

---

# On Theoretical Properties of Sum-Product Networks

---

**Robert Peharz**  
BEE-PRI BioTechMed-Graz  
iDN, Inst. of Physiology  
Medical University of Graz

**Sebastian Tschiatschek**  
Dept. of Computer Science  
ETH Zurich

**Franz Pernkopf**  
SPSC Lab  
Graz University  
of Technology

**Pedro Domingos**  
Dept. of Computer Science  
and Engineering  
University of Washington

## Abstract

Sum-product networks (SPNs) are a promising avenue for probabilistic modeling and have been successfully applied to various tasks. However, some theoretic properties about SPNs are not yet well understood. In this paper we fill some gaps in the theoretic foundation of SPNs. First, we show that the weights of any complete and consistent SPN can be transformed into locally normalized weights without changing the SPN distribution. Second, we show that consistent SPNs cannot model distributions significantly (exponentially) more compactly than decomposable SPNs. As a third contribution, we extend the inference mechanisms known for SPNs with finite states to generalized SPNs with arbitrary input distributions.

## 1 INTRODUCTION

Sum-product networks (SPNs) are a promising type of probabilistic model and showed impressive results for image completion [Poon and Domingos, 2011, Dennis and Ventura, 2012, Peharz et al., 2013], computer vision [Amer and Todorovic, 2012], classification [Gens and Domingos, 2012] and speech/language modeling [Peharz et al., 2014, Cheng et al., 2014]. The potentially deep structure of SPNs allows them to capture complex interaction of the model variables, while at the same time still guaranteeing efficient inference. Inference cost is directly related to the network size, opening the door for inference-aware learning. However, some aspects of SPNs are not yet well understood and there are many open questions worth investigating. In this paper, we revisit the foundations of SPNs

---

Appearing in Proceedings of the 18<sup>th</sup> International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. Copyright 2015 by the authors.

and present new theoretic insights, consisting of three main contributions.

The first contribution is concerned with normalized weights: SPNs have been defined to have *nonnegative* parameters (sum weights), where most work so far has additionally assumed locally normalized parameters, i.e. that for each sum node the associated weights sum up to 1. This assures that the SPN distribution is readily correctly normalized. Up to now it is not clear if this somehow restricts our modeling abilities, i.e. are there “fewer” locally normalized SPNs than unnormalized SPNs? The answer is no, and furthermore we provide an algorithm which transforms any unnormalized parameter set into a locally normalized one, *without changing the modeled distribution*. Thus, from a representational point of view we can always assume locally normalized parameters.

As a second contribution we investigate the two notions of *consistency* and *decomposability*, either of which is sufficient to make inference in SPNs efficient, given that the SPN is also *complete*. Consistency is the more general condition, i.e. any decomposable SPN is also consistent, but not vice versa. So far, only decomposable SPNs have been considered in practice, since decomposability is easier to ensure. An interesting question is: How much do we lose when we go for decomposability? Can we model our distribution exponentially more concise, using consistency? The answer we give is negative. We show that any distribution represented by a consistent SPN can also be represented by a decomposable SPN, using only polynomially more arithmetic operations. Furthermore, we show that consistent SPNs are *not amenable* to the differential approach [Darwiche, 2003], i.e. simultaneously obtaining all marginal posteriors given evidence. This fact was not mentioned in the literature so far. Decomposable SPNs, on the other hand, represent network polynomials and *can* be used in the differential approach.

SPNs were originally defined for finitely many states using indicator variables as leaves [Poon and Domin-

gos, 2011]. However, SPNs can be generalized and equipped with arbitrary distributions as inputs. As a third contribution, we extend the inference mechanisms known for finite state SPNs to these generalized SPNs. We show that marginalization in generalized SPNs reduces to marginalization over the input distributions. Furthermore, we propose a generalized differential approach, which yields the marginal posteriors of all variables given some evidence.

Throughout the paper, we use the following notation. Random variables (RVs) are denoted as  $X$ ,  $Y$  and  $Z$ . The set of values an RV  $X$  can assume is  $\mathbf{val}(X)$ , where corresponding lower-case letters denote their values, e.g.  $x$  is an element of  $\mathbf{val}(X)$ . Sets of random variables are denoted by boldface letters, e.g.  $\mathbf{X} = \{X_1, \dots, X_N\}$ . We define  $\mathbf{val}(\mathbf{X})$  as the set of compound values, i.e. the Cartesian product  $\mathbf{val}(\mathbf{X}) = \times_{n=1}^N \mathbf{val}(X_n)$ , and use  $\mathbf{x}$  for elements of  $\mathbf{val}(\mathbf{X})$ . For  $\mathbf{Y} \subseteq \mathbf{X}$  and  $X \in \mathbf{X}$ ,  $\mathbf{x}[\mathbf{Y}]$  and  $\mathbf{x}[X]$  denote the projections of  $\mathbf{x}$  onto  $\mathbf{Y}$  and onto  $X$ , respectively. The elements of  $\mathbf{val}(\mathbf{X})$  represent *complete* evidence, assigning each RV in  $\mathbf{X}$  a value. *Partial* evidence about  $X$  is represented as *subsets*  $\mathcal{X} \subseteq \mathbf{val}(X)$ , which is an element of the sigma-algebra  $\mathcal{A}_X$  induced by RV  $X$ . For discrete RVs, we assume  $\mathcal{A}_X = 2^{\mathbf{val}(X)}$ , i.e. the power-set of  $\mathbf{val}(X)$ . For continuous RVs, we use  $\mathcal{A}_X = \{\mathcal{X} \in \mathcal{B} \mid \mathcal{X} \subseteq \mathbf{val}(X)\}$ , where  $\mathcal{B}$  are the Borel-sets over  $\mathbb{R}$ . For sets of RVs  $\mathbf{X} = \{X_1, \dots, X_N\}$ , we use the product sets  $\mathcal{H}_{\mathbf{X}} := \{\times_{n=1}^N \mathcal{X}_n \mid \mathcal{X}_n \in \mathcal{A}_{X_n}\}$  to represent partial evidence about  $\mathbf{X}$ . Elements of  $\mathcal{H}_{\mathbf{X}}$  are denoted as  $\mathcal{X}$ . When  $\mathbf{Y} \subseteq \mathbf{X}$ ,  $X \in \mathbf{X}$  and  $\mathcal{X} \in \mathcal{H}_{\mathbf{X}}$ , we define  $\mathcal{X}[\mathbf{Y}] := \{\mathbf{x}[\mathbf{Y}] \mid \mathbf{x} \in \mathcal{X}\}$  and  $\mathcal{X}[X] = \{x \mid \mathbf{x} \in \mathcal{X}\}$ . An example for partial evidence is provided in the supplementary paper.

The paper is organized as follows: In Section 3 we discuss SPNs over random variables with finitely many states. Inference for generalized SPNs is discussed in Section 4. Section 5 concludes the paper. In the main paper, we provide proofs only for the main results. Proofs for all results can be found in the supplementary paper.

## 2 RELATED WORK

Darwiche [2003] introduced *network polynomials* (NPs) for Bayesian networks over RVs  $\mathbf{X}$  with finitely many states. Poon and Domingos [2011] generalized them to unnormalized distributions, i.e. any nonnegative function  $\Phi(\mathbf{x})$  where  $\exists \mathbf{x}: \Phi(\mathbf{x}) > 0$ . We introduce the so-called indicator variables (IVs) for each RV and each state, where  $\lambda_{X=x} \in \mathbb{R}$  is the IV for  $X$  and state  $x$ . Let  $\boldsymbol{\lambda}$  be a vector collecting all IVs of  $\mathbf{X}$ .

**Definition 1** (Network Polynomial). *Let  $\Phi$  be an unnormalized probability distribution over RVs  $\mathbf{X}$  with*

*finitely many states and  $\boldsymbol{\lambda}$  be their IVs. The network polynomial  $f_{\Phi}$  of  $\Phi$  is defined as*

$$f_{\Phi}(\boldsymbol{\lambda}) := \sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \Phi(\mathbf{x}) \prod_{X \in \mathbf{X}} \lambda_{X=\mathbf{x}[X]}. \quad (1)$$

The NP contains exponentially many term in the number of RVs. As shown by Darwiche [2003], it represents the distribution  $\Phi$  in the following sense. Restrict the IVs to  $\{0, 1\}$  and as a function of  $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ :

$$\lambda_{X=x}(\mathbf{x}) = \begin{cases} 1 & \text{if } x = \mathbf{x}[X] \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Let  $\boldsymbol{\lambda}(\mathbf{x})$  be the corresponding vector-valued function, collecting all  $\lambda_{X=x}(\mathbf{x})$ . When we input  $\boldsymbol{\lambda}(\mathbf{x})$  to  $f_{\Phi}$ , all but one of the terms evaluate to 0, i.e.  $f_{\Phi}(\boldsymbol{\lambda}(\mathbf{x})) = \Phi(\mathbf{x})$ . Now, let us extend (2) to a function of  $\mathcal{X} \in \mathcal{H}_{\mathbf{X}}$ :

$$\lambda_{X=x}(\mathcal{X}) = \begin{cases} 1 & \text{if } x \in \mathcal{X}[X] \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Let  $\boldsymbol{\lambda}(\mathcal{X})$  be the corresponding vector-valued function. It is easily verified that  $f_{\Phi}(\boldsymbol{\lambda}(\mathcal{X})) = \sum_{\mathbf{x} \in \mathcal{X}} \Phi(\mathbf{x})$ , i.e. *the NP returns the unnormalized probability measure for arbitrary sets in  $\mathcal{H}_{\mathbf{X}}$* . Thus, the NP compactly describes marginalization over arbitrary domains of the RVs by simply setting the corresponding IVs to 1. In particular, when we set  $\boldsymbol{\lambda} \equiv 1$ , it returns the normalization constant of  $\Phi$ . A direct implementation of the NP is not practical due the exponentially many terms. In [Darwiche, 2002, 2003, Lowd and Domingos, 2008, Lowd and Rooshenas, 2013] exponentially more compact representations were learned using *arithmetic circuits*.

It is important to note that, although the IVs are called *indicator* variables and set to values out of  $\{0, 1\}$  by functions (2) and (3), they are in fact *real-valued* variables – this difference is essential when we take *derivatives* w.r.t. the IVs. These derivatives are used in the so-called *differential approach to inference* [Darwiche, 2003]. Taking the first derivative with respect to some  $\lambda_{X=x}$  yields

$$\frac{\partial f_{\Phi}}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\mathcal{X})) = \Phi(x, \mathcal{X}[\mathbf{X} \setminus X]), \quad (4)$$

since the derivatives of sum-terms in (1) with  $\mathbf{x}[X] \neq x$  are 0, and the derivatives of sum-terms with  $\mathbf{x}[X] = x$  are  $\Phi(\mathbf{x}) \prod_{X' \in \mathbf{X} \setminus X} \lambda_{X'=\mathbf{x}[X']}$ . Thus, the derivative in (4), which is indeed the standard derivative known from calculus, evaluates  $\Phi$  for *modified evidence*  $x, \mathcal{X}[\mathbf{X} \setminus X]$  and is proportional to the *marginal posterior*  $\Phi(x \mid \mathbf{X} \setminus X)$ . The marginal posteriors are especially useful for approximate MAP solvers [Park, 2002,

Park and Darwiche, 2004]. Given a compact representation of the NP, e.g. using an arithmetic circuit, *all* derivatives of the form (4) can be computed *simultaneously* using back-propagation, i.e. using a single upward and downward pass in the circuit.

### 3 FINITE STATE SPNS

For some node  $N$  in an *acyclic directed graph*, we denote the set of parents and children as  $\mathbf{pa}(N)$  and  $\mathbf{ch}(N)$ , respectively. The descendants  $\mathbf{desc}(N)$  are recursively defined as the set containing  $N$  and the children of any descendant. We now define SPNs over RVs with finitely many states [Poon and Domingos, 2011]:

**Definition 2** (Sum-Product Network). *Let  $\mathbf{X}$  be a set of RVs with finitely many states and  $\lambda$  their IVs. A sum-product network  $\mathcal{S} = (\mathcal{G}, \mathbf{w})$  over  $\mathbf{X}$  is a rooted acyclic directed graph  $\mathcal{G} = (V, E)$  and a set of nonnegative parameters  $\mathbf{w}$ . All leaves of  $\mathcal{G}$  are IVs and all internal nodes are either sums or products. A sum node  $S$  computes a weighted sum  $\mathcal{S}(\lambda) = \sum_{C \in \mathbf{ch}(S)} w_{S,C} C(\lambda)$ , where the weight  $w_{S,C} \in \mathbf{w}$  is associated with the edge  $(S, C) \in E$ . A product node computes  $\mathcal{P}(\lambda) = \prod_{C \in \mathbf{ch}(P)} C(\lambda)$ . The output of  $\mathcal{S}$  is the function computed by the root node and denoted as  $\mathcal{S}(\lambda)$ . The number of additions and multiplications performed by SPN  $\mathcal{S}$  are denoted as  $A_S$  and  $M_S$ , respectively, and given as  $A_S = \sum_{S \in V} |\mathbf{ch}(S)|$ ,  $M_S = \sum_{P \in V} (|\mathbf{ch}(P)| - 1) + \sum_{S \in V} |\mathbf{ch}(S)|$ .*

We use symbols  $\lambda$ ,  $S$ ,  $P$ ,  $N$ ,  $C$ ,  $F$  to refer to nodes in SPNs, where  $\lambda$  denotes an IV,  $S$  always refers to a sum and  $P$  always to a product.  $N$ ,  $C$ ,  $F$  refer to nodes without specifying their type, where  $N$  can be any node,  $C$  and  $F$  are used to highlight their roles as children and parents of other nodes, respectively.

The *scope* of  $N$ , denoted as  $\mathbf{sc}(N)$ , is defined as

$$\mathbf{sc}(N) = \begin{cases} \{X\} & \text{if } N \text{ is some IV } \lambda_{X=x} \\ \bigcup_{C \in \mathbf{ch}(N)} \mathbf{sc}(C) & \text{otherwise.} \end{cases} \quad (5)$$

For any  $N$  in an SPN, we define the *sub-SPN*  $\mathcal{S}_N = (\mathcal{G}_N, \mathbf{w}_N)$ , resulting from the graph  $\mathcal{G}_N$  induced by  $\mathbf{desc}(N)$  together with the corresponding weights  $\mathbf{w}_N$ .

The nodes in an SPN are functions over the IVs  $\lambda$ . Using (2) and (3), we define  $N(\mathbf{x}) := N(\lambda(\mathbf{x}))$  and  $N(\mathcal{X}) := N(\lambda(\mathcal{X}))$ . To avoid pathological cases, we assume that for each  $N$  there exist  $\mathbf{x} \in \mathbf{val}(\mathbf{X})$  such that  $N(\mathbf{x}) > 0$ . The SPN *distribution* is defined as [Poon and Domingos, 2011]:

**Definition 3** (SPN Distribution). *Let  $\mathcal{S}$  be an SPN over  $\mathbf{X}$ . The distribution represented by  $\mathcal{S}$  is*

$$P_{\mathcal{S}}(\mathbf{x}) := \frac{\mathcal{S}(\mathbf{x})}{\sum_{\mathbf{x}' \in \mathbf{val}(\mathbf{X})} \mathcal{S}(\mathbf{x}')}. \quad (6)$$

Using sub-SPNs, we associate to each node  $N$  a distribution  $P_N := P_{\mathcal{S}_N}$  over  $\mathbf{sc}(N)$ .

Inference in structurally unconstrained SPNs is generally intractable. Therefore, Poon and Domingos [2011] introduce the notion of *validity*:

**Definition 4** (Valid SPN). *An SPN  $\mathcal{S}$  over  $\mathbf{X}$  is valid if for each  $\mathcal{X} \in \mathcal{H}_{\mathbf{X}}$*

$$\sum_{\mathbf{x} \in \mathcal{X}} P_{\mathcal{S}}(\mathbf{x}) = \frac{\mathcal{S}(\mathcal{X})}{\sum_{\mathbf{x}' \in \mathbf{val}(\mathbf{X})} \mathcal{S}(\mathbf{x}')}. \quad (7)$$

A valid SPN performs marginalization in similar manner as an NP, cf. Section 2. Two conditions are given for guaranteeing validity of an SPN, *completeness* and *consistency*, which are defined as follows:

**Definition 5** (Completeness). *A sum node  $S$  in SPN  $\mathcal{S}$  is complete if  $\mathbf{sc}(C') = \mathbf{sc}(C'')$ ,  $\forall C', C'' \in \mathbf{ch}(S)$ .  $\mathcal{S}$  is complete if every sum node in  $\mathcal{S}$  is complete.*

**Definition 6** (Consistency). *A product node  $P$  in SPN  $\mathcal{S}$  is consistent if  $\forall C', C'' \in \mathbf{ch}(P), C' \neq C''$ , it holds that  $\lambda_{X=x} \in \mathbf{desc}(C') \Rightarrow \forall x' \neq x : \lambda_{X=x'} \notin \mathbf{desc}(C'')$ .  $\mathcal{S}$  is consistent if every product node in  $\mathcal{S}$  is consistent.*

Thus, for complete and consistent SPNs, we can perform arbitrary marginalization tasks of the form (7) with computational cost linear in the network size. A more restrictive condition implying consistency is *decomposability*:

**Definition 7** (Decomposability). *A product node  $P$  in SPN  $\mathcal{S}$  is decomposable if  $\forall C', C'' \in \mathbf{ch}(P), C' \neq C''$ , it holds that  $\mathbf{sc}(C') \cap \mathbf{sc}(C'') = \emptyset$ .  $\mathcal{S}$  is decomposable if every product node in  $\mathcal{S}$  is decomposable.*

Decomposability requires that the scopes of a product node's children do not overlap, while the notion of consistency is somewhat harder to grasp. The following definition and proposition provide an equivalent condition to consistency.

**Definition 8** (Shared RVs). *The shared RVs of some product node  $P$  are defined as*

$$\mathbf{Y} = \bigcup_{\substack{C', C'' \in \mathbf{ch}(P) \\ C' \neq C''}} \mathbf{sc}(C') \cap \mathbf{sc}(C''), \quad (8)$$

*i.e. the part of  $\mathbf{sc}(P)$  shared by at least two children.*

**Proposition 1.** *Let  $P$  be a product node and  $\mathbf{Y}$  be its shared RVs.  $P$  is consistent iff for each  $Y \in \mathbf{Y}$  there exists a unique  $y^* \in \mathbf{val}(Y)$  with  $\lambda_{Y=y^*} \in \mathbf{desc}(P)$ .*

We call  $y^*$  the *consistent state* of the shared RV  $Y$ , and collect all  $y^*$  in  $\mathbf{y}^* \in \mathbf{val}(\mathbf{Y})$ . The following theorem shows that the distribution represented by a consistent product is *deterministic with respect to the shared RVs*.

Furthermore, the distributions represented by descendants of this product are deterministic with respect to the RVs which overlap with  $\mathbf{Y}$ .

**Theorem 1.** *Let  $\mathcal{S}$  be a complete and consistent SPN and  $\mathbf{P}$  be a non-decomposable product in  $\mathcal{S}$ ,  $\mathbf{Y}$  be the shared RVs of  $\mathbf{P}$  and  $\mathbf{y}^*$  the consistent state of  $\mathbf{Y}$ . For  $\mathbf{N} \in \text{desc}(\mathbf{P})$  define  $\mathbf{Y}^{\mathbf{N}} := \mathbf{Y} \cap \text{sc}(\mathbf{N})$  and  $\mathbf{X}^{\mathbf{N}} := \text{sc}(\mathbf{N}) \setminus \mathbf{Y}^{\mathbf{N}}$ . Then for all  $\mathbf{N} \in \text{desc}(\mathbf{P})$ , and all  $\mathbf{x} \in \text{val}(\text{sc}(\mathbf{N}))$ :*

$$P_{\mathbf{N}}(\mathbf{x}) = \mathbb{1}(\mathbf{x}[\mathbf{Y}^{\mathbf{N}}] = \mathbf{y}^*[\mathbf{Y}^{\mathbf{N}}]) P_{\mathbf{N}}(\mathbf{x}[\mathbf{X}^{\mathbf{N}}]), \quad (9)$$

where  $\mathbb{1}$  is the indicator function.

For the proof we require two lemmas.

**Lemma 1.** *Let  $\mathbf{N}$  be a node in some complete and consistent SPN over  $\mathbf{X}$ ,  $X \in \text{sc}(\mathbf{N})$  and  $x \in \text{val}(X)$ . When  $\lambda_{X=x} \notin \text{desc}(\mathbf{N})$ , then  $\forall \mathbf{x} \in \text{val}(\mathbf{X})$  with  $\mathbf{x}[X] = x$  we have  $\mathbf{N}(\mathbf{x}) = 0$ .*

**Lemma 2.** *Let  $P$  be a probability mass function (PMF) over  $\mathbf{X}$  and  $\mathbf{Y} \subseteq \mathbf{X}$ ,  $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$  such that there exists a  $\mathbf{y}^* \in \text{val}(\mathbf{Y})$  with  $P(\mathbf{z}, \mathbf{y}) = 0$  when  $\mathbf{y} \neq \mathbf{y}^*$ . Then we have  $P(\mathbf{z}, \mathbf{y}) = \mathbb{1}(\mathbf{y} = \mathbf{y}^*) P(\mathbf{z})$ .*

*Proof of Theorem 1.* From Proposition 1 we know that  $\forall Y \in \mathbf{Y}: \lambda_{Y=\mathbf{y}^*[Y]} \in \text{desc}(\mathbf{P})$  and  $\forall y \neq \mathbf{y}^*[Y]: \lambda_{Y=y} \notin \text{desc}(\mathbf{P})$ . Consequently, for any  $\mathbf{N} \in \text{desc}(\mathbf{P})$  we have for all  $Y \in \mathbf{Y}^{\mathbf{N}}$  that  $\lambda_{Y=\mathbf{y}^*[Y]} \in \text{desc}(\mathbf{N})$  and  $\forall y \neq \mathbf{y}^*[Y]: \lambda_{Y=y} \notin \text{desc}(\mathbf{N})$ . With Lemma 1 it follows that for all  $\mathbf{x} \in \text{val}(\text{sc}(\mathbf{N}))$  with  $\mathbf{x}[\mathbf{Y}^{\mathbf{N}}] \neq \mathbf{y}^*[\mathbf{Y}^{\mathbf{N}}]$ , we have  $P_{\mathbf{N}}(\mathbf{x}) = 0$ . Theorem 1 follows with Lemma 2.  $\square$

**Corollary 1.** *Let  $\mathbf{P}$ ,  $\mathbf{Y}$ ,  $\mathbf{y}^*$  be as in Theorem 1. For  $\mathbf{C} \in \text{ch}(\mathbf{P})$ , let  $\mathbf{X}^{\mathbf{C}} := \text{sc}(\mathbf{C}) \setminus \mathbf{Y}$ , i.e. the part of  $\text{sc}(\mathbf{C})$  which belongs exclusively to  $\mathbf{C}$ . Then*

$$P_{\mathbf{P}}(\mathbf{x}) = \mathbb{1}(\mathbf{x}[\mathbf{Y}] = \mathbf{y}^*) \prod_{\mathbf{C} \in \text{ch}(\mathbf{P})} \mathbf{C}(\mathbf{x}[\mathbf{X}^{\mathbf{C}}]). \quad (10)$$

A decomposable product has the intuitive interpretation of a distribution assuming independence among the scopes of its children. Corollary 1 shows that a consistent product assumes independence among the shared RVs  $\mathbf{Y}$  and the RVs which are exclusive to  $\mathbf{P}$ 's children. Independence of  $\mathbf{Y}$  stems from the fact that  $\mathbf{Y}$  is deterministically set to the consistent state  $\mathbf{y}^*$ .

### 3.1 Locally Normalized SPNs

In this section we present our first main contribution. In valid SPNs, the normalization constant  $\mathcal{Z}_{\mathcal{S}} = \sum_{\mathbf{x}} \mathcal{S}(\mathbf{x})$  can be determined efficiently by a single upwards pass, setting  $\lambda \equiv 1$ . We call SPNs with  $\mathcal{Z}_{\mathcal{S}} = 1$  *normalized* SPNs, for which we have  $P_{\mathcal{S}}(\mathbf{x}) = \mathcal{S}(\mathbf{x})$ . When the sum weights are normalized for each sum

$\mathcal{S}$ , i.e.  $\forall \mathcal{S}: \sum_{\mathbf{C} \in \text{ch}(\mathcal{S})} w_{\mathcal{S}, \mathbf{C}} = 1$ , the SPN is automatically normalized, since i) IVs can be interpreted as distributions, ii) complete sum nodes with normalized weights are normalized if their children are normalized, and iii) consistent product nodes are normalized if their children are normalized, following from Corollary 1. We call such SPNs, whose weights are normalized for each sum, *locally normalized* SPNs. Clearly, any sub-SPN of a locally normalized SPNs is also locally normalized. Thus, for any complete, consistent and locally normalized SPN  $\mathcal{S}$  and any  $\mathbf{N}$  of  $\mathcal{S}$ , we have  $\forall \mathbf{x} \in \text{sc}(\mathbf{N}): P_{\mathbf{N}}(\mathbf{x}) = \mathcal{S}_{\mathbf{N}}(\mathbf{x})$ . The question rises, if locally normalized SPNs are a weaker class of models than non-normalized SPNs, i.e. are there distributions which can be represented by an SPN with structure  $\mathcal{G}$ , but *not* by a locally normalized SPN sharing the same structure? The answer is no, as stated in the following theorem.

**Theorem 2.** *For each complete and consistent SPN  $\mathcal{S}' = (\mathcal{G}', \mathbf{w}')$ , there exists a complete, consistent and locally normalized SPN  $\mathcal{S} = (\mathcal{G}, \mathbf{w})$  with  $\mathcal{G}' = \mathcal{G}$ , such that  $\forall \mathbf{N} \in \mathcal{G}: \mathcal{S}_{\mathbf{N}} \equiv P_{\mathcal{S}'_{\mathbf{N}}}$ .*

---

#### Algorithm 1 Locally Normalize SPN

---

```

1: Let  $\mathbf{N}_1, \dots, \mathbf{N}_K$  be a topologically ordered list of
   all sum and product nodes
2: For all product nodes  $\mathbf{P}$  initialize  $\alpha_{\mathbf{P}} \leftarrow 1$ 
3: for  $k = 1 : K$  do
4:   if  $\mathbf{N}_k$  is a sum node then
5:      $\alpha \leftarrow \sum_{\mathbf{C} \in \text{ch}(\mathbf{N}_k)} w_{\mathbf{N}_k, \mathbf{C}}$ 
6:      $\forall \mathbf{C} \in \text{ch}(\mathbf{N}_k): w_{\mathbf{N}_k, \mathbf{C}} \leftarrow \frac{w_{\mathbf{N}_k, \mathbf{C}}}{\alpha}$ 
7:   end if
8:   if  $\mathbf{N}_k$  is a product node then
9:      $\alpha \leftarrow \alpha_{\mathbf{N}_k}$ 
10:     $\alpha_{\mathbf{N}_k} \leftarrow 1$ 
11:   end if
12:   for  $\mathbf{F} \in \text{pa}(\mathbf{N}_k)$  do
13:     if  $\mathbf{F}$  is a sum node then
14:        $w_{\mathbf{F}, \mathbf{N}_k} \leftarrow \alpha w_{\mathbf{F}, \mathbf{N}_k}$ 
15:     end if
16:     if  $\mathbf{F}$  is a product node then
17:        $\alpha_{\mathbf{F}} \leftarrow \alpha \alpha_{\mathbf{F}}$ 
18:     end if
19:   end for
20: end for

```

---

*Proof.* Algorithm 1 finds locally normalized weights without changing the distribution of any node. For deriving the algorithm, we introduce a *nonnegative correction factor*  $\alpha_{\mathbf{P}}$  for each product node  $\mathbf{P}$ , initialized to  $\alpha_{\mathbf{P}} = 1$ . We redefine the product node  $\mathbf{P}$  as  $\mathbf{P}(\lambda) := \alpha_{\mathbf{P}} \mathbf{P}(\lambda)$ . At the end of the algorithm, all  $\alpha_{\mathbf{P}}$  will be 1 and can effectively be removed.

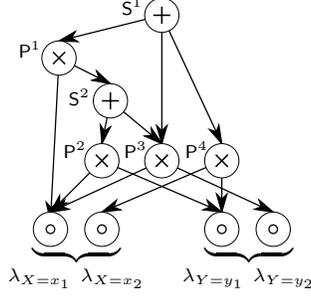


Figure 1: Example of a complete and consistent (but not decomposable) SPN over two binary RVs  $\{X, Y\}$ .

Let  $N'_1, \dots, N'_K$  be a topologically ordered list of all sums and products in the unnormalized SPN  $\mathcal{S}'$ , i.e.  $N'_k \notin \text{desc}(N'_l)$  if  $k > l$ . Let  $N_1, \dots, N_K$  be the corresponding list of  $\mathcal{S}$ , which will be the normalized version after the algorithm has terminated. We have the following loop invariant for the main loop. Given that at the  $k^{\text{th}}$  entrance of the main loop, i)  $P_{N'_l} = \mathcal{S}_{N_l}$ , for  $1 \leq l < k$ , and ii)  $\mathcal{S}'_{N'_m} = \mathcal{S}_{N_m}$ , for  $k \leq m \leq K$ , the same will hold for  $k+1$  at the end of the loop.

Point i) holds since we normalize  $N_k$  during the main loop: All nodes prior in the topological order, and therefore all children of  $N_k$  are already normalized. If  $N_k$  is a sum, then it represents a mixture distribution after step 6. If  $N_k$  is a product, then it will be normalized after step 10, due to Corollary 1 and since we set  $\alpha_{N_k} = 1$ . Point ii) holds since the modification of  $N_k$  can change any  $N_m$ ,  $m > k$ , only via  $\text{pa}(N_k)$ . The change of  $N_k$  is compensated for all parents either in step 14 or step 17, depending on whether the parent is a sum or a product. From this loop invariance it follows by induction that all  $N_1, \dots, N_K$  compute the normalized distributions of  $N'_1, \dots, N'_K$  after the  $K^{\text{th}}$  iteration.  $\square$

Theorem 2 shows that from a representational point of view, we can always assume locally normalized SPNs. It might be advantageous during *learning* to assume the parameter space of non-normalized SPNs. However, after learning is completed we can use Algorithm 1 to find a set of locally normalized parameters.

### 3.2 Consistency vs. Decomposability

As already noted, consistency together with completeness guarantees validity of an SPN, i.e. we can perform efficient marginalization, similarly to NPs. However, a valid but non-decomposable SPN does in general not compute an NP, as the following example shows. The SPN depicted in Figure 1 is complete and consistent, but not decomposable, since  $P^1$  is not decomposable.

It computes the function

$$\begin{aligned} \mathcal{S}(\lambda) = & w_{S^1, P^1} w_{S^2, P^2} \lambda_{X=x_1}^2 \lambda_{Y=y_1} \\ & + w_{S^1, P^1} w_{S^2, P^3} \lambda_{X=x_1}^2 \lambda_{Y=y_2} \\ & + w_{S^1, P^3} \lambda_{X=x_1} \lambda_{Y=y_2} \\ & + w_{S^1, P^4} \lambda_{X=x_2} \lambda_{Y=y_1}, \end{aligned} \quad (11)$$

which is clearly no NP since  $\lambda_{X=x_1}$  is raised to the power of 2 in two terms. This generally happens in non-decomposable SPNs and as a consequence, *Darwiche's differential approach is not applicable to consistent SPNs*. For example, consider the derivative

$$\begin{aligned} \frac{\partial \mathcal{S}}{\partial \lambda_{X=x_1}}(\lambda) = & 2 w_{S^1, P^1} w_{S^2, P^2} \lambda_{X=x_1} \lambda_{Y=y_1} \\ & + 2 w_{S^1, P^1} w_{S^2, P^3} \lambda_{X=x_1} \lambda_{Y=y_2} \\ & + w_{S^1, P^3} \lambda_{Y=y_2}, \end{aligned} \quad (12)$$

which does *not* have the probabilistic interpretation of evaluation of modified evidence, such as the derivatives of an NP, cf. (4). A complete and consistent, but non-decomposable SPN is valid and thus identical to some NP on the *domain of binary vectors*. However, when taking the *derivative*, intuitively speaking, we also consider a small  $\epsilon$ -environment around the 0's and 1's, corrupting the differential approach.

However, complete and *decomposable* SPNs compute NPs and are thus amenable for the differential approach. This is stated in the following proposition.

**Proposition 2.** *A complete and decomposable SPN computes the NP of some unnormalized distribution.*

Therefore, when evaluations of modified evidence (4) are required, we should use decomposable SPNs instead of consistent SPNs. When these are not required, we might want to use the more general condition of consistency instead of decomposability, since we could possibly represent distributions more compactly, i.e. using smaller networks and fewer arithmetic operations. However, the following theorem shows that this potential saving is relatively modest.

**Theorem 3.** *Every complete and consistent SPN  $\mathcal{S} = ((V, E), \mathbf{w})$  over  $\mathbf{X}$  can be transformed into a complete and decomposable SPN  $\mathcal{S}' = ((V', E'), \mathbf{w}')$  over  $\mathbf{X}$  such that  $P_{\mathcal{S}} \equiv P_{\mathcal{S}'}$ , and where  $|V'| \in \mathcal{O}(|V|^2)$ ,  $A_{\mathcal{S}'} = A_{\mathcal{S}}$  and  $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}} |\mathbf{X}|)$ .*

*Proof.* Due to Theorem 2 we assume w.l.o.g. that  $\mathcal{S}'$  is locally normalized, and thus  $P_{\mathcal{S}} \equiv \mathcal{S}$ . Algorithm 2 transforms  $\mathcal{S}$  into a complete and decomposable SPN, representing the same distribution. First it finds a topologically ordered list  $N_1, \dots, N_K$  of all sum and product nodes, i.e.  $k > l \Rightarrow N_k \notin \text{desc}(N_l)$ . Then, in steps 2–7, it considers all sum nodes  $S$  and all children

$C \in \mathbf{ch}(S)$ ; if the child  $C$  has further parents except  $S$ , a newly generated product node  $P^{S,C}$  is interconnected between  $S$  and  $C$ , i.e.  $P^{S,C}$  is connected as child of  $S$  with weight  $w_{S,C}$ ,  $C$  is disconnected from  $S$  and connected as child of  $P^{S,C}$ . To  $P^{S,C}$  we refer as *link* between  $S$  and  $C$ . Note that the link has *only*  $S$  as parent, i.e. the link represents a *private* copy of child  $C$  for sum node  $S$ . Clearly, after step 7, the SPN still computes the same function.

In each iteration of the main loop 8–26, the algorithm finds the lowest non-decomposable product node  $N_k = P$  w.r.t. the topological ordering. We distinguish two cases:  $\mathbf{sc}(P) = \mathbf{Y}$  and  $\mathbf{sc}(P) \neq \mathbf{Y} \Leftrightarrow \mathbf{Y} \subset \mathbf{sc}(P)$ .

In the first case, we know from Corollary 1 that  $P(\mathbf{y}) = \mathbb{1}(\mathbf{y} = \mathbf{y}^*)$ , which is equivalent to the decomposable product  $\prod_{Y \in \mathbf{Y}} \lambda_{Y=\mathbf{y}^*[Y]}$  replacing  $P$ , i.e. this new product is connected as child of all parents of  $P$ , and  $P$  itself is deleted. Deletion of  $P$  might render some nodes unreachable; however, these unreachable nodes do not “influence” the root node and will be safely deleted in step 27.

In the second case, when  $\mathbf{Y} \subset \mathbf{sc}(P)$ , the algorithm first finds the set  $\mathbf{N}^d$  of all sum and product descendants of  $P$ . It also finds the subset  $\mathbf{N}^o$  of  $\mathbf{N}^d$ , containing all nodes whose scope overlaps with  $\mathbf{Y}$ , but is no subset of  $\mathbf{Y}$ . Clearly,  $P$  is contained in  $\mathbf{N}^o$ . The basic strategy is to “cut”  $\mathbf{Y}$  from the scope of  $P$ , i.e. that  $\mathbf{Y}$  is marginalized, rendering  $P$  decomposable. Then, by re-connecting all indicators  $\lambda_{Y=\mathbf{y}^*[Y]}$  to  $P$  in step 24,  $P$  computes the same distribution as before due to Corollary 1, but is rendered *decomposable* now. Steps 21–23 cut  $\mathbf{Y}$  from *all* nodes in  $\mathbf{N}^o$ , in particular from  $P$ , but leave all sub-SPNs rooted at any node in  $\mathbf{N}^d \setminus \mathbf{N}^o$  unchanged. This can be shown by induction over a topological ordering of  $\mathbf{N}^o$ . Due to space reasons, we omit this sub-proof here, but provide it in the supplementary. Although we achieve our primary goal to render  $P$  decomposable, steps 21–23 also cut  $\mathbf{Y}$  from any other node in  $N \in \mathbf{N}^o$ , which would modify the SPN output via  $N$ ’s parents *outside* of  $\mathbf{N}^d$ , i.e. via  $\mathbf{F} = \mathbf{pa}(N) \setminus \mathbf{N}^d$ . Note that all nodes in  $\mathbf{F}$  must be products. To see this, assume that  $\mathbf{F}$  contains a sum  $S$ . This would imply that  $N$  is a link, which can be reached from  $P$  only via its single parent  $S$ . This implies  $S \in \mathbf{N}^d$ , a contradiction. By Theorem 1, the distribution of  $N$  is deterministic w.r.t.  $\mathbf{sc}(N) \cap \mathbf{Y}$ . Steps 21–23 cut  $\mathbf{Y}$  from  $N$  (see supplementary), which could change the distribution of the nodes in  $\mathbf{F}$ . Thus, in step 19 the IVs corresponding to  $\mathbf{Y} \cap \mathbf{sc}(N)$  are connected to all  $\mathbf{F}$ , such that they still “see” the same distribution after steps 21–23. It is easy to see that if some  $F \in \mathbf{F}$  was decomposable beforehand, it will also be decomposable after step 23, i.e. steps 15–24 do not render other products non-decomposable. Thus, in each iteration, one

---

**Algorithm 2** Transform to decomposable SPN
 

---

```

1: Let  $\mathbf{N} = N_1, \dots, N_K$  be a topologically ordered list
   of all sums and products
2: for all sum nodes  $S$  and all  $C \in \mathbf{ch}(S)$  do
3:   if  $\mathbf{pa}(C) > 1$  then
4:     Generate a new product node  $P^{S,C}$ 
5:     Interconnect  $P^{S,C}$  between  $S$  and  $C$ 
6:   end if
7: end for
8: while exist non-decomposable products in  $\mathbf{N}$  do
9:    $P \leftarrow N_{\min\{k' \mid N_{k'} \text{ is a non-decomposable product}\}}$ 
10:   $\mathbf{Y} \leftarrow$  shared RVs of  $P$ 
11:   $\mathbf{y}^* \leftarrow$  consistent state of  $\mathbf{Y}$ 
12:  if  $\mathbf{sc}(P) = \mathbf{Y}$  then
13:    Replace  $P$  by  $\prod_{Y \in \mathbf{Y}} \lambda_{Y=\mathbf{y}^*[Y]}$ 
14:  else
15:     $\mathbf{N}^d \leftarrow$  sums and products in  $\mathbf{desc}(P)$ 
16:     $\mathbf{N}^o \leftarrow \{N \in \mathbf{N}^d : \mathbf{sc}(N) \not\subseteq \mathbf{Y}, \mathbf{sc}(N) \cap \mathbf{Y} \neq \emptyset\}$ 
17:    for  $N \in \mathbf{N}^o \setminus \{P\}$  do
18:       $\mathbf{F} \leftarrow \mathbf{pa}(N) \setminus \mathbf{N}^d$ 
19:       $\forall Y \in \mathbf{Y} \cap \mathbf{sc}(N)$ :
20:        connect  $\lambda_{Y=\mathbf{y}^*[Y]}$  as child of all  $\mathbf{F}$ 
21:    end for
22:    for  $P^o \in \mathbf{N}^o$  do
23:      Disconnect  $C \in \mathbf{ch}(P^o)$  if  $\mathbf{sc}(C) \subseteq \mathbf{Y}$ 
24:    end for
25:     $\forall Y \in \mathbf{Y}$ : connect  $\lambda_{Y=\mathbf{y}^*[Y]}$  as child of  $P$ 
26:  end if
27: end while
28: Delete all unreachable sums and products
    
```

---

non-decomposable product is rendered decomposable, without changing the SPN distribution.

Since the only new introduced nodes are the links between sum nodes and their children, and the number of sum-edges is in  $\mathcal{O}(|V|^2)$ , we have  $|V'| \in \mathcal{O}(|V|^2)$ . The number of summations is the same in  $\mathcal{S}$  and  $\mathcal{S}'$ , i.e.  $A_{\mathcal{S}'} = A_{\mathcal{S}}$ . Furthermore, introducing the links cannot introduce more than twice the number of multiplications, since we already require one multiplication per sum-edge. Thus, after step 7 we have  $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}})$ . Since the while-loop in Algorithm 2 cannot connect more than one IV per  $X \in \mathbf{X}$  to each product node, we have  $M_{\mathcal{S}'} \in \mathcal{O}(M_{\mathcal{S}} |\mathbf{X}|)$ .  $\square$

A graphical example of Algorithm 2 is given in the supplementary. We see that any complete and consistent SPN  $\mathcal{S}$  can be transformed into a complete and decomposable SPN  $\mathcal{S}'$  with the same number of addition operations, and whose number of multiplication operations grows at most linearly with  $M_{\mathcal{S}} |\mathbf{X}|$ . Any distribution which can be encoded by a consistent SPN using polynomially many arithmetic operations in  $|\mathbf{X}|$ , can also be polynomially encoded by a decomposable

SPN. Consequently, the class of consistent SPNs is *not exponentially* more compact than the class of decomposable SPNs.

Furthermore, Algorithm 2 shows that using consistent but non-decomposable products is actually *wasteful* for a particular sub-class of SPNs we denote as *sum-product trees*:

**Definition 9** (Sum-Product Tree). *A sum-product tree (SPT) is an SPN where each sum and product has at most one parent.*

**Proposition 3.** *Every complete and consistent, but non-decomposable SPT  $\mathcal{S} = ((V, E), \mathbf{w})$  over  $\mathbf{X}$  can be transformed into a complete and decomposable SPT  $\mathcal{S}' = ((V', E'), \mathbf{w}')$  over  $\mathbf{X}$  such that  $P_{\mathcal{S}} \equiv P_{\mathcal{S}'}$ , and where  $|V'| \leq |V|$ ,  $A_{\mathcal{S}'} \leq A_{\mathcal{S}}$  and  $M_{\mathcal{S}'} < M_{\mathcal{S}}$ .*

The proof and an example of applying Algorithm 2 to an SPT is given in the supplementary.

## 4 GENERALIZED SPNS

So far, we considered SPNs over finite states using IVs. However, as pointed out in [Gens and Domingos, 2013, Peharz et al., 2013, Rooshenas and Lowd, 2014], SPNs can be generalized by replacing the IVs with arbitrary distributions over arbitrary large scopes. From now on, we assume that each RV is either continuous or discrete, where the latter can also have countably infinitely many states (e.g. Poisson distributed). PMFs and PDFs can be mixed within the same distribution: For example, consider a binary RV  $X$  with  $\mathbf{val}(X) = \{x_1, x_2\}$  and a continuous RV  $Y$  with  $\mathbf{val}(Y) = \mathbb{R}$ . The function

$$D_{\{X, Y\}}(x, y) = 0.4 \times \mathbf{1}(x = x_1) \times \mathcal{N}(y; \mu_1, \sigma_1) + 0.6 \times \mathbf{1}(x = x_2) \times \mathcal{N}(y; \mu_2, \sigma_2), \quad (13)$$

is a PMF with respect to  $X$  and a PDF with respect to  $Y$ . We refer to  $D_{\mathbf{Y}}$  as distribution nodes or simply as distributions. We define generalized SPNs as in Section 3, but now using distributions as leaves. The scope is now defined as

$$\mathbf{sc}(\mathbf{N}) = \begin{cases} \mathbf{Y} & \text{if } \mathbf{N} \text{ is some } D_{\mathbf{Y}} \\ \bigcup_{C \in \mathbf{ch}(\mathbf{N})} \mathbf{sc}(C) & \text{otherwise.} \end{cases} \quad (14)$$

We require that all sums are complete and all products are decomposable. Note that we do allow *heterogeneous* scopes of the input distributions, i.e. when we have a  $D_{\mathbf{Y}}$  with  $Y \in \mathbf{Y}$ , we can have another  $D_{\mathbf{Y}'}$  with  $Y \in \mathbf{Y}'$  and  $\mathbf{Y} \neq \mathbf{Y}'$ . In the most extreme case we can have distributions  $D_{\mathbf{Y}}$  for *all* possible subsets of  $\mathbf{Y} \subseteq \mathbf{X}$ , as long as completeness and decomposability hold. It is easily verified that Theorem 2 also

holds for generalized SPNs, so w.l.o.g. we assume locally normalized weights. In generalized SPNs, each node clearly represents a distribution over its scope: i) leaves are distributions per definition, ii) products are distributions assuming independence, and iii) sums are mixture distributions.

In Section 3, we saw that for finite state SPNs we can efficiently evaluate

1.  $\mathcal{S}(\mathbf{x})$  for  $\mathbf{x} \in \mathbf{val}(\mathbf{X})$  (probability of sample).
2.  $\mathcal{S}(\mathcal{X})$  for  $\mathcal{X} \in \mathcal{H}_{\mathbf{X}}$  (marginalization).
3.  $\mathcal{S}(X, \mathcal{X}[\mathbf{X} \setminus X])$ , simultaneously for all  $X \in \mathbf{X}$  (modified evidence).

We want to have the same inference mechanisms for generalized SPNs. (1.) Evaluation of  $\mathcal{S}(\mathbf{x})$  clearly works in the same way as for SPNs over finite states. (2.) Marginalization also works in a similar way, for which we need to compute

$$\mathcal{S}(\mathcal{X}) = \int_{x_1 \in \mathcal{X}_1} \dots \int_{x_N \in \mathcal{X}_N} \mathcal{S}(x_1, \dots, x_N) dx_1 \dots dx_N, \quad (15)$$

where integrals have to be replaced by sums for discrete RVs. Given that integrals can be computed efficiently for the input distributions, the integral (15) can be evaluated easily, since we can pull the integrals over all sums and products down to the input distributions: At sum nodes, we can interchange the integrals with the sum, due to completeness. At product nodes, we can interchange the integrals with the product, due to decomposability, i.e. since the factors are functions over non-overlapping variable sets. Therefore, we simply perform marginalization at the input distributions (over the respective scopes), and evaluate sums and products in the usual way. Note that in the case of finitely many states and using IVs, this precisely reproduces the mapping (3).

(3.) For evaluation of modified evidence  $\mathcal{S}(X, \mathcal{X}[\mathbf{X} \setminus X])$  we develop an extension of the differential approach. Let  $\mathbf{D}^X$  be the set of input distributions which have  $X$  in their scope. We assume an arbitrary fixed ordering of  $\mathbf{D}^X = \{D^{X,1}, \dots, D^{X,|\mathbf{D}^X|}\}$  and define  $[D]^X = k$  if  $D = D^{X,k}$ . Note that the sets  $\mathbf{D}^X$  do overlap, since for  $D_{\mathbf{Y}}$  we have  $\forall X \in \mathbf{Y}: D_{\mathbf{Y}} \in \mathbf{D}^X$ . For each  $X$ , we introduce a latent RV  $Z_X$  with  $\mathbf{val}(Z_X) = \{1, \dots, |\mathbf{D}^X|\}$ . We denote these RVs as *distribution selectors* (DS) and let  $\mathbf{Z} = \{Z_X \mid X \in \mathbf{X}\}$  be the set of all DS. Now we replace each distribution node  $D_{\mathbf{Y}}$  in the SPN by a product node:

$$D_{\mathbf{Y}} \rightarrow D_{\mathbf{Y}} \times \prod_{X \in \mathbf{Y}} \lambda_{Z_X = [D_{\mathbf{Y}}]^X}, \quad (16)$$

and denote the result as *gated* SPN  $\mathcal{S}^g$ . An example of this process is shown in Figure 2. It is easy to see that

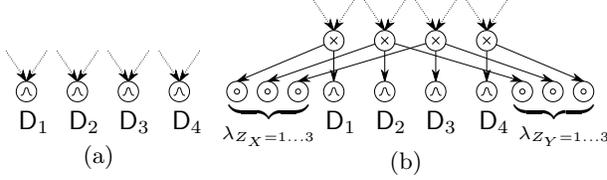


Figure 2: Construction of a gated SPN. We assume that  $\text{sc}(D_1) = \{X\}$ ,  $\text{sc}(D_2) = \text{sc}(D_3) = \{X, Y\}$  and  $\text{sc}(D_4) = \{Y\}$ . (a): Excerpt showing the distribution nodes over  $X$  and  $Y$ . (b): Same excerpt with IVs  $\lambda_{Z_X=1}, \lambda_{Z_X=2}, \lambda_{Z_X=3}, \lambda_{Z_Y=1}, \lambda_{Z_Y=2}, \lambda_{Z_Y=3}$  of DSs.

the gated SPN is a complete and decomposable SPN and thus represents a distribution over  $\mathbf{X} \cup \mathbf{Z}$ . Furthermore, we have  $\mathcal{S}(\mathbf{X}) = \mathcal{S}^g(\mathbf{X}) = \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} \mathcal{S}^g(\mathbf{X}, \mathbf{z})$ , since when the DS are marginalized, i.e.  $\lambda_{\mathbf{Z}} \equiv 1$ , the introduced products simply copy the original distribution nodes. The following proposition establishes a conditional factorization property of the gated SPN.

**Proposition 4.** *Let  $\mathcal{S}^g$  be a gated SPN. For any  $X \in \mathbf{X}$  and  $k \in \{1, \dots, |\mathbf{D}^X|\}$ , let  $\mathbf{X}^D = \text{sc}(\mathbf{D}^{X,k})$  and  $\mathbf{X}^R = \mathbf{X} \setminus \mathbf{X}^D$ . It holds that  $\mathcal{S}^g(\mathbf{X}, Z_X = k, \mathbf{Z} \setminus Z_X) = \mathbf{D}^{X,k}(\mathbf{X}^D) \mathcal{S}^g(\mathbf{X}^R, Z_X = k, \mathbf{Z} \setminus Z_X)$ .*

We define a modified notion of network polynomial, the *extended network polynomial* (ENP).

**Definition 10** (Extended Network Polynomial). *Let  $P(\mathbf{X}, \mathbf{Z})$  be a probability distribution over RVs  $\mathbf{X}, \mathbf{Z}$ , where  $\mathbf{Z}$  have finitely many states. The extended network polynomial  $f_P^{\mathbf{z}}$  is defined as  $f_P^{\mathbf{z}}(\mathbf{X}, \boldsymbol{\lambda}) = \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(\mathbf{X}, \mathbf{z}) \prod_{Z \in \mathbf{Z}} \lambda_{Z=\mathbf{z}[Z]}$ .*

The ENP is an NP over only a part of the RVs, namely  $\mathbf{Z}$ . Similar as for the NP, we can use it to evaluate and marginalize over  $\mathbf{Z}$  and apply the differential approach. Now, using a modification of the proof for Theorem 2, one can show that a gated SPN  $\mathcal{S}^g$  over model RVs  $\mathbf{X}$  and DS  $\mathbf{Z}$  computes the ENP  $f_{\mathcal{S}^g}^{\mathbf{z}}(\mathbf{X}, \boldsymbol{\lambda})$ .

We are now ready to derive the differential approach for generalized SPNs, i.e. to evaluate  $\mathcal{S}(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X])$  for all  $X$  simultaneously. Using Proposition 4, we have

$$\begin{aligned} \mathcal{S}(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X]) &= \sum_{\mathbf{z}} \mathcal{S}^g(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X], \mathbf{z}) \\ &= \sum_{k=1}^{|\mathbf{D}^X|} \mathbf{D}^{X,k}(X, \boldsymbol{\lambda}[\mathbf{X}^D \setminus X]) \sum_{\mathbf{z}'} \mathcal{S}^g(\boldsymbol{\lambda}[\mathbf{X}^R], Z_X = k, \mathbf{z}') \\ &= \sum_{k=1}^{|\mathbf{D}^X|} \mathbf{D}^{X,k}(X, \boldsymbol{\lambda}[\mathbf{X}^D \setminus X]) \mathcal{S}^g(\boldsymbol{\lambda}[\mathbf{X}^R], Z_X = k), \end{aligned} \quad (17)$$

where the sets  $\mathbf{X}^D$  and  $\mathbf{X}^R$ , depending on  $X$  and  $k$ , are defined as in Proposition 4 and  $\mathbf{z}'$  runs over

$\text{val}(\mathbf{Z} \setminus Z_X)$ . We see that  $\mathcal{S}(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X])$  is computed by a linear combination of the input distributions in  $\mathbf{D}^X$ , evaluated for modified evidence  $\boldsymbol{\lambda}[\mathbf{X}^D \setminus X]$  weighted by factors  $\mathcal{S}^g(\boldsymbol{\lambda}[\mathbf{X}^R], Z_X = k)$ . These factors are obtained using the differential approach. When  $\mathbf{Z}$  is marginalized, i.e. when  $\lambda_{\mathbf{Z}} \equiv 1$ , the differential approach tells us that

$$\mathcal{S}^g(\mathbf{X}, Z_X = k) = \frac{\partial \mathcal{S}^g}{\partial \lambda_{Z_X=k}} = \frac{\partial \mathcal{S}^g}{\partial \mathbf{P}} \mathbf{D}^{X,k}(\mathbf{X}^D), \quad (18)$$

where  $\mathbf{P}$  is the parent product of  $\lambda_{Z_X=k}$  and  $\mathbf{D}^{X,k}$  in  $\mathcal{S}^g$  (see Figure 2). Note that  $\frac{\partial \mathcal{S}^g}{\partial \mathbf{P}}$  equals  $\frac{\partial \mathcal{S}}{\partial \mathbf{D}^{X,k}}$ , since the structure above  $\mathbf{D}^{X,k}$  in  $\mathcal{S}$  is identical to the structure in  $\mathcal{S}^g$  above  $\mathbf{P}$ . Comparing (17) and (18) we note that  $\mathcal{S}^g(\boldsymbol{\lambda}[\mathbf{X}^R], Z_X = k) = \frac{\partial \mathcal{S}^g}{\partial \mathbf{P}} = \frac{\partial \mathcal{S}}{\partial \mathbf{D}^{X,k}}$ , yielding

$$\mathcal{S}(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X]) = \sum_{k=1}^{|\mathbf{D}^X|} \mathbf{D}^{X,k}(X, \boldsymbol{\lambda}[\mathbf{X}^D \setminus X]) \frac{\partial \mathcal{S}}{\partial \mathbf{D}^{X,k}}.$$

Inference scenario (3.) is summarized in Algorithm 3. Note, that although we used gated SPNs and ENPs

---

#### Algorithm 3 Infer marginal posteriors

---

- 1: Evaluate input distributions for evidence  $\boldsymbol{\lambda}$
  - 2: Evaluate all sums and products (upwards pass)
  - 3: For all nodes  $\mathbf{N}$ , compute  $\frac{\partial \mathcal{S}}{\partial \mathbf{N}}$  (backpropagation)
  - 4:  $\forall X, k$  let  $f^k(X) = \mathbf{D}^{X,k}(X, \boldsymbol{\lambda}[\text{sc}(\mathbf{D}^{X,k}) \setminus X])$
  - 5:  $\forall X: \mathcal{S}(X, \boldsymbol{\lambda}[\mathbf{X} \setminus X]) \leftarrow \sum_{k=1}^{|\mathbf{D}^X|} \frac{\partial \mathcal{S}}{\partial \mathbf{D}^{X,k}} f^k(X)$
- 

for deriving this result, the required quantities can be computed in the original non-gated SPN. Note that Algorithm 3 reproduces the classical differential approach when considering SPNs over RVs with finitely many states and using IVs as distributed nodes.

## 5 CONCLUSION

In this paper, we aimed to develop a deeper understanding of SPNs and summarize our main results. We now summarize our main results. First, we do not lose any model power when we assume that the weights of SPNs are locally normalized, i.e. normalized for each sum node. Second, we do not lose much model power when we prefer the simpler condition of decomposability over consistency. Third, the inference scenarios known from network polynomials and Darwiche's differential approach can be applied in a similar way to general SPNs, i.e. SPNs with more or less arbitrary input distributions.

#### Acknowledgements

This work was supported by the Austrian Science Fund (project number P25244-N15).

## References

- M. R. Amer and S. Todorovic. Sum-Product Networks for Modeling Activities with Stochastic Structure. In *Proceedings of CVPR*, pages 1314 – 1321, 2012.
- W. C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, and K. M. A. Chai. Language Modeling with Sum-Product Networks. In *Proceedings of Interspeech*, 2014.
- A. Darwiche. A logical approach to factoring belief networks. In *Proceedings of KR*, pages 409–420, 2002.
- A. Darwiche. A Differential Approach to Inference in Bayesian Networks. *ACM*, 50(3):280–305, 2003.
- A. Dennis and D. Ventura. Learning the Architecture of Sum-Product Networks Using Clustering on Variables. In *Advances in Neural Information Processing Systems 25*, pages 2042–2050, 2012.
- R. Gens and P. Domingos. Discriminative Learning of Sum-Product Networks. In *Proceedings of NIPS*, pages 3248–3256, 2012.
- R. Gens and P. Domingos. Learning the Structure of Sum-Product Networks. *Proceedings of ICML*, pages 873–880, 2013.
- D. Lowd and P. Domingos. Learning Arithmetic Circuits. In *Proceedings of UAI*, pages 383–392, 2008.
- D. Lowd and A. Rooshenas. Learning Markov Networks with Arithmetic Circuits. *Proceedings of AIS-TATS*, pages 406–414, 2013.
- J. D. Park. MAP Complexity Results and Approximation Methods. In *Proceedings of UAI*, pages 338–396, 2002.
- J. D. Park and A. Darwiche. Complexity Results and Approximation Strategies for MAP Explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.
- R. Peharz, B. Geiger, and F. Pernkopf. Greedy Part-Wise Learning of Sum-Product Networks. In *ECML/PKDD*, volume 8189, pages 612–627. Springer Berlin, 2013.
- R. Peharz, G. Kapeller, P. Mowlae, and F. Pernkopf. Modeling Speech with Sum-Product Networks: Application to Bandwidth Extension. In *Proceedings of ICASSP*, 2014.
- H. Poon and P. Domingos. Sum-Product Networks: A New Deep Architecture. In *Proceedings of UAI*, pages 337–346, 2011.
- A. Rooshenas and D. Lowd. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. *ICML – JMLR W&CP*, 32:710–718, 2014.