On the Relationship between Sum-Product Networks and Bayesian Networks (Supplementary Material)

Han Zhao HAN.ZHAO@UWATERLOO.CA
Mazen Melibari MMELIBAR@UWATERLOO.CA
Pascal Poupart PPOUPART@UWATERLOO.CA
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

In this supplementary material we provide the proofs for theorems, lemmas as well as propositions in the main paper. Pseudocode to illustrate the transformation from complete and consistent SPN to *normal* SPN is also provided. We start by introducing the notations used in this paper.

1. Notations

We use $1:N$ to abbreviate the notation $\{1, 2, \ldots, N\}$. We use a capital letter X to denote a random variable and a bolded capital letter $\mathbf{X}_{1:N}$ to denote a set of random variables $\mathbf{X}_{1:N} = \{X_1, \ldots, X_N\}$. Similarly, a lowercase letter x is used to denote a value taken by X and a bolded lowercase letter $\mathbf{x}_{1:N}$ denotes a joint value taken by the corresponding vector $\mathbf{X}_{1:N}$ of random variables. We may omit the subscript $1:N$ from $\mathbf{X}_{1:N}$ and $\mathbf{x}_{1:N}$ if it is clear from the context. For a random variable X_i, we use $x_i^j, j \in 1:J$ to enumerate all the values taken by X_i. For simplicity, we use $\Pr(x)$ to mean $\Pr(X = x)$ and $\Pr(\mathbf{x})$ to mean $\Pr(\mathbf{X} = \mathbf{x})$. We use calligraphic letters to denote graphs (e.g., $\mathcal{G}$). In particular, BNs, SPNs and ADDs are denoted respectively by $\mathcal{B}$, $\mathcal{S}$ and $\mathcal{A}$. For a DAG $\mathcal{G}$ and a node v in $\mathcal{G}$, we use $\mathcal{G}_v$ to denote the subgraph of $\mathcal{G}$ induced by v and all its descendants. Let $\mathbf{V}$ be a subset of the nodes of $\mathcal{G}$, then $\mathcal{G}|_{\mathbf{V}}$ is a subgraph of $\mathcal{G}$ induced by the node set $\mathbf{V}$. Similarly, we use $\mathbf{X}|_A$ or $\mathbf{x}|_A$ to denote the restriction of a vector to a subset A. Other notation will be introduced when needed.

We revisit the definition of *normal* SPN:

Definition 7. An SPN is said to be normal if

1. It is complete and decomposable.
2. For each sum node in the SPN, the weights of the edges emanating from the sum node are nonnegative and sum to 1.
3. Every terminal node in an SPN is a univariate distribution over a Boolean variable and the size of the scope of a sum node is at least 2 (sum nodes whose scope is of size 1 are reduced into terminal nodes).

2. Proof of Theorem 3

Theorem 3. For any complete and consistent SPN $\mathcal{S}$, there exists a normal SPN $\mathcal{S}'$ such that $\Pr_{\mathcal{S}}(\cdot) = \Pr_{\mathcal{S}'}(\cdot)$ and $|\mathcal{S}'| = O(|\mathcal{S}|^2)$.

To show this, we first prove the following lemmas.

Lemma 1. For any complete and consistent SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$, there exists a complete and decomposable SPN $\mathcal{S}'$ over $\mathbf{X}_{1:N}$ such that $f_{\mathcal{S}}(\mathbf{x}) = f_{\mathcal{S}'}(\mathbf{x}), \forall \mathbf{x}$ and $|\mathcal{S}'| = O(|\mathcal{S}|^2)$.

Proof. Let $\mathcal{S}$ be a complete and consistent SPN. If it is also decomposable, then simply set $\mathcal{S}' = \mathcal{S}$ and we are done. Otherwise, let $v_1, \ldots, v_M$ be an inverse topological ordering of all the nodes in $\mathcal{S}$, including both terminal nodes and internal nodes, such that for any $v_m, m \in 1:M$, all the ancestors of v_m in the graph appear after v_m in the ordering. Let v_m be the first product node in the ordering that violates decomposability. Let $v_{m_1}, v_{m_2}, \ldots, v_{m_l}$ be the children of v_m where $m_1 < m_2 < \cdots < m_l < m$ (due to the inverse topological ordering). Let $(v_{m_i}, v_{m_j}), i < j, i, j \in 1:l$ be the first ordered pair of nodes such that $\text{scope}(v_{m_i}) \bigcap \text{scope}(v_{m_j}) \neq \varnothing$. Hence, let $X \in \text{scope}(v_{m_i}) \bigcap \text{scope}(v_{m_j})$. Consider $f_{v_{m_i}}$ and $f_{v_{m_j}}$ which are the network polynomials defined by the sub-SPNs rooted at v_{m_i} and v_{m_j}.

Expand network polynomials $f_{v_{m_i}}$ and $f_{v_{m_j}}$ into a *sum-of-product* form by applying the distributive law between products and sums. For example, if $f(X_1, X_2) = (\mathbb{I}_{x_1} + 9\mathbb{I}_{\bar{x}_1})(4\mathbb{I}_{x_2} + 6\mathbb{I}_{\bar{x}_2})$, then the expansion of f is $f(X_1, X_2) = 4\mathbb{I}_{x_1}\mathbb{I}_{x_2} + 6\mathbb{I}_{x_1}\mathbb{I}_{\bar{x}_2} + 36\mathbb{I}_{\bar{x}_1}\mathbb{I}_{x_2} + 54\mathbb{I}_{\bar{x}_1}\mathbb{I}_{\bar{x}_2}$. Since $\mathcal{S}$ is complete, then sub-SPNs rooted at v_{m_i} and v_{m_j} are also complete, which means that each monomial in the expansion of $f_{v_{m_i}}$ must share the same scope. The same applies to $f_{v_{m_j}}$. Since $X \in \text{scope}(v_{m_i}) \bigcap \text{scope}(v_{m_j})$, then every monomial in the expansion of $f_{v_{m_i}}$ and $f_{v_{m_j}}$ must contain an indicator variable over X, either $\mathbb{I}_x$ or $\mathbb{I}_{\bar{x}}$. Furthermore, since $\mathcal{S}$ is consistent, then the sub-SPN rooted at v_m is also consistent. Consider $f_{v_m} = \prod_{k=1}^{l} f_{v_{m_k}} =$

$f_{v_{m_i}} f_{v_{m_j}} \prod_{k\neq i,j} f_{v_{m_k}}$. Because v_m is consistent, we know that each monomial in the expansions of $f_{v_{m_i}}$ and $f_{v_{m_j}}$ must contain the same indicator variable of X, either $\mathbb{I}_x$ or $\mathbb{I}_{\bar{x}}$, otherwise there will be a term $\mathbb{I}_x\mathbb{I}_{\bar{x}}$ in f_{v_m} which violates the consistency assumption. Without loss of generality, assume each monomial in the expansions of $f_{v_{m_i}}$ and $f_{v_{m_j}}$ contains $\mathbb{I}_x$. Then we can re-factorize f_{v_m} in the following way:

$$\begin{aligned} f_{v_m} &= \prod_{k=1}^{l} f_{v_{m_k}} = \mathbb{I}_x^2 \frac{f_{v_{m_i}}}{\mathbb{I}_x}\frac{f_{v_{m_j}}}{\mathbb{I}_x}\prod_{k\neq i,j} f_{v_{m_k}} \\ &= \mathbb{I}_x \frac{f_{v_{m_i}}}{\mathbb{I}_x}\frac{f_{v_{m_j}}}{\mathbb{I}_x}\prod_{k\neq i,j} f_{v_{m_k}} = \mathbb{I}_x \tilde{f}_{v_{m_i}} \tilde{f}_{v_{m_j}} \prod_{k\neq i,j} f_{v_{m_k}} \end{aligned} \tag{1}$$

where we use the fact that indicator variables are idempotent, i.e., $\mathbb{I}_x^2 = \mathbb{I}_x$ and $\tilde{f}_{v_{m_i}}(\tilde{f}_{v_{m_j}})$ is defined as the function by factorizing $\mathbb{I}_x$ out from $f_{v_{m_i}}(f_{v_{m_j}})$. Eq. 1 means that in order to make v_m decomposable, we can simply remove all the indicator variables $\mathbb{I}_x$ from sub-SPNs rooted at v_{m_i} and v_{m_j} and later link $\mathbb{I}_x$ to v_m directly. Such a transformation will not change the network polynomial f_{v_m} as shown by Eq. 1, but it will remove X from $\text{scope}(v_{m_i}) \bigcap \text{scope}(v_{m_j})$. In principle, we can apply this transformation to all ordered pairs $(v_{m_i}, v_{m_j}), i < j, i, j \in 1 : l$ with nonempty intersections of scope. However, this is not algorithmically efficient and more importantly, for local components containing $\mathbb{I}_x$ in f_{v_m} which are reused by other nodes v_n outside of $\mathcal{S}_{v_m}$, we cannot remove $\mathbb{I}_x$ from them otherwise the network polynomials for each such v_n will be changed due to the removal. In such a case, we need to duplicate the local components to ensure that local transformations with respect to f_{v_m} do not affect network polynomials f_{v_n}. We present the transformation in Alg. 1.

Alg. 1 transforms a complete and consistent SPN $\mathcal{S}$ into a complete and decomposable SPN $\mathcal{S}'$. Informally, it works using the following identity:

$$f_{v_m} = \left(\prod_{X\in\Omega(v_m)} \mathbb{I}_{x^*}\right) \prod_{k=1}^{l} \frac{f_{v_{m_k}}}{\prod_{X\in\Omega(v_m)\cap\text{scope}(v_{m_k})} \mathbb{I}_x^*} \tag{2}$$

where $\Omega(v_m) \triangleq \bigcup_{i,j\in 1:l, i\neq j} \text{scope}(v_{m_i})\cap\text{scope}(v_{m_j})$, i.e., $\Omega(v_m)$ is the union of all the shared variables between pairs of children of v_m and $\mathbb{I}_{x^*}$ is the indicator variable of $X \in \Omega(v_m)$ appearing in $\mathcal{S}_{v_m}$. Based on the analysis above, we know that for each $X \in \Omega(v_m)$ there will be only one kind of indicator variable $\mathbb{I}_{x^*}$ that appears inside $\mathcal{S}_{v_m}$, otherwise v_m is not consistent. In Line 6, $\mathcal{S}_{v_m}|_{\mathbf{V}}$ is defined as the sub-SPN of $\mathcal{S}_{v_m}$ induced by the node set $\mathbf{V}$, i.e., a subgraph of $\mathcal{S}_{v_m}$ where the node set is restricted to $\mathbf{V}$. In Lines 5-6, we first extract the induced sub-SPN $\mathcal{S}_{\mathbf{V}}$ from $\mathcal{S}_{v_m}$ rooted at v_m using the node set in which nodes have nonempty

Algorithm 1 Decomposition Transformation

Input: Complete and consistent SPN $\mathcal{S}$.
Output: Complete and decomposable SPN $\mathcal{S}'$.

1: Let $v_1, v_2, \ldots, v_M$ be an inverse topological ordering of nodes in $\mathcal{S}$.
2: **for** $m = 1$ to M **do**
3: **if** v_m is a non-decomposable product node **then**
4: $\Omega(v_m) \leftarrow \bigcup_{i\neq j} \text{scope}(v_{m_i}) \bigcap \text{scope}(v_{m_j})$
5: $\mathbf{V} \leftarrow \{v \in \mathcal{S}_{v_m} \mid \text{scope}(v) \bigcap \Omega(v_m) \neq \varnothing\}$
6: $\mathcal{S}_{\mathbf{V}} \leftarrow \mathcal{S}_{v_m}|_{\mathbf{V}}$
7: $D(v_m) \leftarrow$ descendants of v_m
8: **for** node $v \in \mathcal{S}_{\mathbf{V}} \backslash \{v_m\}$ **do**
9: **if** $Pa(v) \backslash D(v_m) \neq \varnothing$ **then**
10: Create $p \leftarrow v \otimes \prod_{X\in\Omega(v_m)\cap\text{scope}(v)} \mathbb{I}_{x^*}$
11: Connect p to $\forall f \in Pa(v) \backslash D(v_m)$
12: Disconnect v from $\forall f \in Pa(v) \backslash D(v_m)$
13: **end if**
14: **end for**
15: **for** node $v \in \mathcal{S}_{\mathbf{V}}$ in bottom-up order **do**
16: Disconnect $\tilde{v} \in Ch(v)$ $\forall \text{scope}(\tilde{v}) \subseteq \Omega(v_m)$
17: **end for**
18: Connect $\prod_{X\in\Omega(v_m)} \mathbb{I}_{x^*}$ to v_m directly
19: **end if**
20: **end for**
21: Delete all nodes unreachable from the root of $\mathcal{S}$
22: Delete all product nodes with out-degree 0
23: Contract all product nodes with out-degree 1

intersections with $\Omega(v_m)$. We disconnect the nodes in $\mathcal{S}_{\mathbf{V}}$ from their children if their children are indicator variables of a subset of $\Omega(v_m)$ (Lines 15-17). At Line 18, we build a new product node by multiplying all the indicator variables in $\Omega(v_m)$ and link it to v_m directly. To keep unchanged the network polynomials of nodes outside $\mathcal{S}_{v_m}$ that use nodes in $\mathcal{S}_{\mathbf{V}}$, we create a duplicate node p for each such node v and link p to all the parents of v outside of $\mathcal{S}_{v_m}$ and at the same time delete the original link (Lines 9-13).

In summary, Lines 15-17 ensure that v_m is decomposable by removing all the shared indicator variables in $\Omega(v_m)$. Line 18 together with Eq. 2 guarantee that f_{v_m} is unchanged after the transformation. Lines 9-13 create necessary duplicates to ensure that other network polynomials are not affected. Lines 21-23 simplify the transformed SPN to make it more compact. An example is depicted in Fig. 1 to illustrate the transformation process.

We now analyze the size of the SPN constructed by Alg. 1. For a graph $\mathcal{S}$, let $\mathfrak{V}(\mathcal{S})$ be the number of nodes in $\mathcal{S}$ and let $\mathfrak{E}(\mathcal{S})$ be the number of edges in $\mathcal{S}$. Note that in Lines 8-17 we only focus on nodes that appear in the induced SPN $\mathcal{S}_{\mathbf{V}}$, which clearly has $|\mathcal{S}_{\mathbf{V}}| \leq |\mathcal{S}_{v_m}|$. Furthermore, we create a new product node p at Line 10 iff v is reused by other nodes which do not appear in $\mathcal{S}_{v_m}$. This means

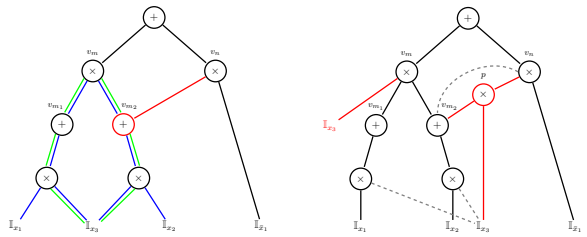

Figure 1. Transformation process described in Alg. 1 to construct a complete and decomposable SPN from a complete and consistent SPN. The product node v_m in the left SPN is not decomposable. Induced sub-SPN $\mathcal{S}_{v_m}$ is highlighted in blue and $\mathcal{S}_{\mathbf{V}}$ is highlighted in green. v_{m_2} highlighted in red is reused by v_n, which is outside $\mathcal{S}_{v_m}$. To compensate for v_{m_2}, we create a new product node p in the right SPN and connect it to indicator variable $\mathbb{I}_{x_3}$ and v_{m_2}. Dashed gray lines in the right SPN denote deleted edges and nodes while red edges and nodes are added during Alg. 1.

that the number of nodes created during each iteration between Lines 2 and 20 is bounded by $\mathfrak{V}(\mathcal{S}_{\mathbf{V}}) \leq \mathfrak{V}(\mathcal{S}_{v_m})$. Line 10 also creates 2 new edges to connect p to v and the indicator variables. Lines 11 and 12 first connect edges to p and then delete edges from v, hence these two steps do not yield increases in the number of edges. So the increase in the number of edges is bounded by $2\mathfrak{V}(\mathcal{S}_{\mathbf{V}}) \leq 2\mathfrak{V}(\mathcal{S}_{v_m})$. Combining increases in both nodes and edges, during each outer iteration the increase in size is bounded by $3|\mathcal{S}_{\mathbf{V}}| \leq 3|\mathcal{S}_{v_m}| = O(|\mathcal{S}|)$. There will be at most $M = \mathfrak{V}(\mathcal{S})$ outer iterations hence the total increase in size will be bounded by $O(M|\mathcal{S}|) = O(|\mathcal{S}|^2)$. □

Lemma 2. For any complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$ that satisfies condition 2 of Def. 7, $\sum_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x}) = 1$.

Proof. We give a proof by induction on the height of $\mathcal{S}$. Let R be the root of $\mathcal{S}$.

- Base case. SPNs of height 0 are indicator variables over some Boolean variable whose network polynomials immediately satisfy Lemma 2.
- Induction step. Assume Lemma 2 holds for any SPN with height $\leq k$. Consider an SPN $\mathcal{S}$ with height $k+1$. We consider the following two cases:
  - The root R of $\mathcal{S}$ is a product node. Then in this case the network polynomial $f_{\mathcal{S}}(\cdot)$ for $\mathcal{S}$ is defined as $f_{\mathcal{S}} = \prod_{v \in Ch(R)} f_v$. We have

$$\sum_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x}) = \sum_{\mathbf{x}} \prod_{v \in Ch(R)} f_v(\mathbf{x}|_{\text{scope}(v)}) \quad (3)$$

$$= \prod_{v \in Ch(R)} \sum_{\mathbf{x}|_{\text{scope}(v)}} f_v(\mathbf{x}|_{\text{scope}(v)}) \quad (4)$$

$$= \prod_{v \in Ch(R)} 1 = 1 \quad (5)$$

    where $\mathbf{x}|_{\text{scope}(v)}$ means that $\mathbf{x}$ is restricted to the set $\text{scope}(v)$. Eq. 4 follows from the decomposability of R and Eq. 5 follows from the induction hypothesis.
  - The root R of $\mathcal{S}$ is a sum node. The network polynomial is $f_{\mathcal{S}} = \sum_{v \in Ch(R)} w_{R,v} f_v$. We have

$$\sum_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x}) = \sum_{\mathbf{x}} \sum_{v \in Ch(R)} w_{R,v} f_v(\mathbf{x}) \quad (6)$$

$$= \sum_{v \in Ch(R)} w_{R,v} \sum_{\mathbf{x}} f_v(\mathbf{x}) \quad (7)$$

$$= \sum_{v \in Ch(R)} w_{R,v} = 1 \quad (8)$$

    Eq. 7 follows from the commutative and associative law of addition and Eq. 8 follows by the induction hypothesis.

□

Corollary 3. For any complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$ that satisfies condition 2 of Def. 7, $\Pr_{\mathcal{S}}(\cdot) = f_{\mathcal{S}}(\cdot)$.

Lemma 4. For any complete and decomposable SPN $\mathcal{S}$, there exists an SPN $\mathcal{S}'$ where the weights of the edges emanating from every sum node are nonnegative and sum to 1, and $\Pr_{\mathcal{S}}(\cdot) = \Pr_{\mathcal{S}'}(\cdot)$, $|\mathcal{S}'| = |\mathcal{S}|$.

Proof. Alg. 2 runs in one pass of $\mathcal{S}$ to construct the required SPN $\mathcal{S}'$.

We proceed to prove that the SPN $\mathcal{S}'$ returned by Alg. 2 satisfies $\Pr_{\mathcal{S}'}(\cdot) = \Pr_{\mathcal{S}}(\cdot)$, $|\mathcal{S}'| = |\mathcal{S}|$ and that $\mathcal{S}'$ satisfies condition 2 of Def. 7. It is clear that $|\mathcal{S}'| = |\mathcal{S}|$ because we only modify the weights of $\mathcal{S}$ to construct $\mathcal{S}'$ at Line 7. Based on Lines 6 and 7, it is also straightforward to verify that for each sum node v in $\mathcal{S}'$, the weights of the edges emanating from v are nonnegative and sum to 1. We now show that $\Pr_{\mathcal{S}'}(\cdot) = \Pr_{\mathcal{S}}(\cdot)$. Using Corollary 3, $\Pr_{\mathcal{S}'}(\cdot) = f_{\mathcal{S}'}(\cdot)$. Hence it is sufficient to show that $f_{\mathcal{S}'}(\cdot) = \Pr_{\mathcal{S}}(\cdot)$. Before deriving a proof, it is helpful to note that for each node $v \in \mathcal{S}$, $val(v) = \sum_{\mathbf{x}|_{\text{scope}(v)}} f_v(\mathbf{x}|_{\text{scope}(v)})$. We give a proof by induction on the height of $\mathcal{S}$.

- Base case. SPNs with height 0 are indicator variables which automatically satisfy Lemma 4.

- Induction step. Assume Lemma 4 holds for any SPN of height $\leq k$. Consider an SPN $\mathcal{S}$ of height $k+1$. Let R be the root node of $\mathcal{S}$ with out-degree l. We discuss the following two cases.
  - R is a product node. Let $R_1, \ldots, R_l$ be the children of R and $\mathcal{S}_1, \ldots, \mathcal{S}_l$ be the corresponding sub-SPNs. By induction, Alg. 2 returns $\mathcal{S}'_1, \ldots, \mathcal{S}'_l$ that satisfy Lemma 4. Since R is a product node, we have

$$f_{\mathcal{S}'}(\mathbf{x}) = \prod_{i=1}^{l} f_{\mathcal{S}'_i}(\mathbf{x}|_{\text{scope}(R_i)}) \tag{9}$$

$$= \prod_{i=1}^{l} \Pr_{\mathcal{S}_i}(\mathbf{x}|_{\text{scope}(R_i)}) \tag{10}$$

$$= \prod_{i=1}^{l} \frac{f_{\mathcal{S}_i}(\mathbf{x}|_{\text{scope}(R_i)})}{\sum_{\mathbf{x}|_{\text{scope}(R_i)}} f_{\mathcal{S}_i}(\mathbf{x}|_{\text{scope}(R_i)})} \tag{11}$$

$$= \frac{\prod_{i=1}^{l} f_{\mathcal{S}_i}(\mathbf{x}|_{\text{scope}(R_i)})}{\sum_{\mathbf{x}} \prod_{i=1}^{l} f_{\mathcal{S}_i}(\mathbf{x}|_{\text{scope}(R_i)})} \tag{12}$$

$$= \frac{f_{\mathcal{S}}(\mathbf{x})}{\sum_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x})} = \Pr_{\mathcal{S}}(\mathbf{x}) \tag{13}$$

    Eq. 10 follows from the induction hypothesis and Eq. 12 follows from the distributive law due to the decomposability of $\mathcal{S}$.
  - R is a sum node with weights $w_1, \ldots, w_l \geq 0$. We have

$$f_{\mathcal{S}'}(\mathbf{x}) = \sum_{i=1}^{l} w'_i f_{\mathcal{S}'_i}(\mathbf{x}) \tag{14}$$

$$= \sum_{i=1}^{l} \frac{w_i val(R_i)}{\sum_{j=1}^{l} w_j val(R_j)} \Pr_{\mathcal{S}_i}(\mathbf{x}) \tag{15}$$

$$= \sum_{i=1}^{l} \frac{w_i val(R_i)}{\sum_{j=1}^{l} w_j val(R_j)} \frac{f_{\mathcal{S}_i}(\mathbf{x})}{\sum_{\mathbf{x}} f_{\mathcal{S}_i}(\mathbf{x})} \tag{16}$$

$$= \sum_{i=1}^{l} \frac{w_i val(R_i)}{\sum_{j=1}^{l} w_j val(R_j)} \frac{f_{\mathcal{S}_i}(\mathbf{x})}{val(R_i)} \tag{17}$$

$$= \frac{\sum_{i=1}^{l} w_i f_{\mathcal{S}_i}(\mathbf{x})}{\sum_{j=1}^{l} w_j val(R_j)} = \frac{f_{\mathcal{S}}(\mathbf{x})}{\sum_{\mathbf{x}} f_{\mathcal{S}}(\mathbf{x})} \tag{18}$$

$$= \Pr_{\mathcal{S}}(\mathbf{x}) \tag{19}$$

    where Eqn. 15 follows from the induction hypothesis, Eq. 17 and 18 follow from the fact that $val(v) = \sum_{\mathbf{x}|_{\text{scope}(v)}} f_v(\mathbf{x}|_{\text{scope}(v)}), \forall v \in \mathcal{S}$.

This completes the proof since $\Pr_{\mathcal{S}'}(\cdot) = f_{\mathcal{S}'}(\cdot) = \Pr_{\mathcal{S}}(\cdot)$. □

Algorithm 2 Weight Normalization

Input: SPN $\mathcal{S}$
Output: SPN $\mathcal{S}'$

```
S' ← S
val(I_x) ← 1, ∀I_x ∈ S
Let v_1, ..., v_M be an inverse topological ordering of the nodes in S
for m = 1 to M do
    if v_m is a sum node then
        val(v_m) ← Σ_{v∈Ch(v_m)} w_{v_m,v} val(v)
        w'_{v_m,v} ← w_{v_m,v} val(v) / val(v_m),   ∀v ∈ Ch(v_m)
    else if v_m is a product node then
        val(v_m) ← Π_{v∈Ch(v_m)} val(v)
    end if
end for
```

Lemma 5. Given a complete and decomposable SPN $\mathcal{S}$, there exists an SPN $\mathcal{S}'$ satisfying condition 3 in Def. 7 such that $\Pr_{\mathcal{S}'}(\cdot) = \Pr_{\mathcal{S}}(\cdot)$ and $|\mathcal{S}'| = O(|\mathcal{S}|)$.

Proof. We give a proof by construction. First, if $\mathcal{S}$ is not weight normalized, apply Alg. 2 to normalize the weights (i.e., the weights of the edges emanating from each sum node sum to 1).

Now check each sum node v in $\mathcal{S}$ in a bottom-up order. If $|\text{scope}(v)| = 1$, by Corollary 3 we know the network polynomial f_v is a probability distribution over its scope, say, $\{X\}$. Reduce v into a terminal node which is a distribution over X induced by its network polynomial and disconnect v from all its children. The last step is to remove all the unreachable nodes from $\mathcal{S}$ to obtain $\mathcal{S}'$. Note that in this step we will only decrease the size of $\mathcal{S}$, hence $|\mathcal{S}'| = O(|\mathcal{S}|)$. □

Proof of Thm. 3. The combination of Lemma 1, 4 and 5 completes the proof of Thm. 3. □

3. Proof of Proposition 1

Proposition 1. Given a normal SPN $\mathcal{S}$, let p be a product node in $\mathcal{S}$ with l children. Let $v_1, \ldots, v_k$ be sum nodes which lie on a path from the root of $\mathcal{S}$ to p. Then

$$\Pr_{\mathcal{S}}(\mathbf{x}|_{\text{scope}(p)} \mid H_{v_1} = v_1^*, \ldots, H_{v_k} = v_k^*) = \prod_{i=1}^{l} \Pr_{\mathcal{S}}(\mathbf{x}|_{\text{scope}(p_i)} \mid H_{v_1} = v_1^*, \ldots, H_{v_k} = v_k^*) \tag{20}$$

where $H_v = v^*$ means the sum node v selects its v^*th branch and $\mathbf{x}|_A$ denotes restricting $\mathbf{x}$ by set A, p_i is the ith child of product node p.

Proof. Consider the sub-SPN $\mathcal{S}_p$ rooted at p. $\mathcal{S}_p$ can be obtained by restricting $H_{v_1} = v_1^*, \ldots, H_{v_k} = v_k^*$, i.e., going from the root of $\mathcal{S}$ along the path $H_{v_1} = v_1^*, \ldots, H_{v_k} = v_k^*$. Since p is a decomposable product node, $\mathcal{S}_p$ admits the above factorization by the definition of a product node and Corollary 3. □

4. Proof of Theorem 4

We first list the construction algorithms presented in the main paper and show the following lemma:

Algorithm 3 Build BN Structure

Input: normal SPN $\mathcal{S}$
Output: BN $\mathcal{B} = (\mathcal{B}_V, \mathcal{B}_E)$

1: $R \leftarrow$ root of $\mathcal{S}$
2: **if** R is a terminal node over variable X **then**
3: Create an observable variable X
4: $\mathcal{B}_V \leftarrow \mathcal{B}_V \cup \{X\}$
5: **else**
6: **for** each child R_i of R **do**
7: **if** BN has not been built for $\mathcal{S}_{R_i}$ **then**
8: Recursively build BN Structure for $\mathcal{S}_{R_i}$
9: **end if**
10: **end for**
11: **if** R is a sum node **then**
12: Create a hidden variable H_R associated with R
13: $\mathcal{B}_V \leftarrow \mathcal{B}_V \cup \{H_R\}$
14: **for** each observable variable $X \in \mathcal{S}_R$ **do**
15: $\mathcal{B}_E \leftarrow \mathcal{B}_E \cup \{(H_R, X)\}$
16: **end for**
17: **end if**
18: **end if**

Lemma 6. Given a normal SPN $\mathcal{S}$, the ADDs constructed by Alg. 4 and 5 encode local CPDs at each node in $\mathcal{B}$.

Proof. It is easy to verify that for each hidden variable H in $\mathcal{B}$, $\mathcal{A}_H$ represents a local CPD since $\mathcal{A}_H$ is a decision stump with normalized weights.

For any observable variable X in $\mathcal{B}$, let $Pa(X)$ be the set of parents of X. By Alg. 3, every node in $Pa(X)$ is a hidden variable. Furthermore, $\forall H$, $H \in Pa(X)$ iff there exists one terminal node over X in $\mathcal{S}$ that appears in the sub-SPN rooted at H. Hence given any joint assignment $\mathbf{h}$ of $Pa(X)$, there will be a path in $\mathcal{A}_X$ from the root to a terminal node that is consistent with the joint assignment of the parents. Also, the leaves in $\mathcal{A}_X$ contain normalized weights corresponding to the probabilities of X (see Def. 7) induced by the creation of decision stumps over X in Lines 5-6 of Alg. 4. □

Theorem 4. For any normal SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$, the BN $\mathcal{B}$ constructed by Alg. 3, 4 and 5 encodes the same probability distribution, i.e., $\Pr_{\mathcal{S}}(\mathbf{x}) = \Pr_{\mathcal{B}}(\mathbf{x}), \forall \mathbf{x}$.

Algorithm 4 Build CPD using ADD, observable variable

Input: normal SPN $\mathcal{S}$, variable X
Output: ADD $\mathcal{A}_X$

1: **if** ADD has already been created for $\mathcal{S}$ and X **then**
2: $\mathcal{A}_X \leftarrow$ retrieve ADD from cache
3: **else**
4: $R \leftarrow$ root of $\mathcal{S}$
5: **if** R is a terminal node **then**
6: $\mathcal{A}_X \leftarrow$ decision stump rooted at R
7: **else if** R is a sum node **then**
8: Create a node H_R into $\mathcal{A}_X$
9: **for** each $R_i \in Ch(R)$ **do**
10: Link BuildADD$(\mathcal{S}_{R_i}, X)$ as ith child of H_R
11: **end for**
12: **else if** R is a product node **then**
13: Find child $\mathcal{S}_{R_i}$ such that $X \in \text{scope}(R_i)$
14: $\mathcal{A}_X \leftarrow$ BuildADD$(\mathcal{S}_{R_i}, X)$
15: **end if**
16: store $\mathcal{A}_X$ in cache
17: **end if**

Algorithm 5 Build CPD using ADD, hidden variable

Input: normal SPN $\mathcal{S}$, variable H
Output: ADD $\mathcal{A}_H$

1: Find the sum node H in $\mathcal{S}$
2: $\mathcal{A}_H \leftarrow$ decision stump rooted at H in $\mathcal{S}$

Proof. Again, we give a proof by induction on the height of $\mathcal{S}$.

- Base case. The height of SPN $\mathcal{S}$ is 0. In this case, $\mathcal{S}$ will be a single terminal node over X and $\mathcal{B}$ will be a single observable node with decision stump $\mathcal{A}_X$ constructed from the terminal node by Lines 5-6 in Alg. 4. It is clear that $\Pr_{\mathcal{S}}(x) = \Pr_{\mathcal{B}}(x), \forall x$.
- Induction step. Assume $\Pr_{\mathcal{B}}(\mathbf{x}) = \Pr_{\mathcal{S}}(\mathbf{x}), \forall \mathbf{x}$ for any $\mathcal{S}$ with height $\leq k$, where $\mathcal{B}$ is the corresponding BN constructed by Alg. 3, 4 and 5 from $\mathcal{S}$. Consider an SPN $\mathcal{S}$ with height $k+1$. Let R be the root of $\mathcal{S}$ and $R_i, i \in 1 : l$ be the children of R in $\mathcal{S}$. We consider the following two cases:
  - R is a product node. Let $\text{scope}(R_t) = \mathbf{X}_t, t \in 1 : l$. Claim: there is no edge between $\mathcal{S}_i$ and $\mathcal{S}_j, i \neq j$, where $\mathcal{S}_i(\mathcal{S}_j)$ is the sub-SPN rooted at $R_i(R_j)$. If there is an edge, say, from v_j to v_i where $v_j \in \mathcal{S}_j$ and $v_i \in \mathcal{S}_i$, then $\text{scope}(v_i) \subseteq \text{scope}(v_j) \subseteq \text{scope}(R_j)$. On the other hand, $\text{scope}(v_i) \subseteq \text{scope}(R_i)$. So we have $\varnothing \neq \text{scope}(v_i) \subseteq \text{scope}(R_i) \bigcap \text{scope}(R_j)$, which contradicts the decomposability of the

product node R. Hence the constructed BN $\mathcal{B}$ will be a forest of l disconnected components, and each component $\mathcal{B}_t$ will correspond to the sub-SPN $\mathcal{S}_t$ rooted at $R_t, \forall t \in 1 : l$, with height $\leq k$. By the induction hypothesis we have $\Pr_{\mathcal{B}_t}(\mathbf{x}_t) = \Pr_{\mathcal{S}_t}(\mathbf{x}_t), \forall t \in 1 : l$. Consider the whole BN $\mathcal{B}$, we have:

$$\Pr_{\mathcal{B}}(\mathbf{x}) = \prod_t \Pr_{\mathcal{B}_t}(\mathbf{x}_t) = \prod_t \Pr_{\mathcal{S}_t}(\mathbf{x}_t) = \Pr_{\mathcal{S}}(\mathbf{x})$$

where the first equation is due to the d-separation rule in BNs by noting that each component $\mathcal{B}_t$ is disconnected from all other components. The second equation follows from the induction hypothesis. The last equation follows from the definition of a product node.

– R is a sum node. In this case, due to the completeness of $\mathcal{S}$, all the children of R share the same scope as R. By the construction process presented in Alg. 3, 4 and 5, there is a hidden variable H corresponding to R that takes l different values in $\mathcal{B}$. Let $w_{1:l}$ be the weights of the edges emanating from R in $\mathcal{S}$. For the tth branch of R, we use $\mathbf{H}_t$ to denote the set of hidden variables in $\mathcal{B}$ that also appear in $\mathcal{B}_t$, and let $\mathbf{H}_{-t} = \mathbf{H} \backslash \mathbf{H}_t$, where $\mathbf{H}$ is the set of all hidden variables in $\mathcal{B}$ except H. First, we show the following identity:

$$\begin{aligned}
\Pr_{\mathcal{B}}(\mathbf{x}|h_t) &= \sum_{\mathbf{h}_t} \sum_{\mathbf{h}_{-t}} \Pr_{\mathcal{B}}(\mathbf{x}, \mathbf{h}_t, \mathbf{h}_{-t}|H = h_t) \\
&= \sum_{\mathbf{h}_t} \sum_{\mathbf{h}_{-t}} \Pr_{\mathcal{B}}(\mathbf{x}, \mathbf{h}_t|h_t, \mathbf{h}_{-t}) \Pr_{\mathcal{B}}(\mathbf{h}_{-t}|h_t) \\
&= \sum_{\mathbf{h}_t} \sum_{\mathbf{h}_{-t}} \Pr_{\mathcal{B}}(\mathbf{x}, \mathbf{h}_t|h_t) \Pr_{\mathcal{B}}(\mathbf{h}_{-t}|h_t) \qquad (21) \\
&= \sum_{\mathbf{h}_t} \Pr_{\mathcal{B}}(\mathbf{x}, \mathbf{h}_t|h_t) \sum_{\mathbf{h}_{-t}} \Pr_{\mathcal{B}}(\mathbf{h}_{-t}|h_t) \qquad (22) \\
&= \sum_{\mathbf{h}_t} \Pr_{\mathcal{B}}(\mathbf{x}, \mathbf{h}_t|h_t) \\
&= \sum_{\mathbf{h}_t} \Pr_{\mathcal{B}_t}(\mathbf{x}, \mathbf{h}_t) = \Pr_{\mathcal{B}_t}(\mathbf{x}) \qquad (23)
\end{aligned}$$

Using this identity, we have

$$\begin{aligned}
\Pr_{\mathcal{B}}(\mathbf{x}) &= \sum_{t=1}^{l} \Pr_{\mathcal{B}}(h_t) \Pr_{\mathcal{B}}(\mathbf{x}|H = h_t) \qquad (24) \\
&= \sum_{t=1}^{l} w_t \Pr_{\mathcal{B}_t}(\mathbf{x}) \qquad (25) \\
&= \sum_{t=1}^{l} w_t \Pr_{\mathcal{S}_t}(\mathbf{x}) = \Pr_{\mathcal{S}}(\mathbf{x}) \qquad (26)
\end{aligned}$$

Eq. 21 follows from the fact that $\mathbf{X}$ and $\mathbf{H}_t$ are independent of $\mathbf{H}_{-t}$ given $H = h_t$, i.e., we take advantage of the CSI described by ADDs of $\mathbf{X}$. Eq. 22 follows from the fact that $\mathbf{H}_{-t}$ appears only in the second term. Combined with the fact that $H = h_t$ is given as evidence in $\mathcal{B}$, this gives us the induced subgraph $\mathcal{B}_t$ referred to in Eq. 23. Eq. 25 follows from Eq. 23 and Eq. 26 follows from the induction hypothesis.

Combing the base case and the induction step completes the proof for Thm. 4. □

5. Proof of Theorem 5

Theorem 5. $|\mathcal{B}| = O(N|\mathcal{S}|)$, where BN $\mathcal{B}$ is constructed by Alg. 3, 4 and 5 from normal SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$.

Proof. For each observable variable X in $\mathcal{B}$, $\mathcal{A}_X$ is constructed by first extracting from $\mathcal{S}$ the induced sub-SPN $\mathcal{S}_X$ that contains all nodes whose scope includes X and then contracting all the product nodes in $\mathcal{S}_X$ to obtain $\mathcal{A}_X$. By the decomposability of product nodes, each product node in $\mathcal{S}_X$ has out-degree 1 otherwise the original SPN $\mathcal{S}$ violates the decomposability property. Since contracting product nodes does not increase the number of edges in $\mathcal{S}_X$, we have $|\mathcal{A}_X| \leq |\mathcal{S}_X| \leq |\mathcal{S}|$.

For each hidden variable H in $\mathcal{B}$, $\mathcal{A}_H$ is a decision stump constructed from the internal sum node corresponding to H in $\mathcal{S}$. Hence, we have $\sum_H \mathcal{A}_H \leq |\mathcal{S}|$.

Now consider the size of the graph $\mathcal{B}$. Note that only terminal nodes and sum nodes will have corresponding variables in $\mathcal{B}$. It is clear that the number of nodes in $\mathcal{B}$ is bounded by the number of nodes in $\mathcal{S}$. Furthermore, a hidden variable H points to an observable variable X in $\mathcal{B}$ iff X appears in the sub-SPN rooted at H in $\mathcal{S}$, i.e., there is a path from the sum node corresponding to H to one of the terminal nodes in X. For a sum node H (which corresponds to a hidden variable $H \in \mathcal{B}$) with scope size s, each edge emanating from H in $\mathcal{S}$ will correspond to directed edges in $\mathcal{B}$ at most s times, since there are exactly s observable variables which are children of H in $\mathcal{B}$. It is clear that $s \leq N$, so each edge emanating from a sum node in $\mathcal{S}$ will be counted at most N times in $\mathcal{B}$. Edges from product nodes will not occur in the graph of $\mathcal{B}$, instead, they have been counted in the ADD representations of the local CPDs in $\mathcal{B}$. So again, the size of the graph $\mathcal{B}$ is bounded by $\sum_H \text{scope}(H) \times \deg(H) \leq \sum_H N \deg(H) \leq 2N|\mathcal{S}|$.

There are N observable variables in $\mathcal{B}$. So the total size of $\mathcal{B}$, including the size of the graph and the size of all the ADDs, is bounded by $N|\mathcal{S}|+|\mathcal{S}|+2N|\mathcal{S}| = O(N|\mathcal{S}|)$. □

6. Proof of Theorem 6

Theorem 6. For any normal SPN $\mathcal{S}$ over $\mathbf{X}_{1:N}$, Alg. 3, 4 and 5 construct an equivalent BN in time $O(N|\mathcal{S}|)$.

Proof. First consider Alg. 3. Alg. 3 recursively visits each node and its children in $\mathcal{S}$ if they have not been visited (Lines 6-10). For each node v in $\mathcal{S}$, Lines 7-9 cost at most $2 \cdot$ out-degree(v). If v is a sum node, then Lines 11-17 create a hidden variable and then connect the hidden variable to all observable variables that appear in the sub-SPN rooted at v, which is clearly bounded by the number of all observable variables, N. So the total cost of Alg. 3 is bounded by $\sum_v 2 \cdot \text{out-degree}(v) + \sum_{v \text{ is a sum node}} N \leq 2\mathfrak{V}(\mathcal{S}) + 2\mathfrak{E}(\mathcal{S}) + N\mathfrak{V}(\mathcal{S}) \leq 2|\mathcal{S}| + N|\mathcal{S}| = O(N|\mathcal{S}|)$. Note that we assume that inserting an element into a set can be done in $O(1)$ by using hashing.

The analysis for Alg. 4 and 5 follows from the same analysis as in the proof for Thm. 5. The time complexity for Alg. 4 and Alg. 5 is then bounded by $N|\mathcal{S}| + |\mathcal{S}| = O(N|\mathcal{S}|)$. □

7. Multiplication and Summing-Out of Symbolic ADD

Here we provide the algorithms to multiply two symbolic ADDs and to sum out one hidden variable in a symbolic ADD. Then we analyze the time and space complexity of these operations in detail. These two operations will be applied in the Variable Elimination algorithm in Alg. 8.

Given two symbolic ADDs $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$, Alg. 6 recursively visits nodes in $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ simultaneously. In general, there are 3 cases: 1) the roots of $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ are both variable nodes (Lines 2-14); 2) one of the two roots is a variable node and the other is a product node (Lines 15-30); 3) both roots are product nodes or at least one of them is a sum node (Lines 31-34). We discuss these 3 cases.

If both roots of $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ are variable nodes, there are two subcases to be considered. First, if they are nodes labeled with the same variable (Lines 3-10), then the computation related to the common variable is shared and the multiplication is recursively applied to all the children, otherwise we simply create a symbolic product node $\otimes$ and link $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ as its two children (Lines 11-14). Once we find $R_1 \in \mathcal{A}_{X_1}$ and $R_2 \in \mathcal{A}_{X_2}$ such that $R_1 \neq R_2$, there will be no common node that is shared by the sub-ADDs rooted at R_1 and R_2. To see this, note that Alg. 6 recursively calls itself as long as the roots of $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ are labeled with the same variable. Let R be the last variable shared by the roots of $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ in Alg. 6. Then R_1 and R_2 must be the children of R in the original SPN $\mathcal{S}$. Since R_1 does not appear in $\mathcal{A}_{X_2}$, then $X_2 \notin \text{scope}(R_1)$, otherwise R_1 will occur in $\mathcal{A}_{X_2}$ and R_1 will be a new shared variable below R, which is a contradiction to the fact that R is the last shared variable. Since R_1 is the root of the sub-ADD of $\mathcal{A}_{X_1}$ rooted at R, hence no variable whose scope contains X_2 will occur as a descendant of R_1, otherwise the scope of R_1 will also contain X_2, which is again a contradiction. On the other hand, each node appearing in $\mathcal{A}_{X_2}$ corresponds to a variable whose scope intersects with $\{X_2\}$ in the original SPN, hence no node in $\mathcal{A}_{X_2}$ will appear in $\mathcal{A}_{X_1}$. The same analysis also applies to R_2. Hence no node will be shared between $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$.

If one of the two roots, say, R_1, is a variable node and the other root, say, R_2, is a product node, then we consider two subcases. If R_1 appears as a child of R_2 then we recursively

Algorithm 6 Multiplication of two symbolic ADDs, $\otimes$

Input: Symbolic ADD $\mathcal{A}_{X_1}$, $\mathcal{A}_{X_2}$
Output: Symbolic ADD $\mathcal{A}_{X_1,X_2} = \mathcal{A}_{X_1} \otimes \mathcal{A}_{X_2}$
1: $R_1 \leftarrow$ root of $\mathcal{A}_{X_1}$, $R_2 \leftarrow$ root of $\mathcal{A}_{X_2}$
2: **if** R_1 and R_2 are both variable nodes **then**
3: **if** $R_1 = R_2$ **then**
4: Create a node $R = R_1$ into $\mathcal{A}_{X_1,X_2}$
5: **for** each $r \in dom(R)$ **do**
6: $\mathcal{A}^r_{X_1} \leftarrow Ch(R_1)|_r$
7: $\mathcal{A}^r_{X_2} \leftarrow Ch(R_2)|_r$
8: $\mathcal{A}^r_{X_1,X_2} \leftarrow \mathcal{A}^r_{X_1} \otimes \mathcal{A}^r_{X_2}$
9: Link $\mathcal{A}^r_{X_1,X_2}$ to the rth child of R in $\mathcal{A}_{X_1,X_2}$
10: **end for**
11: **else**
12: $\mathcal{A}_{X_1,X_2} \leftarrow$ create a symbolic node $\otimes$
13: Link $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ as two children of $\otimes$
14: **end if**
15: **else if** R_1 is a variable node and R_2 is $\otimes$ **then**
16: **if** R_1 appears as a child of R_2 **then**
17: $\mathcal{A}_{X_1,X_2} \leftarrow \mathcal{A}_{X_2}$
18: $\mathcal{A}^{R_1}_{X_1,X_2} \leftarrow \mathcal{A}_{X_1} \otimes \mathcal{A}^{R_1}_{X_2}$
19: **else**
20: Link $\mathcal{A}_{X_1}$ as a new child of R_2
21: $\mathcal{A}_{X_1,X_2} \leftarrow \mathcal{A}_{X_2}$
22: **end if**
23: **else if** R_1 is $\otimes$ and R_2 is a variable node **then**
24: **if** R_2 appears as a child of R_1 **then**
25: $\mathcal{A}_{X_1,X_2} \leftarrow \mathcal{A}_{X_1}$
26: $\mathcal{A}^{R_2}_{X_1,X_2} \leftarrow \mathcal{A}_{X_2} \otimes \mathcal{A}^{R_2}_{X_1}$
27: **else**
28: Link $\mathcal{A}_{X_2}$ as a new child of R_1
29: $\mathcal{A}_{X_1,X_2} \leftarrow \mathcal{A}_{X_1}$
30: **end if**
31: **else**
32: $\mathcal{A}_{X_1,X_2} \leftarrow$ create a symbolic node $\otimes$
33: Link $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ as two children of $\otimes$
34: **end if**
35: Merge connected product nodes in $\mathcal{A}_{X_1,X_2}$

multiply R_1 with the child of R_2 that is labeled with the same variable as R_1 (Lines 16-18). If R_1 does not appear as a child of R_2, then we link the ADD rooted at R_1 to be a new child of the product node R_2 (Lines 19-22). Again, let R be the last shared node between $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ during the multiplication process. Then both R_1 and R_2 are children of R, which corresponds to a sum node in the original SPN $\mathcal{S}$. Furthermore, both R_1 and R_2 lie in the same branch of R in $\mathcal{S}$. In this case, since $\text{scope}(R_1) \subseteq \text{scope}(R)$, $\text{scope}(R_1)$ must be a strict subset of $\text{scope}(R)$ otherwise we would have $\text{scope}(R_1) = \text{scope}(R)$ and R_1 will also appear in $\mathcal{A}_{X_2}$, which contradicts the fact that R is the last shared node between $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$. Hence, here we only need to discuss the two cases where either their scope is disjoint (Lines 16-18) or the scope of one root is a strict subset of another (Lines 19-22).

If the two roots are both product nodes or at least one of them is a sum node, then we simply create a new product node and link $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ to be children of the product node. The above analysis also applies here since sum nodes in symbolic ADDs are created by summing out processed variable nodes and we eliminate all the hidden variables using the inverse topological ordering.

The last step in Alg. 6 (Line 35) simplifies the symbolic ADD by merging all the connected product nodes without changing the function it encodes. This can be done in the following way: suppose $\otimes_1$ and $\otimes_2$ are two connected product nodes in symbolic ADD $\mathcal{A}$ where $\otimes_1$ is the parent of $\otimes_2$, then we can remove the link between $\otimes_1$ and $\otimes_2$ and connect $\otimes_1$ to every child of $\otimes_2$. It is easy to verify that such an operation will remove links between connected product nodes while keeping the encoded function unchanged.

To sum-out one hidden variable H, Alg. 7 replaces H in $\mathcal{A}$ by a symbolic sum node $\oplus$ and labels each edge of $\oplus$ with weights obtained from $\mathcal{A}_H$.

Algorithm 7 Summing-out a hidden variable H from $\mathcal{A}$ using $\mathcal{A}_H$, $\oplus$

Input: Symbolic ADDs $\mathcal{A}$ and $\mathcal{A}_H$
Output: Symbolic ADD with H summed out
1: **if** H appears in $\mathcal{A}$ **then**
2: Label each edge emanating from H with weights obtained from $\mathcal{A}_H$
3: Replace H by a symbolic $\oplus$ node
4: **end if**

Note that Alg. 6 and 7 apply only to ADDs constructed from normal SPNs by Alg. 4 and 5 because such ADDs naturally inherit the topological ordering of sum nodes (hidden variables) in the original SPN $\mathcal{S}$. Otherwise we need to predefine a global variable ordering of all the sum nodes and then arrange each ADD such that its topological ordering is consistent with the pre-defined ordering. Note also that Alg. 6 and 7 should be implemented with caching of repeated operations in order to ensure that directed acyclic graphs are preserved.

8. Proof of Theorem 7

Theorem 7. Alg. 8 builds SPN $\mathcal{S}$ from BN $\mathcal{B}$ with ADDs in $O(N|\mathcal{S}|)$.

Proof. First, it is easy to verify that Alg. 6 takes at most $|\mathcal{A}_{X_1}| + |\mathcal{A}_{X_2}|$ operations to compute the multiplication of $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$. More importantly, the size of the generated $\mathcal{A}_{X_1,X_2}$ is also bounded by $|\mathcal{S}|$. This is because all the common nodes and edges in $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ are shared (not duplicated) in $\mathcal{A}_{X_1,X_2}$. Also, all the other nodes and edges which are not shared between $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ will be in two branches of a product node in $\mathcal{S}$, otherwise they will be shared by $\mathcal{A}_{X_1}$ and $\mathcal{A}_{X_2}$ as they have the same scope which contain both X_1 and X_2. This means that $\mathcal{A}_{X_1,X_2}$ can be viewed as a sub-SPN of $\mathcal{S}$ induced by the node set $\{X_1, X_2\}$ with some product nodes contracted out. So we have $|\mathcal{A}_{X_1,X_2}| \leq |\mathcal{S}|$.

Now consider the for loop (Lines 3-6) in Alg. 8. The loop ends once we have summed out all the hidden variables and there is only one ADD left. Note that there may be only one ADD in Φ during some intermediate steps, in which case we do not have to do any multiplication. In such steps, we only need to perform the sum out procedure without multiplying ADDs. Since there are N ADDs at the beginning of the loop and after the loop we only have one ADD, then there is exactly $N-1$ multiplications during the for loop, which costs at most $(N-1)|\mathcal{S}|$ operations. Furthermore, in each iteration there is exactly one hidden variable being summed out. So the total cost for summing out all the hidden variables in Lines 3-6 is bounded by $|\mathcal{S}|$.

Overall, the operations in Alg. 8 are bounded by $(N-1)|\mathcal{S}| + |\mathcal{S}| = O(N|\mathcal{S}|)$. □

Algorithm 8 Variable Elimination for BN with ADDs

Input: BN $\mathcal{B}$ with ADDs for all observable variables and hidden variables
Output: Original SPN $\mathcal{S}$
1: $\pi \leftarrow$ the inverse topological ordering of all the hidden variables present in the ADDs
2: $\Phi \leftarrow \{\mathcal{A}_X \mid X \text{ is an observable variable}\}$
3: **for** each hidden variable H in π **do**
4: $P \leftarrow \{\mathcal{A}_X \mid H \text{ appears in } \mathcal{A}_X\}$
5: $\Phi \leftarrow \Phi \backslash P \cup \{\oplus_H \otimes_{\mathcal{A} \in P} \mathcal{A}\}$
6: **end for**
7: **return** Φ

9. Proof of Main Theorems

Theorem 1. There exists an algorithm that converts any complete and decomposable SPN $\mathcal{S}$ over Boolean variables $\mathbf{X}_{1:N}$ into a BN $\mathcal{B}$ with CPDs represented by ADDs in time $O(N|\mathcal{S}|)$. Furthermore, $\mathcal{S}$ and $\mathcal{B}$ represent the same distribution and $|\mathcal{B}| = O(N|\mathcal{S}|)$.

Proof. The combination of Thm. 4, 5 and 6 proves Thm. 1. □

Corollary 1. There exists an algorithm that converts any complete and consistent SPN $\mathcal{S}$ over Boolean variables $\mathbf{X}_{1:N}$ into a BN $\mathcal{B}$ with CPDs represented by ADDs in time $O(N|\mathcal{S}|^2)$. Furthermore, $\mathcal{S}$ and $\mathcal{B}$ represent the same distribution and $|\mathcal{B}| = O(N|\mathcal{S}|^2)$.

Proof. Given a complete and consistent SPN $\mathcal{S}$, we can first transform it into a normal SPN $\mathcal{S}'$ with $|\mathcal{S}'| = O(|\mathcal{S}|^2)$ if it is not normal. After this the analysis follows from Thm. 1. □

Theorem 2. Given the BN $\mathcal{B}$ with ADD representation of CPDs generated from a complete and decomposable SPN $\mathcal{S}$ over Boolean variables $\mathbf{X}_{1:N}$, the original SPN $\mathcal{S}$ can be recovered by applying the Variable Elimination algorithm to $\mathcal{B}$ in $O(N|\mathcal{S}|)$.

Proof. Thm. 2 follows from Thm. 7 and the analysis for Alg. 6 and Alg. 7. □