., 2014), and normalizing flows (NF) (Rezende and Mohamed, 2015), linear-time exact algorithms on PCs are available for important inference tasks such as computing marginal probabilities, or the maximum probability assignment of variables, for certain types of PCs. Meanwhile, probabilistic circuit structures have recently been shown to scale to high-dimensional datasets such as CelebA (Peharz et al., 2020a).

A distinguishing feature of the PC framework is the ability to trade off *expressive efficiency* for *tractability* by imposing various properties on the PC. Broadly, these properties can be divided into *scope properties*, such as decomposability and structured decomposability (Pipatsrisawat and Darwiche, 2008), and *support properties*, such as determinism, strong determinism (Pipatsrisawat and Darwiche, 2010), and marginal determinism (Choi et al., 2020). As we impose more properties on a PC, more inference tasks become tractable (i.e. computable in polynomial time), but we also lose some expressive efficiency and generality.

In this paper, we aim to extend the boundaries of advanced inference queries that can be tackled with PCs. In particular, we consider probabilistic inference queries specified as compositions of basic operations such as marginalization, products, and reciprocals, building upon the approach of Vergari et al. (2021). We find that current PC classes are not sufficient to analyze *arbitrary* compositions of these operations, and thus propose a novel class of circuits (md-vtrees & MDNet) and accompanying rules (MD-calculus) to derive in a sound and generalizable manner tractability conditions and algorithms for such compositions. Exploiting this, we design the first efficient exact algorithms for causal inference on probabilistic circuits.

Our first contribution is to introduce a unifying formulation of support properties in structured decomposable circuits using *md-vtrees*. We show that md-vtrees generalize previously proposed PC families such as probabilistic sentential decision diagrams (PSDD) (Kisa et al., 2014) and structured decomposable and deterministic circuits (Dang et al., 2020; Di Mauro et al., 2021). Notably, we also show that PSDDs are not *optimal* in that we can impose weaker support properties while maintaining tractability for the same inference tasks.

Next, as a practical instantiation of the framework, we propose MDNets, a novel architecture for PCs which can be easily configured to conform to any md-vtree. Cru-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

cially, this allows us to enforce arbitrary support properties needed for tractable inference, including those not covered by existing PC architectures. We show how to learn MDNets simply and efficiently using randomized structures and parameter learning (Peharz et al., 2020b,a; Di Mauro et al., 2021).

Finally, for inference, we derive a set of rules for analyzing arbitrary compositions of basic operations using md-vtrees, which we call the MD-calculus. In particular, MD-calculus rules can be applied backward through a given composition, to derive sufficient conditions for tractability on the inputs to the query, which we can enforce during learning through our MDNets. As an application, we demonstrate how the MD-calculus can be applied to derive tractability conditions and algorithms for causal inference estimands on PCs, including the backdoor and frontdoor formulae and (an extension of) the napkin formula.[1]

1.1 Related Work

Support Properties The property of determinism was first introduced in the context of Boolean circuits, specifically, those in negation normal form (Darwiche, 2001; Darwiche and Marquis, 2002), before being naturally extended to arithmetic/probabilistic circuits (Darwiche, 2003). Later, a stronger property known as *strong determinism* was introduced (Pipatsrisawat and Darwiche, 2010; Darwiche, 2011; Kisa et al., 2014) as a convenient means of enforcing determinism in structured decomposable circuits by tying determinism to the scope decomposition, resulting in the (probabilistic) sentential decision diagram (SDD). Oztok et al. (2016) further introduced the notion of *constrained vtrees*, which restricts the structure (scopes, and thus support) of the SDD vtree in order to solve problems on weighted Boolean formulae. Finally, Choi et al. (2020) recently introduced a more general support property called *marginal determinism*, which applies to general probabilistic circuits and is not directly tied to the scope decomposition; our work shows how to construct marginal deterministic circuits, previously considered an intractable task (Choi et al., 2022). Marginal determinism is sufficient for tractability of some marginal MAP queries (Huang et al., 2006).

Causality and Probabilistic Circuits The relationship between probabilistic circuits and causality has its roots in the seminal *compilation* methods of Darwiche (2003), which described an inference approach for (causal) Bayesian networks that involved compiling their graphs into tractable arithmetic circuits; subsequent work has further examined causality and compiled circuits (Butz et al., 2020; Wang et al., 2021; Darwiche, 2021; Chen and Darwiche, 2022). However, obtaining an exact causal interpretation of more general, learned probabilistic circuits has remained an open problem (Zhao et al., 2015; Papantonis and Belle, 2020). The only practical prior causal inference method for such circuits is the neural parameterization of Zecevic et al. (2021), but this lacks exactness guarantees and is only applicable to fully observed settings. In contrast, we consider exact causal inference using do-calculus, where circuits encode the observed probability distribution.

2 PRELIMINARIES

Notation We use uppercase to denote a random variable (e.g., V) and lowercase for an instantiation of a variable (e.g., v). Sets of variables (and their instantiations) are denoted using bold font (e.g., $\boldsymbol{V}, \boldsymbol{v}$), and we use val for the set of all instantiations of a set of variables (e.g., $val(\boldsymbol{V})$).

Probabilistic circuits (PC) (Choi et al., 2020) are computational graphs which encode a non-negative function over a set of variables; in particular, they are often used to model (possibly unnormalized) probability distributions.

Definition 1 (Probabilistic Circuit)**.** *A circuit C over variables $\boldsymbol{V}$ is a parameterized rooted graph, consisting of three types of nodes N: leaf L, sum T and product P. Leaf nodes L are leaves of the graph, while each internal node (sum or product) N has a set of children, denoted $ch(N)$. Sum nodes have a parameter/weight $\theta_i \in \mathbb{R}^{\geq 0}$ associated with each of their children N_i.*

Each leaf node L encodes a non-negative function $p_L : \phi(L) \to \mathbb{R}^{\geq 0}$ over a subset of variables $\phi(L) \subseteq \boldsymbol{V}$, known as its scope*. The function encoded by each internal node N is then given by:*

$$p_N(\boldsymbol{V}) := \begin{cases} \prod_{N_i \in ch(N)} p_{N_i}(\boldsymbol{V}) & \text{if } N \text{ is a product}^2 \\ \sum_{N_i \in ch(N)} \theta_i p_{N_i}(\boldsymbol{V}) & \text{if } N \text{ is a sum} \end{cases}$$

The function encoded by the circuit, $p_C(\boldsymbol{V})$, is defined to be the function encoded by its root node R. The size of a circuit, denoted $|C|$, is defined to be the number of edges in the circuit.

Definition 2 (Scope and support of PC node)**.** *The scope of an internal node N is the set of variables p_N specifies a function over, recursively defined by $\phi(N) := \bigcup_{N_i \in ch(N)} \phi(N_i)$. The support of any node N is the set of all instantiations of its scope s.t. p_N is positive, defined as* $\text{supp}(N) := \{\boldsymbol{w} \in val(\phi(N)) : p_N(\boldsymbol{w}) > 0\}$.

The tractability of probabilistic circuits depends on the scope and support properties they satisfy. A PC is *decomposable* if the children of a product node have distinct scopes (and thus partition the scope of the product node),

[1] Further details and proofs can be found in the Appendices of the extended version of this paper at `http://www.fun2model.org/papers/wk23.pdf`

[2] We assume in this paper that each product node has exactly two children; this does not lose generality as any product node can be converted into a sequence of binary products.

and is *smooth* if the children of a sum node have the same scope. Decomposability and smoothness together enable tractable marginal inference; that is, for any subset $\boldsymbol{W} \subseteq \phi(N)$ of the scope of a node N, we can compute $p_N(\boldsymbol{W})$ efficiently, where $p_N(\boldsymbol{W}) := \sum_{\phi(N) \setminus \boldsymbol{W}} p_N(\phi(N))$ is the *marginal* of the function. A stronger version of decomposability known as *structured decomposability* (Pipatsrisawat and Darwiche, 2008; Kisa et al., 2014) requires the scope of product nodes to decompose according to a *vtree*. Structured decomposability enables efficient computation of additional operations/queries, notably the product of two circuits respecting the same vtree (Pipatsrisawat and Darwiche, 2008; Shen et al., 2016).

As for support properties, a PC is *deterministic* if, for every instantiation $\boldsymbol{w}$ of the scope of a sum node, at most one of its children N_i evaluates to a non-zero value $p_{N_i}(\boldsymbol{w})$ (equivalently, the supports of the children are distinct). Determinism enables tractability of the MAP inference query, i.e. computing $\max_{\boldsymbol{Y} \setminus \boldsymbol{E}} p_N(\boldsymbol{Y} \setminus \boldsymbol{E}, \boldsymbol{e})$ for some instantiation $\boldsymbol{e}$ of a set of evidence variables $\boldsymbol{E} \subseteq \boldsymbol{Y}$.

3 A UNIFYING FRAMEWORK FOR SUPPORT PROPERTIES IN STRUCTURED DECOMPOSABLE CIRCUITS

In this section, we describe our md-vtree framework, which integrates support properties into the vtree formulation of structured decomposable circuits. Using this unifying perspective, we derive a trade-off between the generality of md-vtree circuit classes and tractability, and necessary conditions for optimality of this trade-off.

3.1 Structured Decomposability

The property of structured decomposability is defined with respect to a variable tree known as a *vtree*.

Definition 3 (Vtree)**.** *A vtree $v = (M, E)$ for a set of variables $\boldsymbol{Y}$ is a rooted binary tree with nodes M and edges E, whose leaves m each correspond to a subset $\phi(m) \subseteq \boldsymbol{Y}$, such that the subsets for all leaves form a partition of $\boldsymbol{Y}$.*

We define the *scope* of a leaf m to be $\phi(m)$, and the scope of any other node to be $\phi(m) = \cup_{m_i \in ch(m)} \phi(m_i)$. Further, we write $v_m = (M_m, E_m)$ to denote the vtree rooted at m.

Intuitively, a vtree specifies how the scope of product nodes decompose in a circuit. However, our definition of structured decomposability differs from typical recursive definitions (Pipatsrisawat and Darwiche, 2008; Darwiche, 2011; Shih et al., 2019) in that the key condition is on the scopes of the sum/leaf nodes, without directly placing conditions on the product nodes (besides decomposability):

Definition 4 (PC respecting vtree)**.** *Let C be a PC and $v = (M, E)$ be a vtree, both over variables $\boldsymbol{Y}$. We say that C respects v if (1) C is smooth[3] and decomposable; and (2) for every leaf node and non-trivial[4] sum node $N \in C$, there exists a vtree node $m \in M$ such that $\phi(N) = \phi(m)$.*

Definition 5 (Structured Decomposability)**.** *A PC C is structured decomposable if it respects some vtree v.*

Structured decomposability enables tractable products of circuits respecting compatible vtrees, i.e. those which have the same structure when projected onto their common variables $\boldsymbol{C} := \boldsymbol{Y}^{(1)} \cap \boldsymbol{Y}^{(2)}$ (see Appendix for details). In the next subsection, we will show how support properties, which are also specified on the sum nodes in the circuit, can be neatly integrated into vtrees.

3.2 Structured Marginal Determinism

Our new definition of structured decomposability based on sum nodes provides the basis for us to specify a novel systematic characterization of support properties in structured decomposable circuits, which we call *structured marginal determinism*. First, we reformulate the definitions of marginal determinism from Choi et al. (2020).

Definition 6 (Restricted Scope)**.** *For a PC node N (resp. vtree node m), and given a set $\boldsymbol{C} \subseteq \boldsymbol{Y}$, the restricted scope is defined as $\phi_{\boldsymbol{C}}(N) := \phi(N) \cap \boldsymbol{C}$ (resp. $\phi_{\boldsymbol{C}}(m) := \phi(m) \cap \boldsymbol{C}$).*

Definition 7 (Marginalized Support)**.** *For any PC node N and subset of variables $\boldsymbol{Q} \subseteq \boldsymbol{Y}$, we define the marginalized support of N with respect to $\boldsymbol{Q}$ as* $\text{supp}_{\boldsymbol{Q}}(N) := \{\boldsymbol{q} \in val(\boldsymbol{Q}) : p_N(\boldsymbol{q}) > 0\}$.

Note that $\boldsymbol{Q}$ can contain variables outside of $\phi(N)$; in a slight abuse of notation, we write $p_N(\boldsymbol{q})$ for $p_N(\boldsymbol{q} \cap \phi(N))$.

Definition 8 (Marginal Determinism)**.** *A sum node is marginal deterministic with respect to a subset $\boldsymbol{Q} \subseteq \boldsymbol{Y}$ (written $\boldsymbol{Q}$-deterministic) if the children of the sum node have distinct marginalized support, i.e.* $\text{supp}_{\boldsymbol{Q}}(N_i) \cap \text{supp}_{\boldsymbol{Q}}(N_j) = \emptyset$ *for N_i, N_j distinct children of T.*

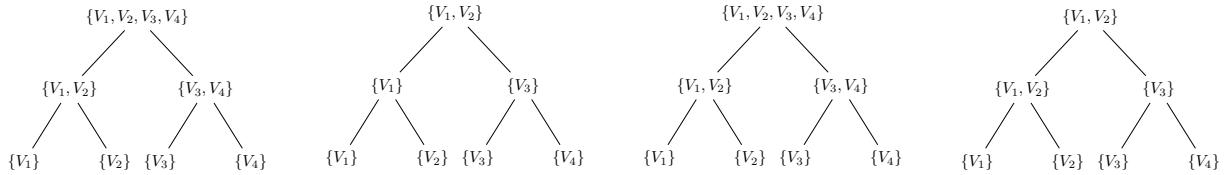
Definition 9 (Marginal Determinism of PC)**.** *A PC is marginal deterministic with respect to a subset $\boldsymbol{Q} \subseteq \boldsymbol{Y}$ (written $\boldsymbol{Q}$-deterministic) if for every sum node T, either:*

- *$\phi(T)$ does not overlap with $\boldsymbol{Q}$, i.e. $\phi_{\boldsymbol{Q}}(T) = \emptyset$; or*
- *The sum node T is $\boldsymbol{Q}$-deterministic.*

For example, normal determinism is equivalent to C being marginal deterministic with respect to $\boldsymbol{Y}$. In general, there is no straightforward relation between $\boldsymbol{Q}$-determinism and $\boldsymbol{Q}'$-determinism for different sets $\boldsymbol{Q}, \boldsymbol{Q}'$. In particular, neither determinism (i.e. $\boldsymbol{Y}$-determinism) nor $\boldsymbol{Q}$-determinism

[3] A structured decomposable circuit can be smoothed in near-linear time (Shih et al., 2019).

[4] A sum node is non-trivial iff it has more than one child.



(a) vtree with scope function ϕ (b) vtree with label $\psi^{(psdd)}$ (c) vtree with label $\psi^{(det)}$ (d) Optimal labels for PSDD

Figure 1: Example of md-vtree with scope function, and three different labelling functions.

imply each other in general; for example, a circuit can be $\boldsymbol{Q}$-deterministic but not deterministic if there exist some sum nodes with $\phi_{\boldsymbol{Q}}(T) = \emptyset$. Thus, we use $\mathcal{Q}(C)$ to denote the set of all sets $\boldsymbol{Q} \subseteq \boldsymbol{V}$ such that C is $\boldsymbol{Q}$-deterministic; this provides a characterization of the support properties of the circuit.

Now, for a given PC C, and any sum node T in that PC, let $\psi(T)$ be the set of all sets $\boldsymbol{Q}$ such that T is $\boldsymbol{Q}$-deterministic; we call this a *labelling function*. Note that the label function ψ is a *specification* of marginal determinism for the circuit; that is, it is sufficient to deduce $\mathcal{Q}(C)$. We make two observations that allow us to simplify the labelling function, one straightforward, and one more subtle. Firstly, we note that $\boldsymbol{Q}$-determinism for the circuit imposes the same requirement on all nodes with the same scope; thus we restrict ψ to have the same value for all sum nodes with the same scope. For structured decomposable circuits, we can thus write $\psi(m)$ as a function of the vtree node m.

The second observation is that, under some assumptions, we can actually specify $\psi(m)$ using a *single* set $\boldsymbol{Q} \subseteq \boldsymbol{V}$.

Proposition 1 (Conflicting $\boldsymbol{Q}$-Determinisms for Sum Nodes)**.** *Let C be a PC, and let $\boldsymbol{Q}, \boldsymbol{Q}' \subseteq \boldsymbol{V}$ such that neither is a subset of the other. Suppose that there exists a non-trivial sum node T in C that is $\boldsymbol{Q}$-deterministic and $\boldsymbol{Q}'$-deterministic, but not $(\boldsymbol{Q} \cap \boldsymbol{Q}')$-deterministic. Then C cannot have full support, i.e.* $\text{supp}(R) \subset val(\boldsymbol{V})$.

Proposition 1 says that, if we want $\psi(m)$ to contain two sets $\boldsymbol{Q}, \boldsymbol{Q}'$ which are not subsets of each other, then this necessarily restricts the support of the circuit. While it can be beneficial to enforce a restricted support on a PC if we have prior knowledge (Kisa et al., 2014), it is undesirable in our case where restricting support comes as a *side effect* of enforcing tractability, as this can result in bias when learning. As such, we only consider labellings $\psi(m)$ where, for every $\boldsymbol{Q}, \boldsymbol{Q}' \in \psi(m)$, we have $\boldsymbol{Q} \subseteq \boldsymbol{Q}'$ or $\boldsymbol{Q}' \subseteq \boldsymbol{Q}$.

Proposition 2 (Superset $\boldsymbol{Q}$-Determinisms for Sum Nodes)**.** *Suppose that a sum node T is $\boldsymbol{Q}$-deterministic. Then it is also $\boldsymbol{Q}'$-deterministic for any $\boldsymbol{Q} \subseteq \boldsymbol{Q}' \subseteq \boldsymbol{V}$.*

Using Proposition 2, it now follows that $\psi(m)$ must take the form $\{\boldsymbol{Q}' | \boldsymbol{Q} \subseteq \boldsymbol{Q}' \subseteq \boldsymbol{V}\}$ for some $\boldsymbol{Q}$. As a result, we can just label our vtree node m with $\boldsymbol{Q}$, i.e. $\psi(m) = \boldsymbol{Q}$.

This motivates our characterization of structural marginal determinism based on the concept of a *md-vtree*, which provides a means of specifying the support properties that a structured decomposable circuit satisfies.

Definition 10 (md-vtree)**.** *A md-vtree $w = (v, \psi)$ for a set of variables $\boldsymbol{V}$ consists of a vtree $v = (M, E)$ over $\boldsymbol{V}$, together with a labelling function ψ.*

The labelling function maps a vtree node $m \in M$ to some element in $\mathcal{P}(\phi(m)) \cup \{U\}$, where U is the universal set.[5]

Definition 11 (PC respecting md-vtree)**.** *Let C be a PC and $w = (v, \psi)$ be a md-vtree, both over variables $\boldsymbol{V}$. Then we say that C respects w if 1) C respects v; and 2) for any sum unit $T \in C$, T is marginally deterministic with respect to $\psi(m)$, where m is the vtree node such that $\phi(T) = \phi(m)$.*

We denote the class of circuits respecting w by $\mathcal{C}_w$.

Intuitively, md-vtrees capture both structured decomposability, as well as a marginal determinism "pattern" that the circuit must follow:

Definition 12 (Implied $\boldsymbol{Q}$-Determinisms)**.** *For any set $\boldsymbol{Q} \subseteq \boldsymbol{V}$, we say that $\boldsymbol{Q}$-determinism is implied by a md-vtree w if, for every vtree node $m \in M$ such that $\phi(m) \cap \boldsymbol{Q} \neq \emptyset$, it is the case that $\boldsymbol{Q} \supseteq \psi(m)$. We write $\mathcal{Q}(w)$ to denote the set of all sets $\boldsymbol{Q}$ s.t. $\boldsymbol{Q}$-determinism is implied by w.*

Proposition 3 (Validity of Implied $\boldsymbol{Q}$-Determinisms)**.** *For any PC C respecting md-vtree w, both over $\boldsymbol{V}$, and any $\boldsymbol{Q} \subseteq \boldsymbol{V}$ s.t. w implies $\boldsymbol{Q}$-determinism, it follows that C is $\boldsymbol{Q}$-deterministic.*

Notice that there is a trade-off between generality (expressivity) of the PC class, and the support properties it supports. Increasing the size of the labelling sets will improve the former, but hurt the latter.

Theorem 1 (Generality-Tractability Tradeoff)**.** *Let $w = (v, \psi)$ and $w' = (v, \psi')$ be two md-vtrees, such that $\psi'(m) \supseteq \psi(m)$ for all $m \in M$. Then we have that $\mathcal{Q}(w) \supseteq \mathcal{Q}(w')$, and $\mathcal{C}_w \subseteq \mathcal{C}_{w'}$.*

It is worth commenting on the two extremes of possible labels; namely, the universal set, and the empty set. The role of the universal set label U is to indicate that no $\boldsymbol{Q}$-determinism properties hold (including normal determinism). On the other hand, the empty set indicates that any

[5]The universal set satisfies, for any set S, $U \supseteq S$, $U \not\subseteq S$ (unless S is U), $U \cap S = S$, and $U \cup S = U$.

sum node T corresponding to m can only have one child N_i which is not zero, i.e. $p_{N_i} \equiv 0$; thus the sum node must be trivial. In practice, this means that it must represent a factorized distribution with factors corresponding to the scopes of the children of m.

Examples To the best of our knowledge, the only concrete type of probabilistic circuit proposed in the literature that imposes non-trivial marginal determinism constraints (i.e., not just normal determinism) is the probabilistic sentential decision diagram (PSDD) (Kisa et al., 2014). PSDDs satisfy structured decomposability and a property known as strong determinism. In the language of md-vtrees, this corresponds to requiring that, for any non-leaf vtree node m, and children m_1, m_2 of m, the label $\psi(m)$ is either $\phi(m_1)$ or $\phi(m_2)$ (which is then referred to as the *left* child). For example, for the vtree over $\boldsymbol{V} = \{V_1, V_2, V_3, V_4\}$, shown in Figure 1a together with the scopes for each vtree node, the label function $\psi^{(psdd)}$ is given on the right in Figure 1b.

Despite implementing strong determinism, recent work has shown that almost all of the tractable queries and operations that PSDDs support require only structured decomposability and determinism (Dang et al., 2020). This raises the question of whether strong determinism adds anything. To analyse this, in Figure 1c we show the labelling function $\psi^{(\text{det})}$ which defines a deterministic circuit. With these representations, we can deduce the $\boldsymbol{Q}$-determinism properties that any PSDD, or structured decomposable and deterministic circuit, must satisfy, by finding the set $\mathcal{Q}(w)$ of sets $\boldsymbol{Q}$ which its md-vtree implies. In this example, by enumerating all sets $\boldsymbol{Q} \subseteq \boldsymbol{V}$ and checking the condition, we can see that $\mathcal{Q}(w^{\text{psdd}}) = \{\{V_1, V_2\}, \{V_1, V_2, V_3\}, \{V_1, V_2, V_3, V_4\}\}$, while $\mathcal{Q}(v^{\text{det}}) = \{\{V_1, V_2, V_3, V_4\}\}$. This shows that PSDDs do have additional $\boldsymbol{Q}$-determinisms, which means, for example, that they are more tractable with regards to MMAP queries. In fact, in the particular case of (P)SDDs, the implied $\boldsymbol{Q}$-determinisms $\mathcal{Q}(w^{\text{psdd}})$ coincide with the definition of $\boldsymbol{Q}$-constrained vtrees (Oztok et al., 2016).

3.3 Regular md-vtrees

Given the trade-off between expressivity and tractability for md-vtrees, one might ask how to choose the labelling function in practice. One reasonable strategy would be to enforce some marginal determinism properties that we require for tractability of some inference task, and optimize for expressivity within this constraint.

Problem 1 (Labelling Selection)**.** *Given a vtree v, and a set $\boldsymbol{S}$ of subsets $\boldsymbol{Q} \subseteq \boldsymbol{V}$, choose a labelling function ψ such that $w = (v, \psi)$ implies $\boldsymbol{Q}$-determinism for all $\boldsymbol{Q} \in \boldsymbol{S}$, i.e. $\mathcal{Q}(w) \supseteq \boldsymbol{S}$, while maximizing expressivity.*

We propose a simple algorithm to tackle this problem (Algorithm 1), which directly enforces the necessary labels for each vtree nodes; it can be seen that it is optimally expressive, in the sense that increasing the size of any labelling set will result in some losing $\boldsymbol{Q}$-determinism for some $\boldsymbol{Q} \in \boldsymbol{S}$. It turns out that such labelling functions have a very specific structure:

Algorithm 1: Optimal Labelling

Input: vtree $v = (M, E)$, required set of marginal determinisms $\boldsymbol{S}$
Result: labelling function ψ for v

```
for m ∈ M do
    ψ(m) ← U
    for Q ∈ S do
        if Q ∩ φ(m) ≠ ∅ then
            ψ(m) ← ψ(m) ∩ Q
    if ψ(m) ≠ U then
        ψ(m) ← ψ(m) ∩ φ(m)
Return w = (v, ψ)
```

Definition 13 (Regular md-vtree)**.** *We say that a md-vtree $w = (v, \psi)$ is regular if for every non-leaf node m, and its children m_1, m_2, it holds that either $\psi(m) = \psi(m_1)$, $\psi(m) = \psi(m_2)$, or $\psi(m) = \psi(m_1) \cup \psi(m_2)$.*

Proposition 4 (Regularity of Algorithm 1)**.** *The output of Algorithm 1 is a regular md-vtree.*

The following Theorem shows that regular md-vtrees are optimal in the sense that for any given marginal determinism requirement $\boldsymbol{S}$, (one of) the most expressive md-vtree is always a regular md-vtree. This means that we can restrict our attention to the much smaller space of regular md-vtrees. In particular, the labelling function of a regular md-vtree is entirely determined by the labelling of each leaf node $\psi(m_{\text{leaf}})$, and a ternary variable over values $\{f, s, b\}$ for each non-leaf node, indicating whether the label depends on the label of the first child, second child, or both.

Theorem 2 (Optimal md-vtrees)**.** *Let $w = (v, \psi)$ be any md-vtree. Then there exists a regular md-vtree $w' = (v, \psi')$ such that $\mathcal{Q}(w) = \mathcal{Q}(w')$, and $\mathcal{C}_{w'} \supseteq \mathcal{C}_w$.*

While Algorithm 1 always returns an optimal labelling for a given vtree satisfying the required marginal determinisms, the expressivity of the circuit class may differ depending on the vtree. For example, for some vtrees, Algorithm 1 may output a labelling function such that $\psi(m)$ is empty for some vtree nodes m, i.e. a factorized distribution. We leave designing optimally expressive vtrees for a given set $\boldsymbol{S}$ of marginal determinisms as an open problem.

Examples Let us return to the PSDD example from Figure 1. It can be easily checked that the corresponding md-vtree w^{psdd} in Figure 1b is not regular. Following Algorithm 1, we therefore construct in Figure 1d an regular md-vtree w^{opt} that retains the same $\mathcal{Q}(w^{\text{opt}}) = \mathcal{Q}(w^{\text{psdd}}) =$

$\{\{V_1, V_2\}, \{V_1, V_2, V_3\}, \{V_1, V_2, V_3, V_4\}\}$. We should prefer w^{opt} over w^{psdd} as it imposes less constraints/more circuits respect w^{opt}. In other words, PSDDs impose more constraints than they "need to" to obtain their marginal determinism properties.

4 MDNETS: A PRACTICAL ARCHITECTURE FOR MD-VTREES

In this section, we show how to construct and learn a probabilistic circuit that respects a particular md-vtree. It is worth noting that, as special cases of md-vtrees, we can use existing architectures and learning algorithms for PCs such as PSDDs (Liang et al., 2017) and structured decomposable and deterministic circuits (Dang et al., 2020; Di Mauro et al., 2021). However, we have seen that PSDDs are not optimally expressive, and to enforce tractability, we may need to target md-vtrees which do not fall into these existing categories, such as those generated from Algorithm 1. We thus propose a novel PC architecture, MDNet, which enforces a given regular md-vtree *by design*.

The key component of MDNets is the *node group*, which is a vector of sum nodes with the same scope (i.e. corresponding to the same vtree node m) with the property that the nodes in the group have disjoint marginalized support $\text{supp}_{\psi(m)}(N)$. Intuitively, the sum nodes in a group provide a partition of the domain of $\psi(m)$, which we use as an invariant in order to enforce the required marginal determinisms throughout the circuit. More formally, suppose that we have a non-leaf vtree node m, and let m_l, m_r be its children. We refer to all sum nodes corresponding to a vtree node as being a *layer*, and assign G groups $\boldsymbol{T}_1, ...\boldsymbol{T}_G$ to the layer for vtree node m, and similarly $\boldsymbol{T}_1^{(l)}, ...\boldsymbol{T}_{G_l}^{(l)}$, $\boldsymbol{T}_1^{(r)}, ...\boldsymbol{T}_{G_r}^{(r)}$ to the layers for m_l, m_r. For regular md-vtrees, the label $\psi(m)$ is either equal to $\psi(m_l)$ or $\psi(m_r)$, or is their union. We handle these cases separately, as *mixing* and *synthesizing* layers.

Mixing Layer If $\psi(m) = \psi(m_l)$ or $\psi(m_r)$, we implement a *mixing* layer. W.l.o.g. we assume $\psi(m) = \psi(m_l)$. For each group $\boldsymbol{T}_i = (T_{i,1}, ..., T_{i,k})$, and for each sum node $T_{i,k}$ in the group, we assign a set of product nodes $\boldsymbol{P}_{i,k}$ to $T_{i,k}$. Each product node has two children; the left child being a node from $\boldsymbol{T}_1^{(l)}, ...\boldsymbol{T}_{G_l}^{(l)}$ and the right child being a node from $\boldsymbol{T}_1^{(r)}, ...\boldsymbol{T}_{G_r}^{(r)}$. We place the following restriction: the left children of the product nodes $\cup_k \boldsymbol{P}_{i,k}$ must all be distinct, and all come from a single group. This ensures that all of the sum nodes are $\psi(m) = \psi(m_l)$-deterministic, and further that each group $\boldsymbol{T}_i$ satisfies the invariant.

Synthesizing Layer If $\psi(m) = \psi(m_l) \cup \psi(m_r)$, we assign product nodes $\boldsymbol{P}_{i,k}$ to each sum node $T_{i,k}$ as before, but with a different restriction; we now require that both the left children and right children of the product nodes $\cup_k \boldsymbol{P}_{i,k}$ are nodes coming from a single group from their respective layers, and that each product node has a unique combination of children from these groups.

Operation	Requirements	Output Encodes
$\texttt{MARG}(C; \boldsymbol{W})$	-	$\sum_{\boldsymbol{W}} p_C(\boldsymbol{V})$
$\texttt{INST}(C; \boldsymbol{w})$	-	$p_C(\boldsymbol{w}, \boldsymbol{V} \setminus \boldsymbol{W})$
$\texttt{PROD}(C_1, C_2)$	Cmp. Vtrees	$p_{C_1}(\boldsymbol{V}) \times p_{C_2}(\boldsymbol{V})$
$\texttt{POW}(C; \alpha)$	Det	$p_C(\boldsymbol{V})^{\alpha}\vert_{\text{supp}(C)}$
$\texttt{MAX}(C)$	Det	$\max_{\boldsymbol{V}} p_C(\boldsymbol{V})$
$\texttt{LOG}(C)$	Det	$\log p_C(\boldsymbol{V})\vert_{\text{supp}(C)}$

Table 1: Definitions of basic operations.

Intuitively, mixing layers achieve their marginal determinism by "copying" the marginal determinism of one of their child layers, while mixing over groups in the other child layer. On the other hand, synthesizing layers enforce marginal determinism by combining, or synthesizing, the marginal determinism properties of both of their children.

For simplicity, we propose to learn MDNets exploiting recent advancements in random structures for PC learning (Peharz et al., 2020a,b; Di Mauro et al., 2021): in particular, we propose to choose the MDNet structure randomly within the constraints, and then learn the parameters using standard MLE estimation if the md-vtree implies ($\boldsymbol{V}$-)determinism, or use EM otherwise (Peharz, 2015).

5 COMPOSITIONAL INFERENCE USING STRUCTURED MARGINAL DETERMINISM

In this section, we will describe a methodology that exploits our md-vtree framework as a language for deriving tractability conditions for arbitrary compositions of basic operations on probabilistic circuits. In particular, we build upon the work of Vergari et al. (2021), showing how to extend their analysis to compositional queries which include marginalization (integration) operations at arbitrary points in the pipeline, and allow for maximization queries (e.g. marginal MAP).

5.1 Support Properties in Compositional Inference

In Table 1, we define the basic probabilistic inference operations, including marginalization, products, instantiation, powers, maximization, and logarithms, along with the properties (*requirements*) under which there exist efficient (polytime) algorithms for computing them on PCs (Vergari et al., 2021); note that we assume decomposability and smoothness by default. These operations produce a circuit encoding the specified function (or scalar in the

case of MAX). We use $|_{\text{supp}(C)}$ to denote the *restricted* power/logarithm, as these functions are not defined at 0:

$$f(p_C(\boldsymbol{V}))|_{\text{supp}(C)} := \begin{cases} f(p_C(\boldsymbol{V})) & \text{if } p_C(\boldsymbol{V}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

We refer to the basic operations in the bottom half of the table as *deterministic* operations, as their tractability depends on the input circuit being deterministic; they are NP-complete (MAX) or #P-hard (POW, LOG) otherwise (Choi and Darwiche, 2017; Vergari et al., 2021).

Many complex inference queries can be expressed as compositions of these basic operations; we show a selection of examples in Table 2. We can interpret such compositions as *pipelines*, or computational graphs, which specify an algorithm for computing the query that uses the efficient algorithms for each basic operation. To show tractability of a pipeline for given input circuits, one needs to show that the inputs to intermediate operations satisfy the requirement for tractability of that operation. For this purpose, Vergari et al. (2021) derive *input-output conditions* for basic operations, which specify a pair of properties such that the output of the operation is guaranteed to satisfy the output property if the input satisfies the input property. This allows properties to be soundly propagated through the pipeline, from the input circuits.

However, there remains a significant unresolved challenge for analyzing general compositions of basic operations; namely, analyzing how operations affect *support properties* of the circuit beyond just determinism. For example, the marginal MAP problem (MMAP) (Huang et al., 2006; Choi et al., 2020) in Table 2 is a canonical inference task that can be decomposed into a composition of a MARG operation $\sum_{\boldsymbol{V} \setminus \boldsymbol{W}} p_C(\boldsymbol{W}, \boldsymbol{V} \setminus \boldsymbol{W})$, and a MAX operation $\max_{\boldsymbol{w}} p_C(\boldsymbol{W})$. The MAX operation $\max_{\boldsymbol{w}} p_C(\boldsymbol{W})$ is known to be tractable for any deterministic input circuit. Unfortunately, however, ensuring that the output of $\texttt{MARG}(C; \boldsymbol{V} \setminus \boldsymbol{W})$ is deterministic is known to be NP-hard (Shen et al., 2016), even if C is deterministic. For similar reasons, any compositional inference task in which a deterministic operation appears after a marginalization operation cannot be currently analyzed, notable examples of which we show in the lower half of Table 2.

5.2 Operations on md-vtrees

To tackle these challenges, we apply our md-vtree framework as a unified language for scope and support in structured decomposable circuits. The first problem that we would like to address is that of determining whether a pipeline is tractable for given input md-vtrees:

Problem 2 (Forward Problem)**.** *Given a query expressed as a pipeline of basic operations, and md-vtree(s) that the input circuits respect, determine if the pipeline is tractable.*

To solve this problem, we propose to derive input-output conditions in terms of md-vtrees. In other words, if we have input circuit(s) that respect some given md-vtree(s), can we obtain a md-vtree that the output of a basic operation is guaranteed to respect? In the Appendix, we detail a set of algorithms for each of the operations in Table 1, which take as input md-vtree(s) and return an output md-vtree that provides exactly such a guarantee, based upon the corresponding algorithms on circuits. Then, given any pipeline, and input md-vtree(s), we can determine if the compositional query is tractable, simply by propagating md-vtree(s) *forward* through the pipeline, and checking that the input md-vtree(s) to any intermediate operation satisfy the requirements in Table 1. Importantly, this can be done *without doing the computation of the query pipeline itself*; all of our algorithms run in polytime in the number of variables $|\boldsymbol{V}|$, which is much smaller than the circuits themselves.

5.3 The MD-calculus

The forward problem allows us to reason about tractability if we already have the input circuits. However, when learning circuits from data, we have the freedom to choose the md-vtree(s) in order to enable tractable inference:

Problem 3 (Backward Problem)**.** *Given any query expressed as a pipeline of basic operations, derive input md-vtree(s) such that the pipeline is tractable.*

To this end, in Table 3 we show a set of input-output conditions called the *MD-calculus*. These are sufficient conditions on the input(s) to an operation to guarantee that the output is $\boldsymbol{Q}$-deterministic. The MD-calculus forms a set of rules that we can apply backwards from deterministic operations (which require $\boldsymbol{V}$-determinism), in order to determine a sufficient set of marginal determinisms $\boldsymbol{S}$ for each intermediate circuit. Finally, we can enforce those marginal determinisms on the input md-vtree(s) using Algorithm 1. We defer proofs of the MD-calculus rules to the Appendix.

Theorem 3 (MD-calculus)**.** *The conditions in Table 3 hold.*

We show examples of compositional queries in the bottom half of Table 2, with the marginal determinism condition on the input circuit C derived using this approach. For MMAP, for the MAX operation, we require $\texttt{MARG}(C; \boldsymbol{V} \setminus \boldsymbol{W})$ to be $\boldsymbol{W}$-deterministic. Using the MD-calculus rule for MARG, we can see that it is sufficient for C to also be $\boldsymbol{W}$-deterministic. For mutual information, applying a similar approach we obtain that C should be both $\boldsymbol{X}$-deterministic and $\boldsymbol{Y}$-deterministic, but we have seen in Proposition 1 that this is not possible without restricting support.

6 APPLICATION: CAUSAL INFERENCE

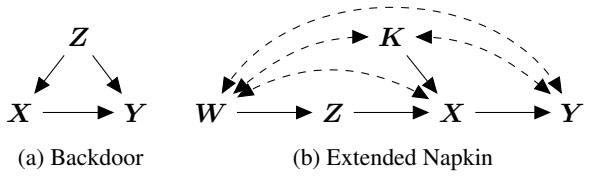
In this section, we use our md-vtree framework to analyse tractability conditions for *exact* causal inference for

Task	Computation	Operations	Condition
Cross-Entropy	$-\sum_{\boldsymbol{V}} p_{C^{(1)}}(\boldsymbol{V}) \log(p_{C^{(2)}}(\boldsymbol{V}))$	MARG, PROD, LOG	$C^{(1)}, C^{(2)}$ Cmp.; $C^{(2)}$ Det.
Var. Elim.	$\sum_{\boldsymbol{W}} p_{C^{(1)}}(\boldsymbol{V}) p_{C^{(2)}}(\boldsymbol{V})$	MARG, PROD	$C^{(1)}, C^{(2)}$ Cmp.
MMAP	$\max_{\boldsymbol{w}} \sum_{\boldsymbol{V} \setminus \boldsymbol{W}} p_C(\boldsymbol{w}, \boldsymbol{V} \setminus \boldsymbol{W})$	MARG, MAX	C Mdet. wrt. $\boldsymbol{W}$
Mut. Inf.	$p_C(\boldsymbol{X}, \boldsymbol{Y}) \log \frac{p_C(\boldsymbol{X}, \boldsymbol{Y})}{p_C(\boldsymbol{X}) p_C(\boldsymbol{Y})}$	MARG, PROD, LOG, POW	-
BD Adj.	$\sum_{\boldsymbol{Z}} p_C(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{Z}) p_C(\boldsymbol{Z})$	MARG, PROD, POW	C Mdet. wrt. $\boldsymbol{X} \cup \boldsymbol{Z}$; Str. Dec.

Table 2: Examples of complex inference tasks expressed as compositions of basic operations.

Operation	Requirement	Input Condition	Output Condition
$\texttt{MARG}(C; \boldsymbol{W})$	-	$\boldsymbol{Q}$-det	$\boldsymbol{Q}$-det
$\texttt{INST}(C; \boldsymbol{w})$	-	$\exists \boldsymbol{W}' \subseteq \boldsymbol{W} : (\boldsymbol{Q} \cup \boldsymbol{W}')$-det	$\boldsymbol{Q}$-det
$\texttt{PROD}(C^{(1)}, C^{(2)})$	$C^{(1)}, C^{(2)}$ respect compatible vtrees	$\exists \boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)} : \boldsymbol{Q}^{(1)}$-det, $\boldsymbol{Q}^{(2)}$-det, and: • Either (a) $\boldsymbol{Q} \subseteq \boldsymbol{V}^{(1)} \cap \boldsymbol{V}^{(2)}$ and $\boldsymbol{Q}^{(1)} = \boldsymbol{Q}^{(2)} = \boldsymbol{Q}$; • Or (b) $\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)} \supseteq \boldsymbol{V}^{(1)} \cap \boldsymbol{V}^{(2)}$ and $\boldsymbol{Q} = \boldsymbol{Q}^{(1)} \cup \boldsymbol{Q}^{(2)}$	$\boldsymbol{Q}$-det
$\texttt{POW}(C; \alpha)$	Det	$\boldsymbol{Q}$-det	$\boldsymbol{Q}$-det
$\texttt{MAX}(C)$	Det	N/A	N/A (scalar output)
$\texttt{LOG}(C)$	Det	-	-

Table 3: MD-calculus: sufficient input-output conditions for each basic operation



(a) Backdoor (b) Extended Napkin

Figure 2: Examples of causal diagrams

PCs. The typical setup in causal inference is that we have access to an observed distribution $p(\boldsymbol{V})$, and some domain assumptions, often conveniently expressed using a *causal diagram* (Pearl, 2009), and we are interested in computing some *interventional distribution* $p_{\boldsymbol{X}}(\boldsymbol{Y})$, where $\boldsymbol{X}, \boldsymbol{Y} \subseteq \boldsymbol{V}$ are disjoint subsets of the observed variables. $p_{\boldsymbol{X}}(\boldsymbol{Y})$ is said to be *identifiable* if the assumptions are sufficient for it to be uniquely determined; and if it is identifiable, a causal formula/estimand can be obtained using the *do-calculus* (Pearl, 1995; Shpitser and Pearl, 2006).

6.1 Hardness of Backdoor Adjustment on Circuits

We assume that the observed data distribution is modelled by a PC C (perhaps learned from data), and we wish to compute some causal query. One of the most common cases where the interventional distribution is identifiable is when there exists a valid *backdoor adjustment set* $\boldsymbol{Z} \subseteq \boldsymbol{V} \setminus (\boldsymbol{X} \cup \boldsymbol{Y})$ (also known as the conditional exchangability/ignorability assumption), an example of which is illustrated in Figure 2a. Whenever such a set exists, the interventional distribution $p_{C,\boldsymbol{X}}(\boldsymbol{Y})$ is given by the backdoor adjustment formula[6]:

$$\sum_{\boldsymbol{Z}} p_C(\boldsymbol{Z}) p_C(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{Z}) = \sum_{\boldsymbol{Z}} p_C(\boldsymbol{Z}) \frac{p_C(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{Z})}{p_C(\boldsymbol{X}, \boldsymbol{Z})}$$

Unfortunately, we now show that the backdoor adjustment is not tractable for existing classes of probabilistic circuits:

Theorem 4 (Hardness of Backdoor Query)**.** *The backdoor query for decomposable and smooth PCs is* #P*-hard, even if the PC is structured decomposable and deterministic.*

We can also view this result as placing a theoretical barrier on interpreting probabilistic circuits as causal models (Zhao et al., 2015; Papantonis and Belle, 2020); namely that, even if such an interpretation exists, it is not possible to tractably perform causal inference with the causal PC. Thus, whether we interpret a PC as itself expressing causality, or merely a model of the observational probability distribution, new PC classes are required. In the following, we take the latter perspective, employing our md-vtrees.

6.2 MD-calculus for Causal Formulae

We now employ the MD-calculus to derive tractability conditions for the backdoor query, given $C(\boldsymbol{V})$ as input. Note that there is only one deterministic operation, namely the reciprocal $\texttt{POW}(\cdot; -1)$. The input to this operation is

[6] As commonly done in causal inference we assume positivity, i.e. $p_C(\boldsymbol{V}) \geq 0$ such that the conditional is well defined.

$C(\boldsymbol{X}, \boldsymbol{Z})$, which we thus require to be deterministic, that is, $(\boldsymbol{X} \cup \boldsymbol{Z})$-deterministic. We then have that $C(\boldsymbol{X}, \boldsymbol{Z}) = \texttt{MARG}(C(\boldsymbol{V}); \boldsymbol{V} \setminus (\boldsymbol{X} \cup \boldsymbol{Z}))$, so, using the MD-calculus rule for MARG, it is sufficient for $C(\boldsymbol{V})$ to be $(\boldsymbol{X} \cup \boldsymbol{Z})$-deterministic. Given a vtree, we can then use Algorithm 1 to derive an md-vtree for the input circuit $C(\boldsymbol{V})$.

It is worth noting that, unlike typical causal inference approaches which derive a scalar $p_{C,\boldsymbol{x}}(\boldsymbol{y})$ (or some causal effect) for a particular intervention $\boldsymbol{X} = \boldsymbol{x}$, the output of the backdoor query here is a *circuit* over variables $\boldsymbol{X}, \boldsymbol{Y}$ encoding $p_{C,\boldsymbol{X}}(\boldsymbol{Y})$. This means that we can do further downstream reasoning over different values of $\boldsymbol{X}, \boldsymbol{Y}$.

As a second example, consider the *extended napkin* causal diagram in Figure 2b. We can use the do-calculus to derive the following expression for $p_{C,\boldsymbol{X}}(\boldsymbol{Y})$ in this case [7]:

$$\sum_{\boldsymbol{K}} f(C, \boldsymbol{K}) \frac{\sum_{\boldsymbol{W}} p_C(\boldsymbol{X}, \boldsymbol{Y} | \boldsymbol{K}, \boldsymbol{z}, \boldsymbol{W}) p_C(\boldsymbol{W}, \boldsymbol{K})}{\sum_{\boldsymbol{W}} p_C(\boldsymbol{X} | \boldsymbol{K}, \boldsymbol{z}, \boldsymbol{W}) p_C(\boldsymbol{W}, \boldsymbol{K})}$$

For this formula, we can try to apply the MD-calculus as usual. The denominator (input to the $\texttt{POW}(\cdot; -1)$ operation) is a function of $\{\boldsymbol{X}, \boldsymbol{K}\}$; thus, passing through the marginalization operation, we require $\texttt{PROD}(C(\boldsymbol{X}|\boldsymbol{K}, \boldsymbol{z}, \boldsymbol{W}), C(\boldsymbol{W}, \boldsymbol{K}))$ to be $(\boldsymbol{X} \cup \boldsymbol{K})$-deterministic. However, if we then look at the conditions for the PROD operation, we see this cannot be achieved. In particular, $C(\boldsymbol{X}|\boldsymbol{K}, \boldsymbol{z}, \boldsymbol{W})$ and $C(\boldsymbol{W}, \boldsymbol{K})$ are circuits over $\boldsymbol{V}^{(1)} = \{\boldsymbol{X}, \boldsymbol{K}, \boldsymbol{W}\}$ and $\boldsymbol{V}^{(2)} = \{\boldsymbol{K}, \boldsymbol{W}\}$ respectively, and there are no possible $\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)}$ such that condition (a) or (b) holds.

However, we *can* derive a tractable pipeline for $p_{C,\boldsymbol{x}}(\boldsymbol{Y})$ where we commit to a specific intervention $\boldsymbol{X} = \boldsymbol{x}$. In this case, the new denominator is a function of just $\boldsymbol{K}$, and so we require $\texttt{PROD}(C(\boldsymbol{x}|\boldsymbol{K}, \boldsymbol{z}, \boldsymbol{W}), C(\boldsymbol{W}, \boldsymbol{K}))$ to be $(\boldsymbol{X} \cup \boldsymbol{K}) \setminus (\boldsymbol{X}) = \boldsymbol{K}$-deterministic. This can be achieved by choosing $\boldsymbol{Q}^{(1)} = \boldsymbol{Q}^{(2)} = \{\boldsymbol{K}\}$, which satisfies condition (a) for PROD. By analyzing the rest of the formula, we can derive a set $\boldsymbol{S}$ of marginal determinisms for $C(\boldsymbol{V})$ that is sufficient for the extended napkin query, and apply Algorithm 1; we defer the full derivations (and analysis of the frontdoor formula) to the Appendix.

7 EMPIRICAL EVALUATION

In this section, we empirically evaluate our tractable algorithm for backdoor adjustment derived using MD-calculus. We generate datasets by sampling 1000 datapoints from the (discrete variable) Bayesian network (BN) models in the *bnrepository* (Scutari, 2022), and learn a MDNet over all variables $\boldsymbol{V}$ from data. For each Bayesian network (causal graph), we select a single treatment variable X and single outcome variable Y, as well as a set of variables $\boldsymbol{Z}$ forming a valid backdoor adjustment set for (X, Y), and seek to estimate $\sum_{\boldsymbol{Z}} p(Y|X, \boldsymbol{Z}) p(\boldsymbol{Z})$. We manually select a vtree which splits the scope into $(X \cup \boldsymbol{Z})$ and Y at the root, and thereafter generates the rest of the vtree randomly. Given a vtree, we use Algorithm 1 to generate a labelling, i.e. regular md-vtree. The required tractability properties for backdoor adjustment are then enforced through the structure of the corresponding MDNet.

Dataset	$\lvert Z \rvert$	Error		Time	
		MD	Counting	MD	Counting
Asia	4	0.269	**0.0143**	**0.5**	1.0
Sachs	4	0.180	**0.0219**	1.7	**0.7**
Child	13	**0.0802**	0.135	1.9	**0.9**
Win95pts	59	**0.0044**	0.0511	3.2	**0.9**
Andes	202	**0.0382**	0.0982	7.9	**1.3**

Table 4: Backdoor Estimation (averaged over 10 runs)

The results are shown in Table 4; for comparison, we show also results for the *counting* approach, where we estimate $p(Y|X, \boldsymbol{Z})$ as $\frac{N_{Y,X,\boldsymbol{Z}}}{N_{X,\boldsymbol{Z}}}$, where N refers to the number of datapoints with the subscripted assignment of variables (this is set to 0 if $N_{X,\boldsymbol{Z}} = 0$). It can be seen that, while the counting approach is generally more robust in lower dimensions, the advantage in terms of learning a full model becomes apparent with the higher-dimensional adjustment sets, as shown by the lower error on the Win95pts and Andes datasets. Remarkably, as the size of the adjustment set $\boldsymbol{Z}$ increases, the time taken for the algorithm based on the MD-calculus increases only approximately *linearly* in the dimension. This illustrates the attractiveness of tractable probabilistic modelling, in that we can systematically control the computational cost of exact inference by restricting the size of the PC model.

8 CONCLUSION

In summary, we introduced the md-vtree framework for support properties in structured decomposable circuits, and showed how it can be employed for reasoning about tractability of compositional inference queries using our MD-calculus rules. Our unifying framework naturally provides insight into the properties of previously proposed PC classes such as PSDDs, as well as inspiring our newly designed MDNet architecture. Nonetheless, there remain a number of interesting challenges for future work. For example, for more challenging datasets, how can we also learn the vtree and/or MDNet structure from data while maintaining tractability? For what other inference queries can we derive tractability conditions using the MD-calculus? We hope that our work will help lay the theoretical foundations for tackling these questions.

[7] Here, $\boldsymbol{z}$ can be any instantiation of $\boldsymbol{Z}$, and $f(C, \boldsymbol{K})$ is a expression over $C, \boldsymbol{K}$ that we omit here for clarity.

Acknowledgements

This project was funded by the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No.834115).

References

Butz, C., S. Oliveira, J., and Peharz, R. (2020). Sum-product network decompilation. In Jaeger, M. and Nielsen, T. D., editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 53–64. PMLR.

Chen, Y. and Darwiche, A. (2022). On the definition and computation of causal treewidth. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*.

Choi, A. and Darwiche, A. (2017). On relaxing determinism in arithmetic circuits. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, page 825–833.

Choi, Y., Friedman, T., and Van den Broeck, G. (2022). Solving marginal map exactly by probabilistic circuit transformations. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Choi, Y., Vergari, A., and Van den Broeck, G. (2020). Probabilistic circuits: A unifying framework for tractable probabilistic models. *arXiv preprint*.

Dang, M., Vergari, A., and Van den Broeck, G. (2020). Strudel: Learning structured-decomposable probabilistic circuits. In Jaeger, M. and Nielsen, T. D., editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 137–148. PMLR.

Darwiche, A. (2001). On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305.

Darwiche, A. (2011). Sdd: A new canonical representation of propositional knowledge bases. In *IJCAI*.

Darwiche, A. (2021). Causal inference using tractable circuits. In *NeurIPS Workshop on Causal Inference and Machine Learning: Why Now? (WHY21)*.

Darwiche, A. and Marquis, P. (2002). A knowledge compilation map. *J. Artif. Int. Res.*, 17(1):229–264.

Di Mauro, N., Gala, G., Iannotta, M., and Basile, T. M. (2021). Random probabilistic circuits. In de Campos, C. and Maathuis, M. H., editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1682–1691. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. volume 27.

Huang, J., Chavira, M., and Darwiche, A. (2006). Solving map exactly by searching on compiled arithmetic circuits. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, page 1143–1148. AAAI Press.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kisa, D., Van den Broeck, G., Choi, A., and Darwiche, A. (2014). Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Liang, Y., Bekker, J., and Van den Broeck, G. (2017). Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Oztok, U., Choi, A., and Darwiche, A. (2016). Solving pppp-complete problems using knowledge compilation. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'16, page 94–103. AAAI Press.

Papantonis, I. and Belle, V. (2020). Interventions and counterfactuals in tractable probabilistic models: Limitations of contemporary transformations. *arXiv preprint*, abs/2001.10905.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Pearl, J. (2009). *Causality*. Cambridge University Press, Cambridge, UK, second edition.

Peharz, R. (2015). *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, Medical University of Graz.

Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Van den Broeck, G., Kersting, K., and Ghahramani, Z. (2020a). Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., and Ghahramani, Z. (2020b). Random sum-product networks: A simple and effective approach to probabilistic deep learning. In Adams, R. P. and Gogate, V., editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115

of *Proceedings of Machine Learning Research*, pages 334–344. PMLR.

Pipatsrisawat, K. and Darwiche, A. (2008). New compilation languages based on structured decomposability. In *AAAI*, pages 517–522. AAAI Press.

Pipatsrisawat, T. and Darwiche, A. (2010). A lower bound on the size of decomposable negation normal form. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Scutari, M. (2022). Bayesian network repository. `https://www.bnlearn.com/bnrepository/`.

Shen, Y., Choi, A., and Darwiche, A. (2016). Tractable operations for arithmetic circuits of probabilistic models. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29.

Shih, A., Van den Broeck, G., Beame, P., and Amarilli, A. (2019). Smoothing structured decomposable circuits. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32.

Shpitser, I. and Pearl, J. (2006). Identification of joint interventional distributions in recursive semi-markovian causal models. AAAI'06, page 1219–1226. AAAI Press.

Vergari, A., Choi, Y., Liu, A., Teso, S., and Van den Broeck, G. (2021). A compositional atlas of tractable circuit operations for probabilistic inference. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13189–13201.

Wang, B., Lyle, C., and Kwiatkowska, M. (2021). Provable guarantees on the robustness of decision rules to causal interventions. In *In Proc. International Joint Conference on Artificial Intelligence (IJCAI-21)*.

Zecevic, M., Dhami, D. S., Karanam, A., Natarajan, S., and Kersting, K. (2021). Interventional sum-product networks: Causal inference with tractable probabilistic models. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Zhao, H., Melibari, M., and Poupart, P. (2015). On the relationship between sum-product networks and bayesian networks. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 116–124, Lille, France. PMLR.