Appeared in: J. Kohlas and S. Moral (eds.), Algorithms for Uncertainty and Defeasible Reasoning, Handbook of Defeasible Reasoning and Uncertainty Managment Systems, Vol. 5, 2000, pp. 5--39, Kluwer Academic Publishers, Dordrecht

Computation in Valuation Algebras

Jürg Kohlas *

Institute of Informatics IIUF
University of Fribourg
CH – 1700 Fribourg (Switzerland)
juerg.kohlas@unifr.ch

Prakash P. Shenoy

Kansas University Business School
Summerfield Hall
Lawrence, KS 66045–2003 USA
pshenoy@ukans.edu

November 15, 1999

Abstract

Many different formalisms for treating uncertainty or, more generally, information and knowledge, have a common underlying algebraic structure. The essential algebraic operations are combination, which corresponds to aggregation of knowledge, and marginalization, which corresponds to focusing of knowledge. This structure is called a *valuation algebra*. Besides managing uncertainty in expert systems, valuation algebras can also be used to to represent constraint satisfaction problems, propositional logic, and discrete optimization problems. This chapter presents an axiomatic approach to valuation algebras. Based on this algebraic structure, different inference mechanisms that use local computations are described. These include the fusion algorithm and, derived from it, the Shenoy-Shafer architecture. As a particular case, computation in idempotent valuation algebras, also called information algebras, is discussed. The additional notion of continuers is introduced and, based on it, two more computational architectures, the Lauritzen-Spiegelhalter and the HUGIN architecture, are presented. Finally, different models of valuation algebras are considered. These include probability functions, Dempster-Shafer belief functions, Spohnian disbelief functions, and possibility functions. As further examples, linear manifolds and systems of linear equations, convex polyhedra and linear inequalities, propositional logic and information systems, and discrete optimization are mentioned.

*Research supported by grant No. 2100–042927.95 of the Swiss National Foundation for Research.

Contents

1 Introduction

The main goal of this chapter is to describe an abstract framework called *valuation algebra* for computing marginals using local computation. The valuation algebra framework is useful in many domains, and especially for managing uncertainty in expert systems using probability, Dempster-Shafer belief functions, Spohnian epistemic belief theory, and possibility theory.

The valuation algebra framework was first introduced by Shenoy (Shenoy, 1989) and was inspired by the formulation of simple axioms that enable local

computation (Shenoy & Shafer, 1990). In valuation algebras, we represent knowledge by entities called valuations, and we make inferences using two operations, called marginalization and combination. Marginalization corresponds to focusing of knowledge to a narrower domain, and combination corresponds to aggregation of different pieces of knowledge. Inferences are made by computing marginals of a combination of several valuations.

The framework of valuation algebras is sufficiently abstract to include many different formalisms. In this chapter, it is shown how probability models (Bayesian networks for example), Dempster-Shafer belief function models, Spohn's epistemic belief models, and possibility theory fit into the framework of valuation algebras.

Besides managing uncertainty in expert systems, valuation algebras can also be used, for example, to represent constraint satisfaction problems, propositional logic (Shenoy, 1994b; Kohlas *et al.*, 1998) and discrete optimization problems (Shenoy, 1991b; Shenoy, 1996).

In many cases, the computation of the combination of several valuations is computationally intractable. However, valuation algebras impose axioms on marginalization and combination which make it possible to compute marginals of a combination of valuations without explicitly computing the combination. This is done by so-called local computation. In this chapter, these axioms are stated. Based on them, a fusion algorithm for computing a marginal using local computation is described. Furthermore, different computational architectures for the computation of multiple marginals, some of them depending on an extended set of axioms, are presented. They all can be formulated as message passing schemes in join trees.

An outline of the chapter is as follows. Section 2 introduces valuation algebras. Section 3 describes the fusion algorithm for computing a marginal. Sections 4 and 5 present different computational architectures, namely Shafer-Shenoy architecture, Lauritzen-Spiegelhalter architecture, and HUGIN architecture. The latter two systems depend on an additional concept, called continuers (Shafer, 1991) (or alternatively division (Shenoy, 1994a; Lauritzen & Jensen, 1997)). Continuation is introduced in section 5. Examples of abstract computation are dispersed in the different sections. Section 6 finally mentions some further examples of abstract computation.

2 Valuation Algebras

2.1 The Framework

We consider reasoning in this chapter to be concerned with a finite set of variables each of which is associated with a finite set of possible values (called its frame) exactly one which is assumed to be true. Thus, the first ingredients for valuation algebras are *variables*, *frames* and *configurations*. Variables will be designated by capital Roman alphabets such as $X, Y, \ldots$. The symbol Ω_X is used to denote the set of possible values of a variable X, called the frame of X. We use lower-case Roman alphabets such as $x, y, \ldots, s, r, t, \ldots$ to denote sets of variables.

Given a nonempty set s of variables, let Ω_s denote the Cartesian product of the frames Ω_X of the variables $X \in s$,

$$\Omega_s = \prod_{X \in s} \Omega_X. \tag{1}$$

Ω_s is called the frame of s. The elements of Ω_s are called configurations of s. We use lower-case, bold-faced letters such as $\mathbf{x}, \mathbf{y}, \ldots$ to denote configurations.

It is convenient to extend this terminology to the case where s is empty. We adopt the convention that the frame of the empty set consists of a single configuration and we use the symbol $\diamond$ to name that configuration; $\Omega_\emptyset = \{\diamond\}$.

The primitive elements of a valuation algebra are *valuations*. Intuitively, a valuation represents some knowledge about the possible values of a set s of variables. So, each valuation refers to some definite set of variables called its *domain*. Given a (possibly empty) set s of variables, there is a set Φ_s of valuations. The elements of Φ_s are called valuations for s. Let r denote the set of all variables, and let Φ denote the set of all valuations, i.e. $\Phi = \cup_{s \subseteq r} \Phi_s$. We use lower-case Greek letters such as $\varphi, \psi, \ldots$ to denote valuations. Finally, let D denote the lattice of subsets of r (i.e. the power set of r).

If φ is a valuation for s, then we write $d(\varphi) = s$ and call s the *domain* of φ. d is a mapping of Φ onto the power set D.

The following are well-known examples of valuation:

- In probability theory, valuations are called probability potentials. A *probability potential* for s is a function $p : \Omega_s \to R^+$, where R^+ denotes

the set of nonnegative real numbers. Usually the frames are assumed to be finite sets. Then the probability potentials are just $|s|$-dimensional tables with $\prod_{X \in s} |\Omega_X|$ entries. If the sum of all entries equals 1, then the potential represents a discrete probability mass function for the variables in s. In this case, the potential is said to be normalized. But a probability potential may also represent a conditional probability table, or an observed event (the indicator function of a subset of Ω_s).

- In Dempster-Shafer theory of belief functions, a valuation for s is a so-called *commonality function*, that is a mapping $q : \mathcal{P}(\Omega_s) \to R^+$ from the power set $\mathcal{P}(\Omega_s)$ of Ω_s into the nonnegative real numbers, such that for

$$m(A) = \sum_{C:\, C \supseteq A} (-1)^{|C-A|} q(C) \tag{2}$$

we have $m(A) \geq 0$ for all subsets A of Ω_s (including the empty set). If the $m(A)$ add up to 1, then they are called *basic probability assignments*, and they determine a *degree of belief*

$$bel(A) = \sum_{C:\, \emptyset \neq C \subseteq A} m(C). \tag{3}$$

This defines a *belief function*. If furthermore, $m(\emptyset) = 0$, then the belief function is said to be normalized. Note furthermore, that commonality functions may also inversely be obtained from m-functions by

$$q(A) = \sum_{C:\, C \supseteq A} m(C) \tag{4}$$

(see (Shafer, 1976)). Note that a commonality function has $2^{|\Omega_s|}$ values. It is thus impractical to work with commonality functions except when $|s|$ is small. On the other hand, $m(A)$ will often be equal to zero for many subsets A. It is therefore a more reasonable representation of a belief function.

- In *Spohn's epistemic belief theory* the valuations are called disbelief potentials. A *disbelief potential* for s is a function $\delta : \Omega_s \to Z^+$, where Z^+ is the set of nonnegative integers (see (Spohn, 1988)).

- Finally, in *possibility theory* valuations are called possibility potentials. A *possibility potential* for s is a function $\pi : \Omega_s \to [0, 1]$ (see (Zadeh, 1978; Zadeh, 1979)).

There are two operations assumed to be defined for valuations. The *combination* is a binary operation $\Phi \times \Phi \to \Phi$ denoted by $(\varphi, \psi) \mapsto \varphi \otimes \psi$. It represents aggregation of knowledge. This operation is assumed to be *associative* and *commutative* (in other words Φ is assumed to be a commutative semigroup under combination). Furthermore, if φ is a valuation for s and ψ a valuation for t, then $\varphi \otimes \psi$ is a valuation for $s \cup t$.

For any s, a set of valuations Φ_s with domain s is itself a semigroup. We may adjoin to it a neutral element e_s such that $\varphi \otimes e_s = e_s \otimes \varphi = \varphi$ for all valuations $\varphi \in \Phi_s$.

Marginalization is another binary operation $\Phi \times D \to \Phi$. With any valuation φ and domain x, we associate a valuation $\varphi^{\downarrow x}$, which is called the *marginal* of φ for x, and is a valuation whose domain is $x \cap d(\varphi)$. Notice that $\varphi^{\downarrow x} = \varphi^{\downarrow x \cap d(\varphi)}$. Intuitively *marginalization* represents focusing of the knowledge captured by φ for $d(\varphi)$ to the smaller domain $x \cap d(\varphi)$.

In the examples above these operations are defined as follows:

- For *probability potentials*, combination is pointwise multiplication. More precisely, if φ, ψ represent two probability potentials for s and t respectively, then, for $\mathbf{x} \in \Omega_{s \cup t}$

  $$(\varphi \otimes \psi)(\mathbf{x}) = \varphi(\mathbf{x}^{\downarrow s})\psi(\mathbf{x}^{\downarrow t}) \tag{5}$$

  where $\mathbf{x}^{\downarrow s}$ denotes the projection of $\mathbf{x}$ to the subset s of variables. Equation 5 defines the semigroup of nonnegative real functions. The neutral elements e_s are defined by $e_s(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega_s$.

  Marginalization to a subset of variables x is defined as summing out all variables not in x. Thus, if φ is a probability potential for s, then for $\mathbf{x} \in \Omega_{x \cap s}$

  $$\varphi^{\downarrow x}(\mathbf{x}) = \sum_{\mathbf{x}^{\downarrow s-x} \in \Omega_{s-x}} \varphi(\mathbf{x}^{\downarrow x \cap s}, \mathbf{x}^{\downarrow s-x}). \tag{6}$$

- For *Dempster-Shafer belief functions*, combination is pointwise multiplication of commonality function. More precisely, if φ, ψ are two commonality functions for s and t respectively, then for a subset A of $\Omega_{s \cup t}$

  $$(\varphi \otimes \psi)(A) = \varphi(A^{\downarrow s})\psi(A^{\downarrow t}) \tag{7}$$

where $A^{\downarrow s}$ denotes the projection of the set A to the subset of variables s. This combination can also be defined in terms of the m-functions as defined in Equation 2. If m_φ and m_ψ denote the m-functions corresponding to φ and ψ respectively, then

$$m_{\varphi\otimes\psi}(A) = \sum_{A_1^{\uparrow s\cup t}\cap A_2^{\uparrow s\cup t}=A} m_\varphi(A_1)m_\psi(A_2) \tag{8}$$

where $A_i^{\uparrow s\cup t}$ denotes the cylindric extension of the set A_i to the set of variables $s \cup t$. This is a variant of the so-called *Dempster's rule of combination* (Dempster, 1967; Shafer, 1976) without normalization. The neutral elements e_s here are defined by the m-function as follows: $m(\Omega_s) = 1, m(A) = 0$ otherwise.

Marginals for Dempster-Shafer belief functions are best explained using the m-functions. If m_φ is the m-function corresponding to a commonality function φ with domain s and A is a subset of $\Omega_{x\cap s}$, then

$$m_{\varphi^{\downarrow x}}(A) = \sum_{C:\, C\subseteq\Omega_s, C^{\downarrow x\cap s}=A} m_\varphi(C). \tag{9}$$

- For *Spohn disbelief potentials*, combination is pointwise addition (Shenoy, 1991a). More precisely, if φ, ψ are two disbelief potentials for s and t, respectively, then

$$(\varphi \otimes \psi)(\mathbf{x}) = \varphi(\mathbf{x}^{\downarrow s}) + \psi(\mathbf{x}^{\downarrow t}) \tag{10}$$

for all $\mathbf{x} \in \Omega_{s\cup t}$.

Marginalization of disbelief potentials φ to t is minimization over the frame $\Omega_{d(\varphi)-t}$ of the variables to be eliminated. Suppose φ is a disbelief potential for s and suppose $t \subseteq s$. Then the marginal of φ for t, denoted by $\varphi^{\downarrow t}$ is given by

$$\varphi^{\downarrow t}(\mathbf{x}) = \min_{\mathbf{y}\in\Omega_{s-t}} \varphi(\mathbf{x}, \mathbf{y}) \tag{11}$$

for all $\mathbf{x} \in \Omega_t$.

- Similarly, for *possibility potentials*, combination will be pointwise multiplication just as for probability potentials, but marginalization will be maximization over the variables to be eliminated. Suppose φ is a possibility potential for s and suppose $t \subseteq s$. Then the marginal of φ for t, denoted by $\varphi^{\downarrow t}$ is given by

$$\varphi^{\downarrow t}(\mathbf{x}) = \max_{\mathbf{y}\in\Omega_{s-t}} \varphi(\mathbf{x}, \mathbf{y}) \tag{12}$$

for all $\mathbf{x} \in \Omega_t$.

2.2 Axiomatics

Given a system Φ of valuations with its operations of combination and marginalization, the problem of inference can be posed in the following way. Suppose we are given a knowledge base consisting of a finite set of valuations $\varphi_1, \varphi_2, \ldots, \varphi_m$. We will refer to the combined knowledge $\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$ as the *joint valuation*. A specified subset x of variables is of particular interest and the question of interest is what is the marginal of the joint valuation for this subset: $(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{\downarrow x}$.

Here is an example drawn from (Lauritzen & Spiegelhalter, 1988):

> Shortness of breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea.

In this example, there are eight binary variables: A (visit to Asia), S (Smoking), T (Tuberculosis), L (Lung cancer), B (Bronchitis), E (Either tuberculosis or lung cancer), X (positive X-ray) and D (Dyspnoea). Prior to any observations, there are eight valuations: α for $\{A\}$, σ for $\{S\}$, τ for $\{A, T\}$, λ for $\{S, L\}$, β for $\{S, B\}$, ϵ for $\{T, L, E\}$, ξ for $\{E, X\}$, and δ for $\{E, B, D\}$. Additional observations will also be modeled by valuations. Suppose for example, a patient is observed, which visited Asia recently and is suffering from dyspnoea. These two observations can be modeled by two valuations o_A for $\{A\}$ and o_D for $\{D\}$ respectively. A question of interest may be: What are the chances that the patient is suffering from Tuberculosis?

A graphical display of such a set of valuations is shown in Figure 1. There, variables are represented by circular nodes, and valuations are represented by square nodes. Each valuation is connected by an edge to each of the variables in its domain. Such a bipartite graph is called a *valuation network*, and it provides a qualitative description of the knowledge base.

A look at the examples of valuations above shows that their sizes increase exponentially in the number of variables in the domain and sometimes, as in the case of belief functions, also in the sizes of the frames. Thus, combination and marginalization is feasible only on domains and frames of small cardinality. In particular, even if all factors φ_i in the joint valuation above are

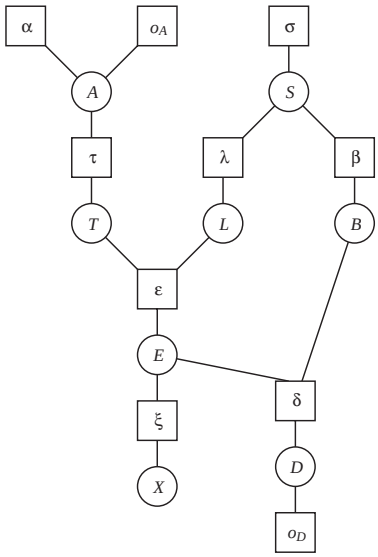


Figure 1: The valuation network for the medical example.

defined on small domains, their combination may soon become intractable as the size of domains and frames increases with each combination. Therefore, a direct solution of the problem posed above is in most cases infeasible.

However, if combination and marginalization satisfy certain conditions, which we call axioms, then we can compute the desired marginal $(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{\downarrow x}$ without explicitly computing the joint valuation $\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$. We state these conditions as part of the following system of axioms for a set Φ of valuations for subsets of a set r of variables, in which the operations of combination and marginalization are defined:

1. *Semigroup.* Φ is associative and commutative under combination. For each $s \in D$ there is an element e_s with $d(e_s) = s$ such that for all $\varphi \in \Phi$ with $d(\varphi) = s, e_s \otimes \varphi = \varphi \otimes e_s = \varphi$.

2. *Domain of Combination.* For $\varphi, \psi \in \Phi$,

$$d(\varphi \otimes \psi) = d(\varphi) \cup d(\psi). \tag{13}$$

3. *Marginalization.* For $\varphi \in \Phi$ and $x \in D$,

$$\varphi^{\downarrow x} = \varphi^{\downarrow x \cap d(\varphi)}, \qquad d(\varphi^{\downarrow x}) = x \cap d(\varphi), \qquad \varphi^{\downarrow d(\varphi)} = \varphi. \tag{14}$$

4. *Transitivity of Marginalization.* For $\varphi \in \Phi$,

$$(\varphi^{\downarrow y})^{\downarrow x} = \varphi^{\downarrow x \cap y}. \tag{15}$$

5. *Distributivity of Marginalization over Combination.* For $\varphi, \psi \in \Phi$ with $d(\varphi) = x$,

$$(\varphi \otimes \psi)^{\downarrow x} = \varphi \otimes \psi^{\downarrow x}. \tag{16}$$

6. *Neutrality.* For $x, y \in D$,

$$e_x \otimes e_y = e_{x \cup y}. \tag{17}$$

A system of valuations Φ with lattice D of domains, combination $\otimes$, and marginalization $\downarrow$, which satisfies these axioms is called a *valuation algebra* and denoted by $(\Phi, D, \otimes, \downarrow)$.

It is especially the distributivity of marginalization over combination axiom, which is crucial for local computation. It says that instead of combining two valuations and then marginalizing the combination to the domain of one of the valuations, we can, as well, first marginalize the other valuation to the intersection of the two domains and then combine. In the first case, the combination leads to a valuation with domain $x \cup y$, whereas in the second case, all operations are done in the smaller domains x and y.

It can be shown that all examples introduced so far (probability potentials, belief functions, disbelief, and possibility potentials) satisfy these axioms (Shenoy & Shafer, 1990; Shenoy, 1991a; Shenoy, 1992a). Thus, they are all examples of valuation algebras.

Instead of marginalization, another primitive operation called *variable elimination* can be defined as follows:

$$\varphi^{-X} = \varphi^{\downarrow d(\varphi) - \{X\}}. \tag{18}$$

Notice that if $X \notin d(\varphi)$, then $\varphi^{-X} = \varphi$. It can be shown, that the transitivity and distributivity axioms translate into the following alternative axioms

1. *Transitivity of Elimination.* For $\varphi \in \Phi$, and $X, Y \in r$,

$$(\varphi^{-X})^{-Y} = (\varphi^{-Y})^{-X}. \tag{19}$$

2. *Distributivity of Elimination over Combination.* For $\varphi, \psi \in \Phi$, with $X \notin d(\varphi)$ and $X \in d(\psi)$,

$$(\varphi \otimes \psi)^{-X} = \varphi \otimes \psi^{-X}. \tag{20}$$

Transitivity of elimination allows to define unambiguously the elimination of several variables $(\cdots((\varphi^{-X_1})^{-X_2})\cdots)^{-X_n}$ as $\varphi^{-\{X_1,X_2,\ldots,X_n\}}$ independently of the actual elimination sequence used. This in turn allows us to express marginalization as variable elimination,

$$\varphi^{\downarrow x} = \varphi^{-(d(\varphi)-x)}. \tag{21}$$

Thus, the two operations of marginalization and variable elimination together with their respective axioms are equivalent and we can use either operation at our convenience.

3 The Fusion Algorithm

In this section, the fusion algorithm for computing the marginal for a subset of variables using local computation in a valuation algebra is described (Cannings *et al.*, 1978; Shenoy, 1992b). The fusion algorithm was called *peeling* by (Cannings *et al.*, 1978) and is also commonly referred to as *variable elimination.*

Let $\varphi_1, \varphi_2, \ldots, \varphi_m$ be valuations from a valuation algebra $(\Phi, D, \otimes, \downarrow)$. Suppose the marginal of the joint valuation for a subset of variables x is to be computed, $(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{\downarrow x}$. The basic idea of the fusion algorithm is to successively delete all variables not in x from the joint valuation. As a consequence of the transitivity of elimination axiom, the variables may be eliminated in any sequence. But different sequences may involve different computational efforts. Comments on good deletion sequences will be given at the end of this section.

First, consider the case of elimination of one variable from a combination of valuations. Suppose $d(\varphi_i) = s_i$. Then $d(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m) = s_1 \cup s_2 \cup \cdots \cup s_m$. Let $Y \in s_1 \cup s_2 \cup \cdots \cup s_m$ and suppose Y is to be deleted from $\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$. Lemma 1, which is a direct consequence of the distributivity axiom, tells that this can be done using local computation.

Lemma 1 *Under the assumptions of the preceding paragraph,*

$$(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{-Y} = \varphi^{-Y} \otimes \left(\bigotimes_{i:\, Y \notin s_i} \varphi_i \right) \tag{22}$$

where

$$\varphi \quad = \quad \bigotimes_{i:\, Y \in s_i} \varphi_i. \tag{23}$$

If $\varphi = \bigotimes_{i:\, Y \in s_i} \varphi_i$, then the φ_i's are called *factors* of φ. If we compare the factors of $(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{-Y}$ in (22) with those of $\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$, we observe that in deleting Y, the factors that do not contain Y in their domains remain unchanged and the factors that do contain Y in their domain are first combined and then Y is eliminated from the combination. This operation is called *fusion*. A formal definition is as follows: Consider a set of m valuations $\varphi_1, \varphi_2, \ldots, \varphi_m$ where $d(\varphi_i) = s_i$. Let $Fus_Y(\{\varphi_1, \varphi_2, \ldots, \varphi_m\})$ denote the set of valuations after fusing the valuations in the set $\{\varphi_1, \varphi_2, \ldots, \varphi_m\}$ with respect to the variable Y:

$$Fus_Y(\{\varphi_1, \varphi_2, \ldots, \varphi_m\}) = \{\varphi^{-Y}\} \cup \{\varphi_i :\ Y \notin s_i\}. \tag{24}$$

where φ is as defined in Equation 23.

Using the definition of fusion, the result of Lemma 1 can be expressed as

$$(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{-Y} = \bigotimes Fus_Y(\{\varphi_1, \varphi_2, \ldots, \varphi_m\}). \tag{25}$$

By successively eliminating variables, we can compute the marginal of the joint valuation for x. This result is stated formally as follows.

Theorem 1 Fusion Algorithm *(Shenoy, 1992b). Suppose $\varphi_1, \varphi_2, \ldots, \varphi_m$ are valuations from a valuation algebra $(\Phi, D, \otimes, \downarrow)$, where $d(\varphi_i) = s_i$. Let s denote $s_1 \cup s_2 \cup \ldots \cup s_m$, and let $X_1, X_2, \ldots, X_n$ be a sequence of the variables in $s - x$ (where $n = |s - x|$). Then*

$$\begin{aligned} &(\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m)^{\downarrow x} \\ &\quad = \bigotimes Fus_{X_n} \left(\cdots \left(Fus_{X_2} \left(Fus_{X_1} \left(\{\varphi_1, \varphi_2, \ldots, \varphi_m\} \right) \right) \right) \cdots \right). \end{aligned} \tag{26}$$

To illustrate the fusion algorithm consider the valuation network of Figure 1. Suppose we need to compute the marginal with respect to $\{T\}$, i.e. $(\alpha \otimes o_A \otimes \sigma \otimes \tau \otimes \lambda \otimes \beta \otimes \epsilon \otimes \xi \otimes \delta \otimes o_D)^{\downarrow \{T\}}$. Consider the deletion sequence A, X, S, D, B, L, E. This gives the following sequence of fusions

1. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, \sigma, \lambda, \beta, \epsilon, \xi, \delta, o_D\}$,
2. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, \sigma, \lambda, \beta, \epsilon, \xi^{-X}, \delta, o_D\}$,
3. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, (\sigma \otimes \lambda \otimes \beta)^{-S}, \epsilon, \xi^{-X}, \delta, o_D\}$,
4. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, (\sigma \otimes \lambda \otimes \beta)^{-S}, \epsilon, \xi^{-X}, (\delta \otimes o_D)^{-D}\}$,
5. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, ((\sigma \otimes \lambda \otimes \beta)^{-S} \otimes (\delta \otimes o_D)^{-D})^{-B}, \epsilon, \xi^{-X}\}$,
6. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, (((\sigma \otimes \lambda \otimes \beta)^{-S} \otimes (\delta \otimes o_D)^{-D})^{-B} \otimes \epsilon)^{-L}, \xi^{-X}\}$,
7. $\{(\alpha \otimes o_A \otimes \tau)^{-A}, [(((\sigma \otimes \lambda \otimes \beta)^{-S} \otimes (\delta \otimes o_D)^{-D})^{-B} \otimes \epsilon)^{-L} \otimes \xi^{-X}]^{-E}\}$

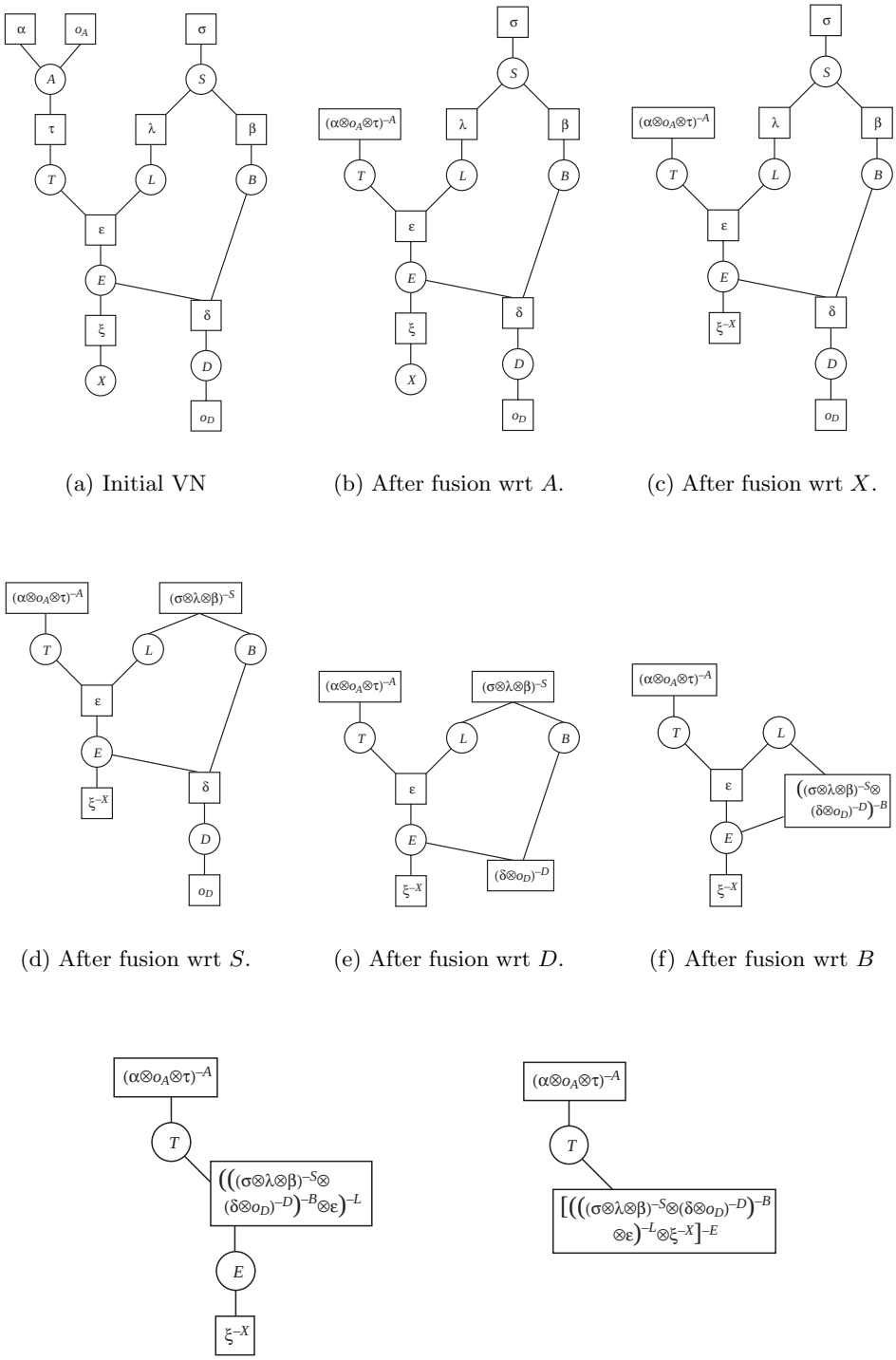
Theorem 1 tell us that

$$\begin{aligned} & (\alpha \otimes o_A \otimes \sigma \otimes \tau \otimes \lambda \otimes \beta \otimes \epsilon \otimes \xi \otimes \delta \otimes o_D)^{\downarrow\{T\}} \\ = \; & (\alpha \otimes o_A \otimes \tau)^{-A} \otimes \\ & [(((\sigma \otimes \lambda \otimes \beta)^{-S} \otimes (\delta \otimes o_D)^{-D})^{-B} \otimes \epsilon)^{-L} \otimes \xi^{-X}]^{-E}. \end{aligned} \tag{27}$$

The fusion algorithm is graphically shown in Figure 2.

As mentioned above, different elimination sequences may involve different computational efforts. A good elimination sequence generates fusions with small domains or frames for the variable to be eliminated. Finding optimal elimination sequences is a secondary optimization problem that has been shown to be NP-hard (Arnborg *et al.*, 1987). But there are several heuristics for finding good elimination sequences (Olmsted, 1983; Kong, 1986; Mellouli, 1988; Zhang, 1988; Kjærulff, 1990).

One such heuristic is called one-step-look-ahead (Olmsted, 1983; Kong, 1986). This heuristic tells us which variable to eliminate next. As per this heuristic, the variable that should be eliminated next is one that leads to combination over the smallest frame with ties broken arbitrarily. In particular, if a variable is in the domain of only one valuation, then this heuristic would pick such a variable first for elimination since no combination is involved. For example, in the valuation algebra of Figure 1, if we assume that each variable has a frame consisting of two configurations, then this heuristic would pick either A or X for the first elimination since elimination of A or X involves combination on the frame for two variables whereas elimination of any other variable would lead to combination over a frame of more than two variables. The elimination sequence A, X, S, D, B, L, E used to illustrate the fusion algorithm is one of the many deletion sequences suggested by the one-step-look-ahead heuristic.



(a) Initial VN (b) After fusion wrt A. (c) After fusion wrt X.

(d) After fusion wrt S. (e) After fusion wrt D. (f) After fusion wrt B

(g) After fusion wrt L. (h) After fusion wrt E.

Figure 2: The fusion algorithm for the valuation network of Figure 1 using the deletion sequence A, X, S, D, B, L, E.

4 Shenoy-Shafer Architecture

4.1 Message-Passing in Join Trees

If we can compute the marginal of the joint valuation for one subset, then we can compute the marginals for all subsets. We simply compute them one after the other. It is obvious, however, that this will involve much repetition of effort. To avoid this repetition of effort, we will redescribe the fusion algorithm as message-passing in join trees so that we can easily generalize the algorithm to computing multiple marginals. A join tree can be thought of as a data structure that allows us to organize the computation, and more importantly, that allows us to cache the computations to avoid repetition of effort. In the next section, we will describe how the join tree data structure allows us to efficiently compute multiple marginals.

A join tree is a tree whose nodes are subsets of r such that if a variable is in two distinct nodes, then it is in every node on the path between the two nodes (Maier, 1983). Join trees are also called qualitative Markov trees (Shafer *et al.*, 1987), hypertrees (Shenoy & Shafer, 1990), clique trees (Lauritzen & Spiegelhalter, 1988), and junction trees (Jensen *et al.*, 1990a). As we will see, join trees are useful data structures to cache computation.

Typically, the edges in a join tree are undirected. In several computational architectures, we will find it useful to pick one node of the join tree as the *root* and direct all edges toward the root. We will call such a directed join tree a *rooted* join tree.

Consider again the definition of fusion. Suppose we have valuations $\varphi_1, \ldots, \varphi_m$, where $d(\varphi_i) = s_i$. Suppose the valuations are labeled such that $s_1, \ldots, s_j$ contain Y and $s_{j+1}, \ldots, s_m$ do not contain Y. Then

$$Fus_Y(\{\varphi_1, \varphi_2, \ldots, \varphi_m\}) = \{(\varphi_1 \otimes \cdots \otimes \varphi_j)^{\downarrow s-\{Y\}}\} \cup \{\varphi_{j+1}, \ldots, \varphi_m\}, \quad (28)$$

where $s = s_1 \cup \cdots \cup s_j$. We will now describe the fusion operation as message passing in a rooted join tree.

We imagine that $s_1, \ldots, s_j, s$, and $s - \{Y\}$ are all processors connected together in a rooted join tree as shown in Figure 4.1 where the arrows point toward the root $s - \{Y\}$. Processors $s_1, \ldots, s_j$ have stored in them the valuations $\varphi_1, \ldots, \varphi_j$, respectively. Processors s and $s - \{Y\}$ have nothing stored in them. First processors $s_1, \ldots, s_j$ send messages to their inward neighbor s consisting of the valuations stored in them. Next, processor s first combines all messages it receives from its outward neighbors, marginalizes the

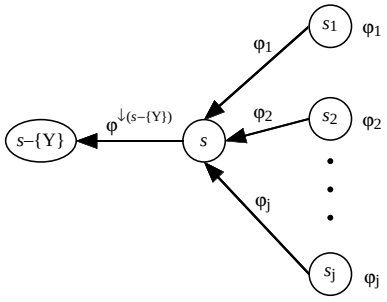


Figure 3: Fusion as message passing in a rooted join tree.

combination $\varphi = \varphi_1 \otimes \cdots \otimes \varphi_j$ to $s - \{Y\}$, and sends the valuation $\varphi^{\downarrow s-\{Y\}}$ as a message to its inward neighbor $s - \{Y\}$.

Notice that the tree constructed using the procedure described above is a join tree. If a variable X different from Y is in s_i, then it is also in s and in $s - \{Y\}$. Variable Y is in all the subsets except $s - \{Y\}$.

At the beginning of the fusion process, we have valuations $\varphi_1, \ldots, \varphi_m$ with their domains $s_1, \ldots, s_m$. At the end of the fusion operation, we are left with valuations $\varphi^{\downarrow s-\{Y\}}, \varphi_{j+1}, \ldots, \varphi_m$ with domains $s - \{Y\}, s_{j+1}, \ldots, s_m$. If we recursively continue this procedure using some elimination sequence until we have eliminated all variables except those in the subset x whose marginal we desire, we are left with either one subset x or a set of subsets whose union is x. In the latter case, we join all such subsets to their union, x, and stop the process. The result will be a join tree with x as the root.

In the procedure described above for constructing join trees starting from domains of valuations in a valuation algebra, it is possible that we have multiple copies of some subsets in the join tree. This could happen if we have more than one valuation for a specific domain. Or it could happen when we add subsets s and $s - \{Y\}$ during the construction process if we have valuations with these subsets as domains. Although having these multiple copies poses no problem, they may simply add to the storage without any benefits. We can easily eliminate the occurrence of multiple copies by making sure we don't introduce any during the construction process. The following procedure in pseudocode does this.

Let Δ denote the set of all domains for which we have valuations, let x denote the subset for which we desire the marginal, and let Ψ denote the

set of all variables:

$$\Psi = (\bigcup_{s_i \in \Delta} s_i) \cup x. \tag{29}$$

Also, let N denote the set of nodes of the join tree, and let E denote the set of directed edges of the join tree.

A Procedure for Constructing a Rooted Join Tree

INPUT: Δ, x
OUTPUT: N, E
INITIALIZATION
$\Psi_u \leftarrow \Psi - x$ % Ψ_u denotes the set of variables that have not yet been eliminated.%
$\Delta_u \leftarrow \Delta \cup \{x\}$ % Δ_u denotes the set of subsets that have not yet been arranged in the join tree.%
$N \leftarrow \emptyset$
$E \leftarrow \emptyset$
DO WHILE $\Psi_u \neq \emptyset$
  Pick a variable $Y \in \Psi_u$ using some heuristic
  $s \leftarrow \bigcup_{s_i \in \Delta_u s.t. Y \in s_i} s_i$
  $N \leftarrow N \cup \{s_i \in \Delta_u | Y \in s_i\} \cup \{s, s - \{Y\}\}$
  $E \leftarrow E \cup \{(s_i, s) | s_i \in (\Delta_u - \{s\}), Y \in s_i\} \cup \{(s, s - \{Y\})\}$
  $\Psi_u \leftarrow \Psi_u - \{Y\}$
  $\Delta_u \leftarrow [\Delta_u - \{s_i \in \Delta_u | Y \in s_i\}] \cup \{s - \{Y\}\}$
END DO
IF $|\Delta_u| > 1$ THEN DO
  $N \leftarrow N \cup \Delta_u$
  $E \leftarrow E \cup \{(s_i, x) | s_i \in \Delta_u, s_i \neq x\}$
END DO
ELSE DO
  $N \leftarrow N \cup \Delta_u$
END DO

A rooted join tree for the medical example using the elimination sequence A, X, S, D, B, L, E is shown in Figure 4. The subset $\{T\}$ is the root of this join tree.

If a factorization $\varphi = \varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$ of a valuation is given such that for all $i = 1, \ldots, m, d(\varphi_i) \subseteq h_j$ for some h_j in the join tree, then the join

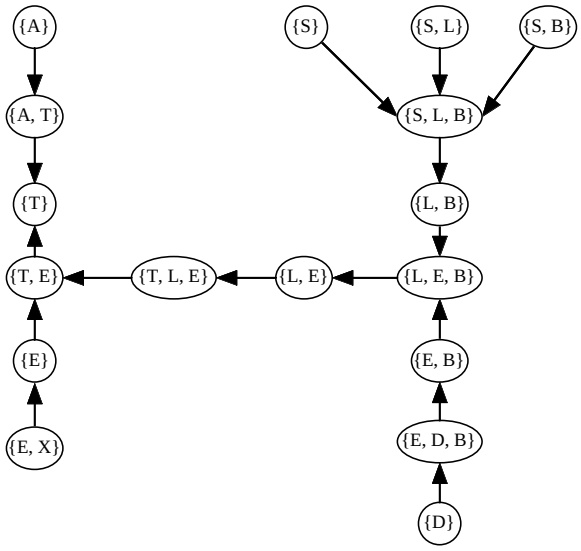


Figure 4: A join tree for the medical example

tree is said to *cover* the factorization of φ. A factorization may have many covering join trees, including the trivial one with the only node r. Interesting covering join trees are those for which the maximal cardinality of the nodes or the frames is as small as possible. This is so, because we will see that marginalization of φ can be done by combination and marginalization within the nodes of the covering join only. Thus, the cardinality of the nodes of the covering join tree bounds the complexity of the operations. Notice that if we start with the domains of all valuation in a system, the join tree constructed using the procedure described above will be a *covering* join tree.

We will now formally describe the architecture for computing the marginal of a product of valuations $\varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$ for some subset x. We start with a rooted join tree whose root is the subset x and where each valuation φ_i is associated with a corresponding subset in the join tree. The protocol for sending messages in the join tree is as follows.

Rule 1 *Messages.* Each node sends a message to its "inward" neighbor (the neighbor toward the root). The root has no inward neighbor and, therefore, does not send a message. The message is computed as follows. First, the node combines all messages it receives from its outward neighbors together with its own valuation (if any), and next, it marginalizes the combination to the domain of the inward neigh-

bor. Thus, each node sends a message to its inward neighbor only *after* it has received a message from each of its outward neighbors. Leaves (nodes without any outward neighbors) can, of course, send messages right away. Notice that if the domain of the inward neighbor is a superset, then no marginalization is involved.

Rule 2 *Marginal.* When the root has received a message from each of its outward neighbors, it combines all messages together with its own valuation (if any) and reports the result as its marginal.

Figure 5 shows the messages sent between neighboring nodes in the join tree of Figure 4 for the example in Section 2.2. The computation of the marginal (Rule 2) is not shown.

This message-passing process is justified by the following lemma.

Lemma 2 *(Shenoy & Shafer, 1990) Suppose l is a leaf node with valuation φ_l, and suppose b is its unique neighbor with valuation φ_b, and suppose φ denotes the joint valuation with domain r. Then*

$$\varphi_l^{\downarrow b} \otimes \varphi_b \otimes \textit{all other } \varphi_j = \varphi^{-(d(\varphi_l)-d(\varphi_b))}. \tag{30}$$

This shows, that a leaf l sends the marginal of its valuation on the domain of its neighbor b to this neighbor according to rule 1. Neighbor b collects the incoming message and combines it with its own valuation. If we remove the leaf from consideration, the lemma applies recursively to the remaining join tree and especially to a leaf in it. Eventually, when all outward neighbors of node b above have been removed, b has combined all incoming messages (rule 1). It becomes a leaf itself and sends its message to the inward neighbor. This corresponds to rule 1. At the end of this process only node x remains, which has then collected and combined all incoming messages. Lemma 2 shows that then all variables outside x have been eliminated and the marginal $\varphi^{\downarrow x}$ is obtained. This justifies finally both rules.

4.2 Computing Multiple Marginals

In this section, we will describe how to modify the algorithm described in the previous subsection so as to compute multiple marginals. The direction of the edges in the join tree construction in the previous subsection indicates the direction of the messages. But we can take any other node b of the join

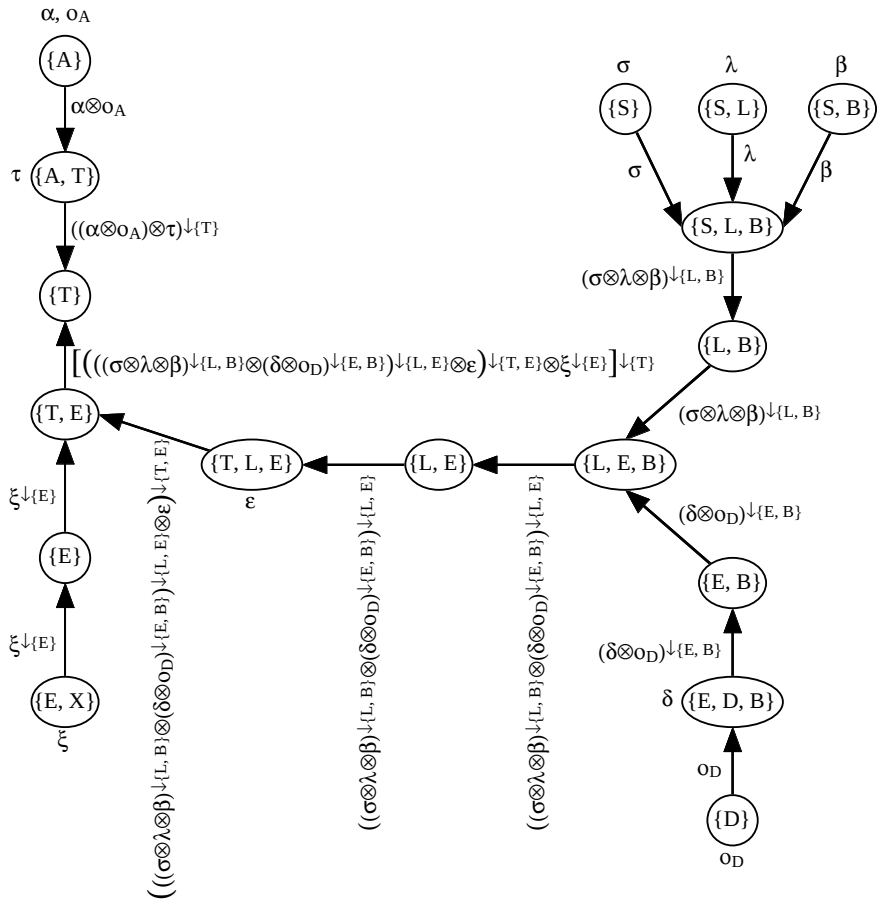


Figure 5: Messages sent between neighboring nodes in the rooted join tree of Figure 4 for the example of Section 2.2. The initial valuation on the nodes are shown next to the respective nodes.

tree as a root and redirect the edges correspondingly. Then we compute the marginal $\varphi^{\downarrow b}$ by the same procedure. But then we may reuse many messages computed before since not all edges will be redirected.

Instead of directing the edges of the join tree, it will be easier to leave the edges undirected and associate directions with the messages. Also, if each node sends a message to each of its neighbors in the undirected join tree (instead of just to its inward neighbor in the rooted join tree), then we can compute marginals for every subset in the join tree. We can achieve this by changing the two rules as follows.

Rule 1SS *Messages.* Each node sends a message to each of its neighbors. Suppose $\mu_{a\to b}$ denotes the message that node a sends to its neighbor b, and suppose $N(a)$ denotes the neighbors of node a in the join tree, and suppose φ_a denotes the valuation associated with node a. Then we have

$$\mu_{a\to b} = \left(\varphi_a \otimes (\bigotimes_{c\in N(a)-\{b\}} \mu_{c\to a})\right)^{\downarrow b}. \tag{31}$$

In words, the message that a sends to b is the combination of all messages that a receives from its *other* neighbors together with its own valuation marginalized to b. Thus, a node sends a message to a neighbor only *after* it has received a message from each of its other neighbors. Leaves (nodes with only one neighbor) can, of course, send messages right away.

Rule 2SS *Marginals.* When a node has received a message from each of its neighbors, it combines all messages together with its own valuation (if any) and reports the result as its marginal,

$$\varphi^{\downarrow a} = \varphi_a \otimes \left(\bigotimes_{c\in N(a)} \mu_{c\to a}\right). \tag{32}$$

In essence, Rules 1SS and 2SS repeat the computations done by Rules 1 and 2 for every node as the root node without repetitions of messages. A formal justification for why Rule 2SS computes the marginal for each node is again given by lemma 2 above.

Consider the timing of the messages as described in Rule 1SS. There is no root, and each node sends a message to a neighbor when the node has

received messages from its other neighbors. Since leaves have only one neighbor, they can send their message right away. So one can imagine the messages starting at the leaves and moving inside the tree and finally moving back to the leaves.

Another possible timing of the messages computed by Rule 1SS is to select (arbitrarily) a root node, direct all edges towards this node and perform first a corresponding inward propagation, computing the messages in the direction of the edges. This first phase, called the *inward* (or *collect*) phase, ends when the root has received a message from each of its outward neighbors. In the second phase, after the root has obtained all messages, it may start sending messages to its outward neighbors and once a node has received a message from its inward neighbor, it sends its messages to all its outward neighbors, computing the messages in the inverse direction of the edges. Once all leaves have received the messages from their (unique) inward neighbor, the procedure stops. This second phase is also called *outward* (or *distribute*) phase. In the subsequent sections, we will describe other computational architectures that need rooted join trees. The timing described in this paragraph will be useful in comparing the above architecture with these other architectures.

Using Rules 1SS and 2SS, we can compute the marginal of the joint valuation for each node in the join tree. If we need the marginal for a particular subset, we can ensure that this subset appears in the join tree by including it among the domains of valuations at the start of the join tree construction process (see also (Xu, 1995)). This setup for the computation of marginals of a factorization of a valuation is called the *Shenoy-Shafer* architecture. It is applicable for computation in all valuation algebras that satisfy the axioms.

Rules 1SS and 2SS suggest a computational architecture as shown in Figure 6. Each node in the join tree would have two storage registers, one for input valuation (the "in-box"), and one for reporting the marginal (the "out-box"). And each edge in the join tree would have two storage registers ("mailboxes") for the two messages, one in each direction.

The Shenoy-Shafer architecture will compute the correct marginals in any join tree. As we discussed before, some join trees are more efficient than others. There are quite a number of potential inefficiencies remaining in the join tree constructed using the algorithm described in the previous subsection. This will be illustrated by a simple example. Consider a valuation network consisting of four variables W, X, Y and Z, and four valuations α for $\{W, X\}$, β for $\{W, Y\}$, γ for $\{W, Z\}$ and δ for $\{X, Y, Z\}$. A join tree

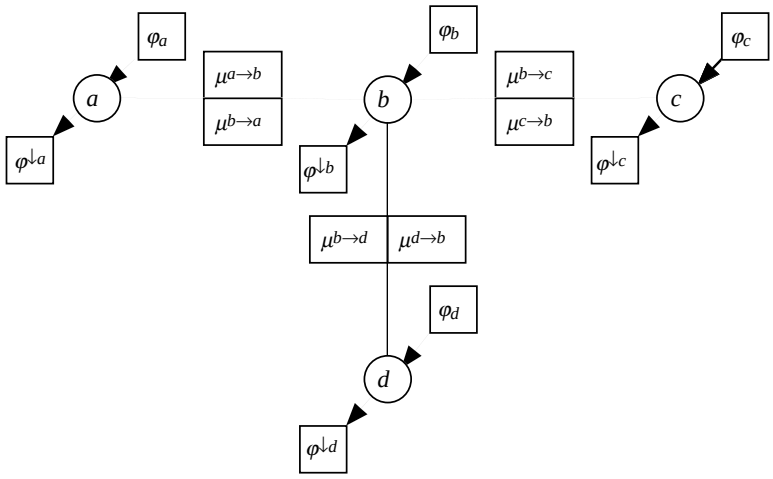


Figure 6: An architecture for storing messages, input valuations, and marginals.

with the messages between nodes is shown in Figure 7. We make some observations about inefficiencies of computation in this join tree.

(1) *Domain of Combination.* First , consider the message $(\alpha \otimes \beta \otimes \gamma)^{\downarrow\{X,Y,Z\}}$ (from $\{W, X, Y, Z\}$ to $\{X, Y, Z\}$). The computation of this message involves combination of the valuations α, β, γ on the domain $\{W, X, Y, Z\}$. However, it would be more efficient to combine first α and β which can be done on the smaller domain $\{W, X, Y\}$ and only then combine γ with $\alpha \otimes \beta$ which must be done on the larger domain $\{W, X, Y, Z\}$ (if, for example W, X, Y, Z have $2, 3, 4, 5$ configurations respectively, then $\{W, X, Y\}$ has 24 configurations whereas $\{W, X, Y, Z\}$ has 120 configurations). A similar observation can be made for the message $(\alpha \otimes \beta \otimes \delta)^{\downarrow\{W,Z\}}$.

(2) *Non-Local Combination.* Second, consider the message $(\beta \otimes \gamma \otimes \delta)^{\downarrow\{W,X\}}$. Notice that Z is in the domain of γ and δ, but not in the domain of β. Thus, it follows from the distributivity axiom that

$$(\beta \otimes \gamma \otimes \delta)^{\downarrow\{W,X\}} = (\beta \otimes (\gamma \otimes \delta)^{\downarrow\{W,X,Y\}})^{\downarrow\{W,X\}}.$$

It is more efficient to compute according to the right hand side of this identity. A similar remark holds also for the message $(\alpha \otimes \gamma \otimes \delta)^{\downarrow\{W,Y\}}$.

(3) *Duplication of Combinations.* Third, consider the messages $(\alpha \otimes \beta \otimes \gamma)^{\downarrow\{X,Y,Z\}}$ and $(\alpha \otimes \beta \otimes \delta)^{\downarrow\{W,Z\}}$. Notice that if these two messages are computed separately, then the combination of α and β is computed twice.

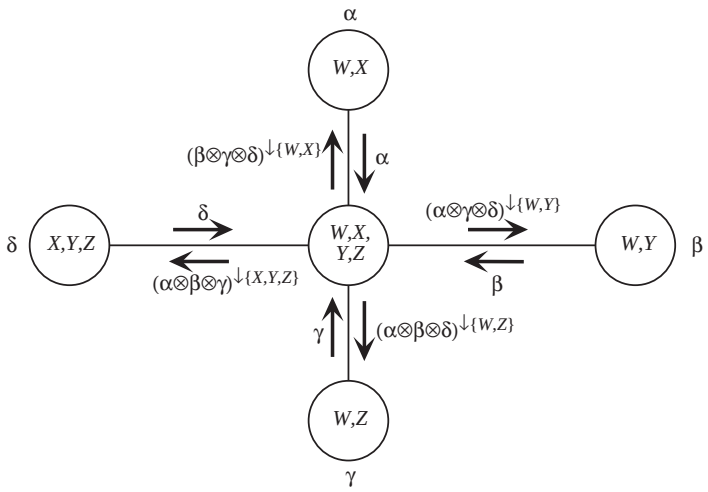


Figure 7: A join tree with messages for a valuation network.

Also for messages $(\beta \otimes \gamma \otimes \delta)^{\downarrow\{W,X\}}$ and $(\alpha \otimes \gamma \otimes \delta)^{\downarrow\{W,Y\}}$, the combination of γ and δ is repeated (see (Xu, 1991; Xu & Kennes, 1994)).

In order to avoid these inefficiencies, and in particular the last one, *binary join trees* were introduced by Shenoy (Shenoy, 1997). A covering join tree for a factorization $\varphi = \varphi_1 \otimes \varphi_2 \otimes \cdots \otimes \varphi_m$ is called *binary*, if no node has more than three neighbors, and if for every factor φ_i there is a node h_j such that $d(\varphi_i) = h_j$. The join tree in Figure 4 is not binary for the example, since the second condition is not satisfied,

Figure 8 shows a binary join tree for the last example. Compared to the join tree of Figure 7 it has an additional node $\{W, X, Y\}$ and an additional edge from this node to $\{W, X, Y, Z\}$.

First, notice that $\alpha \otimes \beta$ is computed on the domain $\{W, X, Y\}$ as the combination of messages arriving from the nodes $\{W, X\}$ and $\{W, Y\}$, before computing $(\alpha \otimes \beta \otimes \gamma)^{\downarrow\{X,Y,Z\}}$ and $(\alpha \otimes \beta \otimes \delta)^{\downarrow\{W,Z\}}$. Thus, we avoid combination of valuations on domains bigger than necessary.

Second, instead of computing $(\beta \otimes \gamma \otimes \delta)^{\downarrow\{W,X\}}$ we compute $(\beta \otimes (\gamma \otimes \delta)^{\downarrow\{W,X,Y\}})^{\downarrow\{W,X\}}$, and instead of computing $(\alpha \otimes \gamma \otimes \delta)^{\downarrow\{W,Y\}}$ we compute $(\alpha \otimes (\gamma \otimes \delta)^{\downarrow\{W,X,Y\}})^{\downarrow\{W,Y\}}$. Thus, the messages are computed locally.

Third, the combination $(\gamma \otimes \delta)^{\downarrow\{W,X,Y\}}$ that appears in the messages $(\beta \otimes (\gamma \otimes \delta)^{\downarrow\{W,X,Y\}})^{\downarrow\{W,X\}}$ and $(\alpha \otimes (\gamma \otimes \delta)^{\downarrow\{W,X,Y\}})^{\downarrow\{W,Y\}}$ is computed only once. Also, the combination $\alpha \otimes \beta$ is computed only once for the messages

$(\alpha \otimes \beta \otimes \gamma)^{\downarrow\{X,Y,Z\}}$ and $(\alpha \otimes \beta \otimes \delta)^{\downarrow\{W,Z\}}$. Thus, repetition of combinations is avoided.

The price one has to pay is more storage since a binary join tree has more nodes and edges. For the construction of binary join trees we refer to (Shenoy, 1997). Other computational improvements to the Shenoy-Shafer architecture are described in (Schmidt & Shenoy, 1998).

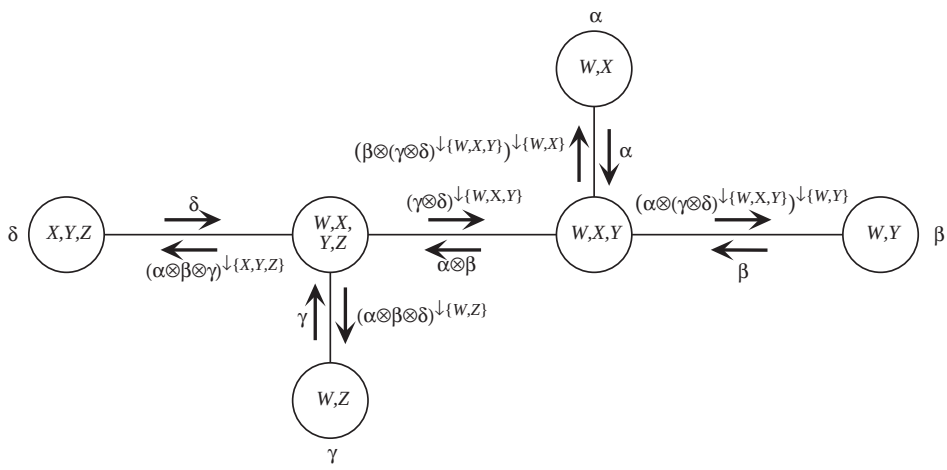


Figure 8: A binary join tree with messages for the same valuation network as Figure 7.

4.3 Idempotent Valuation Algebras

There are valuation algebras $(\Phi, D, \otimes, \downarrow)$ where an additional axiom holds:

7. *Idempotency.* For $\varphi \in \Phi$ and $x \in D$

$$\varphi \otimes \varphi^{\downarrow x} = \varphi. \tag{33}$$

Such valuations can be interpreted as pieces of information. The idempotency axioms says that the combination of an information with itself or with a part of itself gives no new information. Therefore, valuation algebras with this additional idempotency axiom are also called *information algebras*. We refer to (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b) for a discussion of this particular type of valuation algebra.

An example of an information algebra is the algebra of *subsets* of frames Ω_s. The valuations can be represented by indicator functions of subsets. Combination corresponds to set-intersection (or the multiplication of indicator functions). Marginalization is set projection. Clearly this algebra is idempotent. It is shown in (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b) that information algebras are closely related to Scott's information systems and to logic systems. For their connection to logic see (Kohlas *et al.*, 1998; Mengin & Wilson, 1999).

From a computational point of view, idempotency simplifies propagation. In fact, there is no more need for any storage at an edge. One needs storage only at the nodes. Consider the following computational architecture for information algebras. Consider a rooted join tree where all edges are directed toward the root. Let φ_a denote the valuation stored at node a at the start, and let φ denote the joint valuation.

Rule 1IA *Inward Phase.* Each non-root node sends a message to its inward neighbor *after* it has received a message from all its outward neighbors. Each time a node receives a message from an outward neighbor, it replaces the valuation it currently has with the combination of the valuation it currently has and the message. The messages are not stored. After a node has received (and absorbed) messages from all its outward neighbors, it sends a message to its inward neighbor consisting of the marginal of its current valuation to the domain of its inward neighbor (see Figure 9). The inward phase ends when the root has received and absorbed messages from all its outward neighbors.

Rule 2IA *Outward Phase.* Each non-leaf node sends a message to its outward neighbors *after* it has received a message from its inward neighbor. When a node receives a message from its inward neighbor, it replaces the valuation it currently has with the combination of this valuation and the message. The messages are not stored. After a node has received (and absorbed) the message from its inward neighbor, it sends a message to every outward neighbor consisting of the marginal of its current valuation to the domain of the respective outward neighbor (see Figure 9). The outward phase ends when the leaves have received and absorbed messages from their inward neighbors.

The inward phase is essentially the same as before in the general (non-

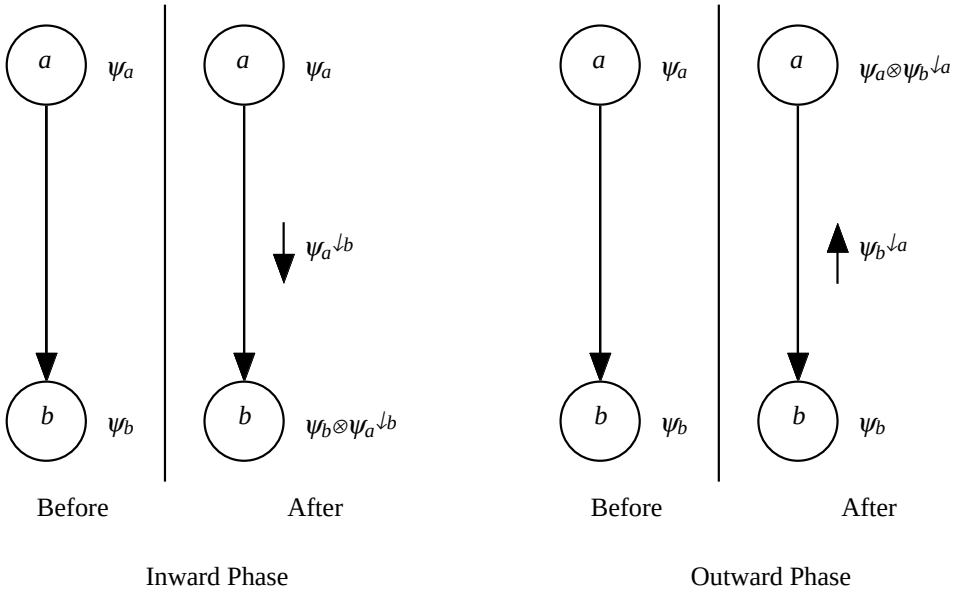


Figure 9: Details of inward and outward propagation in information algebras.

idempotent) case except that we don't save the messages and we do save a valuation at each node.

Notice that both in the inward as well as in the outward phase, all messages have the same form. If ψ_a is the valuation stored in node a when it transmits a message to a neighbor b, then the message is

$$\mu_{a\to b} = \psi_a^{\downarrow b}. \tag{34}$$

When node b receives a message from a neighbor a, then it changes its stored valuation ψ_b to

$$\psi_b \otimes \mu_{a\to b}. \tag{35}$$

If $\varphi_a^{(i)}$ denotes the valuation stored at node a at the end of the inward phase, then for the root node, say c, $\varphi_c^{(i)} = \varphi^{\downarrow c}$. The outward phase is different from the general case. At the end of the outward phase, the valuation stored at each node a is $\varphi^{\downarrow a}$. The reason for this is the following lemma.

Lemma 3 *(Kohlas, 1997) Assume that $(\Phi, D, \otimes, \downarrow)$ is a valuation algebra satisfying the idempotency axiom (an information algebra). Let $\varphi_a^{(i)}$ denote*

the valuation stored at the end of the inward phase at node a of a join tree, and let b denote the inward neighbor of a. Then

$$\varphi^{\downarrow a} = \varphi^{\downarrow a \cap b} \otimes \varphi_a^{(i)}. \tag{36}$$

Thus, at the end of the outward phase, the valuation stored at each node a is $\varphi^{\downarrow a}$.

We will call this architecture information algebra (IA) architecture. In IA architecture there is no need for mailboxes at edges since the messages are not stored. In contrast, however, the valuations $\varphi_a^{(i)}$ computed in the inward propagation must be stored at each node for use in the outward propagation.

Intuitively the reason why the IA architecture works is as follows. The message that a sends to b during the inward phase is returned back to it by b during the outward phase (together with further information). But since combination satisfies the idempotency axiom, double counting of these messages does not matter. In the general (non-idempotent) case, we make sure there is no double counting of messages by removing the inward message from the message sent back from node b to the neighboring node a.

5 Other Computational Architectures

5.1 Continuation

Additional properties of valuation algebras may lead to alternative and possibly more efficient architectures for the computation of marginals. This has been illustrated above for idempotent valuation algebras. In this section, we will consider another assumption that is weaker than the idempotency axiom and that is satisfied by many examples of valuation algebras.

Let $(\Phi, D, \otimes, \downarrow)$ be a valuation algebra. Consider $\varphi \in \Phi$ and $x \subseteq y$. If there exists a $\psi \in \Phi$ such that

$$\varphi^{\downarrow x} \otimes \psi = \varphi^{\downarrow y} \tag{37}$$

we say that ψ *continues* φ from x to y, or we call ψ a *continuer* of φ from x to y. Continuation has been introduced by Shafer in an unpublished paper (Shafer, 1991) in order to generalize some computational architectures proposed for Bayesian networks. Another approach to generalize these architectures has been presented in (Lauritzen & Jensen, 1997).

If φ is a probability potential that is also a discrete probability distribution, then $\varphi^{\downarrow x}, \varphi^{\downarrow y}$ are marginal distributions of φ. Now, if the potential $\varphi^{\downarrow x}$ has no zero entries (the distribution has no zero probabilities), then we may take for $\mathbf{x} \in \Omega_y$

$$\psi(\mathbf{x}) = \frac{\varphi^{\downarrow y}(\mathbf{x})}{\varphi^{\downarrow x}(\mathbf{x}^{\downarrow x})}.$$

If $\varphi^{\downarrow x}$ has some zero entries $\varphi^{\downarrow x}(\mathbf{x}^{\downarrow x})$, then $\varphi^{\downarrow y}(\mathbf{x})$ is also a zero entry and $\psi(\mathbf{x})$ can be selected arbitrarily.

This example shows two things: First, a continuer is something like a quotient representing a *conditional* valuation (in the case of probability distributions, the continuer defined above is a conditional probability distribution derived from the probability distribution $\varphi^{\downarrow y}$). Second, continuers are not unique in general.

In the theory of Dempster-Shafer belief functions, continuers may similarly be obtained by the division of commonality functions. However, it is not guaranteed that the resulting function is a commonality function. That is, the $m(A)$ computed by (2) are not necessarily all nonnegative. In this case the semigroup of the commonality functions must be embedded into the larger semigroup of mappings $c : \mathcal{P}(\Omega_s) \to R^+$.

In the case of idempotent valuation algebras (information algebras), due to idempotency, $\varphi^{\downarrow y}$ is itself a continuer of φ from x to y. Continuation is trivial in this case.

The following lemma enumerates some elementary properties of continuation.

Lemma 4 *(Shafer, 1991)*

1. *If ψ continues φ from x to y, then $d(\psi) \cup x = y$.*
2. *If $w \subseteq x \subseteq y \subseteq z = d(\varphi)$, and ψ continues φ from w to y, then $\psi^{\downarrow d(\psi) \cap x}$ continues φ from w to x.*
3. *If $w \subseteq x \subseteq y \subseteq z = d(\varphi)$, and ψ_1 continues φ from w to x, ψ_2 continues φ from x to y, then $\psi_1 \otimes \psi_2$ continues φ from w to y.*
4. *If $d(\varphi_1) = x, d(\varphi_2) = y$, and ψ continues φ_2 from $x \cap y$ to y, then ψ continues $\varphi_1 \otimes \varphi_2$ from x to $x \cup y$ and from $x \cap y$ to y.*

5.2 Lauritzen-Spiegelhalter Architecture

In this subsection, we assume a valuation algebra $(\Phi, D, \otimes, \downarrow)$ with the additional property that any valuation $\varphi \in \Phi$ has at least one continuer from x to y if $x \subseteq y$. We shall now show how the existence of these continuers helps to rearrange the outward phase.

Assume we have a join tree with valuations φ_a associated with a such that $d(\varphi_a) \subseteq a$ for each node a in the join tree. Let φ denote the joint valuation. As before, let $\varphi_a^{(i)}$ denote the valuation stored at node a at the end of the inward phase.

Rule 1LS *Inward Phase* Each non-root node sends a message to its inward neighbor *after* it has received a message from every outward neighbor. The messages are not stored. Suppose b is the inward neighbor of a. Consider the situation just before a sends a message to b (see Fig. 10). Let ψ_a denote the current valuation associated with node a and let ψ_b denote the current valuation associated with node b. The message that a sends to b is $\psi_a^{\downarrow b}$. b replaces ψ_b with $\psi_b \otimes \psi_a^{\downarrow b}$, and a replaces ψ_a with a continuer ψ of ψ_a from $a \cap b$ to a, i.e.

$$\psi_a^{\downarrow b} \otimes \psi = \psi_a. \tag{38}$$

This phase ends when the root has received a message from all its outward neighbors.

Rule 2LS *Outward Phase.* Each non-leaf node sends a message to its outward neighbors *after* it has received a message from its inward neighbor. The messages are not stored. Suppose a is the outward neighbor of b. Consider the situation just before b sends a message to a (see Fig. 10). Let ψ_b denote the current valuation associated with node b and let ψ_a denote the current valuation associated with node a. The message that b sends to a is $\psi_b^{\downarrow a}$. a replaces ψ_a with $\psi_a \otimes \psi_b^{\downarrow a}$, and b leaves its valuation unchanged. This phase ends when all leaves have received and absorbed messages from their inward neighbors.

Notice that at the end of the inward phase, the valuation stored at the root, say c, is $\varphi^{\downarrow c}$. At the end of the outward phase, the valuation stored at each node a is $\varphi^{\downarrow a}$. The reason for this is the result in the following lemma.

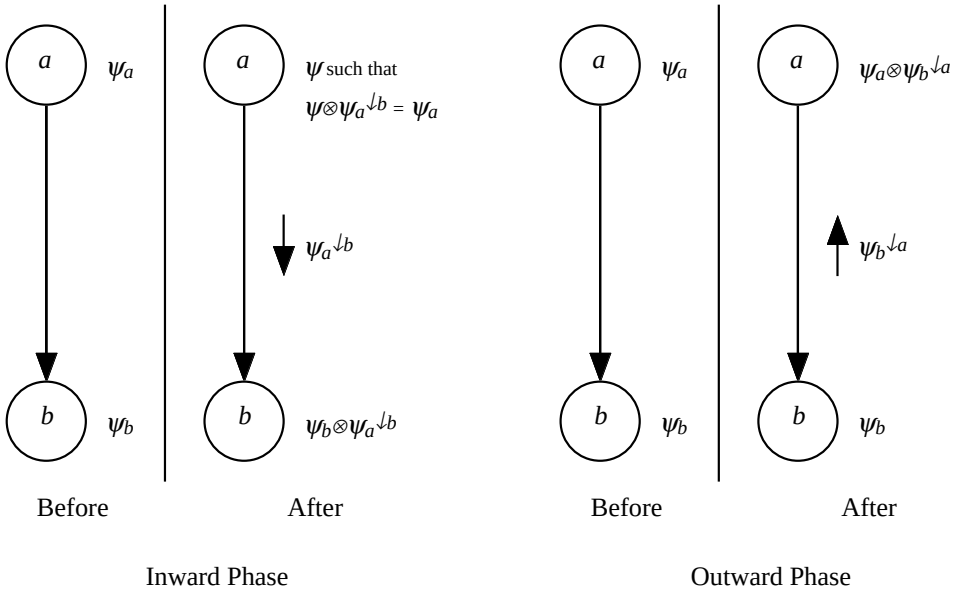


Figure 10: Message passing in the inward and outward phases of the Lauritzen-Spiegelhalter architecture.

Lemma 5 *(Shafer, 1991). Suppose b is the inward neighbor of a, and suppose φ denotes the joint valuation. ψ as defined in (38) continues φ from $a \cap b$ to a.*

Now, at the end of the inward phase, we have $\varphi^{\downarrow c}$ at the root node c. Consider an outward neighbor b of c. As per Lemma 5, the valuation ψ stored at this node is a continuer for φ from $c \cap b$ to b. During the outward propagation, the message that c sends to b is $(\varphi^{\downarrow c})^{\downarrow b} = \varphi^{\downarrow c \cap b}$. b combines this message with its own valuation ψ and replaces it with $\psi \otimes \varphi^{\downarrow c \cap b}$. But since ψ is a continuer for φ from $c \cap b$ to b it follows that

$$\varphi^{\downarrow c \cap b} \otimes \psi = \varphi^{\downarrow b}. \tag{39}$$

This is repeated: Once a node b receives the message from its (unique) inward neighbor, it combines it with the stored continuer to obtain its marginal $\varphi^{\downarrow b}$. Then, for every outward neighbor a, it marginalizes this marginal to $b \cap a$ and sends this marginal to a. Once all the leaves of the directed join tree have processed their incoming messages, the procedure stops. Each node a of the join tree has its marginal $\varphi^{\downarrow a}$ stored in it. In Fig. 10 this message flow of the outward propagation is schematically represented.

This setup is called the *Lauritzen-Spiegelhalter architecture* because it corresponds to the proposal originally made in (Lauritzen & Spiegelhalter, 1988) for computation with probabilities. It works only for valuation algebras with continuers. In contrast to the Shenoy-Shafer architecture, no mailboxes are used. Instead of the messages passed between nodes, a continuer is computed and stored during the inward propagation when a node sends its message to its unique inward neighbor. The computation of the continuer involves division (in the case of probability potentials or commonality functions) and is thus relatively costly.

In the case of idempotent valuation algebras, $\varphi_a^{(i)}$ itself can be taken as the continuer ψ. Then the Lauritzen-Spiegelhalter architecture corresponds exactly to the setup described in section 4.3.

Comparing to the Shenoy-Shafer method, since a node a expects to get the message it sends to neighbor b back during the outward phase, it prepares for it during the inward phase by computing a continuer (essentially by "removing" the message from its valuation) so that there is no double counting of this message.

Finally, a comment about the join tree. The Lauritzen-Spiegelhalter (LS) architecture works for any covering join tree, but not all join trees are equally efficient. In particular, the join trees that are efficient for the Shenoy-Shafer (SS) architecture are not necessarily efficient for the LS architecture and vice-versa. In particular, since in the LS architecture, we have no storage in the edges, but storage and computation at each node, a join tree with many nodes will in general be computationally inefficient. If we construct a join tree using the technique described in Subsection 4.1, then we can "condense" the join tree as follows. If a and b are neighbors in the join tree such that $b \subseteq a$, then we delete b, delete all edges that include b, and add edges from each node in $N(b) - \{a\}$ to a. We recursively repeat this until there are no more pairs of neighboring nodes a, b such that $b \subseteq a$. It is easy to see that the resulting graph is still a join tree. Such join trees are called "clique trees". For the medical example, a clique tree resulting from the join tree in Figure 4 is shown in Figure 11.

5.3 HUGIN Architecture

The Lauritzen-Spiegelhalter architecture uses continuers, hence division (in the case of probabilistic valuations) on the nodes of the join tree. However, it was noted in (Jensen *et al.*, 1990b) that division can be restricted to

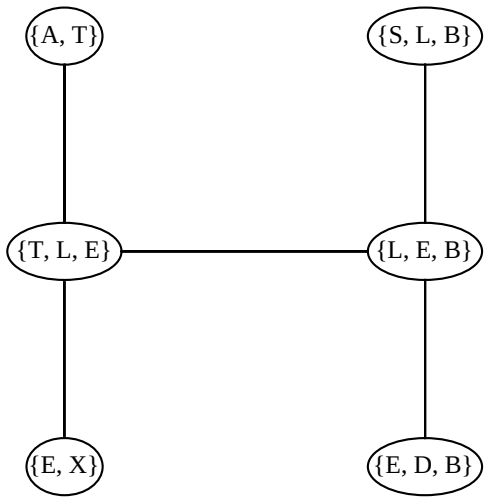


Figure 11: A clique tree for the medical example.

smaller domains. This idea was implemented in the HUGIN software for probabilistic potentials.

The HUGIN architecture can be described as follows. We start with the same situation as in the Lauritzen-Spiegelhalter architecture. Namely we assume we have a rooted join tree with valuations φ_a associated with node a such that $d(\varphi_a) \subseteq a$. Let φ denote the joint valuation.

In contrast to the LS architecture, in the HUGIN architecture, we introduce one new mailbox between every two neighboring nodes. These new mailboxes are called *separators* and they are used to store messages between the two nodes. At the beginning, the separators are empty, i.e., there are no valuations stored in them. A formal description of the algorithm is as follows (see Figure 12).

Rule 1H *Inward Phase.* Each non-root node sends a message to its inward neighbor *after* it has received a message from all its outward neighbors. Suppose b is the inward neighbor of a. Consider the situation just before a sends a message to b. Let ψ_a denote the current valuation associated with node a and let ψ_b denote the current valuation associated with node b. The message that a sends to b is $\psi_a^{\downarrow b}$ and this is stored in the separator. b replaces ψ_b with $\psi_b \otimes \psi_a^{\downarrow b}$, and a leaves its valuation ψ_a unchanged. This phase ends when the root has received a message from all its outward neighbors.

Rule 2H *Outward Phase.* Each non-leaf node sends a message to its outward neighbors via the separator *after* it has received a message from its inward neighbor. Suppose a is the outward neighbor of b. Consider the situation just before b sends a message to a. Let ψ_b denote the current valuation associated with node b, let ψ_a denote the current valuation associated with node a, and let ψ denote the valuation in the separator between a and b. The message that b sends to the separator is $\psi_b^{\downarrow a}$. The separator first computes a valuation χ such that $d(\chi) = b \cap a$ and

$$\chi \otimes \psi = \psi_b^{\downarrow a} \tag{40}$$

and sends this valuation χ to a, and then the separator replaces its valuation ψ with χ. a replaces its valuation ψ_a with $\psi_a \otimes \chi$. This phase ends when the leaves have received and absorbed messages from their inward neighbors.

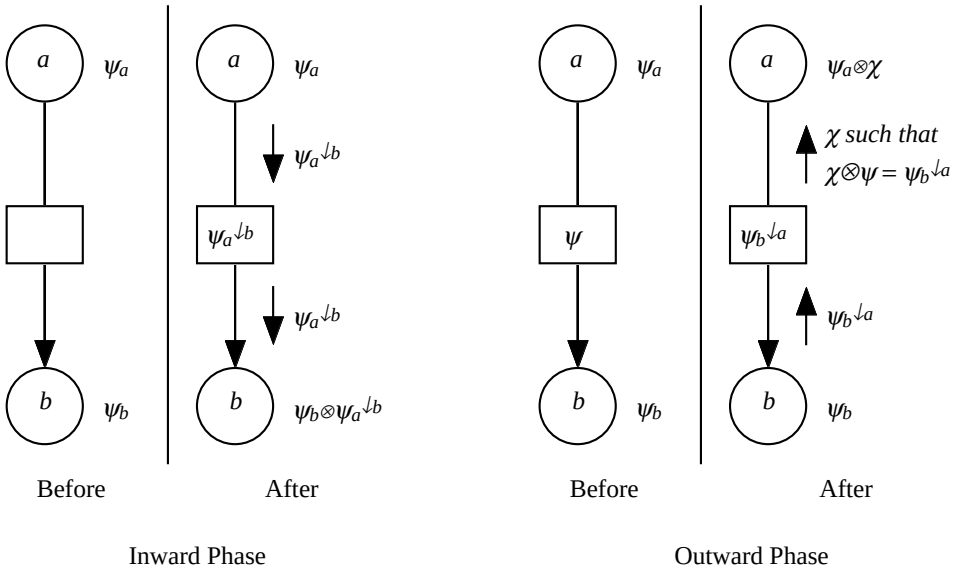


Figure 12: Message passing in the inward and outward propagation for the HUGIN architecture

The outward phase makes use of the following result.

Lemma 6 *(Shafer, 1991) Suppose a and b are neighboring nodes in a rooted join tree such that b is the inward neighbor of a, suppose $\varphi_a^{(i)}$ denotes the*

valuation stored at a at the end of the inward phase, suppose φ denotes the joint valuation, and suppose there exists a continuer of $\varphi_a^{(i)}$ from $a \cap b$ to b. Then there exists a valuation χ with domain $a \cap b$ such that

$$\chi \otimes (\varphi_a^{(i)})^{\downarrow b} = \varphi^{\downarrow b \cap a}. \tag{41}$$

Moreover, if χ is a valuation satisfying Equation 41, then

$$\chi \otimes \varphi_a^{(i)} = \varphi^{\downarrow a}. \tag{42}$$

Note that the prerequisite of this lemma is just what is needed to implement the Lauritzen-Spiegelhalter architecture, namely the existence of the continuer for $\varphi_a^{(i)}$ from $a \cap b$ to a. Thus, the HUGIN architecture is applicable under the same conditions as the LS architecture.

The inward phase is similar to the inward phase of the IA or LS architecture, and therefore at the end of the inward phase, the valuation stored at the root node c is $\varphi^{\downarrow c}$. Based on Lemma 6, one can show that at the end of the outward propagation, the valuation stored at each node a is $\varphi^{\downarrow a}$ and the valuation stored at the separator between a and b is $\varphi^{\downarrow a \cap b}$. To see this, suppose b is an outward neighbor of c, the root node. Since the separator stores the message it receives from c, it is clear that the valuation at the separator is $(\varphi^{\downarrow c})^{\downarrow b} = \varphi^{\downarrow c \cap b}$. Equation 41 guarantees the existence of the valuation χ sent as a message from the separator to b, and it follows from Equation 42, that the valuation stored at the end of the outward phase at b is $\varphi^{\downarrow b}$. By repeating this argument recursively, it is clear that we have the marginal of the joint valuation at every node and every separator.

The main advantage of the HUGIN architecture over the LS architecture is that division (computation of χ) is done on the domains of the separators, which have a smaller domain than the nodes on which division is done in the LS architecture. So at least in the case of probability potentials, HUGIN architecture is clearly more time-efficient than the LS architecture. The comparison with the Shenoy-Shafer architecture based on binary join trees is less obvious. An empirical study (Lepar & Shenoy, 1998; Stärk-Lepar, 1999) for the case of probability potentials shows that the Shenoy-Shafer architecture is slightly more time-efficient than the HUGIN architecture. This time efficiency is achieved at the cost of storage. The Shenoy-Shafer architecture has more storage than HUGIN, which in turn has more storage than the LS architecture.

No comparison of various architectures has been made for other examples of valuation algebras. Since continuers exist in Dempster-Shafer belief function

algebra (Shenoy, 1994a; Lauritzen & Jensen, 1997), the HUGIN architecture applies to this algebra. However, architectures using division do not look very promising in this case. The reason is that division must be done with commonality functions. But representation of belief functions by commonality functions, although very convenient for theoretical purposes, is most inefficient both for storage as well as for the computation of marginals (see however (Bissig *et al.*, 1997)). Using the m-function is more promising and this is no problem in the Shenoy-Shafer architecture, since no division is needed. We refer to the chapter on computation with Dempster-Shafer Belief Functions in this book and also to the chapter on Probabilistic Argumentation Systems, where alternative approaches to treat this case are discussed.

For idempotent valuation algebras, the HUGIN architecture gives no advantage over the Lauritzen-Spiegelhalter architecture since in this case division is trivial (no computation required). The architecture described in section 4.3 seems appropriate for this case.

6 Other Examples of Abstract Computation

We review here shortly several known examples of valuation algebras. Essentially we reconsider examples given already in the text, complement them with variants, and add remarks about a few other systems.

The first class of examples centers around valuation algebras motivated by discrete probability theory. As introduced in section 2.1 potentials are here functions $p : \Omega_s \to R^+$. Marginalization is defined as summing out variables to be eliminated (see Equation 6). Combination is defined as pointwise multiplication of probability marginals (see Equation 5). There is however a variant of combination which involves normalization. If φ, ψ are two probability potentials with domains s and t respectively, then for $\mathbf{x} \in \Omega_{s \cup t}$, define

$$(\varphi \otimes \psi)(\mathbf{x}) = \frac{\varphi(\mathbf{x}^{\downarrow s})\psi(\mathbf{x}^{\downarrow t})}{K}. \tag{43}$$

where

$$K = \sum_{\mathbf{x} \in \Omega_{s \cup t}} \varphi(\mathbf{x}^{\downarrow s)})\psi(\mathbf{x}^{\downarrow t}). \tag{44}$$

If $K = 0$, then $(\varphi \otimes \psi)(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega_{s \cup t}$. This is combination with *normalization*. It means that the combination of two potentials is always normalized if $K \neq 0$, that is, a probability distribution.

This new definition leads to a different valuation algebra than the original one. Often, in the original formulation of the marginalization problem, one wants the final result normalized (to be a probability distribution). This is the case in applications to Bayesian networks or more generally to probabilistic expert systems. In the second, new version of probabilistic valuation algebras, the result is automatically normalized. It can however be shown, that computation in the first version of the probabilistic valuation algebra, and normalization only at the end, leads to the same result. Computationally, the version with normalization only at the end rather than at each combination, is more efficient.

A similar situation exists with respect to Dempster-Shafer belief functions. Valuations in this case may be represented by m-functions and combination can be defined by Equation 8 using these m-functions. A normalized variant of this combination is defined as follows: If m_φ and m_ψ denote the m-functions of two potentials φ and ψ for s and t respectively, then for nonempty subsets A of $\Omega_{s \cup t}$ define

$$m_{\varphi \otimes \psi}(A) = \frac{1}{K} \sum_{A_1^{\uparrow s \cup t} \cap A_2^{\uparrow s \cup t} = A} m_\varphi(A_1) m_\psi(A_2). \tag{45}$$

where

$$K = \sum_{A_1^{\uparrow s \cup t} \cap A_2^{\uparrow s \cup t} \neq \emptyset} m_\varphi(A_1) m_\psi(A_2). \tag{46}$$

Furthermore, define $m_{\varphi \otimes \psi}(\emptyset) = 0$. If $K = 0$, then $m_{\varphi \otimes \psi}(A) = 0$ for all subsets A of $\Omega_{s \cup t}$. In this case, the two potentials are said to be contradictory. This version of combination corresponds to the original *Dempster's rule*. It has a semantic appeal, because it removes contradictory elements (focal sets) in the two belief functions induced by two different sources of information (Dempster, 1967; Kohlas & Monney, 1995). The combination of two belief functions which are not contradictory is always a *normalized* belief function (the $m(A)$ sum up to one and $m(\emptyset) = 0$). With the same marginalization as before and this new rule of combination, we obtain a new valuation algebra.

As with probabilistic valuation algebras, we are often for semantic reasons interested only in normalized belief functions combined with Dempster's

rule. Thus, in principle, we should compute in this second version of a belief function valuation algebra. But it turns out once more that using the first rule of combination, and normalizing only at the end, gives the same result. And again this version is computationally more efficient.

Similar considerations of normalization are also possible in other examples. One has always to be very careful to verify whether normalization at the end gives the same result as combination including normalization. For example in possibility theory, there are several combination rules proposed. One commonly used rule is minimization. For this combination rule, if we don't normalize, then combination satisfies the axioms of valuation algebras. However, if we include normalization, say by division, then such a combination rule is not associative and therefore, does not satisfy the axioms of valuation algebras.

Systems of subsets were identified as an example of idempotent valuation algebras or information algebras. There are a number of interesting subalgebras of these information algebras. These include *linear manifolds*, *convex polyhedra* and *convex subsets*. Linear manifolds are related to *systems of linear equations*. Local computation on join trees corresponds in this case to compute with decomposed matrices (Rose, 1970). Convex polyhedra can be represented by systems of *linear inequalities*. Variable elimination can be performed by Fourier-Motzkin elimination. Again, local computation on join trees corresponds to matrix decomposition. The relation to linear programming remains to be explored.

If the frames are finite, subsets correspond to *relations*. It is therefore not surprising that *relational algebra* as used in relational databases corresponds to information algebras. The *join* is the combination operation and *projection* is marginalization. The importance of join trees in relational database systems has been noted early in (Maier, 1983).

Subsets of a Boolean cube $\{0,1\}^s$ can be considered as possible worlds for formulas of *propositional logic*. This indicates a relationship between information algebras and propositional logic. Variable elimination is based on resolution and the fusion algorithm corresponds to David-Putnam procedure (Kohlas *et al.*, 1998). In this context, assumption-based reasoning (or ATMS, Assumption-Based Truth Maintenance (De Kleer, 1986a; De Kleer, 1986b)) enters into relation to idempotent valuation algebras, and inference in this framework can be considered as computing in valuation algebras.

More generally, cylindric algebras, the algebraic structure behind predicate logic, are also closely related to special (Boolean) information algebras (see

(Henkin *et al.*, 1971)). Similar algebras have also been considered in the study of modularity (Bergstra *et al.*, 1990; Renardel de Lavalette, 1992) in computer science.

Still more general are information systems in the sense of Scott. It has been shown that information systems induce information algebras under some conditions (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b). This brings logic in general into the focus of valuation algebras. It is interesting to note that local computation on join trees has been considered only very recently in this context (Mengin & Wilson, 1999; Kohlas *et al.*, 1998; Kohlas & Moral, 1996). In this field, the relation of local computation in join trees to other procedures of logical deduction and consequence finding remains to be explored.

A last important example of valuation algebras arises in *discrete optimization*. If an objective function over a set of variables factorizes into additive or multiplicative terms whose domains contain only a few variables, the dynamic programming techniques may be applied. They correspond to local computation in join trees. The factors of the objective function are valuations of a valuation algebra. Combination is either multiplication (if the terms multiply) or addition (if the terms add). Marginalization is either maximization or minimization over the variables to be eliminated, depending on the desired sense of optimization. This is the case of non-serial dynamic programming (Bertele & Brioschi, 1972). For further details see (Shenoy, 1991b; Shenoy, 1996).

References

Arnborg, S., Corneil, D., & Proskurowski, A. 1987. Complexity of Finding Embedings in a k-Tree. *SIAM J. of Algebraic and Discrete Methods*, **38**, 277–284.

Bergstra, J.A., Heering, J., & Klint, P. 1990. Module Algebra. *J. of the Association for Computing Machinery*, **37**(2), 335–372.

Bertele, U., & Brioschi, F. 1972. *Nonserial Dynamic Programming*. Academic Press.

Bissig, R., Kohlas, J., & Lehmann, N. 1997. Fast-Division Architecture for Dempster-Shafer Belief Functions. *In:* Gabbay, D., Kruse, R., Nonnengart, A., & Ohlbach, H.J. (eds), *First Int. Joint Conf. on Qualitative*

and Quantitative Practical Reasoning, ECSQARU–FAPR'97, Springer, for Lecture Notes in Artif. Intell.

Cannings, C., Thompson, E.A., & Skolnick, M.H. 1978. Probability Functions on Complex Pedigrees. *Advances in Applied Probability*, **10**, 26–61.

De Kleer, J. 1986a. An Assumption-based TMS. *Artif. Intell.*, **28**, 127–162.

De Kleer, J. 1986b. Extending the ATMS. *Artif. Intell.*, **28**, 163–196.

Dempster, A. 1967. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Ann. Math. Stat.*, **38**, 325–339.

Henkin, L., Monk, J. D., & Tarski, A. 1971. *Cylindric Algebras.* Studies in logic and the foundations of mathematics, vol. 65,115. North-Holland.

Jensen, F.V., Lauritzen, S.L., & Olesen, K.G. 1990a. Bayesian Updating in Causal Probabilistic Networks by Local Computation. *Computational Statistics Quarterly*, **4**, 269–282.

Jensen, F.V., Lauritzen, S.L., & Olesen, K.G. 1990b. Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Comp. Stat. Q.*, **4**, 269–282.

Kjærulff, U. 1990. *Triangulation of Graphs-Algorithms giving Small Total Space.* Tech. rept. R90–09. Department of Mathematics and Computer Science, Aalborg University, Denmark.

Kohlas, J. 1997. *Computational Theory for Information Systems.* Tech. rept. 97–07. University of Fribourg, Institute of Informatics.

Kohlas, J., & Monney, P.A. 1995. *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence.* Lecture Notes in Economics and Mathematical Systems, vol. 425. Springer.

Kohlas, J., & Moral, S. 1996. *Propositional Information Systems.* Tech. rept. 96–01. University of Fribourg, Institute of Informatics.

Kohlas, J., & Stärk, R.F. 1996a. *Eine mathematische Theorie der Informationssysteme.* Tech. rept. 96–12. University of Fribourg, Institute of Informatics.

Kohlas, J., & Stärk, R.F. 1996b. *Information Algebras and Information Systems.* Tech. rept. 96–14. University of Fribourg, Institute of Informatics.

Kohlas, J., Moral, S., & Haenni, R. 1998. *Propositional Information Systems.* Accepted for Publication in the J. of Logic and Computation.

Kong, A. 1986. *Multivariate Belief Functions and Graphical Models.* Ph.D. thesis, Department of Statistics, Harvard University.

Lauritzen, S.L., & Jensen, F.V. 1997. Local Computation with Valuations from a Commutative Semigroup. *Annals of Mathematics and Artificial Intelligence*, **21**(1), 51–70.

Lauritzen, S.L., & Spiegelhalter, D.J. 1988. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *J. of Royal Stat. Soc.*, **50**(2), 157–224.

Lepar, V., & Shenoy, P.P. 1998. A Comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. *Pages 328–337 of:* Cooper, G., & Moral, S. (eds), *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference.* Morgan Kaufmann, San Francisco.

Maier, D. 1983. *The Theory of Relational Databases.* Computer Science Press.

Mellouli, K. 1988. *On the Propagation of Beliefs in Networks Using the Dempster-Shafer Theory of Evidence.* Ph.D. thesis, School of Business, University of Kansas.

Mengin, J., & Wilson, N. 1999. Logical Deduction using the Local Computation Framework. *Pages 386–396 of:* Hunter, A., & Parsons, S. (eds), *European Conf. ECSQARU'99, London*, Springer, for Lecture Notes in Artif. Intell.

Olmsted, S. 1983. *On representing and solving decision problems.* Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University.

Renardel de Lavalette, G.R. 1992. Logical semantics of modularisation. *Pages 306–315 of:* Jäger, G., Kleine Büning, H., & Richter, M.M. (eds), *Computer Science Logic, selected apers from CSL'91.* Springer, Lecture Notes in Computer Science 626.

Rose, D.J. 1970. Triangulated Graphs and the Elimination Process. *J. of Math. Analysis and Applications*, **32**, 597–609.

Schmidt, T., & Shenoy, P.P. 1998. Some Improvements to the Shenoy-Shafer and Hugin Architectures for Computing Marginals. *Artif. Intell.*, **102**, 323–333.

Shafer, G. 1976. *The Mathematical Theory of Evidence.* Princeton University Press.

Shafer, G. 1991. *An Axiomatic Study of Computation in Hypertrees.* Working Paper 232. School of Business, University of Kansas.

Shafer, G., Shenoy, P.P., & Mellouli, K. 1987. Propagating Belief FUnctions in Qualitative Markov Trees. *Int. J. of Approximate Reasoning*, **1**(4), 349–400.

Shenoy, P.P. 1989. A Valuation-Based Language For Expert Systems. *Int. J. of Approximate Reasoning*, **3**, 383–411.

Shenoy, P.P. 1991a. On Spohn's Rule for Revision of Beliefs. *Int. J. of Approximate Reasoning*, **5**(2), 149–181.

Shenoy, P.P. 1991b. Valuation-Based Systems for Discrete optimization. *Pages 385–400 of:* Bonissone, P.P., Henrion, M., Kanal, L.N., & Lemmer, J.F. (eds), *Uncertainty in Artificial Intelligence, 6.* North-Holland, Amsterdam.

Shenoy, P.P. 1992a. Using Possibility Theory in Expert Systems. *Fuzzy Sets and Systems*, **51**(2), 129–142.

Shenoy, P.P. 1992b. Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems. *Pages 83–104 of:* Zadeh, L.A., & Kacprzyk, J. (eds), *Fuzzy Logic for the Management of Uncertainty.* John Wiley & Sons.

Shenoy, P.P. 1994a. Conditional Independence in Valuation-based Systems. *Int. J. of Approximate Reasoning*, **10**, 203–234.

Shenoy, P.P. 1994b. Consistency in Valuation-Based Systems. *ORSA Journal on Computing*, **6**(3), 281–291.

Shenoy, P.P. 1996. Axioms for Dynamic Programming. *Pages 259–275 of:* Gammerman, A. (ed), *Computational Learning and Probabilistic Reasoning.* Wiley, Chichester, UK.

Shenoy, P.P. 1997. Binary Join Trees for Computing Marginals in the Shenoy-Shafer Architecture. *Int. J. of Approximate Reasoning*, **17**(2–3), 239–263.

Shenoy, P.P., & Shafer, G. 1990. Axioms for Probability and Belief Function Propagation. *Pages 169–198 of:* R.D. Shachter, T.S. Levitt, J.F. Lemmer, & Kanal, L.N. (eds), *Uncertainty in Artif. Intell. 4*. North Holland.

Spohn, W. 1988. Ordinal conditional functions: A dynamic theory of epistemic states. *Pages 105–134 of:* Harper, W.L., & Skyrms, B. (eds), *Causation in Decision, Belief Change, and Statistics*, vol. 2. Dordrecht, Netherlands.

Stärk-Lepar, V. 1999. *Performance of Architectures for Local Computations in Bayesian Networks*. Ph.D. thesis, University of Fribourg, Institute of Informatics.

Xu, H. 1991. *An Efficient Implementation of Belief Function Propagation*. Tech. rept. 91–4. IRIDIA, Université Libre de Bruxelles.

Xu, H. 1995. Computing Marginals for Arbitrary Subsets from Marginal Representation in Markov Trees. *Artif. Intell.*, **74**, 177–189.

Xu, H., & Kennes, R. 1994. Steps Toward Efficient Implementation of Dempster-Shafer Theory. *Pages 153–174 of:* Yager, R.R., Fedrizzi, M., & Kacprzyk, J. (eds), *Advances in the Dempster-Shafer Theory of Evidence*. John Wiley & Sons.

Zadeh, L.A. 1978. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, **1**, 3–28.

Zadeh, L.A. 1979. A Theory of Approximate Reasoning. *Pages 149–194 of:* Ayes, J.E., Michie, D., & Mikulich, L.I. (eds), *Machine Intelligence*, vol. 9. Ellis Horwood, Chichester, UK.

Zhang, L. 1988. *Studies on Finding Hypertree Covers of Hupergraphs*. Working Paper 198. School of Business, University of Kansas, Lawrence, KS.

Index