Transparency in Sum-Product Network Decompilation

Ioannis Papantonis[a;*] **and Vaishak Belle**[a,b]

[a]University of Edinburgh
[b]Alan Turing Institute
ORCiD ID: Ioannis Papantonis https://orcid.org/0000-0003-4282-5820,
Vaishak Belle https://orcid.org/0000-0001-5573-8465

Abstract. Sum-product networks guarantee that conditionals and marginals can be computed efficiently, for a wide range of models, bypassing the hardness of inference. However, this advantage comes at the expense of transparency, since it is unclear how variables interact in sum-product networks. Due to this, a series of decompilation algorithms transform sum-product networks back to Bayesian networks. In this work, we first study the transparency and causal utility of the resulting Bayesian networks. We then propose a novel decompilation algorithm to address the identified limitations.

1 Introduction

In recent years, there has been an increasing interest in studying causality-related properties in machine learning models [2, 3]. Broadly speaking [13], 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*, for example, what is the likelihood of being tall given that you play basketball? On the other hand, causal reasoning allows us to reason about *interventions*, e.g., what is the probability of a person being tall given that he/she is made to play basketball?

A fundamental challenge underlying stochastic models, however, is the intractability of inference [6]. This has led to the paradigm of *tractable probabilistic models* (TPMs), 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 [1], recent TPMs, such as sum product networks (SPNs) [17, 10] are derived from arithmetic circuits (ACs) [8], which exploit efficient function representations and also capture high tree-width models. These models can also be learnt from data [14] which leverage the efficiency of inference. The combination of these properties has allowed SPNs to find applications in critical domains, such as in healthcare [18].

Furthermore, SPNs are closely related to Bayesian networks (BNs), as any BN can be compiled into a SPN [7]. However, their internal representation makes it very challenging to identify relationships and dependencies among the variables, in contrast to BNs, where it is immediate to uncover all the connections and conditional independencies within a set of variables. This has led to the widespread view that while BNs are *transparent* models, SPNs act as *black-boxes* with no representational semantics [5]. Having said that, a series of earlier works attempt to mitigate this issue by designing decompilation algorithms that can transform SPNs back to BNs [23, 15, 4], hoping that in addition to rendering SPNs transparent, such decompilations would also enable SPNs to be used in causal inference applications. In fact, on studying the relationship between SPNs and BNs [23], the authors conclude with:

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

In this work, we revisit the problem of decompiling a SPN, seeking to assess the extent to which existing decompilation algorithms enhance transparency in SPNs. In more detail, we begin by studying the qualitative features and causal utility of the BNs generated by the methods in [23, 15, 4], showing that existing decompilations result in BNs with limited causal utility, which only marginally improve the transparency of the underlying SPNs, thus providing a negative answer to the open question in [23]. Following this finding, we study the reasons behind this limitation and based on these insights we propose a novel algorithm that is guaranteed to achieve an exact SPN to BN decompilation, for SPNs compiled from BNs using the *variable elimination reverse topological ordering* (VErto) algorithm. To our knowledge, this is the first decompilation that can accurate handle any such SPN, identifying an equivalent and transparent BN representation that faithfully uncovers a SPN's internal representations.

In what follows, an uppercase letter X denotes a Boolean random variable, while a lowercase letter x denotes an assignment to X; alternatively, X and $\neg X$ represent the events $X = 1$ and $X = 0$, respectively. Sets of variables, $\mathbf{X}$, and joint assignments, $\mathbf{x}$, are in **bold**. Finally, all proofs can be found in the supplementary material.[1]

2 Background

2.1 SPNs

SPNs over binary variables are rooted directed graphical models that provide for an efficient way of representing the network polynomial [7] of a BN [17], 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, the summation is over all possible states, and $\mathbb{1}_{x_n}$ is the indicator function. In its simplest form, the network polynomial contains 2^N terms, but there is a wide array of problems, where it is possible

* Corresponding Author. Email: i.papantonis@sms.ac.ed.uk.

[1] Link: https://github.com/GiannisPapantonis/Transparency-in-Sum-Product-Network-Decompilation.git

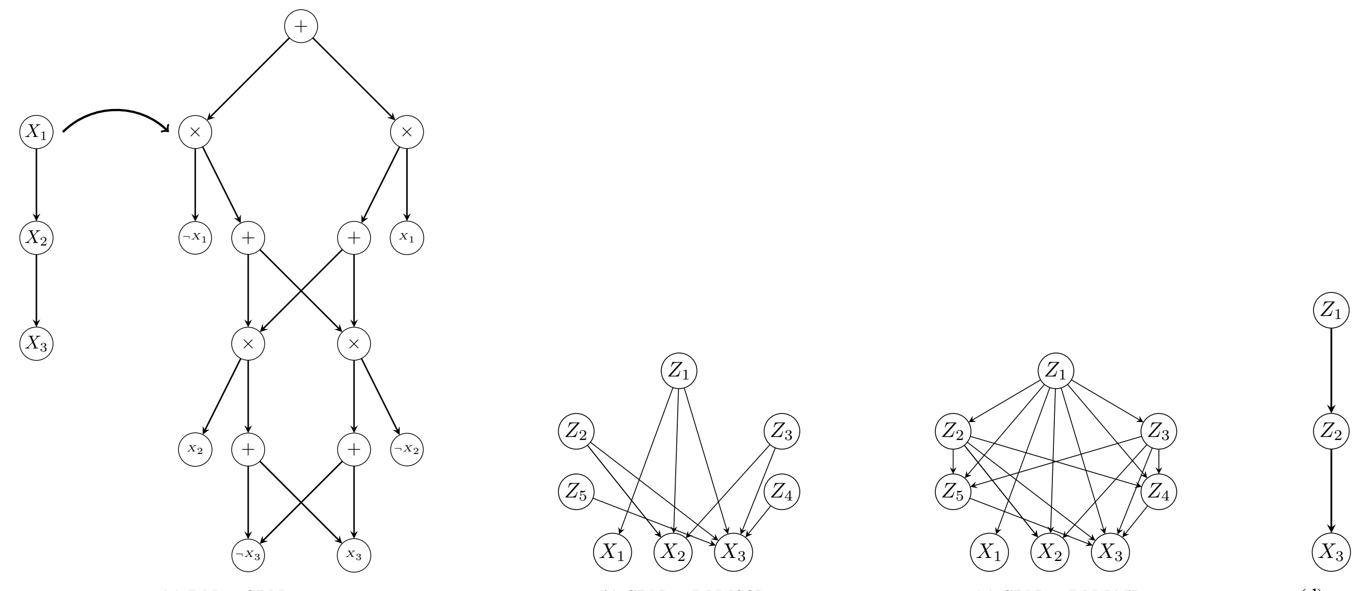

(a) BN to SPN (b) SPN to BN [23] (c) SPN to BN [15] (d) SPN to BN [4]

Figure 1: The final BNs that result from each transformation

to obtain a factorized representation, that is not exponential in the number of the model's variables, see [17] for a discussion. This is exactly the idea behind SPNs, discovering a compact factorization of the network polynomial, and thereby enabling efficient inference.

A central notion in SPNs is that of the *scope* of a node, which is defined as follows:

$$Scope(n) = \begin{cases} \{X\}, & \text{if } n \text{ is an indicator } \mathbb{1}_{X=0} \text{ or } \mathbb{1}_{X=1} \\ \cup_{c \in Ch(N)} Scope(c), & \text{otherwise} \end{cases}$$

where $Ch(n)$ denotes the children of a node, i.e. the set of all nodes that are connected to n with edges exiting from n. Using this notation, two properties that guarantee that efficient inference can be performed are completeness and decomposability:

- A sum node, n, is complete if all of its children have the same scope, meaning that $Scope(j) = Scope(k)$ for all $j, k \in Ch(n)$. In this case, the outcome of the sub-SPN rooted at node n is $S_n(\cdot) = \sum_{j \in Ch(n)} w_{nj} S_j(\cdot)$, where $\sum_j w_{nj} = 1$.
- A product node, n, is decomposable if all of its children have disjoint scopes, meaning that $Scope(j) \cap Scope(k) = \emptyset$ for all $j, k \in Ch(n)$. In this case, there is a partition of $Scope(n) = \{\mathbf{X}_1, \cdots, \mathbf{X}_{|Ch(n)|}\}$ such that we have that $S_n(\mathbf{X}_1 \cup \cdots \cup \mathbf{X}_{|Ch(n)|}) = \prod_{j \in Ch(n)} S_j(\mathbf{X}_j)$.

Putting everything together, SPNs consist of leaf nodes that correspond to tractable univariate distributions, product nodes that compute the product of their children, as well as sum nodes that compute a weighted sum of their children. Moreover, *selective* SPNs are a recently introduced extension [14], where determinism is imposed, meaning that for any variable assignment only one of the children of a sum node is non-zero.

Finally, a BN can be readily transformed (or *compiled*) into an equivalent SPN using the VErto algorithm [7]. The procedure starts by first specifying a topological ordering of the variables in the BN[2], and then eliminating each variable from all the factors it appears, following this order. Intermediate factors are aggregated, until a single polynomial remains. The result is a complete, decomposable, and selective SPN that computes exactly the same distribution as the original BN.

[2] An ordering where if X is a parent of Y, then X appears before Y.

2.2 *Causal Inference*

Traditionally, causal analysis has been based on structural equation models (SEMs) [12], which provide for an effective way to encode dependencies between variables, as well as allow for studying causal queries. In this framework, probabilistic relationships are represented using a BN, which allows for estimating causal queries such as the effect of interventions.

Interventional distributions, i.e. the distribution of a set of variables, after a second set of variables is forced to attain certain values, are of central importance in causal inference. Assuming an intervention on a variable X, denoted by either $do(X = x)$ or $do(x)$, the joint distribution of the remaining variables, $\mathbf{V}_{-x}$, under this intervention, is $\Pr(\mathbf{V}_{-x}|do(X = x)) = \frac{\Pr(\mathbf{V}_{-x}, X=x)}{\Pr(X=x|\mathbf{PA}_x)}$, where $\mathbf{PA_x}$ are the parents of X. However, graphical criteria are extensively used in order to express interventional distributions in terms of conditionals and marginals [12], highlighting how standard observational distributions can be used to estimate the effects of interventions. More on one of the most popular graphical criteria, the *rules of do-Calculus*, can be found in the supplementary material.

3 Related work

The question of how to transform SPNs to BNs was initially studied in [23], where the authors propose an algorithm to achieve this goal, by interpreting sum nodes as latent random variables. Given an SPN defined over variables $X_1, X_2, \cdots, X_k$, the exact procedure is as follows:

- Introduce a set of latent variables, $Z_1, Z_2, \cdots, Z_n$, as many as the sum nodes in the SPN.

- Create an empty BN over the observable variables, $X_1, X_2, \cdots, X_k$, and the new latent variables.
- Draw an arrow from each Z_i to the observable variables belonging in the scope of the sum node for which Z_i was created for.

Figure 1b, depicts the result of applying this algorithm to the SPN in Figure 1a. A similar approach was proposed subsequently in [15], where the difference is that the authors take into account the hierarchical structure of a SPN, so instead of only connecting each latent variable with the observable variables in its scope, they allow for connections between latent variables, $Z_i \rightarrow Z_j$, as long as the sum node corresponding to Z_j is in the scope of the one corresponding to Z_i. The result of applying this procedure to the SPN in 1a is shown in Figure 1c. A closer look shows that the only difference between the BNs in 1c and 1b, is that the former has some additional edges representing the hierarchy between the sum nodes in the SPN.

More recently, a new transformation was proposed in [4], which given a SPN compiled from a BN (using the VErto algorithm), is guaranteed to output a BN with the same structure as the *moral closure*[3] of the original BN, however it requires interpreting sum nodes as latent variables, which may result in losing reference of some of the observable variables. Figure 1d demonstrates this issue, since although the decompilation algorithm accurately recovered the structure of original BN in Figure 1a[4], both X_1 and X_2 have been replaced by latent variables, making the relationship between X_1, X_2, X_3 ambiguous, limiting the overall improvement of the transparency of the corresponding SPN.

Moreover, an important observation about all three approaches, is that while they differ in the specifics of the connections and the number of latent variables they induce, they all generate BNs with bipartite structure, with arrows stemming only from latent to observable variables. This will play a crucial role in proving that these BNs are not suitable for studying causal relationships between variables.

Apart from improving transparency, an additional advantage of accurately decompiling SPNs is that it opens the door for studying causal queries using the SPN distribution. So far, the link between SPNs and causality has been explored in alternative ways, such as in [21], where the authors propose a way for utilizing SPNs in order to estimate interventional distributions, using the SPN as a mapping between BN structures and distributions. Furthermore, in [22], the authors show that SPNs are capable of representing interventional distributions. Finally, in [9], a different approach is presented, where a BN is compiled into a SPN, and then the resulting SPN is used in order to compute both causal and non-causal queries.

4 Graphical Representations of SPNs

4.1 Causal Utility of Existing Decompilations

All the algorithms presented in the previous section result in BNs that represent conditional independencies that hold in the augmented joint probability defined over both the latent and the observable variables. However, in terms of transparency, looking at the figures in 1 it is clear that the introduction of latent variables leads to transformations that result in BNs that obscure the SPN's internal representation. This situation seems rather problematic, since even when a SPN is compiled from an informative BN, existing transformations might fail to recover the original BN or the exact way variables are connected to each other. Furthermore, looking at Figure 1, an intuitive observation is that this should have profound implications in causal inference applications, since if we assume there are causal relationships between X_1, X_2, X_3, the decompiled BNs attribute every correlation to latent factors. This means that if we start with a BN that represents a set of causal assumptions, compiling the BN to a SPN, and then decompiling it using any of the existing transformations, results in a great loss of information. The following result formalizes this intuitive observation, by showing that the fact that there is no edge coming out of the observable variables implies that all interventional distributions are trivial.

Proposition 4.1 *Let $\mathcal{B}$ be a BN, $\boldsymbol{V}$ the set of its nodes, and $\boldsymbol{X} \subseteq \boldsymbol{V}$ such that no node in $\boldsymbol{X}$ has an edge coming out of it, then* $\Pr(\boldsymbol{V}_{-X}|do(\boldsymbol{x})) = \Pr(\boldsymbol{V}_{-X})$.

This immediately leads to the following:

Theorem 1 *The BN, $\mathcal{B}$, that results after transforming an SPN using the procedure described in [23], [15], or [4], satisfies the property* $\Pr(\boldsymbol{V}_{-X}|do(\boldsymbol{x})) = \Pr(\boldsymbol{V}_{-X})$, *for any $\boldsymbol{X} \subseteq \boldsymbol{V}$, where $\boldsymbol{V}$ is the set of the observable variables.*

The above result shows that existing decompilations do not result in BNs that can accommodate for interesting causal queries, providing an answer to the open question in [23]. This limitation stems from the fact that sum nodes are interpreted as latent variables, which leads to every probabilistic dependency being attributed to unobserved confounders, not to direct interaction between the variables. In turn, it is not surprising that all interventions are trivial, since the resulting BNs hold no meaningful information about the relationships between the variables. Furthermore, it is concerning that even when SPNs are compiled from BNs that are both transparent and suitable for causal applications, such as in 1a, the decompiled BNs do not retain these properties.

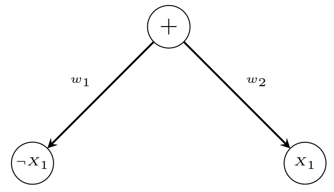
4.2 A New Decompilation Approach

The results in the previous section identify some of the implications of not being able to accurately decompile a SPN, in causality-related applications. However, even if a SPN represents purely associational, non-causal, relationships, the resulting BNs may fail to represent the dependencies between the variables, hindering the SPN's transparency, and limiting its overall utility. It is now reasonable to wonder whether this is due to an inherent limitation of SPNs, or an artifact of the existing transformations. One might argue that a SPN is merely a computational representation of a distribution, meaning that its purpose is to capture the functional characteristics of the joint distribution and provide an efficient way of computing it. Looking at SPNs from this angle, existing decompilation algorithms just reflect the various functions that SPNs define internally in order to perform the necessary computations. For example, Figure 1b represents the fact that the SPN in 1a consists of 5 sub-SPNs; one over all X_1, X_2, X_3, two over X_2, X_3, and two over just X_3.

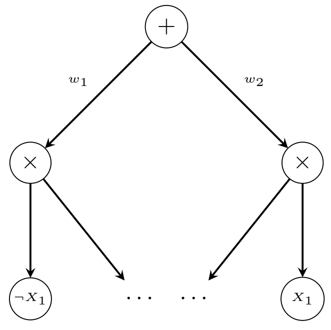
However, in the remaining of this section, we show that under certain conditions it is possible to decompile SPNs to far more informative BNs, that contain no augmented variables, rather they only include the original SPN variables. This involves utilizing alternative interpretations of the SPN sum nodes, as well as considering the SPN parameters. This is a novel aspect of our approach, since none of the existing decompilation algorithms take into account the SPN parameters, although they contain information that cannot be retrieved by

[3] The moral closure of a BN introduces an edge between any two nodes that share common children.

[4] This BN is already morally closed, which is why the decompilation is exact.



(a) Children



(b) Grandchildren

Figure 2: Examples of indicator children and grandchildren

looking at the SPN structure alone [14]. More precisely, we assume that a BN $\mathcal{B}$ is an *I-map* for the distribution it represents (i.e. the independence relationships that can be inferred from the BN, are indeed satisfied by the distribution), and we consider the problem of decompiling the SPN that results from compiling $\mathcal{B}$ using the VErto algorithm. This is a similar setting to the one in [4], where the proposed algorithm can perform an exact decompilation for SPNs compiled from morally closed BNs, which is a relatively small subset of BNs, however in this work we go one step further and introduce an algorithm that is always accurate, *regardless* of the underlying BN.

As shown in section 4.1, the assumption that every sum node corresponds to a latent variable is very restricting, having detrimental effects on the transparency and causal utility of the decompiled BN. Although all existing decompilation algorithms share this assumption, this is not the only way of interpreting sum nodes. As a matter of fact, a meaningful probabilistic interpretation can be given to any sum node of a SPN that *represents* a variable [11]. A sum node, S, represents a variable, V, (denoted by Represent$(S) = V$) if it has as many children as the number of states V has, and an indicator corresponding to one of V's states can be reached either immediately (i.e., it is a child of S) or after one intermediate layer (i.e., it is a grandchild of S). For example, in both figures 2a, 2b the root represents X_1, since it has as many children as X_1's states, and each of them reaches a unique indicator either immediately (2a) or through a product node (2b). It is not hard to see that if a sum node represents a variable, then it is selective, however the converse is not necessarily true. The decompilations in [15, 4] operate on selective SPNs, leading to the BNs discussed in the previous section, while our approach takes a different stance and makes use of the properties that follow from having a node representing a variable, instead. As shown in [11], if a sum node represents V, then it encodes the conditional distribution of V given the context set by its ancestors. For example, in Figure 3b, the sum node in the red double circle (corresp. the blue double circle) models the distribution of $X_2|X_1 = 0$ (corresp. $X_2|X_1 = 1$). Analogously, in Figure 3c, the sum node in red models the distribution of $X_3|X_1 = 0, X_2 = 0$, since it represents X_3, while starting from the root, the path leading to it contains indicators corresponding to the context $X_1 = 0, X_2 = 0$.

An important consequence of having a sum node representing a variable is that it is no longer necessary to introduce augmented variables to define conditional distributions. Furthermore, we can now make two crucial observations about the qualitative properties of SPNs that result from the VErto algorithm. To this end, let $\mathcal{B}$ be a BN, and $\mathcal{SPN_C}$ be the corresponding compiled SPN, then the following properties hold:

- **Property 1** Every sum node in $\mathcal{SPN_C}$ represents a variable [7, 11].
- **Property 2** Let V be a variable in $\mathcal{B}$, and $\mathbf{S}$ be the set of sum nodes representing V in the SPN, $\mathbf{S} = \{S | S \text{ is a sum node, Represent}(S) = V\}$. Then, since $\mathcal{SPN_C}$ is compiled using a topological ordering, each of the paths that start from the root and end in one of the nodes in $\mathbf{S}$ meets sum nodes representing the same variables. For example, looking at Figure 3b, X_2 is represented by the nodes in the red and blue double circles. Starting from the root, all paths leading two these nodes include just the root, which represents X_1. In addition, the double circles in figures 3c,3d correspond to all the nodes representing X_3. It is not difficult to see that all paths ending in one of these nodes, include the root (representing X_1) as well as one of the nodes that represent X_2. This is a direct consequence of having a topological ordering.

At this point we should remind ourselves that the goal of any decompilation algorithm is to recover the parent-child relationships represented by the underlying BN, $\mathcal{B}$. Having this in mind, properties 1 and 2 imply that for every variable, V, $\mathcal{SPN_C}$ defines its conditional distribution by conditioning on all of its SPN ancestors, $X_1, \cdots, X_k$, i.e. the variables that appear earlier than V in $\mathcal{SPN_C}$, denoted by $\mathbf{C}_S$. This resembles the conditional distributions of V in $\mathcal{B}$, however this is defined by conditioning only on its parents, $P_1, \cdots, P_m$, denoted by $\mathbf{C}_B$, not all of its ancestors. Regardless, the crucial observation here is that since $\mathcal{SPN_C}$ was compiled following a topological ordering respecting $\mathcal{B}$, then $\mathbf{C}_B \subseteq \mathbf{C}_S$ and furthermore all variables in $\mathbf{C}_S \setminus \mathbf{C}_B$ are non-descendants of V in $\mathcal{B}$.

In general, distributions that factorize according to a BN respect the *local Markov property* [16], which states that a variable is independent of its non-descendants given its parents. In our case, since $\mathbf{C}_B$ contains exactly the parents of V in $\mathcal{B}$, the aforementioned property makes sure that for any set, $\mathbf{ND}$, comprised of non-descendants of V in $\mathcal{B}$, we have that $\Pr(V|\mathbf{C}_B) = \Pr(V|\mathbf{C}_B, \mathbf{ND})$. Connecting this with the observations in the previous paragraph, we immediately see that $\Pr(V|\mathbf{C}_S) = \Pr(V|\mathbf{C}_B)$, since $\mathbf{C}_B \subseteq \mathbf{C}_S$ and $\mathbf{C}_S \setminus \mathbf{C}_B \subseteq \mathbf{ND}$.

Following this reasoning reveals that the task of decompiling $\mathcal{SPN_C}$ is equivalent to inferring the set $\mathbf{C}_S \setminus \mathbf{C}_B$. This is because $\mathbf{C}_S$ can be easily found by inspecting $\mathcal{SPN_C}$, so if $\mathbf{C}_S \setminus \mathbf{C}_B$ is also known, then their difference $\mathbf{C}_S \setminus (\mathbf{C}_S \setminus \mathbf{C}_B) = \mathbf{C}_B$, uncovering the parents of V in B.

As an example, in Figure 3b the nodes representing X_2 model the distributions $X_2|X_1 = 0, X_2|X_1 = 1$, so $\mathbf{C}_S = \{X_1\}$. Moreover,

Algorithm 1 SPN to BN decompilation

```
Require: A SPN over variables $X_1, \cdots X_n$, compiled using the VErto algorithm, $\mathcal{SPN_C}$
  $\mathcal{B} \leftarrow$ the empty BN over $X_1, \cdots X_n$
  $Not\text{-}visited \leftarrow \{X_1, \cdots X_n\}$
  while $Not\text{-}visited \neq \emptyset$ do
      Pick a variable $X_k \in Not\text{-}visited$
      $\mathbf{Rep}_{X_k} \leftarrow \{S | S$ is a sum node, Represent($S$) $= X_k\}$
      $Ancestors_{X_k} \leftarrow \{X_m | \exists\ S$ such that Represent($S$) $= X_m, S$ is above nodes in $\mathbf{Rep}_{X_k}\}$
      if $Ancestors_{X_k} = \emptyset$ then          ▷ The conditioning set is empty, so no parents exist
          $Not\text{-}visited \leftarrow Not\text{-}visited \setminus \{X_k\}$
          continue
      end if
      for every variable $X_m \in Ancestors_{X_k}$ do
          $\mathbf{Rep}_{X_m} \leftarrow \{S | S$ is a sum node, Represent($S$) $= X_m\}$
          for every $S_m \in \mathbf{Rep}_{X_m}$ do
              $Trees^{S_m} \leftarrow \{T | T \in Subtrees^{S}(\mathcal{SPN_C})$ for some $S \in \mathbf{Rep}_{X_k}, S_m \in T\}$
              $PairedTrees^{S_m} \leftarrow \{(t_0, t_1) | t_0, t_1 \in Trees^{S_m}, \mathbb{1}_{X_m=0} \in t_0, \mathbb{1}_{X_m=1} \in t_1,$
                                   otherwise they contain the same indicators$\}$
              for every $(t_0, t_1) \in PairedTrees$ do
                  $S_0 \leftarrow$ the end node in $t_0$
                  $S_1 \leftarrow$ the end node in $t_1$
                  if $S_0 \neq S_1$ then
                      Add $X_m \rightarrow X_k$ to $\mathcal{B}$
                      break
                  end if
              end for
          end for
      end for
      $Not\text{-}visited \leftarrow Not\text{-}visited \setminus \{X_k\}$
  end while
  return $\mathcal{B}$
```

since $\mathbf{C}_B \subseteq \mathbf{C}_S$, it follows that $\mathbf{C}_S \setminus \mathbf{C}_B$ is equal to either $\{X_1\}$ or $\emptyset$. However, looking at the underlying BN shown in Figure 3a, we immediately see that $\Pr(X_2|X_1) = \Pr(X_2)$, so the local Markov property guarantees that X_1 is a non-descendant of X_2 in $\mathcal{B}$, leading to $\mathbf{C}_S \setminus \mathbf{C}_B = \{X_1\} \Rightarrow \mathbf{C}_B = \emptyset$, concluding that X_2 has no parents in the underlying BN. It only appears as if X_1 influences the distribution of X_2 because all layers of a SPN are connected, but this is not really the case.

We are now ready to give an intuitive description of the decompilation process: looking at the ancestors of any variable, V, in $\mathcal{SPN_C}$, we can immediately identify $\mathbf{C}_S$, which is a superset of $\mathbf{C}_B$. Furthermore, since V's conditional distribution must respect $\mathcal{B}$, the local Markov property guarantees that all variables in $\mathbf{C}_S \setminus \mathbf{C}_B$ can be removed from $\mathbf{C}_S$ without affecting the distribution. This means that once we establish a way to remove redundant variables from $\mathbf{C}_S$, then the ones that are going to remain by the end of the elimination process are going to be exactly the parents of V in $\mathcal{B}$. It should be noted that this interplay between the local Markov property in $\mathcal{B}$ and the conditioning sets in $\mathcal{SPN_C}$ is novel to the presented approach, and is not found in any existing decompilation algorithm. All previous algorithms rely on structural SPN properties alone, where the presented approach utilizes information about $\mathcal{B}$, yet in a way that does not require $\mathcal{B}$ to be known beforehand, rather only involving properties that follow from its existence.

The only thing remaining is to come up with a way for identifying the variables in $\mathbf{C}_S \setminus \mathbf{C}_B$. However, this can be easily done by using the following proposition:

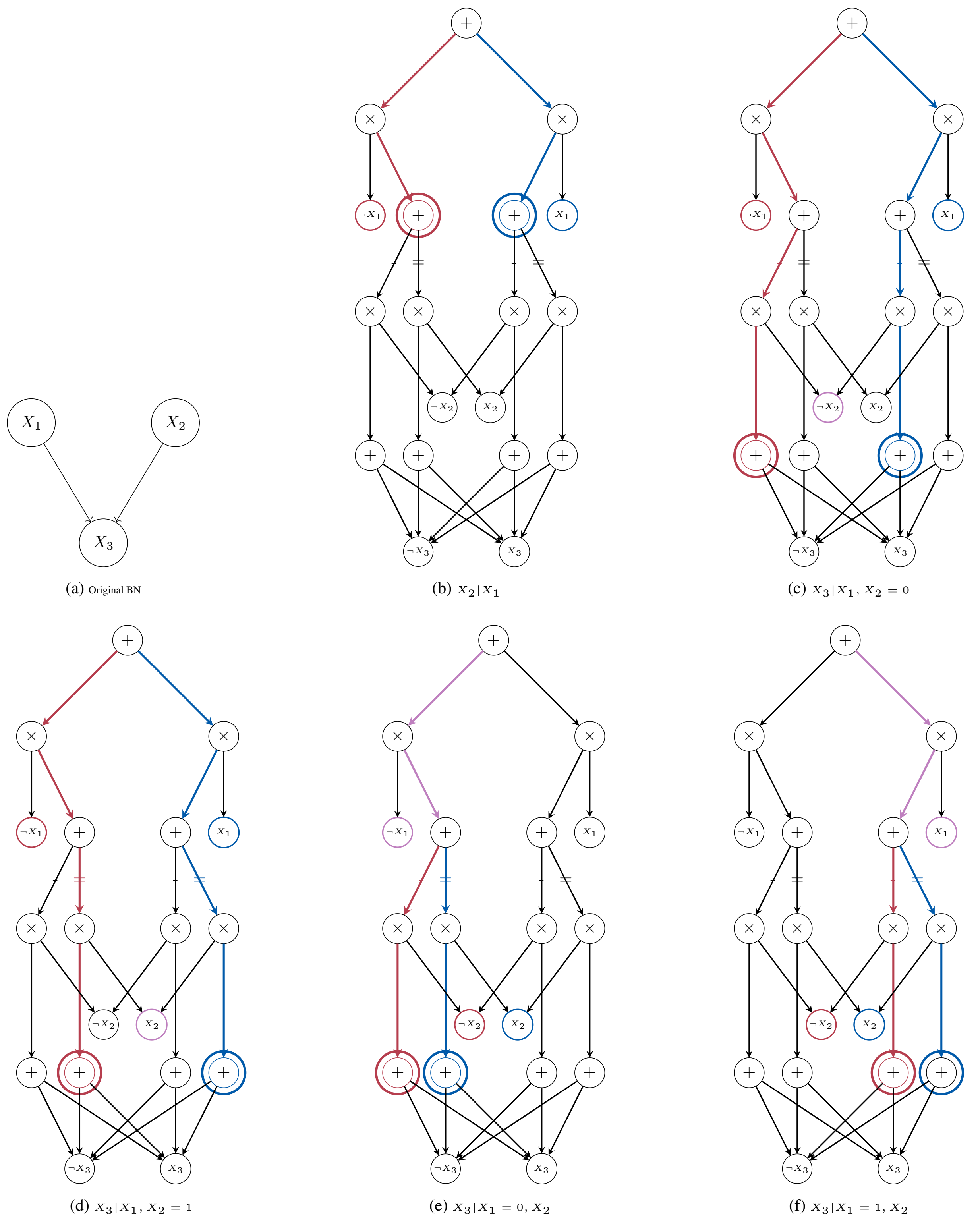
Proposition 4.2 *Let $Y, X_1, X_2, \cdots, X_n$ be binary random variables. If there is a X_k, such $\Pr(Y|X_1, X_2, \cdots, X_k = 0, \cdots, X_n) = \Pr(Y|X_1, X_2, \cdots, X_k = 1, \cdots, X_n)$, then $Y \perp X_k | X_1, X_2, \cdots, X_{k-1}, X_{k+1}, \cdots, X_n$.*

Finally, Proposition 4.2 can be invoked implicitly by considering appropriate *induced trees up to a sum node S'*, which generalize standard *induced trees* [24].

Definition 4.1 *Let $\mathcal{S}$ be a complete and decomposable SPN over variables $X_1, \cdots, X_n$, and $T = (T_V, T_E)$ be a subgraph of $\mathcal{S}$. T is called an induced subtree up to node S' of $\mathcal{S}$ if:*

- *$Root(\mathcal{S}) \in T_V$.*
- *If $v \in T_V$ is a sum node, then exactly one child of v in $\mathcal{S}$ is in T_V, and the corresponding edge is in T_E.*
- *If $v \in T_V$ is a product node, then all children of v in $\mathcal{S}$ are in T_V, and the corresponding edges are in T_E*
- *$S' \in T_V$, and once S' is reached, the tree expansion stops.*

An induced tree up to a sum node S' results from traversing a SPN top-down, such that for every sum node only one of its children is included in the tree, for every product node all of its children are in the tree, and as soon as S' is reached the procedure terminates. For a SPN $\mathcal{S}$, we denote the collection of all induced subtrees up to node S' as $Subtrees^{S'}(\mathcal{S})$. The terminal node models the conditional distribution of the variable represented by S', given the context implied by the indicators in the tree. For example, the sum node in the double red circle in Figure 3d is the terminal node of the induced tree in red, and



(a) Original BN

(b) $x_2 | x_1$

(c) $x_3 | x_1, x_2 = 0$

(d) $x_3 | x_1, x_2 = 1$

(e) $x_3 | x_1 = 0, x_2$

(f) $x_3 | x_1 = 1, x_2$

Figure 3: Illustration of applying Algorithm 1. Indicators in red are reached traversing the red paths, while indicators in blue are reached via blue paths. Indicators and edges in purple are belong to both red and blue paths. Edges marked with the same symbol have equal parameters.

it models $\Pr(X_3|X_1 = 0, X_2 = 1)$, since this tree contains indicators for $X_1 = 0, X_2 = 1$. In turn, if S_0, S_1 represent the same variable V, comparing $T_0 \in Subtrees^{S_0}(\mathcal{S}), T_1 \in Subtrees^{S_1}(\mathcal{S})$, where T_0, T_1 differ only in the state of an indicator made for a single variable X, implicitly invokes Proposition 4.2. If $S_0 \neq S_1$, then the equality in 4.2 does not hold, so by the local Markov property $X \in \mathbf{C}_B$, meaning that X must be a parent of V in $\mathcal{B}$. Alternatively, if all such nodes are identical, then $X \in \mathbf{C}_S \setminus \mathbf{C}_B$. This way, Algorithm 1 utilizes a unique blend of information about both the structure and parameters of $\mathcal{SPN}_\mathcal{C}$ to enable an exact decompilation.

The following theorem is the culmination of our analysis, proving the validity of Algorithm 1[5].

Theorem 2 *Let $\mathcal{B}$ be a BN that is an I-map for the distribution it represents, and let $\mathcal{SPN}_\mathcal{C}$ be the SPN that results from compiling $\mathcal{B}$ using the VErto algorithm. Then applying Algorithm 1 to $\mathcal{SPN}_\mathcal{C}$ outputs exactly $\mathcal{B}$.*

Example: Figure 3 demonstrates the general process with an example. The original BN is shown in Figure 3a, while the remaining figures show the compiled SPN, along with the trees that have to be examined following Algorithm 1. It is well known that compiling the BN in 3a results in a SPN where the sum nodes in double circles in Figure 3b must have identical parameters, which is why the corresponding edges are marked. Apart from that, the parameters of the remaining sum nodes are in general distinct, since they are not bound to satisfy any constraint.

Starting with X_1, we see that $\mathbf{C}_S = \emptyset \Rightarrow \mathbf{C}_B = \emptyset$, so X_1 has no parents in the underlying BN. Next, moving on to X_2, we have that $\mathbf{C}_S = \{X_1\}$. Looking at Figure 3b, we need to compare the sum node in red (which models the distribution $\Pr(X_2|X_1 = 0)$) to the node in blue (which models $\Pr(X_2|X_1 = 1)$). Since these nodes have identical parameters, we have that $\Pr(X_2|X_1 = 0) = \Pr(X_2|X_1 = 1)$, so by Proposition 4.2 we conclude that $\Pr(X_2|X_1) = \Pr(X_2)$, which combined with the local Markov property makes sure that $X_1 \notin \mathbf{C}_B \Rightarrow \mathbf{C}_B = \emptyset$, so neither X_2 has a parent in the underlying BN. Moving on to X_3, we see that $\mathbf{C}_S = \{X_1, X_2\}$. Figures 3c, 3d highlight the trees that need to be considered to decide whether X_1 belongs to $\mathbf{C}_B$. Comparing the corresponding sum nodes, we see that they model distinct distributions (since they do not not have equal parameters), meaning that $X_1 \in \mathbf{C}_B$, so the edge $X_1 \rightarrow X_3$ is added to the BN. Finally, we have to repeat this process for X_2, which is shown in figures 3e, 3f. Again, since the final sum nodes are not identical, the edge $X_2 \rightarrow X_3$ is added, exactly recovering the BN in Figure 3a.

Finally, we would like to note that although in the worst case Algorithm 1 needs to compare all induced trees in $\mathcal{SPN}_\mathcal{C}$, so it has an exponential complexity, various optimizations can help reduce the number of computations. For example, by using the *break* command, Algorithm 1 connects two variables in the underlying BN as soon as it detects a pair of non-identical sum nodes, without needing to explore all the remaining induced trees. Additional enhancements, such as caching intermediate results, should greatly help speed up the computations, however here we focus on proving the feasibility and providing the theoretical means of achieving an exact SPN decompilation.

[5] Although the algorithm is stated for binary variables, it is straightforward to extend it to the non-binary case by adjusting the induced subtrees that need to be considered.

4.3 Discussion and Conclusions

The above example shows how Algorithm 1 combines information about the structure and parameters of the compiled SPN in order to perform an exact decompilation. In contrast, even for relatively simple cases, like those in figures 1a, 3a, all existing algorithms might fail to perform this task. This limitation has had a direct impact on a number of related works, such as the one presented in [9], where the author has to maintain both a BN and the corresponding compiled SPN in order to study interventional queries, despite the fact that all computations are performed using just the SPN. This is because maintaining only the SPN would result in losing all the information about the way variables are connected, hindering any subsequent causal analysis. Nevertheless, using Algorithm 1 it is no longer necessary to maintain the original BN, since all this information can be recovered at any point. This means that the compiled SPN can be used to compute both standard observational queries, as well as interventional ones (by first decompiling it to a BN and then following [9]). This observation is also related to the results in [22], where the authors show that SPNs can be used to represent interventional distributions. However, SPNs compiled from BNs demonstrate that it is possible for a single SPN to represent both observational and interventional distributions, mitigating the need to maintain a separate SPN for each.

Looking at the broader picture, our work challenges the view that BNs are transparent models with representational semantics, while SPNs are black-boxes with only operational ones [5], by showing that SPNs have equivalent transparent representations, however they are embedded into their structure and parameters. Finally, looking back at the algorithm's derivation, we see that properties 1 and 2 are the only essential requirements to guarantee that an exact decompilation is possible. This implies that Algorithm 1 should be applicable to any SPN satisfying them, not only those compiled using the VErto algorithm. An immediate consequence is that even when learning a SPN from data, instead of compiling it from a BN, as long as these two properties are enforced during training, it should be possible to generate a BN without additional hidden variables that agrees with the SPN distribution. Moreover, our results point towards a very interesting future research direction, since they imply that as long as sum nodes can be assigned with a meaningful interpretation, Algorithm 1 can be used to recover and explicate all the relationships learned by the SPN. For example, sum nodes of SPNs learned from data using the algorithm in [19], could be interpreted as representing tree distributions, so applying Algorithm 1 should yield a transparent mixture of distributions. Exploring this approach has the potential to result in a new generation highly flexible, fully transparent models, that combine the efficient inference of SPNs with the representational transparency of BNs, opening the door for utilizing SPNs in applications where model transparency is essential [20].

Acknowledgements

This research was partly supported by a Royal Society University Research Fellowship, UK, and partly supported by a grant from the UKRI Strategic Priorities Fund, UK to the UKRI Research Node on Trustworthy Autonomous Systems Governance and Regulation (EP/V026607/1, 2020–2024).

References

[1] Francis R Bach and Michael I Jordan, 'Thin junction trees', in *Advances in Neural Information Processing Systems*, pp. 569–576, (2002).

[2] 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).

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

[4] Cory J. Butz, Jhonatan de S. Oliveira, and Robert Peharz, 'Sum-product network decompilation', in *International Conference on Probabilistic Graphical Models, PGM 2020, 23-25 September 2020, Aalborg, Hotel Comwell Rebild Bakker, Skørping, Denmark*, eds., Manfred Jaeger and Thomas Dyhre Nielsen, volume 138 of *Proceedings of Machine Learning Research*, pp. 53–64. PMLR, (2020).

[5] YooJung Choi, Antonio Vergari, and Guy Van den Broeck, 'Probabilistic circuits: A unifying framework for tractable probabilistic models', (oct 2020).

[6] Gregory F Cooper, 'The computational complexity of probabilistic inference using bayesian belief networks', *Artificial intelligence*, **42**(2-3), 393–405, (1990).

[7] Adnan Darwiche, 'A differential approach to inference in bayesian networks', *J. ACM*, **50**, 280–305, (2000).

[8] Adnan Darwiche, 'A logical approach to factoring belief networks', in *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 409–420, (2002).

[9] Adnan Darwiche, 'Causal inference using tractable circuits', *arXiv preprint arXiv:2202.02891*, (2022).

[10] R. Gens and P. Domingos, 'Learning the structure of sum-product networks', in *International Conference on Machine Learning*, (2013).

[11] Iago Paris, Raquel Sanchez-Cauce, and Francisco Javier Diez. Sum-product networks: A survey, 2020.

[12] Judea Pearl, 'Causal inference in statistics: An overview', *Statistics Surveys*, **3**, 96–146, (01 2009).

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

[14] Robert Peharz, Robert Gens, and Pedro Domingos, 'Learning selective sum-product networks', in *LTPM workshop*, volume 32, (2014).

[15] 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*, (2016).

[16] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2017.

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

[18] Fabian Rathke, Mattia Desana, and Christoph Schnörr, 'Locally adaptive probabilistic models for global segmentation of pathological oct scans', in *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part I 20*, pp. 177–184. Springer, (2017).

[19] Amirmohammad Rooshenas and Daniel Lowd, 'Learning sum-product networks with direct and indirect variable interactions', in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, p. I–710–I–718. JMLR.org, (2014).

[20] Cynthia Rudin, 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead', *Nature Machine Intelligence*, **1**(5), 206–215, (2019).

[21] Matej Zečević, Devendra Dhami, Athresh Karanam, Sriraam Natarajan, and Kristian Kersting, 'Interventional sum-product networks: Causal inference with tractable probabilistic models', *Advances in Neural Information Processing Systems*, **34**, 15019–15031, (2021).

[22] Matej Zečević, Devendra Singh Dhami, and Kristian Kersting, 'On the tractability of neural causal inference', *arXiv preprint arXiv:2110.12052*, (2021).

[23] Han Zhao, Mazen Melibari, and Pascal Poupart, 'On the relationship between sum-product networks and bayesian networks', in *ICML'15*, (2015).

[24] Han Zhao, Pascal Poupart, and Geoffrey J Gordon, 'A unified approach for learning the parameters of sum-product networks', in *Advances in Neural Information Processing Systems*, eds., D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, volume 29, pp. 433–441. Curran Associates, Inc., (2016).