Probabilistic Sentential Decision Diagrams

Doga Kisa and **Guy Van den Broeck** and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{doga,guyvdb,aychoi,darwiche}@cs.ucla.edu

Abstract

We propose the Probabilistic Sentential Decision Diagram (PSDD): A complete and canonical representation of probability distributions defined over the models of a given propositional theory. Each parameter of a PSDD can be viewed as the (conditional) probability of making a decision in a corresponding Sentential Decision Diagram (SDD). The SDD itself is a recently proposed complete and canonical representation of propositional theories. We explore a number of interesting properties of PSDDs, including the independencies that underlie them. We show that the PSDD is a tractable representation. We further show how the parameters of a PSDD can be efficiently estimated, in closed form, from complete data. We empirically evaluate the quality of PSDDs learned from data, when we have knowledge, a priori, of the domain logical constraints.

Introduction

The interplay between logic and probability has been of great interest throughout the history of AI. One of the earliest proposals in this direction is Nilsson's (1986) probabilistic logic, which aimed at augmenting first-order logic with probabilities. This has prompted similar approaches, including, for example, Halpern (1990). The focus of these approaches, however, was mainly semantical, yielding no effective schemes for realizing them computationally. More recently, the area of lifted probabilistic inference has tackled this interplay, while employing a different compromise (Poole 2003). In these efforts, the focus has been mostly on restricted forms of first-order logic (e.g., function-free and finite domain), but with the added advantage of efficient inference (e.g., algorithms whose complexity is polynomial in the domain size).

On the propositional side, the thrust of the interplay has been largely computational. An influential development in this direction has been the realization that enforcing certain properties on propositional representations, such as decomposability and determinism, provides one with the power to answer probabilistic queries efficiently. This development was actually based on two technical observations. First, that decomposable and deterministic representations allow one to perform weighted model counting efficiently. Second, that probabilistic reasoning can be reduced to weighted model counting. This development, which has its first roots in Darwiche (2002), has been underlying an increasing number of probabilistic reasoning systems in the last decade. This is especially true for representations that employ both logical and probabilistic elements (e.g., Chavira, Darwiche, and Jaeger (2006) and Fierens et al. (2011)). Moreover, the technique has been extended recently to certain first-order representations as well (Van den Broeck et al. 2011).

This paper is concerned with an orthogonal contribution to this interplay between propositional logic and probability theory. The problem we tackle here is that of developing a representation of probability distributions in the presence of massive, logical constraints. That is, given a propositional logic theory which represents domain constraints, our goal is to develop a representation that induces a unique probability distribution over the models of the given theory. Moreover, the proposed representation should satisfy requirements that are sometimes viewed as necessary for the practical employment of such representations. These include a clear semantics of the representation parameters; an ability to reason with the representation efficiently; and an ability to learn its parameters from data, also efficiently.

Our proposal is called a *Probabilistic Sentential Decision Diagram (PSDD).* It is based on the recently proposed *Sentential Decision Diagram (SDD)* for representing propositional theories (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). While the SDD is comprised of *logical* decision nodes, the PSDD is comprised of *probabilistic* decision nodes, which are induced by supplying a distribution over the branches of a logical decision node. Similar to SDDs, the PSDD is a canonical representation, but under somewhat more interesting conditions. Moreover, computing the probability of a term can be done in time linear in the PSDD size. In fact, the probability of each and every literal can be computed in only two passes over the PSDD. It is particularly notable that the local parameters of a PSDD have clear semantics with respect to the global distribution induced by the PSDD. We will also show that these parameters can be learned efficiently from complete data.

This paper is structured as follows. We start by a concrete discussion on some of the applications that have driven the development of PSDDs and follow by an intuitive expo-

L	K	P	A	Students
0	0	1	0	6
0	0	1	1	54
0	1	1	1	10
1	0	0	0	5
1	0	1	0	1
1	0	1	1	0
1	1	0	0	13
1	1	1	0	8
1	1	1	1	3

Table 1: Student enrollment data.

sure of PSDDs and their salient features. We next provide a formal treatment of the syntax, semantics and properties of PSDDs. This allows us to present the main inference algorithm for PSDDs and the one for learning PSDD parameters from complete data. The paper concludes with some experimental results showing the promise of PSDDs in learning probability distributions under logical constraints. Proofs of theorems are delegated to the full version of the paper due to space limitations.

Motivation

PSDDs were inspired by the need to learn probability distributions that are subject to domain constraints. Take for example a computer science department that organizes four courses: Logic (L), Knowledge Representation (K), Probability (P), and Artificial Intelligence (A). Students are asked to enroll for these courses under the following restrictions:

– A student must take at least one of Probability or Logic.

– Probability is a prerequisite for AI.

– The prerequisite for KR is either AI or Logic.

The department may have data on student enrollments, as in Table 1, and may wish to learn a probabilistic model for reasoning about student preferences. For example, the department may need to know whether students are more likely to satisfy the prerequisite of KR using AI or using Logic.

A mainstream approach for addressing this problem is to learn a probabilistic graphical model, such as a Bayesian network. In this case, a network structure is constructed manually or learned from data. The structure is then turned into a Bayesian network by learning its parameters from the data. Other graphical models can also be used. This includes, for example, Markov networks or their variations.

What is common among all these approaches is that they lack a principled and effective method for accommodating the domain constraints into the learning process—that is, ensuring, for example, that a student with a profile $A \wedge K \wedge L \wedge \neg P$, or a profile $\neg A \wedge K \wedge \neg L \wedge P$, has zero probability in the learned model. In principle, the zero parameters of a graphical model can capture logical constraints, although a fixed model structure will not in general accommodate arbitrary logical constraints. We could introduce additional structure into the model to capture such constraints, using, e.g., the method of virtual evidence (Pearl 1988; Mateescu and Dechter 2008). However, incorporating constraints in this manner will in general lead to a highly-connected network, making inference intractable. Even if inference remained tractable, such an approach is not ideal as we now have to learn a distribution that is conditioned on the constraints. This would require new learning algorithms (e.g., gradient methods) for performing parameter estimation as traditional methods may no longer be applicable. For example, in Bayesian networks, the closed-form parameter estimation algorithm under complete data will no longer be valid in this case.

The domain constraints of our example can be expressed using the following CNF.

$$\begin{aligned} P \vee L \\ A \Rightarrow P \\ K \Rightarrow A \vee L \end{aligned} \tag{1}$$

Even though there are 16 combinations of courses, the CNF says that only 9 of them are valid choices. An approach that observes this information must learn a probability distribution that assigns a zero probability to every combination that is not allowed by these constraints.

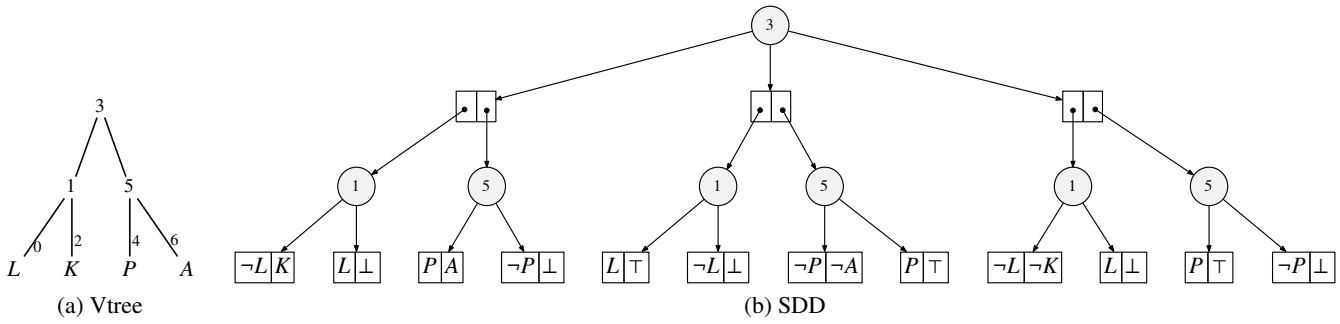
None of the standard learning approaches we are familiar with has been posed to address this problem. The complication here is not strictly with the learning approaches, but with the probabilistic models that are amenable to being learned under these circumstances. In particular, these models are not meant to induce probability distributions that respect a given set of logical constraints.

The simple problem we posed in this section is exemplary of many real-world applications. We mention in particular *configuration* problems that arise when purchasing products, such as cars and computers. These applications give users the option to configure products, but subject to certain constraints. Data is abundant for these applications and there is a clear economic interest in learning probabilistic models under the given constraints. We also mention *reasoning about physical systems,* which includes verification and diagnosis applications. Here, propositional logic is typically used to encode some system functionality, while leaving out some system behaviors which may have a non-deterministic nature (e.g., component failures and probabilistic transitions). There is also an interest here to learn probabilistic models of these systems, subject to the given constraints.

Our goal in this paper is to introduce the PSDD representation for addressing this particular need. We will start by an intuitive (and somewhat informal) introduction to PSDDs, followed by a more formal treatment of their syntax, semantics and the associated reasoning and learning algorithms.

PSDDs

We will refer to domain constraints as the *base* of a probability distribution. Our proposed approach starts by representing this base as a *Sentential Decision Diagram (SDD)* as in Figure 1 (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). An SDD is determined by a *vtree,* which is a full binary tree with leaves corresponding to the domain variables (Pipatsrisawat and Darwiche 2008). The choice of a particular SDD can then be thought of as a choice of a particular vtree. We will later discuss the impact of this



(a) Vtree (b) SDD

Figure 1: A vtree and SDD for the student enrollment problem. Numbers in circles correspond to vtree nodes.

choice on the represented distribution. For now, however, we will develop some further understanding of SDDs as they are the backbones of our probability distributions.

SDDs. An SDD is either a *decision node* or a *terminal node.* A terminal node is a literal, the constant $\top$ (true) or the constant $\bot$ (false). A decision node is a disjunction of the form $(p_1 \wedge s_1) \vee \ldots \vee (p_n \wedge s_n)$, where each pair (p_i, s_i) is called an *element.* A decision node is depicted by a circle and its elements are depicted by paired boxes. Here, $p_1, \ldots, p_n$ are called *primes* and $s_1, \ldots, s_n$ are called *subs.* Primes and subs are themselves SDDs. Moreover, the primes of a decision node are always consistent, mutually exclusive and exhaustive. The SDD in Figure 1 has seven decision nodes. The decision node to the far left has two elements $(\neg L, K)$ and $(L, \bot)$. It represents $(\neg L \wedge K) \vee (L \wedge \bot)$, which is equivalent to $\neg L \wedge K$. There are two primes for this node $\neg L$ and L. The two corresponding subs are K and $\bot$.

Structure. An SDD can be viewed as a *structure* that induces infinitely many probability distributions (all having the same base). By *parameterizing* an SDD, one obtains a PSDD that induces a particular probability distribution.

Parameters. Figure 2 depicts a PSDD which is obtained by parameterizing the SDD in Figure 1. Both decision and terminal SDD nodes are parameterized, but we focus here on decision nodes. Let n be a decision node having elements $(p_1, s_1), \ldots, (p_n, s_n)$. To parameterize node n is to provide a distribution $\theta_1, \ldots, \theta_n$. Intuitively, θ_i is the probability of prime p_i given that the decision of node n has been implied. We will formalize and prove this semantics later. We will also provide an efficient procedure for learning the parameters of a PSDD from complete data. The PSDD parameters in Figure 2 were learned using this procedure from the data in Table 2. The table also depicts the probability distribution induced by the learned PSDD.

Independence. The PSDD structure is analogous to a Bayesian network structure in the following sense. The latter can be parameterized in infinitely many ways, with each parameterization inducing a particular probability distribution. Moreover, all the induced distributions satisfy certain independences that can be inferred from the underlying Bayesian network structure. The same is true for PSDDs. Each parameterization of a PSDD structure yields a unique probability distribution. Moreover, all the induced distributions satisfy independences that can be inferred from the PSDD structure.

L	K	P	A	Students	Learned PSDD Distribution	
0	0	1	0	6	$0.6 \cdot 0.1$	6.0%
0	0	1	1	54	$0.6 \cdot 0.9$	54.0%
0	1	1	1	10	0.1	10.0%
1	0	0	0	5	$0.3 \cdot 0.2 \cdot 0.6$	3.6%
1	0	1	0	1	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.75$	1.8%
1	0	1	1	0	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.25$	0.6%
1	1	0	0	13	$0.3 \cdot 0.8 \cdot 0.6$	14.4%
1	1	1	0	8	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.75$	7.2%
1	1	1	1	3	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.25$	2.4%

Table 2: Student enrollment data and learned distribution.

We will show, however, that PSDD independence is more refined than Bayesian network independence as it allows one to express more qualified independence statements.

The Syntax and Semantics of PSDDs

PSDDs are based on *normalized* SDDs in which every node n is associated with (normalized for) a vtree node v according to the following rules (Darwiche 2011).

- If n is a terminal node, then v is a leaf node which contains the variable of n (if any).
- If n is a decision node, then its primes (subs) are normalized for the left (right) child of v.
- If n is the root SDD node, then v is the root vtree node.

The SDD in Figure 1 is normalized. Each decision node in this SDD is labeled with the vtree node it is normalized for. We are now ready to define the syntax of a PSDD.

Definition 1 (PSDD Syntax) *A PSDD is a normalized SDD with the following parameters.*

- *For each decision node $(p_1, s_1), \ldots, (p_k, s_k)$ and prime p_i, a positive parameter θ_i is supplied such that $\theta_1 + \ldots + \theta_k = 1$ and $\theta_i = 0$ iff $s_i = \bot$.*
- *For each terminal node $\top$, a positive parameter θ is supplied such that $0 < \theta < 1$.*

A terminal node $\top$ with parameter θ will be denoted by $X : \theta$, where X is the variable of leaf vtree node that $\top$ is normalized for. Other terminal nodes (i.e., $\bot$, X and $\neg X$) have fixed, implicit parameters (discussed later) and will be denoted as is. A decision PSDD node will be denoted by

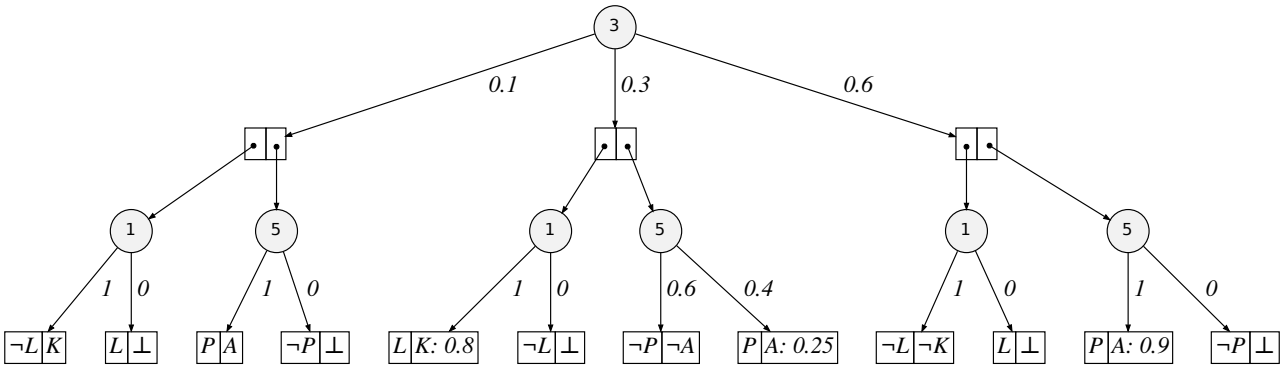


Figure 2: A PSDD for the student enrollment problem, which results from parameterizing the SDD in Figure 1. The parameters were learned from the dataset in Table 1 (also shown in Table 2).

$(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. Graphically, we will just annotate the edge into element (p_i, s_i) with the parameter θ_i. Figure 2 provides examples of this notation.

We next define the distribution of a PSDD, inductively. That is, we first define the distribution induced by a terminal node. We then define the distribution of a decision node in terms of the distributions induced by its primes and subs.

Definition 2 (PSDD Distribution) *Let n be a PSDD node that is normalized for vtree node v. Node n defines a distribution Pr_n over the variables of vtree v as follows.*

– *If n is a terminal node, and v has variable X, then*

n	$\mathsf{Pr}_n(X)$	$\mathsf{Pr}_n(\neg X)$
$X : \theta$	θ	$1-\theta$
$\bot$	0	0
X	1	0
$\neg X$	0	1

– *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ and v has left variables $\mathbf{X}$ and right variables $\mathbf{Y}$, then*

$$\mathsf{Pr}_n(\mathbf{xy}) \stackrel{def}{=} \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i \quad \textit{for } i \textit{ where } \mathbf{x} \models p_i.$$

Applying this definition to the PSDD of Figure 2 leads to the distribution in Table 2 for its root node. The following table depicts the distribution induced by a non-root node in this PSDD, which appears in the middle of Figure 2.

$\mathbf{x}$	$\mathbf{y}$	$\mathsf{Pr}_{p_i}(\mathbf{x})$	$\mathsf{Pr}_{s_i}(\mathbf{y})$	θ_i	$\mathsf{Pr}_n(\mathbf{xy})$
P	A	1	0.25	0.4	0.1
P	$\neg A$	1	0.75	0.4	0.3
$\neg P$	A	1	0	0.6	0.0
$\neg P$	$\neg A$	1	1	0.6	0.6

Table 3: Distribution of node $n = (\neg P, \neg A)(P, \top)$.

The SDD node associated with a PSDD node n is called the *base* of n and is denoted by $[n]$. When there is no ambiguity, we will often not distinguish between a PSDD node n and its base $[n]$.

A PSDD assigns a strictly positive probability to a variable instantiation iff the instantiation satisfies its base. This can be seen, for example, in Table 3. This is also the first key property of PSDDs.

Theorem 1 (Base) *Consider a PSDD node n that is normalized for vtree node v. If $\mathbf{Z}$ are the variables of vtree v, then $\mathsf{Pr}_n(\mathbf{z}) > 0$ iff $\mathbf{z} \models [n]$.*

We will now discuss the second key property of PSDDs, which reveals the *local* semantics of PSDD parameters.

Theorem 2 (Parameter Semantics) *Let n be a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. We have $\theta_i = \mathsf{Pr}_n([p_i])$.*

Consider the PSDD in Figure 2 and its decision node n in Table 3. Prime $\neg P$ of this node has parameter 0.6. According to Theorem 2, we must then have $\mathsf{Pr}_n(\neg P) = 0.6$, which can be verified in Table 3. Similarly, $\mathsf{Pr}_n(P) = 0.4$.

The third key property of PSDDs is the relationship between the local distributions induced by its various nodes (node distributions) and the global distribution induced by its root node (PSDD distribution)—for example, the relationship between the distribution of node n in Table 3 and the PSDD distribution given in Table 2.

Node distributions are linked to the PSDD distribution by the notion of *context.*

Definition 3 (Context) *Let $(p_1, s_1), \ldots, (p_k, s_k)$ be the elements appearing on some path from the SDD root to node n.*[1] *Then $p_1 \wedge \ldots \wedge p_k$ is called a sub-context for node n and is feasible iff $s_i \neq \bot$. The context is a disjunction of all sub-contexts and is feasible iff some sub-context is feasible.*

Consider Figure 1. The three decision nodes normalized for vtree node $v = 5$ have the contexts $\neg L \wedge K$, L and $\neg L \wedge \neg K$. Moreover, the terminal nodes normalized for vtree $v = 6$ have the contexts:

– A: $\neg L \wedge K \wedge P$

– $\neg A$: $L \wedge \neg P$

– $\bot$: $(\neg L \wedge K \wedge \neg P) \vee (\neg L \wedge \neg K \wedge \neg P) = (\neg L \wedge \neg P)$

– $\top$: $(L \wedge P) \vee (\neg L \wedge \neg K \wedge P) = (L \vee \neg K) \wedge P$.

Contexts satisfy interesting properties.

[1] That is, $n = p_k$ or $n = s_k$.

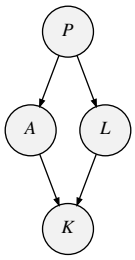


Figure 3: A Bayesian network structure.

Theorem 3 (Context) *A node is implied by its context and the underlying SDD. Nodes normalized for the same vtree node have mutually exclusive and exhaustive contexts. The sub-contexts of a node are mutually exclusive. A context/sub-context is feasible iff it has a strictly positive probability.*

Contexts give a global interpretation to node distributions.

Theorem 4 (Node Distribution) *Consider a PSDD r and let n be one of its nodes. If γ_n is a feasible sub-context or feasible context of node n, then $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \gamma_n)$.*

Contexts also give a *global* interpretation to parameters.

Corollary 1 (Parameter Semantics) *Consider a PSDD r and node n with feasible sub-context or feasible context γ_n.*

- *If n is a terminal node $X:\theta$, then $\theta = \mathsf{Pr}_r(X \mid \gamma_n)$.*
- *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, then $\theta_i = \mathsf{Pr}_r([p_i] \mid \gamma_n)$ for $i = 1, \ldots, k$.*

This corollary says that the parameters of a node are conditional probabilities of the PSDD distribution.

We show in the Appendix that PSDDs are *complete* as they are capable of representing any probability distribution. We also show that PSDDs are *canonical* under a condition known as compression. More precisely, we show that there is a unique compressed PSDD for each distribution and vtree. This is particularly important for learning PSDDs (structure and parameters) as it reduces the problem of searching for a PSDD into the problem of searching for a vtree.

PSDD Independence

Consider the Bayesian network structure in Figure 3, which corresponds to our earlier example. This structure encodes a number of probabilistic independences that hold in any distribution it induces (i.e., regardless of its parameters). These independences are

- A and L are independent given P.
- K and P are independent given AL.

These independences are conditioned on variables. That is, "given AL" reads "given any state of variables A and L." The second independence can therefore be expanded into $2 \times 2 \times 4$ statements of the form "α is independent of β given γ," where α, β and γ are propositional sentences (e.g., $\neg K$ is independent of P given $A \wedge \neg L$).

The structure of a PSDD also encodes independences that hold in every induced distribution. These independences fall into two major categories, the first coming from Theorem 4.

Corollary 2 (Independence I) *Consider a PSDD r and a node n with context γ_n and feasible sub-context β_n.*

- *If n is a terminal node $X:\theta$, then*

$$\mathsf{Pr}_r(X \mid \gamma_n, \beta_n) = \mathsf{Pr}_r(X \mid \gamma_n) = \mathsf{Pr}_r(X \mid \beta_n) = \theta.$$

- *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, then*

$$\mathsf{Pr}_r([p_i] \mid \gamma_n, \beta_n) = \mathsf{Pr}_r([p_i] \mid \gamma_n) = \mathsf{Pr}_r([p_i] \mid \beta_n) = \theta_i$$

for $i = 1, \ldots, k$.

That is, the probability of a prime is independent of a sub-context once the context is known. This is also true for the probability of a terminal sub. Moreover, which specific sub-context we know is irrelevant. All are equivalent as far as defining the semantics of parameters is concerned.

The second category of independences is as follows.

Theorem 5 (Independence II) *Let γ_v be a feasible context for a PSDD node normalized for vtree node v. Variables inside v are independent of those outside v given context γ_v.*

To read the independences characterized by this theorem, one iterates over each vtree node v, identifying its corresponding, feasible contexts γ_v. Consider the PSDD in Figure 2. There are three decision nodes which are normalized for vtree node $v = 5$, with contexts $\neg L \wedge K$, L and $\neg L \wedge \neg K$. Using the second context, we get

variables PA and LK are independent given context L.

This reads as "If we know that someone took Logic, then whether they took KR has no bearing on whether they took Probability or AI."

PSDD independences are conditioned on propositional sentences (contexts) instead of variables. This kind of independence is known to be more expressive and is usually called *context-specific independence* (Boutilier et al. 1996). This kind of independence is beyond the scope of probabilistic graphical models, which can only condition independence statements on variables. Consider the example statement we discussed above. If we were to condition on the variable L instead of the propositional sentence L, we would also get "If we know that someone *did not take* Logic, then whether they took KR has no bearing on whether they took Probability or AI." This is actually contradicted by the logical constraints for this problem ($K \Rightarrow A \vee L$). If someone did not take Logic, but took KR, they must have taken AI.

Reasoning with PSDDs

We now present the main algorithms for reasoning with PSDDs. In particular, given a PSDD r and an instantiation $\mathbf{e}$ of some variables (*evidence*), we provide an algorithm for computing the probability of this evidence $\mathsf{Pr}_r(\mathbf{e})$. We also present an algorithm for computing the conditional probability $\mathsf{Pr}_r(X \mid \mathbf{e})$ for every variable X. Both algorithms run in time which is linear in the PSDD size.

We start with the first algorithm. For variable instantiation $\mathbf{e}$ and vtree node v, we will use $\mathbf{e}_v$ to denote the subset of instantiation $\mathbf{e}$ that pertains to the variables of vtree v, and $\mathbf{e}_{\bar{v}}$ to denote the subset of $\mathbf{e}$ that pertains to variables outside v. The first algorithm is based on the following result.

Algorithm 1: Probability of Evidence

Input: PSDD r and evidence $\mathbf{e}$

$evd(n) \leftarrow 0$ for every node n
// visit children before parents
foreach *node n in the PSDD* **do**
  if *n is a terminal node* **then**
    $v \leftarrow$ leaf vtree node that n is normalized for
    $evd(n) \leftarrow \mathsf{Pr}_n(\mathbf{e}_v)$
  else
    foreach *element (p_i, s_i, θ_i) of node n* **do**
      $evd(n) \leftarrow evd(n) + evd(p_i) \cdot evd(s_i) \cdot \theta_i$

Theorem 6 *Consider a decision node $n = (p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ that is normalized for vtree node v, with left child l and right child r. For evidence $\mathbf{e}$, we have*

$$\mathsf{Pr}_n(\mathbf{e}_v) = \sum_{i=1}^{k} \mathsf{Pr}_{p_i}(\mathbf{e}_l) \cdot \mathsf{Pr}_{s_i}(\mathbf{e}_r) \cdot \theta_i$$

When n is a terminal node, v is a leaf vtree node and $\mathbf{e}_v$ is either a literal or the empty instantiation. In this case, we can just look up the value of $\mathsf{Pr}_n(\mathbf{e}_v)$ from the distribution induced by the terminal node n (from Definition 2).

Theorem 6 leads to Algorithm 1, which traverses the PSDD bottom-up, computing $\mathsf{Pr}_n(\mathbf{e}_v)$ for each node n and storing the result in $evd(n)$. The probability of evidence is then $evd(r)$, where r is the PSDD root.

We now turn to computing the probability $\mathsf{Pr}_r(X, \mathbf{e})$ for each variable X. One can use Algorithm 1 to perform this computation, but the algorithm would need to be called once for each variable X. However, with the following theorem, we can compute all of these node marginals using a single, second pass on the PSDD, assuming that Algorithm 1 did the first pass.

Theorem 7 *Consider a PSDD r, variable X, and its leaf vtree node v. Let $n_1, \ldots, n_k$ be all the terminal nodes normalized for v and let $\gamma_{n_1}, \ldots, \gamma_{n_k}$ be their corresponding contexts. For evidence $\mathbf{e}$, we have*

$$\mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}}) = \sum_{i=1}^{k} \mathsf{Pr}_{n_i}(X) \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}).$$

If $\mathbf{e} \models \neg X$, then $\mathsf{Pr}_r(X, \mathbf{e}) = 0$. Otherwise, $X, \mathbf{e}_{\bar{v}} = X, \mathbf{e}$ and $\mathsf{Pr}_r(X, \mathbf{e}) = \mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}})$.

The term $\mathsf{Pr}_{n_i}(X)$ in Theorem 7 is immediately available. Algorithm 2 computes $\mathsf{Pr}_r(\gamma_n, \mathbf{e}_{\bar{v}})$ for every PSDD node n that has context γ_n and is normalized for vtree node v. The algorithm traverses the PSDD top-down, computing this probability for each visited node n, storing it in $ctx(n)$. If n is a terminal node, the algorithm also computes $\mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}})$ and $\mathsf{Pr}_r(\neg X, \mathbf{e}_{\bar{v}})$, storing them in $mrg(X)$ and $mrg(\neg X)$.

The simplicity of Algorithm 2 is due to the following. The probability of a sub-context can be computed by multiplying the parameters appearing on its corresponding path. Since sub-contexts are mutually exclusive, their probabilities can be added to obtain the context probability. Algorithm 2 does precisely this except that it accounts for evidence as well using quantities computed by Algorithm 1.

Algorithm 2: Probability of Contexts

Input: PSDD r

$ctx(n) \leftarrow 0$ for nodes $n \neq r$ and $ctx(r) \leftarrow 1$
$mrg(X) \leftarrow 0$ and $mrg(\neg X) \leftarrow 0$ for every variable X
// visit parents before children
foreach *node n in the PSDD* **do**
  if *n is a terminal node* **then**
    $X \leftarrow$ variable of node n
    $mrg(X) \leftarrow mrg(X) + ctx(n) \cdot \mathsf{Pr}_n(X)$
    $mrg(\neg X) \leftarrow mrg(\neg X) + ctx(n) \cdot \mathsf{Pr}_n(\neg X)$
  else
    foreach *element (p_i, s_i, θ_i) of node n* **do**
      $ctx(p_i) \leftarrow ctx(p_i) + \theta_i \cdot evd(s_i) \cdot ctx(n)$
      $ctx(s_i) \leftarrow ctx(s_i) + \theta_i \cdot evd(p_i) \cdot ctx(n)$

Learning with PSDDs

We now present an algorithm for learning the parameters of a PSDD from a *complete* dataset. We start first with some basic definitions. An instantiation of all variables is called an *example*. There are 2^n distinct examples over n propositional variables. A complete dataset is a multi-set of examples.[2] That is, an example may appear multiple times in a dataset. Given a PSDD structure (a normalized SDD), and a complete dataset, our goal is to learn the value of each PSDD parameter. More precisely, we wish to learn *maximum likelihood* parameters: ones that maximize the probability of examples in the dataset.

We will use Pr_θ to denote the distribution induced by the PSDD structure and parameters θ. The *likelihood* of these parameters given dataset $\mathbf{D}$ is defined as

$$L(\theta|\mathbf{D}) = \prod_{i=1}^{N} \mathsf{Pr}_\theta(\mathbf{d}_i),$$

where $\mathbf{d}_i$ ranges over all N examples in dataset $\mathbf{D}$. Our goal is then to find the maximum likelihood parameters

$$\theta^{ml} = \operatorname*{argmax}_\theta L(\theta|\mathbf{D}).$$

We will use $\mathbf{D}\#(\alpha)$ to denote the number of examples in dataset $\mathbf{D}$ that satisfy propositional sentence α. For a decision node $n = (p_1, s_1, \theta_1), ..., (p_k, s_k, \theta_k)$ with context γ_n, we propose the following estimate for parameter θ_i:

$$\theta_i^{ml} = \frac{\mathbf{D}\#(p_i, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \quad (2)$$

For terminal node $n = X{:}\theta$ with context γ_n, we propose the following estimate for parameter θ:

$$\theta^{ml} = \frac{\mathbf{D}\#(X, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \quad (3)$$

[2] In an *incomplete* dataset, an example corresponds to an instantiation of some variables (not necessarily all).

We can now show the following.

Theorem 8 *The parameter estimates of Equations 2 and 3 are the only estimates that maximize the likelihood function.*

Our parameter estimates admit a *closed-form*, in terms of the counts $\mathbf{D}\#(\alpha)$ in the data. One can compute these estimates using a single pass through the examples of a dataset. Moreover, each distinct example can be processed in time linear in the PSDD size.[3] These are very desirable properties for a parameter learning algorithm. These properties are shared with Bayesian network representations, but are missing from many others, including Markov networks.

When learning probabilistic graphical models, one makes a key distinction between learning structures versus learning parameters (the former being harder in general). While learning PSDD structures is beyond the scope of this paper, the experimental results we present next do use a basic method for learning structures. In particular, since we compile logical constraints into an SDD (i.e., a PSDD structure), the compilation technique we use is effectively "learning" a structure. We used the publicly available SDD package for this purpose (*http://reasoning.cs.ucla.edu/sdd/*). The SDD package tries to dynamically minimize the size of the compiled SDD and, as a result, tries to minimize the number of PSDD parameters.

Preliminary Experimental Results

In this Section, we empirically evaluate our parameter estimation algorithm for PSDDs. First, we illustrate how our algorithm can effectively recover the parameters of a PSDD from data, assuming that the dataset was indeed generated by the distribution of a PSDD. Second, we highlight how knowing the logical constraints underlying a given dataset can impact the accuracy of learned models.

In our first set of experiments, using synthetic data, we show how we can recover the true parameters that generated a dataset, given that the logical constraints are known. As we will be simulating datasets from a known PSDD, we shall use the KL-divergence to compare the original PSDD distribution $\Pr(\mathbf{Z})$, and the PSDD distribution $\Pr'(\mathbf{Z})$ that we learned from the data:

$$KL(\Pr, \Pr') = \sum_{\mathbf{z}} \Pr(\mathbf{z}) \log \frac{\Pr(\mathbf{z})}{\Pr'(\mathbf{z})}.$$

Note that the KL-divergence is non-negative, and zero iff the two distributions are equivalent.[4]

[3] A dataset may not include every example that is consistent with the domain constraints. If this is the case, a parameter θ for prime p may be estimated to zero, even though its sub s may not be $\bot$; see Theorem 1. To address this, one can assume a pseudo-count for each distinct example, which can be thought of as providing a prior distribution on parameters. In our experiments, we assumed a pseudo-count of $1/\mathsf{mc}$ for each distinct example, where mc is the model count of the SDD. This corresponds to a very weak prior since, in aggregate, these pseudo-counts contribute a total count that is equivalent to one real example in the dataset.

[4] The KL–divergence between two PSDDs can be computed efficiently if they share the same structure.

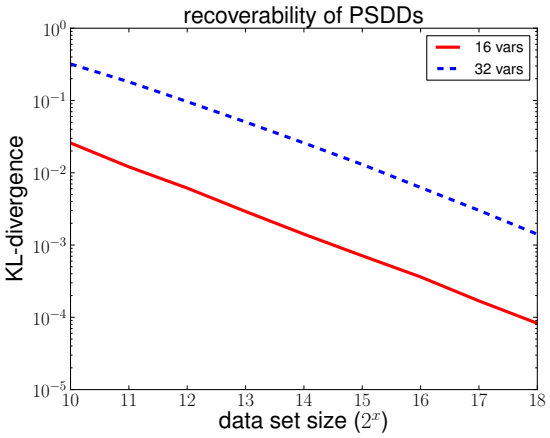


Figure 4: We observe how our parameter estimation algorithm can recover the original PSDD as we increase the size of the training dataset (x-axis), measured by the KL-divergence (y-axis, where lower is better).

Consider Figure 4, where each point in the plot represents an average of 100 randomly generated PSDDs. We generated 10 random PSDD structures by compiling random 3-CNFs into SDDs (the used clause to variable ratio is 3). For each SDD, we randomly generated 10 different parameterizations, giving us $10 \times 10 = 100$ PSDDs total. The SDDs were compiled using dynamic minimization, which reduces the size of the SDD, and hence, the number of parameters in the corresponding PSDD. We evaluated PSDDs over 16 variables and 32 variables (one curve each).

As we go right on the x-axis, we increase the size of the dataset that we are learning our PSDD from. We clearly see that as more data becomes available, the parameters that we learn more closely match the true parameters of the original PSDD (measured using the KL-divergence, on the y-axis). Indeed, both curves are steadily tending to zero. The PSDDs over 16 variables converge faster than those over 32 variables. The PSDDs over 16 variables have fewer parameters, roughly, a tenth as many.

Our final experiment tries to put PSDD learning in the context of learning probabilistic graphical models. It is hard to make a direct comparison here since existing approaches do not factor domain constraints into the learning process. Hence, there are no available systems that tackle the learning problem we posed in this paper, nor are there reported experimental results that are directly relevant. Still, we conceived an experiment that is revealing in this regard, which we explain next.

A standard experiment for evaluating a learning approach goes as follows. The dataset is divided into a *training* portion and *testing* one. The learning algorithm is then given the training dataset and asked to find a probabilistic model. The quality of the model is then measured by computing the likelihood of learned parameters, but using the testing dataset. The literature contains many experiments of this form (e.g., Table 4). The question we posed is this: What if the dataset embedded the domain constraints, in that any example that

Table 4: Log Likelihoods of Testing Datasets. A question mark indicates the unavailability of likelihood for the corresponding approach/dataset.

benchmark	#var	ACMN	ACBN	LTM	CLT	PSDD
NLTCS	16	-6.01	-6.02	-6.49	-6.76	-6.31
MSNBC	17	-6.04	-6.04	-6.52	-6.54	-6.20
KDDCup	64	-2.15	-2.16	-2.18	-2.29	-2.11
Plants	69	-12.89	-12.85	-16.39	-16.52	-10.99
Audio	100	-40.32	-41.13	-41.90	-44.37	-19.34
Jester	100	-53.35	-54.43	-55.17	-58.23	-18.49
Netflix	100	-57.26	-57.75	-58.53	-60.25	-19.41
Accidents	111	?	?	?	-33.19	-19.09
Retail	135	?	?	?	-10.94	-10.68
Pumsb Star	163	?	?	?	-30.79	-18.25
DNA	180	?	?	?	-87.68	-14.75
MSWeb	294	-9.77	-9.81	-10.21	-10.19	-9.25
Book	500	-35.62	-36.02	-34.22	-37.83	-17.38
EachMovie	500	?	?	?	-64.83	-14.89
WebKB	839	-161.30	-159.85	-156.84	-163.43	-15.89
Kosarak	883	?	?	?	-19.87	-9.03
Reuters-52	889	-89.54	-89.27	-91.23	-94.41	-17.58
20 NG	910	-159.56	-159.65	-156.77	-164.13	-18.91
BBC	1058	?	?	?	-261.80	-14.22
Ad	1556	?	?	?	-16.41	-10.31

does not appear in the dataset (training or testing) is an impossible example? Under this assumption, the domain constraints correspond to a sentence which is obtained by disjoining all examples in the dataset. This is indeed what we did in our last experiment. We compiled the mentioned disjunction (which is a DNF) into an SDD, learned its parameters using the training dataset, and measured the quality of learned PSDD using the testing dataset.

We compared the obtained results on the corpus in Table 4; see, e.g., (Davis and Domingos 2010) for more statistics on these datasets. These real-world datasets are commonly used to evaluate algorithms for learning probabilistic graphical models as we just discussed.[5] Here, ACMN are Arithmetic Circuits for Bayesian networks, ACBN are Arithmetic Circuits for Markov networks, LTM are Latent Tree Models, and CLT are Chow-Liu trees. Each algorithm learns its corresponding model using the training dataset, and evaluates it using the testing dataset. The likelihoods of learned models are given in Table 4; these likelihoods can also be found in (Lowd and Rooshenas 2013). For the two datasets over a small number of variables, our PSDDs obtain competitive likelihoods (smaller magnitude is better). In datasets over a larger number of variables, the PSDDs obtain better, and often significantly better, likelihoods than other learned models.

While this experiment is contrary to what is usual, it is both revealing and suggestive. First, it highlights the potential impact of accommodating logical constraints into the learning process. Second, it begs the question: What if the data was indeed indicative of the domain constraints? For example, when the dataset is large enough, and the domain is known to be constrained enough, it is not unreasonable to assume that the dataset is indicative of domain constraints. Moreover, theoretically, this assumption does hold in the limit. As far as we know, there is no existing approach that can exploit this assumption as done by this work. In fact, this observation and the need to address it is what prompted us to develop PSDDs in the first place.

[5]The models that were learned in Table 4 correspond to restricted Bayesian or Markov networks (they have a corresponding arithmetic circuit (AC) or have a tree structure). These restrictions are meant to allow one to compute likelihoods efficiently.

Related Work

The PSDD can be viewed from two angles: As a knowledge representation and reasoning formalism, and as a learning formalism. We next discuss related work across both angles.

As a KR formalism, the PSDD is related to work on extending Binary Decision Diagrams (BDDs) to represent real-valued functions. The relationship between SDDs (which underly PSDDs) and BDDs is known (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). In sum though, BDDs branch on literals, instead of sentences, which leads to limitations that are inherited by their extensions (Xue, Choi, and Darwiche 2012). Algebraic Decision Diagrams (ADDs) (R.I. Bahar et al. 1993) in particular are often used to represent joint probability distributions. However, they are not factorized representations and, as such, they need to explicitly represent every distinct probability in a distribution. Affine ADDs (Sanner and McAllester 2005) and Edge-valued Binary Decision Diagrams (Lai and Sastry 1992) do offer a factorized representation, but their parameters are not interpretable as in PSDDs (i.e., as probabilities of the global distribution).

There is also a body of work on facilitating the representation of logical constraints in the context of probabilistic representations. This includes Richardson and Domingos (2006), who facilitate the representation of logical constraints in Markov networks. Additionally, Mateescu and Dechter (2008) separate an unconstrained probabilistic model (e.g., Bayesian network) from the logical constraints and define the target distribution as the former conditioned on the latter. Hence, individual parameters in their framework have no local semantics in the target distribution. Several more first-order probabilistic languages can directly express logical dependencies (De Raedt et al. 2008; Getoor and Taskar 2007). Within these representations, there has been considerable interest in probabilistic reasoning in the presence of logical constraints. See for example Poon and Domingos (2006) and Gogate and Dechter (2007).

Perhaps the most strongly related formalism is the probabilistic decision graph (Bozga and Maler 1999; Jaeger 2004). These circuits are governed by a variable forest instead of a vtree. Their parameters represent contextual conditional probabilities. However, for a fixed variable forest, probabilistic decision graphs are not a complete representation. There are distributions that cannot be represented because the variable forest already encodes certain conditional independencies. Hence, these representations are canonical only in a weaker sense although they have been learned from data (Jaeger, Nielsen, and Silander 2006). As a logical representation, they are situated in between BDDs and SDDs:

They provide stronger decompositions than BDDs but still branch over single variables instead of sentences.

Viewing PSDDs as a learning formalism brings up two potential connections to existing work. First, as a formalism that aims at learning *generative* probabilistic models with well defined bases (i.e., logical constraints), the PSDD is somewhat unique as this learning problem has not been posed this explicitly before. The only possible exception is (Chang et al. 2008), which learns a *discriminative* probabilistic model given logical constraints (they propose a generalization of linear models called Constrained Conditional Models).[6] Second, as a formalism that aims at learning *tractable* representations of distributions, the PSDD falls into the recent body of work on *deep learning*. The closest connection here is to the sum-product network (Poon and Domingos 2011), which is a new (but also tractable) class of deep architectures. These networks represent probability distributions as a (deep) hierarchy of mixtures (essentially, an arithmetic circuit with latent variables). However, they do not take logical constraints as an input. Moreover, even though they provide a less constrained representation of distributions, compared to the PSDD, the implication is that learning becomes harder (e.g., there is no closed form for parameter learning under complete data).

Conclusion

We presented the PSDD as a representation of probability distributions that respect a given propositional theory. The PSDD is a complete and canonical representation, with parameters that are interpretable as conditional probabilities. The PSDD encodes context-specific independences, which can be derived from its structure. The PSDD is a tractable representation, allowing one to compute the probability of any term in time linear in its size. The PSDD has unique maximum likelihood parameters under complete data, which can be learned efficiently using a closed form. Preliminary experimental results suggest that the PSDD can be quite effective in learning distributions under domain constraints.

Acknowledgments

This work has been partially supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen).

Appendix

Completeness and Canonicity of PSDDs

We will now address two connected questions with regards to the representational power of PSDDs. The first question asks whether every distribution can be represented by a PSDD. The second question asks whether the PSDD representation is canonical (given a vtree). The answers to both questions rely on the notion of *probabilistic* $\mathbf{XY}$*-partitions,* which are based on the $\mathbf{XY}$-partitions underlying SDDs.

We review the latter notion first. Consider a propositional sentence α over disjoint variables $\mathbf{X}$ and $\mathbf{Y}$. One can always express this sentence as

$$\alpha = (p_1 \wedge s_1) \vee \ldots \vee (p_k \wedge s_k) \tag{4}$$

where $p_1, \ldots, p_k$ mention only variables $\mathbf{X}$ and $s_1, \ldots, s_k$ mention only variables $\mathbf{Y}$. One can also always ensure that each p_i is consistent, that $p_i \wedge p_j$ is inconsistent for $i \neq j$, and that $p_1 \vee \ldots \vee p_k$ is valid. Under these conditions, the form in (4) is called an $\mathbf{XY}$-partition of α (Darwiche 2011). One obtains an SDD for some sentence α by recursively decomposing it into $\mathbf{XY}$-partitions (Darwiche 2011). A parallel notion exists for distributions.

Definition 4 *Let* $\mathsf{Pr}(\mathbf{XY})$ *be a distribution with base* α*. A probabilistic* $\mathbf{XY}$*-partition of this distribution has the form*

$$(\mathsf{Pr}_{p_1}(\mathbf{X}), \mathsf{Pr}_{s_1}(\mathbf{Y}), \theta_1), \ldots, (\mathsf{Pr}_{p_k}(\mathbf{X}), \mathsf{Pr}_{s_k}(\mathbf{Y}), \theta_k)$$

such that

- $(p_1 \wedge s_1) \vee \ldots \vee (p_k \wedge s_k)$ *is an* $\mathbf{XY}$*-partition of base* α*.*
- *Each* Pr_{p_i} *is a (prime) distribution with base* p_i*.*
- *Each* Pr_{s_i} *is a (sub) distribution with base* s_i*.*
- $\theta_i \geq 0$*,* $\theta_i = 0$ *iff* $s_i = \bot$*, and* $\theta_1 + \cdots + \theta_k = 1$*.*
- $\mathsf{Pr}(\mathbf{xy}) = \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i$ *where* p_i *satisfies* $\mathbf{x} \models p_i$*.*

The above definition shows how one can decompose a distribution $\mathsf{Pr}(\mathbf{XY})$ into a number of smaller distributions $\mathsf{Pr}(\mathbf{X})$ and $\mathsf{Pr}(\mathbf{Y})$. More importantly though, the bases of these smaller distributions correspond to a decomposition of the base for distribution $\mathsf{Pr}(\mathbf{XY})$.

One can always decompose a distribution this way.

Theorem 9 *Let* Pr *be a probability distribution and suppose that its variables are partitioned into* $\mathbf{X}$ *and* $\mathbf{Y}$*. There must exist a probabilistic* $\mathbf{XY}$*-partition of distribution* Pr*.*

Corollary 3 *Every probability distribution can be represented by a PSDD.*

Let v be a vtree over the distribution variables. The root of this vtree partitions the variables into $\mathbf{X}$ (variables in left subtree of v) and $\mathbf{Y}$ (variables in right subtree of v). By Theorem 9, we can construct a probabilistic $\mathbf{XY}$-partition of the distribution, which defines the root PSDD node. By repeating the process recursively, we can construct PSDDs for the prime and sub distributions of each constructed $\mathbf{XY}$-partition, until we reach distributions over single variables.

A distribution may have multiple (or even many) probabilistic $\mathbf{XY}$-partitions for the same sets of variables $\mathbf{X}$ and $\mathbf{Y}$. However, exactly one of these is *compressed.*

Theorem 10 *A probabilistic* $\mathbf{XY}$*-partition is compressed iff its sub distributions are distinct. A distribution has a unique compressed* $\mathbf{XY}$*-partition for each* $\mathbf{X}$ *and* $\mathbf{Y}$*.*[7]

[6] Generative learning is concerned with learning a distribution that is optimized for generating the data, yet is not connected to any particular reasoning task. Discriminative learning is concerned with learning a distribution that is optimized for a classification task (i.e., a probabilistic classifier).

[7] Consider a prime distribution Pr_{p_i} where sub $s_i = \bot$. Since $\theta_i = 0$ in this case, the non-zero probabilities of the prime distribution Pr_{p_i} are irrelevant to the distribution represented by the probabilistic $\mathbf{XY}$-partition. The uniqueness claimed in this theorem is modulo such prime distributions.

A decision PSDD node is compressed iff its subs (not their bases) are distinct. A PSDD is compressed iff all its decision nodes are compressed. We now have our canonicity result.

Corollary 4 *Every distribution is induced by a unique, compressed PSDD (given a vtree).*

We close this section by stressing the following point. The notion of compression in PSDDs generalizes a corresponding notion of compression for SDDs (Darwiche 2011). In particular, an SDD is said to be compressed if every decision node has distinct subs. The main observation here is that the SDD of a compressed PSDD may itself be uncompressed. That is, a compressed PSDD may have two distinct sub distributions with equal bases.[8] In fact, the PSDD representation is complete only if one allows the underlying SDDs to be uncompressed.

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context–specific independence in Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 115–123.

Bozga, M., and Maler, O. 1999. On the representation of probabilities over structured domains. In *Computer Aided Verification*, 261–273. Springer.

Chang, M.-W.; Ratinov, L.-A.; Rizzolo, N.; and Roth, D. 2008. Learning and inference with constraints. In Fox, D., and Gomes, C. P., eds., *AAAI*, 1513–1518. AAAI Press.

Chavira, M.; Darwiche, A.; and Jaeger, M. 2006. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning* 42(1-2):4–20.

Choi, A., and Darwiche, A. 2013. Dynamic minimization of sentential decision diagrams. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 187–194.

Darwiche, A. 2002. A logical approach to factoring belief networks. In *Proceedings of KR*, 409–420.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, 819–826.

Davis, J., and Domingos, P. 2010. Bottom-up learning of Markov network structure. In *ICML*, 271–278.

De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic inductive logic programming: theory and applications*. Berlin, Heidelberg: Springer-Verlag.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, 211–220.

Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Dechter, R. 2007. Samplesearch: A scheme that searches for consistent samples. In *International Conference on Artificial Intelligence and Statistics*, 147–154.

[8]This is never true in our experiments since we start off with a compressed SDD (the SDD package outputs compressed SDDs).

Halpern, J. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46(3):311–350.

Jaeger, M.; Nielsen, J. D.; and Silander, T. 2006. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning* 42(1):84–100.

Jaeger, M. 2004. Probabilistic decision graphscombining verification and ai techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12(supp01):19–42.

Lai, Y.-T., and Sastry, S. 1992. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, 608–613. IEEE Computer Society Press.

Lowd, D., and Rooshenas, A. 2013. Learning Markov networks with arithmetic circuits. In *AISTATS*, 406–414.

Mateescu, R., and Dechter, R. 2008. Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence* 54(1-3):3–51.

Nilsson, N. 1986. Probabilistic logic. *Artificial intelligence* 28(1):71–87.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

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

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of IJCAI*, 985–991.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, 458–463.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. *Proc. 12th Conf. on Uncertainty in Artificial Intelligence* 337–346.

R.I. Bahar; E.A. Frohm; C.M. Gaona; G.D. Hachtel; E. Macii; A. Pardo; and F. Somenzi. 1993. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, 188–191. Santa Clara, California: IEEE Computer Society Press.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1):107–136.

Sanner, S., and McAllester, D. A. 2005. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1384–1390.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of IJCAI*, 2178–2185.

Xue, Y.; Choi, A.; and Darwiche, A. 2012. Basing decisions on sentences in decision diagrams. In *AAAI*, 842–849.

Appendix: Proofs

Theorem 1 (Base) *Consider a PSDD node n that is normalized for vtree node v. If $\mathbf{Z}$ are the variables of vtree v, then $\mathsf{Pr}_n(\mathbf{z}) > 0$ iff $\mathbf{z} \models [n]$.*

Proof We proceed by induction on the structure of the vtree. As our base case, our theorem holds for leaf vtree nodes, by Definition 2. Suppose now that we have a vtree node v with variables $\mathbf{X}$ in the left vtree and variables $\mathbf{Y}$ in the right vtree. We consider two cases for $\mathbf{z} = \mathbf{xy}$:

- if $\mathbf{xy} \models [n]$: the unique prime p_i of n, where $\mathbf{x} \models [p_i]$, has a sub s_i where $\mathbf{y} \models [s_i]$ (and hence $s_i \neq \bot$). By induction, $\mathsf{Pr}_{p_i}(\mathbf{x}) > 0$ and $\mathsf{Pr}_{s_i}(\mathbf{y}) > 0$. Since $\theta_i > 0$ when $s_i \neq \bot$, we have $\mathsf{Pr}_n(\mathbf{xy}) > 0$, by Definition 2.
- if $\mathbf{xy} \not\models [n]$: the unique prime p_i of n, where $\mathbf{x} \models [p_i]$, has a sub s_i where $\mathbf{y} \not\models [s_i]$. By induction, $\mathsf{Pr}_{s_i}(\mathbf{y}) = 0$, hence $\mathsf{Pr}_n(\mathbf{xy}) = 0$, by Definition 2. □

Theorem 2 (Parameter Semantics) *Let n be a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. We have $\theta_i = \mathsf{Pr}_n([p_i])$.*

Proof Let n be normalized for vtree v, with variables $\mathbf{X}$ in the left vtree and variables $\mathbf{Y}$ in the right vtree. Then:

$$\begin{aligned}
\mathsf{Pr}_n([p_i]) &= \sum_{\mathbf{xy} \models [p_i]} \mathsf{Pr}_n(\mathbf{xy}) \\
&= \sum_{\mathbf{x} \models [p_i]} \sum_{\mathbf{y}} \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i \\
&= \Big(\sum_{\mathbf{x} \models [p_i]} \mathsf{Pr}_{p_i}(\mathbf{x}) \Big) \cdot \Big(\sum_{\mathbf{y}} \mathsf{Pr}_{s_i}(\mathbf{y}) \Big) \cdot \theta_i \\
&= \mathsf{Pr}_{p_i}([p_i]) \cdot \mathsf{Pr}_{s_i}(\top) \cdot \theta_i = \theta_i
\end{aligned}$$

where $\mathsf{Pr}_{p_i}([p_i]) = 1$ (which follows from Theorem 1). □

Before we proceed to Theorem 3, we first consider the following Lemma, which relates an SDD to the decisions made in a sub-context.

Lemma 1 *For a root SDD node r and a node n, let $(p_1, s_1), \ldots, (p_k, s_k)$ be the elements appearing on some path from r to n. Let $\beta = p_1 \wedge \cdots \wedge p_k$ denote the corresponding sub-context. Then:*

$$[r] \wedge \beta = [p_1] \wedge [s_1] \wedge \cdots \wedge [p_k] \wedge [s_k].$$

Proof Note first that given an SDD node n and one of its primes p_i, we have

$$[n] \wedge [p_i] = [p_i] \wedge [s_i] = [n] \wedge [p_i] \wedge [s_i]. \tag{5}$$

To prove our lemma, we use induction. In our base case, when $k = 1$, node p_1 is a prime of the root r, so we have $[r] \wedge [p_1] = [p_1] \wedge [s_1]$, by Equation 5. Assume now that we have a sub-context β_{k+1} with $k+1$ primes. We then have

$$\begin{aligned}
[r] \wedge \beta_{k+1} &= [r] \wedge \beta_k \wedge [p_{k+1}] \\
&= [p_1] \wedge [s_1] \wedge \cdots \wedge [p_k] \wedge [s_k] \wedge [p_{k+1}],
\end{aligned}$$

by induction. On the path from root r to node n, node p_{k+1} appears as a prime in exactly one of p_k or s_k. Without loss of generality, assume it was a prime of s_k. Then by Equation 5, we have $[s_k] \wedge [p_{k+1}] = [s_k] \wedge [p_{k+1}] \wedge [s_{k+1}]$. We substitute into the previous equation, and obtain the desired result. □

Theorem 3 (Context) *A node is implied by its context and the underlying SDD. Nodes normalized for the same vtree node have mutually exclusive and exhaustive contexts. The sub-contexts of a node are mutually exclusive. A context/sub-context is feasible iff it has a strictly positive probability.*

Proof Consider a root SDD r and a node n with a sub-context $\beta = p_1 \wedge \cdots \wedge p_k$. By Lemma 1, we have

$$[r] \wedge \beta = [p_1] \wedge [s_1] \wedge [p_2] \wedge [s_2] \wedge \cdots \wedge [p_k] \wedge [s_k].$$

On the last step of the path from root r to node n, we either have $n = p_k$ or $n = s_k$. Hence, $[r] \wedge \beta \models [n]$, for any sub-context β of node n. Now, if γ is the context of node n, where $\gamma = \bigvee_\beta \beta$, then since $[r] \wedge \beta \models [n]$ for all sub-contexts β, we have $[r] \wedge \gamma \models [n]$, i.e., the node n is implied by its context and the SDD r.

Next, we show that all nodes normalized for a vtree node v, have contexts that are mutually exclusive and exhaustive. The root vtree node, our base case, has only one node, and one context, $\top$, which is mutually exclusive and exhaustive. Consider then a vtree node v with SDD nodes n, and the parent vtree node u with SDD nodes m. First, any sub-context of a node n in vtree v was constructed by conjoining a prime to some sub-context of a node m in vtree u. Since primes are mutually exclusive, and the contexts of nodes m are mutually exclusive (by induction), the sub-contexts of two different nodes n must be mutually exclusive. Second, any sub-context of a node m in vtree u can be extended to a sub-context of a node n in vtree v, by conjoining some prime p of m. Hence, since all primes are exhaustive, we can disjoin all sub-contexts of nodes n to obtain a disjunction over all contexts of nodes m (which are exhaustive, by induction). Since all sub-contexts of nodes in vtree v can be partitioned according to the contexts, the contexts of nodes in vtree v are mutually exclusive and exhaustive.

Next, we show that the sub-contexts of an individual node n are mutually exclusive. If node n has a unique sub-context (like the root node r), the above statement holds trivially. If a node n has two different sub-contexts, then denote them by $\beta_1 = [p_1] \wedge \cdots \wedge [p_k]$ and $\beta_2 = [q_1] \wedge \cdots \wedge [q_k]$. Sub-contexts β_1 and β_2 correspond to two distinct paths from the root r to node n, which must diverge at some point, by branching on two distinct primes p_i and q_i. Since primes are mutually exclusive, the sub-contexts β_1 and β_2 must also be mutually exclusive.

Finally, say we have an SDD r, with a node n and sub-context $\beta = [p_1] \wedge \cdots \wedge [p_k]$. By Lemma 1, we have:

$$[r] \wedge \beta = [p_1] \wedge [s_1] \wedge \cdots \wedge [p_k] \wedge [s_k].$$

Sub-context β is feasible iff $[r] \wedge \beta$ is consistent (in both cases, no sub can be false). Moreover, $[r] \wedge \beta$ is consistent iff $\mathsf{Pr}_r(\beta) > 0$, by Theorem 1. If the context of n is $\gamma = \bigvee_\beta \beta$, then since all sub-contexts are mutually exclusive, $\mathsf{Pr}_r(\gamma) = \sum_\beta \mathsf{Pr}_r(\beta)$. Hence, context γ is feasible iff some sub-context β is feasible, iff $\mathsf{Pr}_r(\gamma) \geq \mathsf{Pr}_r(\beta) > 0$. □

Before we proceed to Theorem 4, we first relate a node distribution with its prime and sub distributions.

Proposition 1 *Given a decision node n with some element (p, s, θ), where $[s] \neq \bot$, we have:*

– $\mathsf{Pr}_p(.) = \mathsf{Pr}_n(. \mid [p])$,
– $\mathsf{Pr}_s(.) = \mathsf{Pr}_n(. \mid [p])$.

Proof Let the sets $\mathbf{X}$ and $\mathbf{Y}$ denote the variables of our primes and subs, respectively. First, if $\mathbf{x} \not\models [p]$, then $\mathsf{Pr}_p(\mathbf{x}) = \mathsf{Pr}_n(\mathbf{x} \mid [p])$, since $\mathsf{Pr}_p(\mathbf{x}) = 0$ by Theorem 1, and $\mathsf{Pr}_n(\mathbf{x} \mid [p]) = 0$ since we condition on $[p]$. Otherwise, if $\mathbf{x} \models [p]$:

$$\begin{aligned}
\mathsf{Pr}_n(\mathbf{x} \mid [p]) &= \frac{\mathsf{Pr}_n(\mathbf{x})}{\mathsf{Pr}_n([p])} = \frac{1}{\theta} \cdot \mathsf{Pr}_n(\mathbf{x}) && \text{by Theorem 2} \\
&= \frac{1}{\theta} \sum_{\mathbf{y}} \mathsf{Pr}_n(\mathbf{xy}) \\
&= \frac{1}{\theta} \sum_{\mathbf{y}} \mathsf{Pr}_p(\mathbf{x}) \cdot \mathsf{Pr}_s(\mathbf{y}) \cdot \theta && \text{by Definition 2} \\
&= \mathsf{Pr}_p(\mathbf{x}) \sum_{\mathbf{y}} \mathsf{Pr}_s(\mathbf{y}) = \mathsf{Pr}_p(\mathbf{x})
\end{aligned}$$

Next, we have

$$\begin{aligned}
\mathsf{Pr}_n(\mathbf{y} \mid [p]) &= \frac{\mathsf{Pr}_n(\mathbf{y}, [p])}{\mathsf{Pr}_n([p])} \\
&= \frac{1}{\theta} \cdot \mathsf{Pr}_n(\mathbf{y}, [p]) && \text{by Theorem 2} \\
&= \frac{1}{\theta} \sum_{\mathbf{x} \models [p]} \mathsf{Pr}_n(\mathbf{xy}) \\
&= \frac{1}{\theta} \sum_{\mathbf{x} \models [p]} \mathsf{Pr}_p(\mathbf{x}) \cdot \mathsf{Pr}_s(\mathbf{y}) \cdot \theta && \text{by Definition 2} \\
&= \mathsf{Pr}_s(\mathbf{y}) \sum_{\mathbf{x} \models [p]} \mathsf{Pr}_p(\mathbf{x}) = \mathsf{Pr}_s(\mathbf{y})
\end{aligned}$$

where $\mathsf{Pr}_{p_i}([p_i]) = 1$ (which follows from Theorem 1). □

Theorem 4 (Node Distribution) *Consider a PSDD r and let n be one of its nodes. If γ_n is a feasible sub-context or feasible context of node n, then $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \gamma_n)$.*

Proof We first consider the case of a feasible sub-context β_n. Let the set $\mathbf{Z}$ denote the variables of r. For the purposes of induction, consider the base case where the node n is the root r of the PSDD. The only sub-context of r is β_r is true, which we shall just denote here by $\top$. As $\mathsf{Pr}_r(\top) = 1$, we have $\mathsf{Pr}_r(\mathbf{z}) = \mathsf{Pr}_r(\mathbf{z}, \top)/\mathsf{Pr}_r(\top) = \mathsf{Pr}_r(\mathbf{z} \mid \top)$.

Consider a non-root node n. A feasible sub-context β_n of node n has the form $[p] \wedge \beta_m$, where p is the prime of an element (p, s) of some parent node m with sub-context β_m. If β_n is feasible, then the sub s is non-false, and the sub-context β_m is feasible. Moreover, the node n is either the prime p or sub s of element (p, s). Let the sets $\mathbf{X}$ and $\mathbf{Y}$ denote the variables of our primes and subs. By induction, we have:

$$\mathsf{Pr}_m(\mathbf{xy}) = \mathsf{Pr}_r(\mathbf{xy} \mid \beta_m)$$

for all instantiations $\mathbf{xy}$. Hence,

$$\begin{aligned}
\mathsf{Pr}_m([p]) &= \sum_{\mathbf{xy} \models [p]} \mathsf{Pr}_m(\mathbf{xy}) = \sum_{\mathbf{xy} \models [p]} \mathsf{Pr}_r(\mathbf{xy} \mid \beta_m) \\
&= \mathsf{Pr}_r([p] \mid \beta_m). \qquad (6)
\end{aligned}$$

If $\mathbf{xy} \models [p]$, we have $\mathbf{xy} = \mathbf{xy} \wedge [p]$ and so

$$\mathsf{Pr}_m(\mathbf{xy} \wedge [p]) = \mathsf{Pr}_r(\mathbf{xy} \wedge [p] \mid \beta_m) \qquad (7)$$

Combining Equations 6 & 7, we get:

$$\mathsf{Pr}_m(\mathbf{xy} \mid [p]) = \mathsf{Pr}_r(\mathbf{xy} \mid [p] \wedge \beta_m) = \mathsf{Pr}_r(\mathbf{xy} \mid \beta_n).$$

Finally, by Proposition 1, we have both

$$\mathsf{Pr}_p(\mathbf{x}) = \mathsf{Pr}_m(\mathbf{x} \mid [p]) = \mathsf{Pr}_r(\mathbf{x} \mid \beta_n)$$
$$\mathsf{Pr}_s(\mathbf{y}) = \mathsf{Pr}_m(\mathbf{y} \mid [p]) = \mathsf{Pr}_r(\mathbf{y} \mid \beta_n)$$

and hence $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \beta_n)$ (either $n = p$ or $n = s$).

We now consider the case of a feasible context $\gamma_n = \bigvee_{\beta_n} \beta_n$ of a node n, composed of sub-contexts β_n. In the base case where n is the root of the PSDD, the context is just the single sub-context $\top$, where $\mathsf{Pr}_r(\mathbf{z}) = \mathsf{Pr}_r(\mathbf{z} \mid \top)$ as we shown already. For a non-root node n, where $\mathbf{xy} \models \beta_n$,:

$$\mathsf{Pr}_r(\mathbf{x} \wedge \beta_n) = \mathsf{Pr}_r(\mathbf{x} \mid \beta_n)\mathsf{Pr}_r(\beta_n) = \mathsf{Pr}_p(\mathbf{x})\mathsf{Pr}_r(\beta_n)$$
$$\mathsf{Pr}_r(\mathbf{y} \wedge \beta_n) = \mathsf{Pr}_r(\mathbf{y} \mid \beta_n)\mathsf{Pr}_r(\beta_n) = \mathsf{Pr}_s(\mathbf{y})\mathsf{Pr}_r(\beta_n)$$

First, since all sub-contexts β_n are mutually exclusive, all events $\mathbf{x} \wedge \beta_n$ are mutually exclusive, all events $\mathbf{y} \wedge \beta_n$ are mutually exclusive, and hence:

$$\sum_{\beta_n} \mathsf{Pr}_r(\mathbf{x} \wedge \beta_n) = \mathsf{Pr}_r(\mathbf{x} \wedge \bigvee_{\beta_n} \beta_n) = \mathsf{Pr}_r(\mathbf{x} \wedge \gamma_n)$$
$$\sum_{\beta_n} \mathsf{Pr}_r(\mathbf{y} \wedge \beta_n) = \mathsf{Pr}_r(\mathbf{y} \wedge \bigvee_{\beta_n} \beta_n) = \mathsf{Pr}_r(\mathbf{y} \wedge \gamma_n)$$

Moreover, we have $\sum_{\beta_n} \mathsf{Pr}_r(\beta_n) = \mathsf{Pr}_r(\gamma_n)$, and thus:

$$\sum_{\beta_n} \mathsf{Pr}_p(\mathbf{x})\mathsf{Pr}_r(\beta_n) = \mathsf{Pr}_p(\mathbf{x})\mathsf{Pr}_r(\gamma_n)$$
$$\sum_{\beta_n} \mathsf{Pr}_s(\mathbf{y})\mathsf{Pr}_r(\beta_n) = \mathsf{Pr}_s(\mathbf{y})\mathsf{Pr}_r(\gamma_n).$$

Putting the last two sets of equations together, we obtain:

$$\mathsf{Pr}_r(\mathbf{x} \wedge \gamma_n) = \mathsf{Pr}_p(\mathbf{x})\mathsf{Pr}_r(\gamma_n)$$
$$\mathsf{Pr}_r(\mathbf{y} \wedge \gamma_n) = \mathsf{Pr}_s(\mathbf{y})\mathsf{Pr}_r(\gamma_n)$$

and $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \gamma_n)$ (either $n = p$ or $n = s$). □

Before proceeding to Theorem 5, we first consider the following Lemma.

Proposition 2 (Independence III) *Let node n be a decision node with prime variables $\mathbf{X}$ and sub variables $\mathbf{Y}$. For each element (p, s) of n where s is non-false, the variables $\mathbf{X}$ are independent of variables $\mathbf{Y}$ given prime p.*

Proof If $\mathbf{x} \not\models [p]$, then $\mathsf{Pr}_n(\mathbf{xy} \mid [p]) = \mathsf{Pr}_n(\mathbf{x} \mid [p])\mathsf{Pr}_n(\mathbf{y} \mid [p]) = 0$. Otherwise, if $\mathbf{x} \not\models [p]$, then

$$\begin{aligned}
\mathsf{Pr}_n(\mathbf{xy} \mid [p]) &= \frac{1}{\mathsf{Pr}_n([p])}\mathsf{Pr}_n(\mathbf{xy}, [p]) \\
&= \frac{1}{\mathsf{Pr}_n([p])}\mathsf{Pr}_n(\mathbf{xy}) && \text{as } \mathbf{x} \models [p] \\
&= \frac{1}{\theta}\mathsf{Pr}_n(\mathbf{xy}) && \text{by Theorem 2} \\
&= \frac{1}{\theta}\mathsf{Pr}_p(\mathbf{x}) \cdot \mathsf{Pr}_s(\mathbf{y}) \cdot \theta && \text{by Definition 2} \\
&= \mathsf{Pr}_p(\mathbf{x}) \cdot \mathsf{Pr}_s(\mathbf{y}) \\
&= \mathsf{Pr}_n(\mathbf{x} \mid [p]) \cdot \mathsf{Pr}_n(\mathbf{y} \mid [p]) && \text{by Proposition 1}
\end{aligned}$$

as desired. □

Theorem 5 (Independence II) *Let γ_v be a feasible context for a PSDD node normalized for vtree node v. Variables inside v are independent of those outside v given context γ_v.*

Proof We first consider independence given sub-contexts, via induction. First, the base case is for a root vtree node v, which is trivial, since there are no variables outside the root. Consider then a non-root node v with a parent u and a sibling w. Let $\mathbf{V}$ denote the variables in vtree v, let $\mathbf{W}$ denote the variables in the sibling vtree w, and let $\mathbf{Z}$ denote the variables outside of parent vtree node u. Let β_v be the sub-context for the PSDD node normalized for vtree node v. The sub-context $\beta_v = \beta_u \wedge [p]$ is an extension of a sub-context β_u by a prime p, both belonging to some node normalized for parent vtree node u. When $\beta_v \wedge \mathbf{wz}$ is consistent:

$$\begin{aligned}
&\Pr_r(\mathbf{v} \mid \beta_v, \mathbf{wz}) = \Pr_r(\mathbf{v} \mid \beta_u \wedge [p], \mathbf{wz}) \\
&= \frac{\Pr_r(\mathbf{vw}, [p] \mid \beta_u, \mathbf{z})}{\Pr_r(\mathbf{w}, [p] \mid \beta_u, \mathbf{z})} \\
&= \frac{\Pr_r(\mathbf{vw}, [p] \mid \beta_u)}{\Pr_r(\mathbf{w}, [p] \mid \beta_u)} && \text{by induction} \\
&= \frac{\Pr_u(\mathbf{vw}, [p])}{\Pr_u(\mathbf{w}, [p])} && \text{by Theorem 4} \\
&= \Pr_u(\mathbf{v} \mid [p], \mathbf{w}) = \Pr_u(\mathbf{v} \mid [p]) && \text{by Prop. 2} \\
&= \frac{\Pr_u(\mathbf{v}, [p])}{\Pr_u([p])} = \frac{\Pr_r(\mathbf{v}, [p] \mid \beta_u)}{\Pr_r([p] \mid \beta_u)} && \text{by Theorem 4} \\
&= \Pr_r(\mathbf{v} \mid \beta_u \wedge [p]) = \Pr_r(\mathbf{v} \mid \beta_v)
\end{aligned}$$

as desired.

Suppose now that $\gamma_v = \bigvee_{\beta_v} \beta_v$ is the context of a node n normalized for our (non-root) vtree node v, composed of sub-contexts β_v. All sub-contexts β_v for a node are mutually exclusive, by Theorem 3, hence the probability of a context γ_v is simply the sum of the probabilities of the individual sub-contexts β_v. Similarly, all $\alpha \wedge \beta_v$ are mutually exclusive, for any fixed event α. By Theorem 4, we have that

$$\Pr_r(\mathbf{v} \mid \beta_v) = \Pr_n(\mathbf{v})$$

for all sub-contexts β_v. Hence:

$$\begin{aligned}
&\Pr_r(\mathbf{v} \mid \gamma_v, \mathbf{wz}) = \frac{\Pr_r(\mathbf{v}, \gamma_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} = \frac{\sum_{\beta_v} \Pr_r(\mathbf{v}, \beta_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} \\
&= \frac{\sum_{\beta_v} \Pr_r(\mathbf{v} \mid \beta_v, \mathbf{wz}) \Pr_r(\beta_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} \\
&= \frac{\sum_{\beta_v} \Pr_r(\mathbf{v} \mid \beta_v) \Pr_r(\beta_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} \\
&= \frac{\sum_{\beta_v} \Pr_n(\mathbf{v}) \Pr_r(\beta_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} = \frac{\Pr_n(\mathbf{v}) \sum_{\beta_v} \Pr_r(\beta_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} \\
&= \frac{\Pr_n(\mathbf{v}) \Pr_r(\gamma_v, \mathbf{wz})}{\Pr_r(\gamma_v, \mathbf{wz})} = \Pr_n(\mathbf{v}) = \Pr_r(\mathbf{v} \mid \gamma_v)
\end{aligned}$$

with the last step following by Theorem 4. □

Theorem 6 *Consider a decision node $n = (p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ that is normalized for vtree node v, with left child l and right child r. For evidence $\mathbf{e}$, we have*

$$\Pr_n(\mathbf{e}_v) = \sum_{i=1}^{k} \Pr_{p_i}(\mathbf{e}_l) \cdot \Pr_{s_i}(\mathbf{e}_r) \cdot \theta_i$$

Proof Let $\mathbf{X}$ denote the variables of the left vtree l, and $\mathbf{Y}$ denote the variables of right vtree r. We then have:

$$\begin{aligned}
&\Pr_n(\mathbf{e}_v) = \sum_{\mathbf{x} \models \mathbf{e}_l} \sum_{\mathbf{y} \models \mathbf{e}_r} \Pr_n(\mathbf{xy}) \\
&= \sum_{i=1}^{k} \sum_{\mathbf{x} \models [p_i] \wedge \mathbf{e}_l} \sum_{\mathbf{y} \models \mathbf{e}_r} \Pr_{p_i}(\mathbf{x}) \cdot \Pr_{s_i}(\mathbf{y}) \cdot \theta_i \\
&= \sum_{i=1}^{k} \Big(\sum_{\mathbf{x} \models [p_i] \wedge \mathbf{e}_l} \Pr_{p_i}(\mathbf{x}) \Big) \cdot \Big(\sum_{\mathbf{y} \models \mathbf{e}_r} \Pr_{s_i}(\mathbf{y}) \Big) \cdot \theta_i \\
&= \sum_{i=1}^{k} \Pr_{p_i}(\mathbf{e}_l) \cdot \Pr_{s_i}(\mathbf{e}_r) \cdot \theta_i
\end{aligned}$$

□

Theorem 7 *Consider a PSDD r, variable X, and its leaf vtree node v. Let $n_1, \ldots, n_k$ be all the terminal nodes normalized for v and let $\gamma_{n_1}, \ldots, \gamma_{n_k}$ be their corresponding contexts. For evidence $\mathbf{e}$, we have*

$$\Pr_r(X, \mathbf{e}_{\bar{v}}) = \sum_{i=1}^{k} \Pr_{n_i}(X) \cdot \Pr_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}).$$

Proof By Theorem 3, the contexts $\gamma_{n_1}, \ldots, \gamma_{n_k}$ are mutually exclusive and exhaustive. Thus:

$$\begin{aligned}
&\Pr_r(X, \mathbf{e}_{\bar{v}}) = \sum_{i=1}^{k} \Pr_r(X, \gamma_{n_i}, \mathbf{e}_{\bar{v}}) \\
&= \sum_{i=1}^{k} \Pr_r(X \mid \gamma_{n_i}, \mathbf{e}_{\bar{v}}) \cdot \Pr_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) \\
&= \sum_{i=1}^{k} \Pr_r(X \mid \gamma_{n_i}) \Pr_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) && \text{by Theorem 5} \\
&= \sum_{i=1}^{k} \Pr_{n_i}(X) \cdot \Pr_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) && \text{by Theorem 4}
\end{aligned}$$

as desired. □

Next we prove the correctness of Algorithm 2.

Theorem 11 *Given PSDD r and evidence $\mathbf{e}$, Algorithm 2 computes marginals for each variable X, and the marginals for each context γ_n of decision node n that is normalized for vtree node v:*

$$\begin{aligned}
mrg(X) &= \Pr_r(X, \mathbf{e}_{\bar{v}}) \\
ctx(n) &= \Pr_r(\gamma_n, \mathbf{e}_{\bar{v}})
\end{aligned}$$

Proof The correctness of the variable marginals $mrg(X)$ follows from Theorem 7.

As for the correctness of context marginals, we show this by induction. For the base case, of root node r, there is one context $\top$, and there is no evidence outside of the root vtree

node, so the context has probability 1 (see Line 1 of Algorithm 2).

Next, we consider the context marginals $ctx(p)$ for a prime p normalized for the left vtree node v_l of a vtree node v, with right vtree node v_r. Let $n_1, \ldots, n_k$ be all nodes normalized for v which contain an element (p, s_i, θ_i) mentioning prime p. Moreover, let γ_p be the context of prime p, let γ_{n_i} be the context of node n_i, and let $\gamma_{s_i} = \gamma_{n_i} \wedge [p]$ be the disjunction of sub-contexts of the sub s_i that go through node n_i and prime p. We then have

$$\begin{aligned}
&\mathsf{Pr}_r(\gamma_p, \mathbf{e}_{\bar{v}_l}) = \sum_{i=1}^{k} \mathsf{Pr}_r(\gamma_{n_i} \wedge [p], \mathbf{e}_{\bar{v}_l}) \\
&= \sum_{i=1}^{k} \mathsf{Pr}_r(\gamma_{n_i} \wedge [p], \mathbf{e}_{\bar{v}}, \mathbf{e}_{v_r}) \\
&= \sum_{i=1}^{k} \mathsf{Pr}_r(\mathbf{e}_{v_r} \mid \gamma_{n_i}, [p], \mathbf{e}_{\bar{v}}) \mathsf{Pr}_r([p] \mid \gamma_{n_i}, \mathbf{e}_{\bar{v}}) \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) \\
&= \sum_{i=1}^{k} \mathsf{Pr}_r(\mathbf{e}_{v_r} \mid \gamma_{s_i}, \mathbf{e}_{\bar{v}}) \cdot \mathsf{Pr}_r([p] \mid \gamma_{n_i}, \mathbf{e}_{\bar{v}}) \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) \\
&= \sum_{i=1}^{k} \mathsf{Pr}_r(\mathbf{e}_{v_r} \mid \gamma_{s_i}) \cdot \mathsf{Pr}_r([p] \mid \gamma_{n_i}) \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}) \\
&= \sum_{i=1}^{k} \mathsf{Pr}_{s_i}(\mathbf{e}_{v_r}) \cdot \theta_i \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}})
\end{aligned}$$

which corresponds to the accumulated sum by Algorithm 2 in Line 10 (where $ctx(n)$ corresponds to $\mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}})$, by induction).

We can find an analogous form for the context marginals $ctx(s)$ for a sub s normalized for the right vtree node v_r of a vtree node v, with left vtree node v_l:

$$\mathsf{Pr}_r(\gamma_s, \mathbf{e}_{\bar{v_r}}) = \sum_{i=1}^{k} \mathsf{Pr}_{p_i}(\mathbf{e}_{v_l}) \cdot \theta_i \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}})$$

which corresponds to the accumulated sum by Algorithm 2 in Line 11 (where $ctx(n)$ corresponds to $\mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}})$, by induction). Here, $n_1, \ldots, n_k$ are all nodes normalized for v which contain an element (p_i, s, θ_i) mentioning sub s. Moreover, γ_s is the context of sub s, and the γ_{n_i} are the contexts of nodes n_i. □

Theorem 8 *The parameter estimates of Equations 2 and 3 are the only estimates that maximize the likelihood function.*

Proof Maximizing the likelihood function is equivalent to maximizing the log-likelihood function:

$$LL(\theta \mid \mathbf{D}) = \log L(\theta \mid \mathbf{D}) = \sum_{i=1}^{N} \log \mathsf{Pr}_\theta(\mathbf{d}_i) \qquad (8)$$

where $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ is a complete dataset consisting of N examples $\mathbf{d}_i$. The probability $\mathsf{Pr}_\theta(\mathbf{d}_i)$ is a product of PSDD parameters, one parameter per vtree node v. For each vtree node, there is a unique node n with context γ_n where $\mathbf{d}_i \models \gamma_n$ (since the contexts of nodes in vtree v are mutually exclusive and exhaustive, by Theorem 3). If this node n is a decision node $(p_1, s_1, \theta_1), ..., (p_k, s_k, \theta_k)$, and if $\mathbf{d}_i \models p_j$, then our parameter is θ_j. If n is a terminal node $X : \theta$, then our parameter is θ if $\mathbf{d}_i \models X$, and otherwise it is $1 - \theta$ if $\mathbf{d}_i \models \neg X$. If n is a terminal node X or $\neg X$, then our parameter is just 1. Otherwise, our node n can not be $\bot$, as we would have $\mathsf{Pr}_\theta(\mathbf{d}_i) = 0$ (all examples of the data should be consistent with logical base of the PSDD, which is the underlying assumption made when learning a PSDD from data). Let t denote a terminal node $X : \theta$. We then have

$$\begin{aligned}
LL(\theta \mid \mathbf{D}) &= \sum_{i=1}^{N} \log \mathsf{Pr}_\theta(\mathbf{d}_i) \\
&= \sum_{i=1}^{N} \log \prod_{\mathbf{d}_i \models \gamma_n \wedge p_j} \theta_j \prod_{\mathbf{d}_i \models \gamma_t} \theta^t \\
&= \sum_{i=1}^{N} \sum_{\mathbf{d}_i \models \gamma_n \wedge p_j} \log \theta_j + \sum_{i=1}^{N} \sum_{\mathbf{d}_i \models \gamma_t} \log \theta^t
\end{aligned}$$

Across the examples of the data, we can collect the common terms mentioning the same decision node parameters θ_j and terminal node parameters θ^t. This amounts to counting how many times each parameter is used to compute $\mathsf{Pr}_\theta(\mathbf{d}_i)$ across all examples $\mathbf{d}_i$:

$$\begin{aligned}
LL(\theta \mid \mathbf{D}) &= \sum_{n} \sum_{j} \mathbf{D}\#(p_j, \gamma_n) \cdot \log \theta_j \\
&+ \sum_{t} \mathbf{D}\#(X, \gamma_t) \cdot \log \theta + \mathbf{D}\#(\neg X, \gamma_t) \cdot \log (1 - \theta)
\end{aligned}$$

where $\mathbf{D}\#(\alpha)$ denotes the number of examples in dataset $\mathbf{D}$ that satisfy propositional sentence α.

Optimizing the log-likelihood now amounts to optimizing a set of *independent* components, one for each decision node n and terminal node t. For each decision node n, component $\sum_j \mathbf{D}\#(p_j, \gamma_n) \cdot \log \theta_j$ is maximized uniquely by $\theta_j = \frac{\mathbf{D}\#(p_j, \gamma_n)}{\mathbf{D}\#(\gamma_n)}$. Similarly, for each terminal node $t = X : \theta$, component $\mathbf{D}\#(X, \gamma_t) \cdot \log \theta + \mathbf{D}\#(\neg X, \gamma_t) \cdot \log (1 - \theta)$ is maximized uniquely by $\theta = \frac{\mathbf{D}\#(X, \gamma_t)}{\mathbf{D}\#(\gamma_t)}$ (Darwiche 2009, Lemma 17.1). □

Theorem 9 *Let* Pr *be a probability distribution and suppose that its variables are partitioned into* $\mathbf{X}$ *and* $\mathbf{Y}$*. There must exist a probabilistic* $\mathbf{XY}$*-partition of distribution* Pr*.*

Proof Say that the probability distribution $\mathsf{Pr}(\mathbf{XY})$ has a logical base α. We can construct an (uncompressed) probabilistic $\mathbf{XY}$-partition of distribution Pr, as follows. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be the set of all $2^{|\mathbf{X}|}$ complete instantiations over $\mathbf{X}$. Our probabilistic $\mathbf{XY}$-partition has:

- primes p_i, one prime for each instantiation $\mathbf{x}_i$, and subs s_i, where s_i is the base of distribution $\mathsf{Pr}(\mathbf{Y} \mid \mathbf{x}_i)$ if $\mathbf{x}_i \wedge \alpha$ is consistent, and where $s_i = \bot$ otherwise;
- prime distributions $\mathsf{Pr}_{p_i}(\mathbf{X})$, where $\mathsf{Pr}_{p_i}(\mathbf{x}) = 1$ if $\mathbf{x} = \mathbf{x}_i$ and 0 otherwise;

- sub distributions $\mathsf{Pr}_{s_i}(\mathbf{Y})$, where $\mathsf{Pr}_{s_i}(\mathbf{y}) = \mathsf{Pr}(\mathbf{y} \mid \mathbf{x}_i)$ when $\mathbf{x}_i \wedge \alpha$ is consistent, and is undefined otherwise ($s_i = \bot$ when $\mathbf{x}_i \wedge \alpha$ is not consistent);
- parameters θ_i where $\theta_i = \mathsf{Pr}(\mathbf{x}_i)$ if $\mathbf{x}_i \wedge \alpha$ is consistent, and where $\theta_i = 0$ otherwise.

By construction, our primes p_i are mutually exclusive and exhaustive, so the elements (p_i, s_i) constitute an $\mathbf{XY}$-partition of base α. Further, $\theta_1 + \cdots + \theta_k = \sum_i \mathsf{Pr}(\mathbf{x}_i) = 1$. Finally, the probabilistic partition is equivalent to the original distribution:

$$\mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i = 1 \cdot \mathsf{Pr}(\mathbf{y} \mid \mathbf{x}) \cdot \mathsf{Pr}(\mathbf{x}) = \mathsf{Pr}(\mathbf{xy})$$

when $\mathbf{x} \models p_i$ (i.e., $\mathbf{x} = \mathbf{x}_i$). □

Before proceeding to the proof of Theorem 10, we consider the following proposition which will tell us how to compress a probabilistic $\mathbf{XY}$-partition.

Proposition 3 *Say we have a probability distribution* Pr *with a probabilistic* $\mathbf{XY}$*-partition*

$$(\mathsf{Pr}_{p_1}, \mathsf{Pr}_{s_1}, \theta_1), \ldots, (\mathsf{Pr}_{p_k}, \mathsf{Pr}_{s_k}, \theta_k).$$

If sub distributions Pr_{s_i} *and* Pr_{s_j} *define the same distribution over variables* $\mathbf{Y}$*, then the probabilistic* $\mathbf{XY}$*-partition we obtain by replacing the two elements* $(\mathsf{Pr}_{p_i}, \mathsf{Pr}_{s_i}, \theta_i)$ *and* $(\mathsf{Pr}_{p_j}, \mathsf{Pr}_{s_j}, \theta_j)$ *by the single element* $(\mathsf{Pr}_{p_{ij}}, \mathsf{Pr}_{s_i}, \theta_i + \theta_j)$ *induces the same distribution* Pr*, when* $\mathsf{Pr}_{p_{ij}}$ *is:*

$$\mathsf{Pr}_{p_{ij}}(\mathbf{x}) = \begin{cases} \frac{\theta_i}{\theta_i + \theta_j} \mathsf{Pr}_{p_i}(\mathbf{x}), & \text{if } \mathbf{x} \models p_i; \\ \frac{\theta_j}{\theta_i + \theta_j} \mathsf{Pr}_{p_j}(\mathbf{x}), & \text{if } \mathbf{x} \models p_j; \\ 0, & \text{otherwise}. \end{cases}$$

Proof Let $\mathsf{Pr}^\star$ denote the probability distribution induced by the new probabilistic partition. First, note that the bases of Pr_{p_i} and Pr_{p_j} are mutually exclusive. Hence, if $\mathbf{x} \models p_i$:

$$\begin{aligned} \mathsf{Pr}^\star(\mathbf{xy}) &= \mathsf{Pr}_{p_{ij}}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot (\theta_i + \theta_j) \\ &= \frac{\theta_i}{(\theta_i + \theta_j)} \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot (\theta_i + \theta_j) \\ &= \theta_i \cdot \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) = \mathsf{Pr}(\mathbf{xy}). \end{aligned}$$

If instead $\mathbf{x} \models p_j$, then

$$\begin{aligned} \mathsf{Pr}^\star(\mathbf{xy}) &= \mathsf{Pr}_{p_{ij}}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot (\theta_i + \theta_j) \\ &= \frac{\theta_j}{(\theta_i + \theta_j)} \mathsf{Pr}_{p_j}(\mathbf{x}) \cdot \mathsf{Pr}_{s_j}(\mathbf{y}) \cdot (\theta_i + \theta_j) \\ &= \theta_j \cdot \mathsf{Pr}_{p_j}(\mathbf{x}) \cdot \mathsf{Pr}_{s_j}(\mathbf{y}) = \mathsf{Pr}(\mathbf{xy}). \end{aligned}$$

Otherwise, $\mathbf{x} \models p_l$ for some $l \notin \{i, j\}$. As both probabilistic partitions contain the same element $(\mathsf{Pr}_{p_l}, \mathsf{Pr}_{s_l}, \theta_l)$, we have $\mathsf{Pr}^\star(\mathbf{xy}) = \mathsf{Pr}(\mathbf{xy})$. □

Theorem 10 *A probabilistic* $\mathbf{XY}$*-partition is compressed iff its sub distributions are distinct. A distribution has a unique compressed* $\mathbf{XY}$*-partition for each* $\mathbf{X}$ *and* $\mathbf{Y}$.[9]

[9]Consider a prime distribution Pr_{p_i} where sub $s_i = \bot$. Since $\theta_i = 0$ in this case, the non-zero probabilities of the prime distribution Pr_{p_i} are irrelevant to the distribution represented by the probabilistic $\mathbf{XY}$-partition. The uniqueness claimed in this theorem is modulo such prime distributions.

Proof Let $(\mathsf{Pr}_{p_1}, \mathsf{Pr}_{s_1}, \theta_1), \ldots, (\mathsf{Pr}_{p_k}, \mathsf{Pr}_{s_k}, \theta_k)$ be a probabilistic $\mathbf{XY}$-partition for Pr, as defined in the proof of Theorem 9. This probabilistic partition is in general not compressed, but we can compress it as follows. If the sub distributions Pr_{s_i} and Pr_{s_j} define the same distribution over variables $\mathbf{Y}$, then according to Proposition 3, we can create a new probabilistic partition by replacing the two elements $(\mathsf{Pr}_{p_i}, \mathsf{Pr}_{s_i}, \theta_i)$ and $(\mathsf{Pr}_{p_j}, \mathsf{Pr}_{s_j}, \theta_j)$ by a single element $(\mathsf{Pr}_{p_{ij}}, \mathsf{Pr}_{s_i}, \theta_i + \theta_j)$ where $\mathsf{Pr}_{p_{ij}}$ is a prime distribution with base $p_i \vee p_j$. The global distribution Pr is invariant to this operation, hence we can repeatedly apply this operation until all sub distributions are distinct. Moreover, the final probabilistic partition that we obtain is invariant to the order in which we apply these operations. Hence, this process yields the unique and compressed probabilistic $\mathbf{XY}$-partition for Pr. □