Generic Local Computation

Marc Pouly, SnT, University of Luxembourg

Cesar Schneuwly, University of Fribourg (Switzerland)

Jürg Kohlas, University of Fribourg (Switzerland)

29 March 2011

978-2-87971-034-1

Generic Local Computation

Marc Pouly
marc.pouly@uni.lu
University of Luxembourg

Cesar Schneuwly
cesar.schneuwly@unifr.ch
University of Fribourg (Switzerland)

Juerg Kohlas
juerg.kohlas@unifr.ch
University of Fribourg (Switzerland)

January 19, 2011

Abstract

Many problems of artificial intelligence, or more generally, many problems of information processing, have a generic solution based on local computation on join trees or acyclic hypertrees. There are several variants of this method all based on the algebraic structure of a valuation algebra. A strong requirement underlying this approach is that the elements of a problem decomposition form a join tree. Although it is always possible to construct covering join trees, if the requirement is originally not satisfied, it is not always possible or not efficient to extend the elements of the decomposition to the covering join tree. Therefore in this paper different variants of an axiomatic framework of valuation algebras are introduced which prove sufficient for local computation without the need of an extension of the factors of a decomposition. This framework covers the axiomatic system proposed by (Shenoy & Shafer, 1990). A particular emphasis is laid on the important special cases of idempotent algebras and algebras with some notion of division. It is shown that all well-known architectures for local computation like the Shenoy-Shafer architecture, Lauritzen-Spiegelhalter and HUGIN architectures may be adapted to this new framework. Further a new architecture for idempotent algebras is presented. As examples, in addition to the classical instances of valuation algebras, semiring induced valuation algebras, Gaussian potentials and the relational algebra are presented.

History

This paper pursues a technical report (Schneuwly *et al.*, 2004) from the University of Fribourg. It was submitted to a journal in 2005 where it was forgotten for more than two years. Later, the paper was rejected, mainly because its content flowed into other publications in the meantime, in particular into (Pouly, 2008) and (Pouly & Kohlas, 2011).

Contents

1 Introduction **3**

2 Valuation Algebra **5**

2.1 Axiomatic 6

2.2 A Few Examples 8

2.2.1 Semiring-Valued Potentials 9

2.2.2 Gaussian Potentials 10

2.2.3 Densities 10

2.2.4 Relational Algebra 11

2.2.5 Logic 11

2.3 Neutral Elements 12

3 Local Computation **13**

3.1 Factorizations and Join Trees 13

3.2 Collect Algorithm 15

3.3 Shenoy-Shafer Architecture 20

4 Division in Valuation Algebras **22**

4.1 Separative Valuation Algebras 22

4.1.1 Gaussian Potentials 24

4.1.2 Densities 24

4.2 Regular Valuation Algebras 27

4.2.1 Discrete Probability Potentials 28

4.2.2 Information Algebras 28

4.2.3 Regular Semirings 29

5 Architectures using Division **29**

5.1 Lauritzen-Spiegelhalter Architecture 29

5.2 HUGIN Architecture 33

5.3 Local Computation in Information Algebras 35

6 Conclusion **36**

References **37**

1 Introduction

Local computation or tree-decomposition techniques were originally introduced for probability networks by (Lauritzen & Spiegelhalter, 1988) to provide a solution for an otherwise computationally intractable problem. Based on this work, (Shenoy & Shafer, 1990) formulated a set of sufficient axioms for the application of this algorithm, and it was also shown that other formalisms as belief functions for example satisfy these axioms and therefore qualify for the application of local computation. This laid the foundation for a generic approach to inference, reasoning and combination of information based on local computation.

The mathematical base for local computation is provided by the algebraic structure determined by the axioms introduced in (Shenoy & Shafer, 1990). This structure is now called a *valuation algebra* (Kohlas & Shenoy, 2000; Kohlas, 2003) and was first introduced in (Shenoy, 1989), see also (Shenoy, 1992b). A first algebraic study of these structures and local computation was laid down in an unpublished paper (Shafer, 1991). Valuation algebras provide a unifying approach to reasoning, covering many very different formalisms ranging from different uncertainty calculi like probability, possibility theory and belief functions to various logical systems, passing by relational algebra, constraint systems, systems of equations and inequalities, formalisms related to path problems and many more. A related unifying approach to reasoning is also given in (Dechter, 1999).

In the valuation algebra framework, pieces of information are represented by *valuations*, and a set of valuations is called *knowledgebase*. Each valuation refers to a certain *domain* which reflects the set of questions that is generally associated with a piece of information. A knowledgebase then specifies a computational task called *inference problem* that requires to combine or aggregate all its valuations and to project, focus or marginalize the combination onto some *queries* of interest. This will be formulated more precisely in Section 2 and 3. It turns out that a direct solution of inference problems according to this description is computationally intractable in most cases. If, however, the domains of the knowledgebase valuations and the queries form a *hypertree*, then the axioms of the valuation algebra allow to define a procedure for the solution of inference problems where the domains of valuations are always bounded by the hyperedges of the hypertree. This technique called *local computation* is computationally feasible, if the cardinalities of the hyperedges are not too large. The condition that the domains of a set of valuations together with the queries form a hypertree is very strong and only incidentally, but by no means generally, satisfied. On the other hand, it is always possible to construct a *covering hypertree* (also called *tree-decomposition*) whose hyperedges cover the domains of the given valuations and queries. Then, if the valuation algebra contains *neutral elements*, the valuations may be changed such that their domains coincide with the hyperedges of the covering hypertree. This makes the application of local computation on the modified knowledgebase possible. In the literature, it was tacitly assumed that such neutral elements always exist, which is indeed the case for popular systems such as probability networks or belief functions. But it will be argued in this paper that important systems without neutral elements exist. A typical example are

Gaussian potentials, for which the approach using covering hypertrees as proposed so far is thus not applicable. The main contribution of this paper is to show that local computation on covering hypertrees can still be exploited in these cases. Moreover, it will be argued that even if neutral elements exist, it is not efficient to use them for the modification of the original knowledgebase valuations. Finally, there are also cases where neutral elements exist, but having no finite representation. This makes their use not only inefficient but infeasible. To sum it up, this paper shows for the first time that local computation on covering hypertrees is possible without the presence and use of neutral elements in the underlying valuation algebra.

It is worth noting that there is a related approach to local computation for the treatment of *Boolean conjunctive queries* (BCQs) in the domain of relational databases (Gottlob *et al.*, 1999b), which may also be applied to *constraint satisfaction problems* (CSPs) (Gottlob *et al.*, 1999a). Its main subject is to identify so-called *hypertree decompositions of bounded hypertree width*. Strictly speaking, it is a generalization of the query decomposition of the bounded query width concept introduced in the same research field (Chekuri & Rajaraman, 1997). Both are based on covering join trees. Since relational algebra is a prototype algebra where neutral elements are not finitely representable, they can not be used in order to fill the nodes. Instead, the convenient idempotency property of BCQs and CSPs can be exploited for this purpose. In this way, efficient constructions of hypertrees for new queries which are equivalent to the starting becomes possible (Gottlob *et al.*, 1999b; Gottlob *et al.*, 2001). While they make local computation possible, (Gottlob *et al.*, 1999b) showed that queries of bounded query width are, in difference to queries of bounded hypertree width, not efficiently recognizable. Nevertheless, these approaches may only be applied to idempotent algebras.

In Section 2, we formulate carefully the axioms of a valuation algebra as used in this paper. They differ slightly from the original ones given in (Shenoy & Shafer, 1990) and used in other publications (e.g. (Shafer, 1991; Lauritzen & Jensen, 1997; Mengin & Wilson, 1999; Kohlas, 2003)). We claim that these new axioms are sufficient for local computation according to our modified version. Moreover, if neutral elements are present, then this new axiomatic system becomes equivalent to the traditional system. A selection of formalisms that satisfy the valuation algebra axioms will then be given. As a first main result, it will be shown that a *unique* identity element can always be adjoined to such a valuation algebra, if it is not yet present. This identity element will enable us to formulate the modified local computation algorithm. In Section 3, a first version of a local computation scheme based on covering join trees will be introduced. In a first step, the *collect algorithm* for computing the marginal on a given pre-specified domain will be formulated and proved. It will be shown that this coincide essentially with the well-known *fusion algorithm* (originally introduced in (Cannings *et al.*, 1978; Shenoy, 1992a)) and *bucket-elimination scheme* (Dechter, 1999). The latter however are formulated in terms of variable elimination, whereas the collect algorithm is expressed using the more general projections or marginalization operator. In many practical cases, not only a single but multiple queries have to be computed from a given knowledgebase. It is well-known that caching avoids redundant computations in such cases. One organization exploiting this is given by the so-called *Shenoy-Shafer Architecture* (Shenoy & Shafer, 1990), and it will be shown, how local computation based on this architecture can be adapted to covering hypertrees without using neutral elements, even if they exist.

It is known from probability networks that local computation schemes using division can be formulated, provided that some concept of inverse elements exists in a valuation algebra. Sufficient conditions for their presence are studied in Section 4, following the abstract description proposed by (Lauritzen & Jensen, 1997). However, we will present in a more precise way first a general condition for defining division, leading to so-called *separative valuation algebras*. An instance of a separative algebra is the valuation algebra of Gaussian potentials, where division leads to conditional Gaussian distributions. More restricted is a regularity condition, which is for example satisfied for discrete probability potentials, and which leads to *regular valuation algebras* as a special case of separative valuation algebras. This section is a summary of a theory developed in (Kohlas, 2003). Local computation architectures exploiting division are the *Lauritzen-Spiegelhalter Architecture* proposed in (Lauritzen & Spiegelhalter, 1988) and the *HUGIN Architecture* described in (Jensen *et al.*, 1990). In Section 5, we show that these architectures can be adapted for covering hypertrees and regular valuation algebras without neutral elements. An important case of regular valuation algebras are idempotent valuation algebras (also called *information algebras* (Kohlas, 2003)). In this case, both architectures above collapse to a very simple and symmetric new architecture.

We provide in this paper a rigorous base for generic local computation on covering hypertrees for the most general case of valuation algebras without neutral elements. This furthermore implies that even if neutral elements are present, they do not need to be used for the extension of the domains of valuations to hyperedges, which thus enables a more efficient organization of local computation. The theory presented here is implemented in a software framework called NENOK (Pouly, 2008) that offers generic implementations of local computation architectures. This library can be accessed by any implemented formalism that satisfies the valuation algebra axioms.

2 Valuation Algebra

Information or knowledge concerns generally a certain domain. It can be aggregated with other pieces and focused to the part we are interested in. In order to deal with this conception of knowledge or information, a precise formalism is needed given by a system of axioms determining the behavior of the three basic operations: *labeling* for retrieving the domain, *combination* for aggregation and *marginalization* or *projection* for focusing of knowledge. The resulting algebraic system is called a valuation algebra. The concepts and ideas are mainly taken from (Kohlas & Shenoy, 2000; Kohlas, 2003).

2.1 Axiomatic

The basic elements of a valuation algebra are so-called *valuations*. Intuitively, a valuation can be regarded as a representation of knowledge about the possible values of a set of variables. It can be said that each valuation ϕ refers to a finite set of variables $d(\phi)$, called its *domain*. For an arbitrary set s of variables, Φ_s denotes the set of valuations ϕ with $d(\phi) = s$. With this notation, the set of all possible valuations corresponding to a finite set of variables r can be defined as

$$\Phi = \bigcup_{s \subseteq r} \Phi_s.$$

Let D be the lattice of subsets (the powerset) of r. For a single variable X, Ω_X denotes the set of all its possible values. We call Ω_X the *frame* of variable X. In an analogous way, we define the frame of a non-empty variable set $s \in D$ by the Cartesian product of frames Ω_X of each variable $X \in s$,

$$\Omega_s = \prod_{X \in s} \Omega_X. \tag{2.1}$$

The elements of Ω_s are called *configurations* of s. The frame of the empty variable set is defined by convention as $\Omega_\emptyset = \{\diamond\}$.

Let Φ be a set of valuations with their domains in D. We assume the following operations defined on Φ and D:

1. *Labeling:* $\Phi \to D$; $\phi \mapsto d(\phi)$,
2. *Combination:* $\Phi \times \Phi \to \Phi$; $(\phi, \psi) \mapsto \phi \otimes \psi$,
3. *Marginalization:* $\Phi \times D \to \Phi$; $(\phi, x) \mapsto \phi^{\downarrow x}$, for $x \subseteq d(\phi)$.

These are the three basic operations of a valuation algebra. Valuations can be regarded as pieces of information. The label of a valuation determines its domain and it is retrieved by the labeling operation. Combination represents aggregation of pieces of information and marginalization of a valuation (sometimes also called projection) focusing of information, i.e. extraction of the part related to some subdomain.

We impose now the following set of axioms on Φ and D:

(A1) *Commutative Semigroup:* Φ is associative and commutative under $\otimes$.

(A2) *Labeling:* For ϕ, $\psi \in \Phi$,

$$d(\phi \otimes \psi) \quad = \quad d(\phi) \cup d(\psi).$$

(A3) *Marginalization:* For $\phi \in \Phi$, $x \in D$, $x \subseteq d(\phi)$,

$$d(\phi^{\downarrow x}) \quad = \quad x.$$

(A4) *Transitivity:* For $\phi \in \Phi$ and $x \subseteq y \subseteq d(\phi)$,

$$(\phi^{\downarrow y})^{\downarrow x} \quad = \quad \phi^{\downarrow x}.$$

(A5) *Combination:* For ϕ, $\psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$,

$$(\phi \otimes \psi)^{\downarrow z} \quad = \quad \phi \otimes \psi^{\downarrow z \cap y}.$$

(A6) *Domain:* For $\phi \in \Phi$ with $d(\phi) = x$,

$$\phi^{\downarrow x} \quad = \quad \phi.$$

Definition 1 *A system (Φ, D) with the operations of labeling, combination and marginalization satisfying these axioms is called a valuation algebra.*

The axioms express natural properties of pieces of information and their operations. The first axiom says that Φ is a commutative semigroup under combination. It means that if information comes in pieces, the sequence in which the pieces are aggregated does not influence the result, the combined information. The labeling axiom says that the combination of valuations relates to the union of the domains involved. The marginalization axiom expresses what we expect, namely that the domain of a valuation which is focused on some subdomain is exactly this subdomain. Transitivity means that marginalization can be performed in steps. The combination axiom is the most important axiom for local computation. It states that if we have to combine two valuations and then marginalize the result to a domain containing the domain of the first one, we do not need first to combine and then to marginalize. We may as well marginalize the second valuation to the intersection of its domain and the target domain. This avoids the extension to a domain which is the union of the domains of the two factors, according to the labeling axiom, if we combine before marginalization. This, in a nutshell, is what local computation is about. Finally, the domain axiom assures that information is not influenced by trivial projection.

Usually, and especially in the original paper (Shenoy & Shafer, 1990), only axioms (A1), (A4) and (A5) (the latter in a simplified version) are stated. The labeling axiom (A2) and the marginalization axiom (A3) are tacitly assumed (in (Shafer, 1991) the labeling axiom however is formulated). They do however not follow from the other axioms. The domain axiom (A6) also is not a consequence of the other ones, as was already shown in (Shafer, 1991). It expresses some kind of stability of a piece of information under trivial projection which is important for local computation.

Often a *neutral element* is assumed in each semigroup Φ_s, i.e. an element e_s such that $e_s \otimes \phi = \phi \otimes e_s = \phi$ for all valuations $\phi \in \Phi_s$ (e.g. (Shafer, 1991; Kohlas, 2003)). Then it is postulated that the neutrality axiom holds

$$e_s \otimes e_t \quad = \quad e_{s \cup t}. \tag{2.2}$$

However we shall see below (Subsection 2.2) that there are important examples where such an element does not exist, or exists, but is not representable in the chosen framework. That is why we choose not to assume (in general) the existence of neutral elements.

The combination axiom usually is formulated in the following simplified form: If $d(\phi) = x$ and $d(\psi) = y$, then

$$(\phi \otimes \psi)^{\downarrow x} = \phi \otimes \psi^{\downarrow x \cap y}. \tag{2.3}$$

This is a particular case of the combination axiom (A5). If the valuation algebra has neutral elements satisfying the Neutrality Axiom, then the simplified version is equivalent to (A5). But otherwise this does not hold, and for local computation we need the version (A5) of the combination axiom.

Sometimes it is also assumed that each semigroup Φ_s contains a *null* or *absorbing element*, i.e. an element z_s such that $z_s \otimes \phi = \phi \otimes z_s = z_s$ for all valuations $\phi \in \Phi_s$. It represents contradictory information and exist in many (even most) valuation algebra instances. However, there are again important cases where null elements do not exists, and for this reason, their existence is not assumed in the above system.

Finally, we remark that instead of a domain lattice D of subsets, one might consider any lattice of domains as for example partitions of a set which form a non-distributive lattice. Local computation can still be developed in this more general setting (Shafer, 1991; Kohlas & Monney, 1995). But, without distributivity in the lattice of domains, this becomes more involved and will not be considered here.

The following lemma describes a few elementary properties of valuation algebras derived from the set of axioms. We refer to (Kohlas, 2003) for their simple proofs.

Lemma 1

1. *If ϕ, $\psi \in \Phi$ with $d(\phi) = x$ and $d(\psi) = y$, then*

$$(\phi \otimes \psi)^{\downarrow x \cap y} = \phi^{\downarrow x \cap y} \otimes \psi^{\downarrow x \cap y}. \tag{2.4}$$

2. *If ϕ, $\psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$ and $z \subseteq x$, then*

$$(\phi \otimes \psi)^{\downarrow z} = (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow z}. \tag{2.5}$$

2.2 A Few Examples

The axioms for a valuation algebra as proposed above will prove sufficient for local computation. On the other hand, they cover the known interesting examples of knowledge or information representation. This will be illustrated by selected instances in this subsection.

2.2.1 Semiring-Valued Potentials (Kohlas & Wilson, 2008)

A *semiring* is a set A with two binary operations designated by $+$ and $\times$, which satisfy the following conditions:

1. $+$ and $\times$ are commutative[1] and associative,
2. $\times$ is distributive over $+$, i.e. for $a, b, c, \in A$ we have
$$a \times (b + c) \quad = \quad (a \times b) + (a \times c).$$

Examples of semirings are the *Boolean semiring*, where $A = \{0, 1\}$, $a+b = \max\{a, b\}$, $a \times b = \min\{a, b\}$, the *Bottleneck Algebra*, where $+$ is the max operation and $\times$ the min operation on pairs of real numbers, augmented with $+\infty$ and $-\infty$, or $(\max / \min, +)$ semirings, where A consists of the nonnegative integers plus $+\infty$. Addition $+$ is taken as the min or, alternatively, the max-operation, whereas $\times$ is the ordinary multiplication. The nonnegative reals $\mathbb{R}_0^+$ together with the ordinary addition and multiplication form also a semiring. Finally the interval $[0, 1]$ with $+$ as the max operation and any t-norm for multiplication yields another semiring. We remind that a *t-norm* is a binary operation on the unit interval that is associative, commutative and non-decreasing in both arguments.

The associativity of $+$ allows to write expressions like $a_1 + \cdots + a_n$ or $\sum_i a_i$. If now A is a semiring, we define valuations on s by mappings from configurations to semirngs values, $\phi : \Omega_s \to A$. If $\mathbf{x}$ is a configuration of s and $t \subseteq s$, then let $\mathbf{x}^{\downarrow t}$ denote the configuration of t consisting of the components x_i of $\mathbf{x}$ with $i \in t$. Then we define the following operations with respect to semiring-valued valuations:

1. *Labeling:* $d(\phi) = s$ if ϕ is a valuation on s,
2. *Combination:* If $d(\phi) = s$, $d(\psi) = t$ and $\mathbf{x}$ is a configuration of $s \cup t$, then
$$\phi \otimes \psi(\mathbf{x}) = \phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow t}), \tag{2.6}$$
3. *Marginalization:* If $t \subseteq d(\phi)$ and $\mathbf{x}$ is a configuration of t, then
$$\phi^{\downarrow t}(\mathbf{x}) = \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}). \tag{2.7}$$

It is easy to see that these semiring-valued valuations form a valuation algebra.

In the case of $\mathbb{R}_0^+$ with ordinary addition and multiplication, this is the valuation of *discrete probability potentials* as studied in (Lauritzen & Spiegelhalter, 1988; Shenoy & Shafer, 1990). In the case of a Boolean semiring this corresponds to *constraint systems*. If we use t-norms for multiplication and min for $+$ we get various systems of possibility measures. The $(\max / \min, +)$ semirings lead to algebras used in *dynamic optimization*. Thus, semiring-valued valuations cover many important valuation algebras. They may or may not contain neutral elements, depending on whether the underlying semiring has a unit element, i.e. a neutral element of multiplication.

[1]Commutativity of multiplication is not always required in the literature.

2.2.2 Gaussian Potentials (Kohlas, 2003)

Consider a family of variables X_i with $i \in I = \{1, \ldots, n\}$. A Gaussian distribution over a subset of these variables is determined by its mean value vector and the concentration matrix, the inverse of the variance-covariance matrix. If s is a subset of the index set I, then let $\mu : s \to \mathbb{R}$ denote the mean value vector relative to the variables in s and $K : s \times s \to \mathbb{R}$ the concentration matrix, which is assumed to be *positive definite*. If μ and K are defined relative to a set s and $t \subseteq s$, then $\mu^{\downarrow t}$ and $K^{\downarrow t}$ denote the sub-vector or submatrix with only components $\mu(i)$ and $K(i,j)$ belonging to t. If, on the other hand $t \supseteq s$, then $\mu^{\uparrow t}$ and $K^{\uparrow t}$ denote the vector or matrix obtained from μ or K by setting $\mu(i) = 0$ and $K(i,j) = 0$ for $i, j \in t - s$.

A pair (μ, K) where both μ and K are relative to a subset $s \subseteq I$ is called a *Gaussian potential*, and the set s is the label of the potential, $d(\mu, K) = s$. Further, we define the operation of combination between two Gaussian potentials (μ_1, K_1) and (μ_2, K_2) with domains s and t respectively as follows:

$$(\mu_1, K_1) \otimes (\mu_2, K_2) \quad = \quad (\mu, K), \tag{2.8}$$

where

$$K \quad = \quad K_1^{\uparrow s \cup t} + K_2^{\uparrow s \cup t}$$

and

$$\mu \quad = \quad K^{-1} \left(K_1^{\uparrow s \cup t} \cdot \mu_1^{\uparrow s \cup t} + K_2^{\uparrow s \cup t} \cdot \mu_2^{\uparrow s \cup t} \right).$$

For a Gaussian potential (μ, K) on domain s, marginalization to a set $t \subseteq s$ is defined by

$$(\mu, K)^{\downarrow t} \quad = \quad (\mu^{\downarrow t}, ((K^{-1})^{\downarrow t})^{-1}). \tag{2.9}$$

This system satisfies the axioms of a valuation algebra, and there are no neutral elements. This algebra becomes most important when division is introduced, allowing to represent *conditional Gaussian distributions*, see Section 4.1.

2.2.3 Densities (Kohlas, 2003)

A continuous, nonnegative-valued function f on $\mathbb{R}^n$ is called a *density*, if its integral is finite,

$$\int_{-\infty}^{+\infty} f(\mathbf{x}) d\mathbf{x} \quad < \quad \infty.$$

Let $I = \{1, \ldots, n\}$ be an index set. For any subset $s \subseteq I$ we consider the set Φ_s of densities $f : \mathbb{R}^{|s|} \to \mathbb{R}$ with domain $d(f) = s$. If f is a density with domain s and $t \subseteq s$, the marginal of f with respect to t is defined by the integral

$$f^{\downarrow t}(\mathbf{x}) \quad = \quad \int_{-\infty}^{+\infty} f(\mathbf{x}, \mathbf{y}) d\mathbf{y},$$

if $\mathbf{x}$ denotes configurations with respect to t and $\mathbf{y}$ configurations with respect to $s - t$. For two densities f and g on s and t respectively, the combination is defined, for $\mathbf{x}$ configuration of $s \cup t$, by

$$f \otimes g(\mathbf{x}) \quad = \quad f(\mathbf{x}^{\downarrow s}) \cdot g(\mathbf{x}^{\downarrow t}).$$

It can be shown that densities on subsets of I form a valuation algebra. It has no neutral elements, since $f(\mathbf{x}) = 1$ for all $\mathbf{x}$ has no finite integral and is therefore not a density. But it has a null element $f(\mathbf{x}) = 0$ for all $\mathbf{x}$, which is a density according to our definition.

2.2.4 Relational Algebra

An important instance of a valuation algebra is the relational algebra. Let $\mathcal{A}$ be a finite set of symbols called *attributes*. For each $\alpha \in \mathcal{A}$ let U_α be a non-empty set, called the domain of attribute α. Let $s \subseteq \mathcal{A}$. An *s-tuple* is a function f with domain s and $f(\alpha) \in U_\alpha$. The set of all s-tuples is called E_s. For an s-tuple and a subset t of s the restriction $f[t]$ is defined to be the t-tuple g such that $g(\alpha) = f(\alpha)$ for all $\alpha \in t$.

A *relation* R over s is a set of s-tuples, i.e. a subset of E_s. The set of attributes s is called the domain of R and denoted by $d(R) = s$. If R is a relation over s and t a subset of s, then the *projection* of R onto t is defined as follows:

$$\pi_s(R) \quad = \quad \{f[t] : f \in R\}. \tag{2.10}$$

The *natural join* of a relation R over s and a relation S over t is defined as

$$R \bowtie S \quad = \quad \{f \in E_{s \cup t} : f[s] \in R, f[t] \in S\}. \tag{2.11}$$

It is easy to see that the algebra of relations with join as combination and projection as marginalization is a valuation algebra. The full relations E_s are neutral elements in this algebra. It is essentially the same as the constraint algebra introduced in Example 2.2.1. A particularity of this algebra is its property of *idempotency*: A relation combined (joined) by a projection of itself returns the original relation. Such algebras are also called *information algebras* and have a rich theory (Kohlas, 2003).

The family of *finite* relations is closed under projection and join and forms itself a valuation algebra. If however some of the sets U_α are infinite, then the neutral elements do not belong to this algebra. Moreover, such infinite sets are not explicitly representable as relations.

2.2.5 Logic

Many logics have an algebraic theory. Lindenbaum algebras represent propositional logic (Davey & Priestley, 1990) and predicate logic has cylindric algebras (Henkin *et al.*, 1971) as its algebraic counterpart. These algebras are closely related to valuation algebras, where valuations represent logical statements. Cylindric algebras are in fact instances of valuation algebras. For valuation algebras related to propositional logic, we refer to (Kohlas *et al.*, 1999). More general relations between logic and valuation algebra are presented in (Mengin & Wilson, 1999; Kohlas, 2003) and also in a different direction in (Kohlas, 2002).

These examples are by no means exhaustive. Other important instances are provided by belief functions (Shenoy & Shafer, 1990), systems of linear equations or linear inequalities, equivalently represented by affine linear manifolds or convex polyhedra (Kohlas, 2003), where local computation is closely related to sparse matrix techniques, and convex sets of discrete probability distributions (Cano *et al.*, 1992). For even further examples we refer to (Kohlas, 2003; Pouly, 2008).

2.3 Neutral Elements

The existence of neutral elements is not mandatory in a valuation algebra as we have seen. Nevertheless, for computational purposes, it is convenient to have at least one identity or neutral element which serves as placeholder whenever no valuation, no knowledge or information, is available at the moment. In this subsection we show that it is always possible to adjoin such an element, if it is not already provided by a valuation algebra (Schneuwly, 2007; Pouly, 2008).

Let (Φ, D) be an arbitrary valuation algebra as defined in Section 2.1. We add a new valuation e to Φ and denote the resulting system by (Φ', D). The operations of the algebra are extended from Φ to Φ' in the following way:

1. *Labeling:* $\Phi' \rightarrow D$; $\phi \mapsto d'(\phi)$,
   - $d'(\phi) = d(\phi)$, if $\phi \in \Phi$;
   - $d'(e) = \emptyset$;
2. *Combination:* $\Phi' \times \Phi' \rightarrow \Phi'$; $(\phi, \psi) \mapsto \phi \otimes' \psi$,
   - $\phi \otimes' \psi = \phi \otimes \psi$ if $\phi, \psi \in \Phi$;
   - $\phi \otimes' e = e \otimes' \phi = \phi$ if $\phi \in \Phi$;
   - $e \otimes' e = e$;
3. *Marginalization:* $\Phi' \times D \rightarrow \Phi'$; $(\phi, x) \mapsto \phi^{\downarrow' x}$, for $x \subseteq d(\phi)$
   - $\phi^{\downarrow' x} = \phi^{\downarrow x}$ if $\phi \in \Phi$;
   - $e^{\downarrow' \emptyset} = e$.

We claim that the extended algebra is still a valuation algebra.

Lemma 2 *(Φ', D) with the extended operations d', $\otimes'$ and $\downarrow'$ is a valuation algebra.*

Proof. Note that the axioms are satisfied for the elements in Φ. Therefore, we need only to verify them for the adjoined neutral element. This is straightforward using the definitions of the extended operations, except for the combination axiom.

For $\phi \in \Phi$ with $d(\phi) = x$, $d(e) = y = \emptyset$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$ it follows $z = x$ and by the domain axiom in (Φ, D)

$$(\phi \otimes' e)^{\downarrow' z} \;=\; \phi^{\downarrow' z} \;=\; \phi^{\downarrow x} \;=\; \phi \;=\; \phi \otimes' e^{\downarrow' \emptyset} \;=\; \phi \otimes' e^{\downarrow' z \cap \emptyset}.$$

On the other hand, let $d(e) = x = \emptyset$, $d(\phi) = y$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$ and it follows $z \cap y = z$. We get

$$(e \otimes' \phi)^{\downarrow' z} = \phi^{\downarrow' z} \;=\; e \otimes' \phi^{\downarrow' z} \;=\; e \otimes' \phi^{\downarrow' z \cap y}.$$

Finally,

$$(e \otimes' e)^{\downarrow' \emptyset} \;=\; e \;=\; e \otimes' e^{\downarrow' \emptyset \cap \emptyset}.$$

This proves the combination axiom, when the neutral element occurs. □

We usually identify the operators in (Φ', D) like in (Φ, D), i.e. d' by d, $\otimes'$ by $\otimes$ and $\downarrow'$ by $\downarrow$ if they are not used to distinguish between the two algebras. Next, we note that there can only be one neutral element e in a valuation algebra such that $e \otimes \phi = \phi$ for all $\phi \in \Phi$. In fact, assume another element e' with this property. Then

$$e \;=\; e \otimes e' \;=\; e'$$

and the two elements are identical. As a consequence, if the valuation algebra already has neutral elements, we do not need to adjoin a new one. This is expressed in the next lemma.

Lemma 3 *If (Φ, D) is a valuation algebra with neutral elements satisfying the neutrality axiom, then $e_\emptyset \in \Phi$ satisfies the same properties in (Φ, D) as e in (Φ', D).*

Proof. The domain of $e_\emptyset$ is $d(e_\emptyset) = \emptyset$. We get by the neutrality axiom and commutativity

$$\phi \;=\; \phi \otimes e_s \;=\; \phi \otimes e_{s \cup \emptyset} \;=\; \phi \otimes e_s \otimes e_\emptyset \;=\; \phi \otimes e_\emptyset \;=\; e_\emptyset \otimes \phi.$$

The property $e_\emptyset \otimes e_\emptyset = e_\emptyset$ follows by the definition of a neutral element. Finally, marginalization follows from the domain axiom $e_\emptyset^{\downarrow \emptyset} = e_\emptyset$. □

Henceforth, we assume that we always dispose of a neutral element e with domain $d(e) = \emptyset$ in a valuation algebra. Either it is already there, or we may adjoin it.

3 Local Computation

3.1 Factorizations and Join Trees

A basic generic problem within a valuation algebra (Φ, D) is the *projection problem*: Given a finite number of valuations $\phi_1, \ldots, \phi_m \in \Phi$, and a domain $x \in D$, compute the marginal

$$(\phi_1 \otimes \cdots \otimes \phi_m)^{\downarrow x}. \tag{3.1}$$

Let $s_i = d(\phi_i)$ denote the domains of the factors in the combination above. If this marginal is computed naively by first combining all factors, then a valuation of domain $s_1 \cup \cdots \cup s_m$ is obtained. This is often unfeasible, because this domain is much too large. Therefore, more efficient procedures are required, where never valuations on domains which are essentially larger than the domains of the original valuations s_i arise. Computations to solve the projection problem (3.1) which satisfy this requirement are called *local computations*. Local computation is possible, when the domains s_i of the factors satisfy some strong conditions formulated next.

A family of finite subsets $\{s_1, \ldots, s_m\}$ of some index set $I = \{1, \ldots, n\}$ defines a *hypergraph* and the sets s_i are called its *hyperedges*. A hypergraph is a *hypertree* if it contains no cycles, i.e. no sequences of hyperedges $s_{i_1}, s_{i_2}, \ldots, s_{i_k}, s_{i_1}$ such that the intersection of two consecutive hyperedges is not empty. It is well-known that a *join tree* can be associated to every hypertree. A join tree is a tree $T = (V, E)$ with a set of vertices V and a labeling function λ which assigns to each vertex $v \in V$ a subset $\lambda(v)$ of I such that the *running intersection property* is satisfied: If an index i belongs to the label of two vertices v_1 and v_2 of the tree, $i \in \lambda(v_1), \lambda(v_2)$, then it belongs to the label of all nodes on the path between v_1 and v_2. Now in (Lauritzen & Spiegelhalter, 1988) and (Shenoy & Shafer, 1990) it has been shown that local computation is possible, if the family $\{s_1, \ldots, s_m\}$ is a hypertree and x is a subset of one of the domains s_i. Local computation is then based on an associated join tree.

However, this requirement is rarely satisfied for a projection problem (3.1). Furthermore, the projection problem has often to be solved not only for a single target domain x, but rather for a family $\{x_1, \ldots, x_k\}$ of target domains. Usually, the following approach is then proposed in the literature: It is always possible to find a join tree T such that for all s_i and all x_j there is some vertex v such that $s_i \subseteq \lambda(v)$ or $x_j \subseteq \lambda(v)$. Such a join tree is called a *covering join tree* or *tree-decomposition* for the (extended) projection problem. We have then an *assignment mapping* $a : \{1, \ldots, m\} \to V$ which assigns each factor ϕ_i to a node $v \in V$ such that $s_i \subseteq \lambda(v)$. We assume henceforth that the vertices in V are enumerated from $i = 1$ to $|V|$, such that we can access them by their index. Thus, to node $i \in V$ we assign the valuation

$$\psi_i = \bigotimes_{j:a(j)=i} \phi_j$$

if there is at least one valuation ϕ_j assigned by a to node i. Otherwise no valuation is assigned to node i.

Now, if the valuation algebra has neutral elements, then we may assign to all nodes without factor assigned the neutral element $\psi'_i = e_{\lambda(i)}$ and to the other nodes we assign the extended valuation $\psi'_i = \psi_i \otimes e_{\lambda(i)}$. Clearly, by the nature of neutral elements,

$$\bigotimes_{i=1}^{m} \phi_i = \bigotimes_{j=1}^{|V|} \psi'_j.$$

By the labeling axiom, the domains of this transformed factorization are $\lambda(j)$, and they form a join tree. So, local computation techniques can be applied.

The first problem is that, as we have seen in Section 2.2, there are valuation algebras which have no neutral elements, like the Gaussian potentials. But even if neutral elements exist, they may have an infinite representation, like in the relational algebra, and can not be used as proposed in this approach. Finally, in any case, neutral elements represent trivial, but large data, as for example large tables of unit elements in the case of semiring-valued valuations, e.g. in the case of probability potentials. We therefore propose a more general, and in all cases, more efficient approach. In fact, we adjoin a neutral element e, where necessary (or take $e = e_\emptyset$, if neutral elements exist) and define $\psi_i = e$, for nodes of the covering join tree, where no original factor is assigned. In this way, every node of the covering join tree has a valuation assigned and again we have the identity

$$\bigotimes_{i=1}^{m} \phi_i = \bigotimes_{j=1}^{|V|} \psi_j. \tag{3.2}$$

Further any target domain x_i is covered by some node of the join tree. But now the domains of the factorization (3.2) form no join tree in general. The main result of this paper is to show that nevertheless all known architecture for local computation can be adapted to this new situation.

3.2 Collect Algorithm

The original projection problem (3.1) can be solved by the peeling (Cannings *et al.*, 1978), fusion algorithm (Shenoy, 1992a) or *bucket-elimination scheme* (Dechter, 1999). This is a local computation technique, which does not need a join tree (although implicitly it generates one) and which therefore does not suffer from the problems discussed above. The fusion algorithm is however based on *variable elimination* instead of marginalization and solves only the marginalization relative to a unique target domain. Variable elimination is closely related to marginalization, but not identical (Kohlas & Shenoy, 2000; Kohlas, 2003). The fusion algorithm can be translated into a procedure based on covering join trees and using marginalization, which also does not suffer from the problems cited above. This is our starting point.

Consider a covering join tree for a factorization $\psi_1 \otimes \cdots \otimes \psi_m$ with domains $d(\psi_i) = s_i \subseteq \lambda(i)$ as obtained according to the previous Section 3.1 from an original factorization and a number of target domains. We want to compute the marginal of the combination to a target domain which corresponds to one of the vertex domains of the join tree. Without loss of generality we may assume that the target domain is $\lambda(m)$. So, the projection problem considered is

$$(\psi_1 \otimes \cdots \otimes \psi_m)^{\downarrow \lambda(m)}. \tag{3.3}$$

We may always number the nodes of the join tree in such a way that $i < j$ if j is a node on the path from i to m. The vertex m is called the root node. The neighbor of a node i on the path towards m is called the child of i and denoted by $ch(i)$. In order to describe a m-step algorithm on the join tree we assign a storage to each node i of the join tree to store a valuation and a related domain and define

- $\psi_j^{(1)} = \psi_j$ is the initial content of node j;
- $\psi_j^{(i)}$ is the content of node j before step i of the algorithm.

A similar notation is used to refer to the domain of a node:

- $\omega_j^{(1)} = \omega_j = d(\psi_j)$ is the initial domain of node j;
- $\omega_j^{(i)} = d(\psi_j^{(i)})$ is the domain of node j before step i of the algorithm.

The node numbering introduction implies that at step i, node i can send its message to its child. The collect algorithm can now be specified formally:

- At step i, node i computes the message
$$\mu_{i \to ch(i)} = \psi_i^{(i) \downarrow \omega_i^{(i)} \cap \lambda(ch(i))}. \tag{3.4}$$
This message is sent to the child node $ch(i)$ with node label $\lambda(ch(i))$.
- The receiving node $ch(i)$ updates its storage to
$$\psi_{ch(i)}^{(i+1)} = \psi_{ch(i)}^{(i)} \otimes \mu_{i \to ch(i)}. \tag{3.5}$$
Its node domain changes to:
$$\omega_{ch(i)}^{(i+1)} = d(\psi_{ch(i)}^{(i+1)}) = \omega_{ch(i)}^{(i)} \cup \left(\omega_i^{(i)} \cap \lambda(ch(i)) \right). \tag{3.6}$$
The storages of all other nodes do not change at step i,
$$\psi_j^{(i+1)} = \psi_j^{(i)} \tag{3.7}$$
for all $j \neq ch(i)$. The same holds for the node domains: $\omega_j^{(j+1)} = \omega_j^{(i)}$.

This is the *collect algorithm*. It is similar to the fusion algorithm or bucket-elimination scheme: On node i at step i we collect all remaining valuations containing the variables with indices $\omega_i^{(i)} - \lambda(ch(i))$, which are to be eliminated (by marginalization to the intersection $\omega_i^{(i)} \cap \lambda(ch(i))$). Only, instead of eliminating variable by variable, a whole group of variables are eliminated in one step by marginalization.

The justification of the collect algorithm is formulated by the following theorem:

Theorem 1 *At the end of the collect algorithm, the root node m contains the marginal of ϕ relative to $\lambda(m)$,*
$$\psi_m^{(m)} \quad = \quad \phi^{\downarrow \lambda(m)}. \tag{3.8}$$

In order to prove this important theorem, we need the following lemma:

Lemma 4 *For $i = 1, \dots, m$ we define*

$$y_i = \bigcup_{j=i}^{m} \omega_j^{(i)}. \tag{3.9}$$

Then, for i = 1,...,m-1,

$$\left(\bigotimes_{j=i}^{m} \psi_j^{(i)} \right)^{\downarrow y_{i+1}} = \bigotimes_{j=i+1}^{m} \psi_j^{(i+1)} = \phi^{\downarrow y_{i+1}} \tag{3.10}$$

Proof. We show first that $y_{i+1} \subseteq y_i$ to guarantee that the marginalization in Equation (3.10) is well-defined:

$$\begin{aligned} y_i &= \omega_i^{(i)} \cup \omega_{ch(i)}^{(i)} \cup \bigcup_{j=i+1, j \neq ch(i)}^{m} \omega_j^{(i)} \\ y_{i+1} &= \omega_{ch(i)}^{(i+1)} \cup \bigcup_{j=i+1, j \neq ch(i)}^{m} \omega_j^{(i+1)} \end{aligned}$$

From (3.5) we obtain,

$$\omega_{ch(i)}^{(i+1)} = \omega_{ch(i)}^{(i)} \cup (\omega_i^{(i)} \cap \lambda(ch(i))) \subseteq \omega_{ch(i)}^{(i)} \cup \omega_i^{(i)} \tag{3.11}$$

and since $\omega_j^{(i+1)} = \omega_j^{(i)}$ for all $j \neq ch(i)$ we conclude that $y_{i+1} \subseteq y_i$.

Next, we prove the following property:

$$\omega_i^{(i)} \cap y_{i+1} = \omega_i^{(i)} \cap \lambda(ch(i)). \tag{3.12}$$

Assume first that $X \in \omega_i^{(i)} \cap \lambda(ch(i))$. Then, from Equation (3.11) we deduce that $X \in \omega_{ch(i)}^{(i+1)}$ and by the definition of y_{i+1}, $X \in y_{i+1}$, hence $X \in \omega_i^{(i)} \cap y_{i+1}$. On the other hand, assume that $X \in \omega_i^{(i)} \cap y_{i+1}$. Then, by the running intersection property and the definition of y_{i+1}, $X \in \lambda(ch(i))$ and therefore $X \in \omega_i^{(i)} \cap \lambda(ch(i))$.

We conclude from Equation (3.6) and (3.7) that

$$\begin{aligned} y_{i+1} &= \omega_{ch(i)}^{(i+1)} \cup \bigcup_{j=i+1, j \neq ch(i)} \omega_j^{(i+1)} \\ &\supseteq \omega_{ch(i)}^{(i)} \cup \bigcup_{j=i+1, j \neq ch(i)} \omega_j^{(i)}. \end{aligned}$$

Therefore, we can apply the combination axiom and obtain from Property (3.12):

$$\begin{aligned}\left(\bigotimes_{j=i}^{m} \psi_j^{(i)}\right)^{\downarrow y_{i+1}} &= \left(\psi_i^{(i)} \otimes \left(\psi_{ch(i)}^{(i)} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{m} \psi_j^{(i)}\right)\right)^{\downarrow y_{i+1}} \\ &= \psi_i^{(i)\downarrow \omega_i^{(i)} \cap y_{i+1}} \otimes \psi_{ch(i)}^{(i)} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{m} \psi_j^{(i)} \\ &= \psi_i^{(i)\downarrow \omega_i^{(i)} \cap \lambda(ch(i))} \otimes \psi_{ch(i)}^{(i)} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{m} \psi_j^{(i)} \\ &= \psi_{ch(i)}^{(i+1)} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{m} \psi_j^{(i+1)} \\ &= \bigotimes_{j=i+1}^{m} \psi_j^{(i+1)}\end{aligned}$$

This proves the first equality of (3.10). The second is shown by induction over i. For $i = 1$ we have

$$\left(\bigotimes_{j=1}^{m} \psi_j^{(1)}\right)^{\downarrow y_2} = \left(\bigotimes_{j=1}^{m} \psi_j\right)^{\downarrow y_2} = \phi^{\downarrow y_2}.$$

We assume that the same equation holds for i,

$$\bigotimes_{j=i}^{m} \psi_j^{(i)} = \phi^{\downarrow y_i}.$$

Then, by transitivity of marginalization,

$$\bigotimes_{j=i+1}^{m} \psi_j^{(i+1)} = \left(\bigotimes_{j=i}^{m} \psi_j^{(i)}\right)^{\downarrow y_{i+1}} = (\phi^{\downarrow y_i})^{\downarrow y_{i+1}} = \phi^{\downarrow y_{i+1}}$$

which proves (3.10) for all i. □

Theorem 1 can now be proved by applying Lemma 4, in particular (3.10) for $i = m - 1$.

Proof. We observe first that

$$y_m = \omega_m^{(m)}. \tag{3.13}$$

It remains to prove that $\omega_m^{(m)} = \lambda(m)$. For this purpose, it is sufficient to show that if $X \in \lambda(m)$ then $X \in \omega_m^{(m)}$ since $\omega_m^{(m)} \subseteq \lambda(m)$. Let $X \in \lambda(m)$. Then, according to the definition of the covering join tree for a projection problem, there exists a factor ψ_j with $X \in d(\psi_j)$. ψ_j has been assigned to node $r = a(j)$ and therefore $X \in \omega_r^{(r)}$.

The collect algorithm implies that $X \in \omega_{ch(r)}^{(r+1)}$ and by repeating this argument for each node between $r+1$ and the root m, $X \in \omega_m^{(m)}$. $\square$

The marginal $\phi^{\downarrow\lambda(m)}$ can now be used to solve the projection problem for any target domain $x \subseteq \lambda(m)$. For this purpose, it is sufficient to perform one last marginalization to the target domain x, i.e.

$$\phi^{\downarrow x} \quad = \quad \left(\phi^{\downarrow\lambda(m)}\right)^{\downarrow x}.$$

This holds by transitivity because $x \subseteq \lambda(m)$.

By removing an edge $(i, ch(i))$ from the given directed covering join tree, we obtain two parts, where the one which contains the node i is called *sub-tree rooted to node i*. A sub-tree rooted to node i is normally abbreviated by T_i. We remark that the collect theorem can also be applied for each such sub-tree T_i:

Corollary 1 *At the end of the collect algorithm, node i contains*

$$\psi_i^{(i)} \quad = \quad \left(\bigotimes_{j \in T_i} \psi_j\right)^{\downarrow\omega_i^{(i)}}. \tag{3.14}$$

Proof. Node i is the root of the sub-tree T_i. So, due to Equation (3.13), the root node i contains the marginal to $\omega_i^{(i)}$ of the factors associated to T_i. $\square$

Note that only inclusion between $\omega_i^{(i)}$ and $\lambda(i)$ holds, because we cannot guarantee that a corresponding factor for each variable in $\lambda(i)$ has been assigned to a node in the sub-tree T_i. In other words, the root node i of T_i is not necessarily filled.

The following lemma is useful for latter purposes:

Lemma 5 *It holds that*

$$\omega_i^{(m)} \cap \omega_{ch(i)}^{(m)} \quad = \quad \omega_i^{(m)} \cap \lambda(ch(i)). \tag{3.15}$$

Proof. The left part of Equation (3.15) is clearly contained in the right part, because $\omega_{ch(i)}^{(m)} \subseteq \lambda(ch(i))$. The second inclusion is derived as follows:

$$\begin{aligned}
\omega_i^{(m)} \cap \omega_{ch(i)}^{(m)} \quad &\supseteq \quad \omega_i^{(m)} \cap \omega_{ch(i)}^{(i+1)} \\
&= \quad \omega_i^{(m)} \cap \left(\omega_{ch(i)}^{(i)} \cup \left(\omega_i^{(i)} \cap \lambda(ch(i))\right)\right) \\
&= \quad \left(\omega_i^{(m)} \cap \omega_{ch(i)}^{(i)}\right) \cup \left(\omega_i^{(m)} \cap \lambda(ch(i))\right) \\
&= \quad \omega_i^{(m)} \cap \lambda(ch(i)).
\end{aligned}$$

$\square$

As already stated, at the end of the collect algorithm, the interior nodes $i < m$ are not necessarily filled, $\omega_i^{(i)} \subseteq \lambda(i)$. However, their labels can be adapted in such a way that the tree is still a join tree, but all nodes are full after the collect algorithm.

Theorem 2 *At the end of the collect algorithm executed on a join tree T with labels λ, the same tree with labels $\lambda^*(i) = \omega_i^{(m)}$ for $i = 1, \ldots, m$ is still a covering join tree for the factorization $\psi_1 \otimes \cdots \otimes \psi_m$.*

Proof. We will show that the running intersection property is still satisfied between the nodes of the newly labeled tree. Let i and j be two nodes whose reduced labels contain variable X, i.e. $X \in \lambda^*(i)$ and $X \in \lambda^*(j)$. Because T is a join tree relative to the old labels, there exists a common descendant node h, with $X \in \lambda(h)$ and $i, j \leq h$. Also since T is a join tree, we have that $X \in \lambda(ch(i))$. From

$$\lambda^*(ch(i)) \quad = \quad \omega_{ch(i)}^{(i)} \cup (\lambda^*(i) \cap \lambda(ch(i)))$$

it follows that $X \in \lambda^*(ch(i))$ and by induction $X \in \lambda^*(h)$. The same argument applies to the nodes on the path from j to h and therefore, the running intersection property holds in the newly labeled tree. $\square$

The collect algorithm is an important building block in most local computation architectures.

3.3 Shenoy-Shafer Architecture

The collect algorithm offers an adequate method to solve the projection problem efficiently. Nevertheless, a major drawback of this method is that only one single query can be answered at a time. According to the transformation described in Section 3.1, the covering join tree covers also the different target domains we are interested in. We may therefore assume that we want to compute the marginal of the factorization to *all* domains of the covering join tree. It is well-known that one could in turn select each node as a root node and repeat the collect algorithm. This causes a lot of redundant, repeated computations. But already (Shenoy & Shafer, 1990) noted that one can do much better by caching some computations. The corresponding organization is called the *Shenoy-Shafer Architecture* (SSA) and was originally developed for a factorization whose domains form a join tree. Here we show that it can be adapted to join trees covering a factorization $\psi_1 \otimes \cdots \otimes \psi_m$.

The main idea is to install *mailboxes* on each edge between two neighboring nodes of the join tree to store the messages exchanged between the nodes. A schematic representation of this concept is shown in Figure 3.1. Then, the Shenoy-Shafer algorithm can be described by the following two rules:

R1: Node i sends a message to its neighbor j, as soon as it has received all messages from its other neighbors. Leaves can send their messages right away.

R2: When node i is ready to send a message to neighbor j, it combines its initial node content with all messages from all other neighbors. The message is computed by marginalizing this result to the intersection of the result's domain and the receiving neighbor's node label.

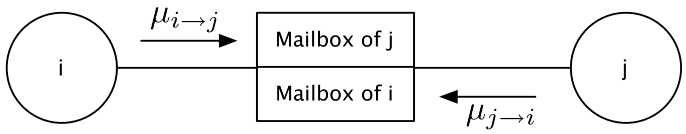

Figure 3.1: Mailboxes to store the messages exchanged by two neighboring nodes.

The algorithm stops when every node has received all messages from its neighbors.

Before a message from i to j can be computed, the domain of the valuation described in Rule R2 must be determined as

$$\omega_{i\to j} \quad = \quad \omega_i \cup \bigcup_{k\in ne(i), j\neq k} d(\mu_{k\to i}). \tag{3.16}$$

Then, the message from node i to a neighboring node j in the Shenoy-Shafer Architecture is defined as follows:

$$\mu_{i\to j} \quad = \quad \left(\psi_i \otimes \bigotimes_{k\in ne(i), j\neq k} \mu_{k\to i}\right)^{\downarrow \omega_{i\to j}\cap\lambda(j)}. \tag{3.17}$$

As in the original Shenoy-Shafer Architecture, there is always a sequence of nodes which allows to compute all the messages between all pairs of neighboring nodes. In fact, it is possible to schedule a first part of the messages in such a way that their sequence corresponds to the execution of a collect algorithm towards a root node m. This node m is the first node which has received the messages of *all* its neighbors, and the according phase of the Shenoy-Shafer Architecture is called the *collect phase*. It is easy to see that in the collect phase we have $\omega_{i\to ch(i)} = \omega_i^{(i)}$, if $\omega_i^{(i)}$ is, as in the previous Section 3.2, the domain of the valuation stored in a node i after step i of the collect algorithm towards m. The ongoing process after the collect phase is called *distribute algorithm* or *distribute phase*. Distribute starts with the root node and halts as soon as all leaves received their messages. Collect is also often called *inward propagation* and distribute *outward propagation*.

Again, we have a procedure which works without the necessity to fill the nodes of the covering join tree. The following theorem justifies the scheme underlying the Shenoy-Shafer Architecture.

Theorem 3 *At the end of the message passing in the Shenoy-Shafer Architecture, we obtain at node i*

$$\phi^{\downarrow\lambda(i)} \quad = \quad \psi_i \otimes \bigotimes_{j\in ne(i)} \mu_{j\to i}. \tag{3.18}$$

Proof. The point is that the messages $\mu_{k\to j}$ do not depend on the actual schedule used to compute them. Due to this fact, we may select node i arbitrarily as root node, direct the edges towards this root and number the nodes as in the collect algorithm. Then, the message passing corresponds to the collect algorithm, and the proposition follows for every node i from Theorem 1. Note that all nodes are filled at the end. □

The marginals to every target domain x_i can now be obtained by a further marginalization in a node j covering x_i of the join tree,

$$\phi^{\downarrow x_i} \quad = \quad \left(\phi^{\downarrow \lambda(j)}\right)^{\downarrow x_i}.$$

It is well-known that the Shenoy-Shafer Architecture may involve redundant computations in the computation of the messages, if there are more than three neighbors to a node (Shenoy, 1997) and some combinations eventually take place on larger domains than necessary (Kohlas & Shenoy, 2000). Binary join trees are trees where every node has at most three neighbors. Any join tree can be transformed into a binary one by adding additional nodes. The join tree becomes bigger, but redundant computations can be avoided. An example can be found in (Lehmann, 2001).

4 Division in Valuation Algebras

4.1 Separative Valuation Algebras

The original scheme of local computation proposed in (Lauritzen & Spiegelhalter, 1988) for discrete probability potentials involves some kind of division. This operation is not defined in general valuation algebras, but under some additional conditions, the necessary operation of (partial) division can be introduced into valuation algebras such that the procedure proposed in (Lauritzen & Spiegelhalter, 1988) can be applied. This has first been discussed in (Lauritzen & Jensen, 1997). The approach is based on well-known results from semigroup theory, which show under what conditions a semigroup can be embedded into a disjoint union of groups. However, in valuation algebras, the operation of marginalization exists too and is through the combination axiom linked to combination. In (Lauritzen & Jensen, 1997) it is tacitly assumed that all required marginals interact properly with the added operation of division. This however is not guaranteed a priori. Therefore further conditions need to be satisfied as shown in (Kohlas, 2003). The following is based on (Kohlas, 2003) and we refer to this reference for further details and the proofs of the theorems.

The sufficient conditions are collected in the following definition:

Definition 2 (Separative Valuation Algebras) *A valuation algebra (Φ, D) is called* separative*, if*

- *there is a congruence γ in (Φ, D), such that for all $\phi \in \Phi$ and $t \subseteq d(\phi)$,*

$$\phi^{\downarrow t} \otimes \phi \quad \equiv \quad \phi \pmod{\gamma}; \tag{4.1}$$

- *for all ϕ, ψ, ψ' which are contained in the equivalence class $[\phi]_\gamma$ of the congruence γ of ϕ and $\phi \otimes \psi = \phi \otimes \psi'$, we have $\psi = \psi'$.*

It can be shown, that the equivalence classes $[\phi]_\gamma$ are semigroups. So, Φ decomposes into a family of disjoint semigroups

$$\Phi = \bigcup_{\phi \in \Phi} [\phi]_\gamma.$$

Semigroups obeying the second property in the definition above are called *cancellative.*

It is known from semigroup theory, that every cancellative semigroup $[\phi]_\gamma$ which is also commutative can be embedded into a commutative group $\gamma(\phi)$ of pairs (ϕ, ψ) of elements of $[\phi]_\gamma$ (Clifford & Preston, 1967; Croisot, 1953; Tamura & Kimura, 1954). Two pairs (ϕ, ψ) and (ϕ', ψ') are identified if $\phi \otimes \psi' = \phi' \otimes \psi$, and multiplication is defined by

$$(\phi, \psi) \otimes (\phi', \psi') = (\phi \otimes \phi', \psi \otimes \psi').$$

This is similar to the construction of rational numbers from integers.

One can prove, that

$$\Phi^* = \bigcup_{\phi \in \Phi} \gamma(\phi),$$

together with the combination

$$(\phi, \psi) \otimes (\phi', \psi') = (\phi \otimes \phi', \psi \otimes \psi'),$$

defined for the elements $(\phi, \psi), (\phi', \psi') \in \Phi^*$, is a commutative semigroup and the mapping from Φ into Φ^*

$$\phi \mapsto (\phi \otimes \phi, \phi)$$

is a semigroup embedding. We thereby identify usually $\phi \in \Phi$ with $(\phi \otimes \phi, \phi)$ in Φ^*.

Every group $\gamma(\phi)$ has an identity element $f_{\gamma(\phi)}$. It belongs not necessarily to Φ, but only to Φ^*. These identity elements satisfy

$$f_{\gamma(\phi)} \otimes f_{\gamma(\psi)} = f_{\gamma(\phi \otimes \psi)}.$$

A partial order between the groups $\gamma(\phi)$ is defined as follows:

$$\gamma(\psi) \leq \gamma(\phi) \text{ if, and only if, } f_{\gamma(\phi)} \otimes f_{\gamma(\psi)} = f_{\gamma(\phi)}.$$

Since $f_{\gamma(\phi)} \otimes f_{\gamma(\phi \otimes \psi)} = f_{\gamma(\phi \otimes \psi)} = f_{\gamma(\psi)} \otimes f_{\gamma(\phi \otimes \psi)}$, it follows that $\gamma(\phi \otimes \psi)$ is an upper bound of $\gamma(\phi)$ and $\gamma(\psi)$. For any other upper bound $\gamma(\eta)$ of both $\gamma(\phi), \gamma(\psi)$, we deduce easily that $f_{\gamma(\eta)} \otimes f_{\gamma(\phi \otimes \psi)} = f_{\gamma(\eta)}$, that is, $\gamma(\phi \otimes \psi)$ is the least upper bound of $\gamma(\phi), \gamma(\psi)$,

$$\gamma(\phi \otimes \psi) \quad = \quad \gamma(\phi) \vee \gamma(\psi).$$

This shows that the groups form a semilattice. Further, every element $\phi \in \Phi$ belongs to some group $\gamma(\phi)$, and in this group it has an inverse ϕ^{-1}, such that

$$\phi \otimes \phi^{-1} \quad = \quad f_{\gamma(\phi)}.$$

The following properties are proved in (Kohlas, 2003):

Lemma 6

1. *If $\gamma(\psi) \leq \gamma(\phi)$, then $\phi' \otimes f_{\gamma(\psi)} = \phi'$ for all $\phi' \in \gamma(\phi)$.*
2. *$\gamma(\phi^{\downarrow t}) \leq \gamma(\phi)$ for all $t \subseteq d(\phi)$.*
3. *For all $\phi, \psi \in \Phi$ it holds that $\gamma(\phi) \leq \gamma(\phi \otimes \psi)$.*
4. *$(\phi \otimes \psi)^{-1} = \phi^{-1} \otimes \psi^{-1}$.*

The following two examples serve to illustrate the situation:

4.1.1 Gaussian Potentials

The semigroup of Gaussian potentials on a fixed domain is clearly cancellative. So, we may consider potentials with the same domain as equivalent. This is the congruence required in the definition of separative valuation algebras. Note that Gaussian potentials correspond to Gaussian density functions. Therefore, the embedding semigroup Φ^* consists essentially of quotients of Gaussian densities, which themselves are no more Gaussian densities. The identity element on a domain s is the identity function $f(\mathbf{x}) = 1$ for all configuration $\mathbf{x}$ of s. It is not itself a Gaussian density. The quotient $g(\mathbf{x})/g^{\downarrow t}(\mathbf{x}^{\downarrow t})$ of a Gaussian density g for example represents a family of conditional Gaussian densities and is a member of Φ^*. So, embedding a separative valuation algebra into a larger semigroup is not just a formal construction, but may well have a significant meaning.

4.1.2 Densities

General continuous densities are a generalization of Gaussian densities or potentials. But here, the situation is a bit more involved. For a density f on a set s (see Example 2.2.2) we define the support as the set

$$supp(f) \quad = \quad \{\mathbf{x} : f(\mathbf{x}) \neq 0\}.$$

We say that two densities f and g are equivalent, $f \equiv g$, if $supp(f) = supp(g)$. Since for continuous densities $f^{\downarrow t}(\mathbf{x}^{\downarrow t}) = 0$ implies $f(\mathbf{x}) = 0$, we have that $f \otimes f^{\downarrow t} \equiv f$. The semigroup of densities on the same support is clearly cancellative. So, the valuation algebra of continuous densities is *separative*. It is, similar to Gaussian densities, embedded in the semigroup of quotients of densities. For any density f the group $\gamma(f)$ consists of quotients of densities with support equal to $supp(f)$. The identity of the group $\gamma(f)$ is the function which is identical 1 on $supp(f)$ and zero outside $supp(f)$. The order between these groups is defined as $\gamma(f) \leq \gamma(g)$ if, and only if, $d(f) \subseteq d(g)$ and $supp(f)^{\uparrow d(g)} \supseteq supp(g)$, where $\uparrow$ denotes the cylindric extension. Combination corresponds then essentially to the intersection of support sets. More precisely, if $d(f) = s$ and $d(g) = t$, then $supp(f \otimes g) = supp(f)^{\uparrow s \cup t} \cap supp(g)^{\uparrow s \cup t}$. The inverse of a density $f(\mathbf{x})$ is the function $1/f(\mathbf{x})$ defined on $supp(f)$ and zero outside $supp(f)$. Further, let f be a density on s and g a density on t, and let $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ be configurations of $s - t$, $s \cap t$ and $t - s$ respectively. Then, $f \otimes g^{-1}$ can be identified with the quotient $f(\mathbf{x},\mathbf{y})/g(\mathbf{y},\mathbf{z})$ defined on $supp(f)^{\uparrow s \cup t} \cap supp(g)^{\uparrow s \cup t}$, and zero outside this set. A special case of such a combination is $f \otimes (f^{\downarrow t})^{-1}$ for a density on s and $t \subseteq s$. The corresponding quotient is

$$\frac{f(\mathbf{x},\mathbf{y})}{\int f(\mathbf{x},\mathbf{y})d\mathbf{x}} \tag{4.2}$$

on $supp(f)$ and zero otherwise. This represents a family of conditional densities, one for each value of $\mathbf{y}$.

Reconsidering general separative algebras, labeling can be extended from the separative valuation algebra Φ to its extension Φ^*. If $\psi \in \Phi^*$ belongs to a group $\gamma(\phi)$ for some $\phi \in \Phi$, then define $d(\psi) = d(\phi)$. This definition is unambiguous, since all valuations ϕ in a group $\gamma(\phi)$ have the same domain.

Marginalization is not necessarily defined for elements in Φ^*, which do not belong to Φ. As an example we refer to the conditional densities introduced in the example above. Integration is only possible over the variable $\mathbf{x}$, but not over $\mathbf{x}$ and $\mathbf{y}$ together. Thus, marginalization can only partially be extended from Φ to Φ^*. We refer to (Kohlas, 2003) for a detailed description how marginalization can be extended to Φ^*. With this extension, Φ^* becomes a *valuation algebra with partial marginalization* (Kohlas, 2003). This means, that for each $\psi \in \Phi^*$ there is a subset $\mathcal{M}(\psi) \subseteq D$ for which marginalization is defined. For a valuation algebra with partial marginalization axioms (A3) to (A5) become now:

(A3') *Marginalization:* For $\phi \in \Phi$ and $x \in \mathcal{M}(\phi)$,

$$d(\phi^{\downarrow x}) \quad = \quad x.$$

(A4') *Transitivity:* If $\phi \in \Phi$ and $x \subseteq y \subseteq d(\phi)$, then $x \in \mathcal{M}(\phi)$ implies $x \in \mathcal{M}(\phi^{\downarrow y}) \wedge y \in \mathcal{M}(\phi)$ and

$$(\phi^{\downarrow y})^{\downarrow x} \quad = \quad \phi^{\downarrow x}.$$

(A5') *Combination:* If $\phi, \psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$, then $z \cap y \in \mathcal{M}(\psi)$ implies $z \in \mathcal{M}(\phi \otimes \psi)$ and

$$(\phi \otimes \psi)^{\downarrow z} \quad = \quad \phi \otimes \psi^{\downarrow z \cap y}.$$

Sometimes, the semilattice of subgroups $\gamma(\phi)$ with domain x has a *minimal element*. Let us call it γ_x. The elements of group γ_x are called *positive*. Denote the identity element of γ_x by e_x. Thus every sub-semigroup

$$\Phi^*_x \quad = \quad \bigcup_{\phi \in \Phi_x} \gamma(\phi)$$

has a neutral element. Then, $e_x \otimes \psi = \psi$ for all $\psi \in \Phi^*_x$. Furthermore, it can be shown that $e_x \otimes e_y = e_{x \cup y}$ (Kohlas, 2003). Thus, in this case the valuation algebra (Φ^*, D) with partial marginalization has neutral elements, and as in Section 2.3, the element $e_\emptyset$ is what we need in local computation in covering join trees. Both Gaussian potentials as well as continuous densities form separative valuations with positive elements and dispose of $e_\emptyset \in \Phi^*$.

If those neutral elements do not exist, we may adjoin a neutral element without loosing separativity. First, we extend the congruence relation γ in (Φ, D) to a congruence relation γ' in (Φ', D). We say that $\phi \equiv \psi \pmod{\gamma'}$ if either

- $\phi, \psi \in \Phi$ *and* $\phi \equiv \psi \pmod{\gamma}$ or
- $\phi = \psi = e$.

It is clear that this is still a congruence, if it is one in (Φ, D). The equivalence class $[e]_{\gamma'}$ consists of the single element e, and the extended algebra (Φ', D) essentially inherits the separativity from (Φ, D).

Lemma 7 *If (Φ, D) is a separative valuation algebra according to the congruence γ, then so is (Φ', D) according to γ' and the extended operators d', $\downarrow'$ and $\otimes'$.*

Proof. We have seen in the Lemma 2 that (Φ', D) is a valuation algebra with the operators d', $\downarrow'$ and $\otimes'$. Further, we have seen above that γ' is a congruence in (Φ', D). It remains to shown that γ' obeys the properties required for separativity, see definition 2. For $\phi \in \Phi$ and $t \subseteq d(\phi)$ the congruence γ induces $\phi^{\downarrow t} \otimes \phi \equiv \phi \pmod{\gamma'}$. For e we get the desired result from $e^{\downarrow \emptyset} \otimes e = e$ and reflexivity of γ',

$$e^{\downarrow \emptyset} \otimes e \quad \equiv \quad e \pmod{\gamma'}.$$

Cancellativity of $[\phi]_{\gamma'}$ for $\phi \in \Phi$ is again implied by γ. Since $[e]_{\gamma'}$ consists of the single element e, cancellativity of $[e]_{\gamma'}$ is trivial. □

It remains to show that in this case, the induced valuation algebra with partial marginalization (Φ^*, D) inherits a neutral element too. In fact, we prove that e is a neutral element of Φ^*.

Lemma 8 *Let (Φ', D) be a separative valuation algebra with an unique identity element e. Then, e is a neutral element in the valuation algebra (Φ^*, D) induced by (Φ', D).*

Proof. The embedding of e into Φ^* is $e^* = (e \otimes e, e) = (e, e)$. We verify the properties imposed on e^*. We have for $\eta = (\phi, \psi) \in \Phi^*$ with $\phi, \psi \in \Phi$

$$\begin{array}{rcl} d(e^*) &=& d(e) \;=\; \emptyset; \\ \eta \otimes e^* &=& (\phi, \psi) \otimes (e, e) \;=\; (\phi \otimes e, \psi \otimes e) \;=\; (\phi, \psi) \;=\; \eta; \\ e^* \otimes e^* &=& (e, e) \otimes (e, e) \;=\; (e \otimes e, e \otimes e) \;=\; (e, e) \;=\; e^* \end{array}$$

and by the domain axiom $(e^*)^{\downarrow\emptyset} = e^*$. □

This completes our short overview of separative valuation algebras. Below, it will be shown that local computation in covering join trees as defined in Section 3 can be applied to valuation algebras with partial marginalization too, and in particular, the architectures using division like those proposed in (Lauritzen & Spiegelhalter, 1988) for discrete probability potentials and others can be used with these valuation algebras.

4.2 Regular Valuation Algebras

An important particular case of a separative algebra is provided by so called *regular valuation algebras*. We require some more structure such that a specific congruence relation exists which induces directly groups and not, as before, semigroups. This makes life easier, since we dispose of full marginalization. Nevertheless, the approach is less general and does, for example, not cover the examples given in the previous Section 4.1. Here is the definition of regular algebras, which is motivated by Croisot's theory of semigroups with inverses (Croisot, 1953), but extended to valuation algebras.

Definition 3 (Regularity)

- *An element $\phi \in \Phi$ is called* regular*, if there exists for all $t \subseteq d(\phi)$ an element $\chi \in \Phi$ with $d(\chi) = t$, such that*

$$\phi \;=\; \phi^{\downarrow t} \otimes \chi \otimes \phi.$$

- *A valuation algebra (Φ, D) is called* regular*, if all its elements are regular.*

The Green relation in a semigroup is defined by

$$\phi \;\equiv\; \psi \pmod{\gamma} \quad \text{if} \quad \phi \otimes \Phi \;=\; \psi \otimes \Phi.$$

Here, $\phi \otimes \Phi$ denotes the set $\{\phi \otimes \eta : \eta \in \Phi\}$, i.e. the principal ideal in Φ generated by ϕ. It is a congruence relation in a regular algebra (Kohlas, 2003). Note that $\phi \otimes \Phi = (\phi \otimes \phi^{\downarrow t}) \otimes \Phi$ for every $t \subseteq d(\phi)$, hence $\phi \equiv \phi \otimes \phi^{\downarrow t} \pmod{\gamma}$. This is implied by regularity because $\phi \otimes \eta = (\phi \otimes \phi^{\downarrow t}) \otimes \chi \otimes \eta$. So, the Green relation in a regular valuation algebra satisfies the first condition of separativity. Further, the equivalence classes $[\phi]_\gamma$ are already groups themselves (Kohlas, 2003). They are cancellative and therefore regular algebras are also separative algebras. But in this particular case Φ itself decomposes into disjoint groups

$$\Phi = \bigcup_{\phi \in \Phi} [\phi]_\gamma.$$

Consequently, marginalization is defined within all groups fully and not only partially as with separative algebras in general.

There are many examples of regular valuation algebras:

4.2.1 Discrete Probability Potentials

This is the prototype example of a regular valuation algebra. We have by definition of combination for a probability potential on s and $t \subseteq s$,

$$p^{\downarrow t} \otimes \chi \otimes p(\mathbf{x}) = p^{\downarrow t}(\mathbf{x}^{\downarrow t}) \cdot \chi(\mathbf{x}^{\downarrow t}) \cdot p(\mathbf{x}).$$

So, we may define

$$\chi(\mathbf{x}) = \begin{cases} \frac{1}{p^{\downarrow t}(\mathbf{x}^{\downarrow t})}, & \text{if } p^{\downarrow t}(\mathbf{x}^{\downarrow t}) > 0, \\ \text{arbitrary}, & \text{otherwise.} \end{cases}$$

Since $p^{\downarrow t}(\mathbf{x}^{\downarrow t}) = 0$ implies $p(\mathbf{x}) = 0$, this is a solution to the regularity equation. Hence, all probability potentials are regular. Similar to continuous densities, the groups are given by quotients of discrete potentials p with the same support $supp(p) = \{\mathbf{x} : p(\mathbf{x}) > 0\}$. Contrary to continuous densities, quotient of potentials with the same support are again discrete probability densities, in particular, the inverse p^{-1} of a potential p is again a potential.

4.2.2 Information Algebras

Many valuation algebras are *idempotent*. This means that for all $\phi \in \Phi$ and $t \subseteq d(\phi)$ it holds that

$$\phi \otimes \phi^{\downarrow t} = \phi.$$

This is a typical property of information: Adding to a piece of information a part of itself gives nothing new. Therefore, idempotent valuation algebras are also called *information algebras* (Kohlas, 2003). Examples of information algebras are relational algebra and valuation algebras related to propositional or predicate logic (cylindric algebras, see (Henkin *et al.*, 1971)). Idempotent algebras are clearly regular: take for example $\chi = e$ in the regularity example. They are regular in a trivial way, the Green relation gives equivalence classes $[\phi]_\gamma = \{\phi\}$ consisting of single elements, since there is only one idempotent per group. Each valuation is the inverse of itself. Nevertheless, local computation architecture with division can be applied to information algebras. In fact, these architectures collapse to some very simple form as we see in Section 5.3.

4.2.3 Regular Semirings

A semiring is called regular, if the semigroup of the operation $\times$ is regular, i.e. if for all $a \in A$ there is a $b \in A$ such that

$$a \times b \times a = a.$$

A semiring is called positive, if it has a neutral element 0 for the operation $+$ and if $a + b = 0$ implies $a = 0$. A positive, regular semiring induces a valuation algebra as described in Example 2.2.1, which is regular (Kohlas & Wilson, 2008). For example, the semiring of nonnegative reals with $+$ and $\times$ as ordinary addition and multiplication is regular (and the induced valuation algebra corresponds to discrete probability potentials). Most of the t-norms are not regular, and so the corresponding possibility potentials are not regular. However the product t-norm is regular, and so is the corresponding possibility potential.

The identity element e can be adjoined without changing the regularity of the valuation algebra. But, as with separative algebras in general, we may have regular algebras which already have neutral elements.

Lemma 9 *If (Φ, D) is a regular valuation algebra, then so is (Φ', D) with the extended operators d', $\downarrow'$ and $\otimes'$.*

Proof. Lemma 2 shows that (Φ', D) is a valuation algebra with the operators d', $\downarrow'$ and $\otimes'$. All elements in Φ are regular. And so is e, since $e = e^{\downarrow\emptyset} \otimes \chi \otimes e$ with $\chi = e$. □

As for separative algebras, this permits to adapt local computation architectures with division to covering join trees.

5 Architectures using Division

5.1 Lauritzen-Spiegelhalter Architecture

By *Lauritzen-Spiegelhalter Architecture* (LSA) we design a local computation method proposed in (Lauritzen & Spiegelhalter, 1988) for discrete probability potentials. Here we show that it can be applied to separative valuation algebras in general, extending thus its domain of applicability considerably. As always we consider a covering join tree of a factorization

$$\phi = \bigotimes_{j=1}^{m} \psi_j \qquad (5.1)$$

such that $\psi_j \subseteq \lambda(j)$ for all nodes j of the join tree. Here we assume that ϕ belongs to Φ, whereas the factors ψ_j belong to Φ^*. This means that all marginals of the combination are well defined, but the factors ψ_j may only have partial marginals.

Again, we assume the nodes of the join tree numbered such that $i < j$ if j is on the path of i to the root m. The LSA consists of an execution of the collect algorithm towards the root node m. The messages $\mu_{i \to ch(i)}$ sent of a node i towards its child $ch(i)$ is defined in (3.4). But in contrast to the collect algorithm described in Section 3.2, each node i divides its message $\mu_{i \to ch(i)}$ out of the valuation stored at the node. Let us denote the content of the store of node i by η_i. Before sending the message to its child, this node content is $\eta_i = \psi_i^{(i)}$ according to the collect algorithm. After sending the message, it is changed to

$$\eta_i \quad := \quad \psi_i^{(i)} \otimes \mu_{i \to ch(i)}^{-1}.$$

This facilitates somewhat the distribute algorithm. In fact, this second phase starts, when the collect phase terminates. Node i sends its message

$$\mu_{i \to j} \quad = \quad \eta_i^{\downarrow \lambda(i) \cap \lambda(j)}$$

to all its neighbors $j \neq ch(i)$ (to its parents), once it has received its message from its unique child $ch(i)$. The receiving node combines this message to its current valuation

$$\eta_j \quad := \quad \eta_j \otimes \mu_{i \to j}.$$

The distribute phase starts with the root node m sending its messages outwards.

The following theorem claims that this procedure ends with the marginal of the factorization to each node domain in the store of each node. This is true provided that the required marginals all exist. For regular valuation algebras this is guaranteed. For separative ones, where only partial marginalization is possible, sufficient conditions for this are given below. First, we assume that all messages occurring in the SSA exist and show that then LSA gives the correct result.

Theorem 4 *Assume that all SSA messages for the factorization and the covering join tree exist. Then, all messages for the LSA exist and, at the end of the LSA, each node $i \in V$ contains $\phi^{\downarrow \lambda(i)}$.*

Proof. Let μ' denote the messages during an execution of *SSA* for the given factorization and covering join tree,

$$\mu'_{j \to i} \quad = \quad \left(\psi_j \otimes \bigotimes_{k \in ne(j), k \neq i} \mu'_{k \to j} \right)^{\downarrow \omega_{j \to i} \cap \lambda(i)}.$$

We assume that all these messages exist. Now, for the LSA we have that $\mu_{i \to j} = \mu'_{i \to j}$ during inward propagation and the messages in LSA exist too in the collect phase. Further, the theorem is correct by the collect algorithm for the root node m, see Theorem 1.

We prove that it is correct for all nodes by induction over the outward propagation phase using the correctness of SSA. The outward propagation phase is scheduled in the reverse numbering of the nodes. When a node $j > i$ is ready to send a message towards i, node i stores

$$\psi_i \otimes (\mu'_{i\to j})^{-1} \otimes \bigotimes_{k\in ne(i),k\neq j} \mu'_{k\to i}. \tag{5.2}$$

By the induction hypothesis, the sending node j stores $\phi^{\downarrow\lambda(j)}$. Hence, the messages $\mu_{j\to i} = \phi^{\downarrow\lambda(j)\cap\lambda(i)}$ occurring in the distribute phase do exist. Then, since the SSA messages exist too, by the correctness of SSA, Theorem 3, and the combination axiom, we obtain

$$\begin{aligned}
\mu_{j\to i} &= \phi^{\downarrow\lambda(j)\cap\lambda(i)} \\
&= \left(\psi_j \otimes \bigotimes_{k\in ne(j)} \mu'_{k\to j}\right)^{\downarrow\lambda(j)\cap\lambda(i)} \\
&= \left(\psi_j \otimes \bigotimes_{k\in ne(j),k\neq i} \mu'_{k\to j}\right)^{\downarrow\omega_{j\to i}\cap\lambda(i)} \otimes \mu'_{i\to j} \\
&= \mu'_{j\to i} \otimes \mu'_{i\to j} \;=\; \mu'_{j\to i} \otimes \mu_{i\to j}.
\end{aligned} \tag{5.3}$$

So, we obtain at node i, when we combine the incoming message $\mu_{j\to i}$ to its actual content and use Theorem 3,

$$\psi_i \otimes \bigotimes_{k\in ne(i),k\neq j} \mu'_{k\to i} \otimes (\mu'_{i\to j})^{-1} \otimes \mu'_{j\to i} \otimes \mu'_{i\to j} \;=\; \phi^{\downarrow\lambda(i)} \otimes f_{\gamma(\mu'_{i\to j})}.$$

But by Lemma 6 we obtain

$$\gamma(\mu'_{i\to j}) \;\leq\; \gamma(\mu'_{j\to i} \otimes \mu'_{i\to j}) \;=\; \gamma(\phi^{\downarrow\lambda(j)\cap\lambda(i)}) \;\leq\; \gamma(\phi^{\downarrow\lambda(i)})$$

and the theorem is proved. □

The proof is based on the messages used in the SSA. If they exist, the LSA works and gives correct results. It is therefore interesting to examine the messages in the SSA. We first show how the original factorization can be changed without affecting its marginals, but in such a way that the working of the collect algorithm guarantees the existence of all SSA messages. This will then be sufficient for the working of the whole LSA too.

Lemma 10 *Let ϕ be defined by (5.1). Then, for all $i = 1,\ldots,m-1$,*

$$\phi \;=\; \phi \otimes f_{\gamma(\mu_{i\to ch(i)})},$$

if the messages $\mu_{i\to ch(i)}$ exist.

Proof. For any $i = 1,\ldots,m-1$, by Corollary 1,

$$\mu_{i\to ch(i)} \;=\; \left(\bigotimes_{k\in T_i} \psi_k\right)^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i))}$$

where T_i is the sub-tree rooted to node i. By Lemma 6 it follows further that

$$\gamma(\mu_{i \to ch(i)}) \;\leq\; \gamma(\otimes_{k \in T_i} \psi_k) \;\leq\; \gamma(\phi)$$

and this is sufficient for $\phi = \phi \otimes f_{\gamma(\mu_{i \to ch(i)})}$. □

Define now

$$\psi'_i \;=\; \psi \otimes \bigotimes_{j:ch(j)=i} f_{\gamma(\mu_{j \to i})}.$$

Then, according to the theorem just proved,

$$\phi \;=\; \bigotimes_{i=1}^{m} \psi'_i.$$

So, the adding of the new identity elements as factors does not change the value of the original factorization. Also, at the end of the collect algorithm with the new factorization, we have the same valuations stored in the nodes $i = 1, \ldots, m$.

Lemma 11 *A run of the collect algorithm with the valuations $\psi_1, \psi_2, \ldots, \psi_m$ assigned to the nodes of the given join tree towards a root node m ends with the same node stores at the end as a run of the collect algorithm with the new assignments $\psi'_1, \psi'_2, \ldots, \psi'_m$.*

Proof. In the collect algorithm with the new assignments every message $\mu_{i \to ch(i)}$ meets its neutral element $f_{\gamma(\mu_{i \to ch(i)})}$ in the store of the node $ch(i)$ by construction. Every $f_{\gamma(\mu_{i \to ch(i)})}$ is therefore absorbed by Lemma 6 during the algorithm. □

A direct consequence of this lemma is that the messages are in both cases the same. If the collect algorithm can be executed for the original factorization, then so it can in its changed version. But, what is more, in this case the whole of SSA works with the new assignment.

Lemma 12 *If the collect algorithm works with the valuations $\psi_1, \psi_2, \ldots, , \psi_m$ assigned to the nodes of the given join tree towards a root node m, then so does the* complete *SSA with the new assignments $\psi'_1, \psi'_2, \ldots, \psi'_m$.*

Proof. By Lemma 11, all messages during the inward propagation phase of SSA towards the root node m exist with respect to the new factorization ψ'. It remains to show that the outward SSA messages exist too. The root node m is the first node which is ready to send messages towards its parents in the outward propagation phase. Let j be such a neighbor. Then $\mu_{m \to j}$ is a marginal, if it exists, of

$$\begin{aligned}
\psi'_m \otimes \bigotimes_{k \in ne(m), k \neq j} \mu_{k \to m} &= \psi_m \otimes \bigotimes_{k \in pa(m)} f_{\gamma(\mu_{k \to m})} \otimes \bigotimes_{k \in ne(m), k \neq j} \mu_{k \to m} \\
&= \psi_m \otimes f_{\gamma(\mu_{j \to m})} \otimes \bigotimes_{k \in ne(m), k \neq j} \mu_{k \to m} \\
&= \psi_m \otimes \mu_{j \to m} \otimes \mu_{j \to m}^{-1} \otimes \bigotimes_{k \in ne(m), k \neq j} \mu_{k \to m} \\
&= \phi^{\downarrow \lambda(m)} \otimes \mu_{j \to m}^{-1}. \qquad (5.4)
\end{aligned}$$

From the definition of $\omega_{m\to j}$, see Equation (3.16), we obtain using the labeling axiom

$$\begin{aligned}\omega_{m\to j} &= d\left(\psi'_m\right) \cup \bigcup_{k\in ne(m),k\neq j} d(\mu_{k\to m}) \\ &= d\left(\psi'_m \otimes \bigotimes_{k\in ne(m),k\neq j} \mu_{k\to m}\right) \\ &= d\left(\phi^{\downarrow\lambda(m)} \otimes \mu^{-1}_{j\to m}\right) = d\left(\phi^{\downarrow\lambda(m)}\right) = \lambda(m).\end{aligned}$$

Therefore, from the combination axiom and Equation (5.4), it follows that

$$\begin{aligned}\phi^{\downarrow\lambda(m)\cap\lambda(j)} \otimes \mu^{-1}_{j\to m} &= \left(\phi^{\downarrow\lambda(m)} \otimes \mu^{-1}_{j\to m}\right)^{\downarrow\lambda(m)\cap\lambda(j)} \\ &= \left(\phi^{\downarrow\lambda(m)} \otimes \mu^{-1}_{j\to m}\right)^{\downarrow\omega_{m\to j}\cap\lambda(j)}\end{aligned}$$

But the last term defines $\mu_{m\to j}$ and $\mu_{m\to j}$ therefore exists. All messages from the neighbors to the node j are thus defined. We may now select node j as a new root node for a collect phase. So, the procedure above can be applied to j in order to prove that the messages sent towards its parents exist too. We conclude by induction that the whole SSA can be executed. □

The definition of the new valuations ψ' is dependent on the selection of the root node m. The last lemma implies now that it is possible to execute inward propagation towards any node i and not only towards m. So, finally, if only the collect algorithm with respect to a certain root node can be executed, it can be executed towards any node of the join tree. All SSA messages will then exist and by Theorem 4, LSA works for any node as root node in the join tree. The question arises if there are factorizations of a valuation whose factors have only partially defined marginals, but such that nevertheless the messages during the collect algorithm exist. The answer is affirmative but goes beyond the scope of this paper. We refer to Bayesian networks as an example (Cowell *et al.*, 1999; Kohlas, 2003).

5.2 HUGIN Architecture

There is a modification of the LSA which postpones division from the collect phase to the outward propagation and which also carries out division on smaller domains than LSA. This architecture is called HUGIN according to a software for Bayesian networks (Jensen *et al.*, 1990). A covering join tree for a factorization like (5.1) can be extended by adding a new node between i and j on any edge $\{i, j\}$ of the original join tree and associating the label $\lambda(i) \cap \lambda(j)$. These nodes are called *separators.* Let's denote the separator between i and $ch(i)$ by $\sigma(i)$. Clearly, the extended tree is still a join tree, i.e. the running intersection property holds.

Inward propagation is exactly like the collect algorithm, but we store every message $\mu_{i\to j}$ in the *separator* situated between the neighboring nodes i and j. The separators can actually be seen as a passive memory. In the following outward propagation phase, the messages are computed like in the outward phase of the LSA. But they have to pass through the separator lying in-between the sending and receiving node. The separator becomes activated by the crossing message and holds it back in order to divide out its current content. Finally, the mutated message is send towards the destination and the *original* incoming message stored in the separator. Formally, let i and j be two neighboring nodes where i sends the message

$$\mu_{i\to j} \quad = \quad \left(\psi_i \otimes \bigotimes_{k\in pa(i)} \mu_{k\to i}\right)^{\downarrow\omega_i\cap\lambda(j)}$$

towards j in the inward propagation phase. This message is stored in the separator. In the outward propagation phase node j sends the message

$$\mu'_{j\to i} \quad = \quad \left(\psi_j \otimes \bigotimes_{k\in ne(i)} \mu_{k\to j}\right)^{\downarrow\lambda(i)\cap\lambda(j)}$$

towards i. In the separator between them, this message is changed to $\mu_{j\to i} = \mu'_{j\to i} \otimes \mu_{i\to j}^{-1}$. The message $\mu_{j\to i}$ is finally combined to the store of the node i. The advantage over the LSA is that the divisions are performed in the separators which usually have smaller labels than the nodes.

Theorem 5 *Assume that for an original factorization (5.1) and a corresponding covering join tree, the collect algorithm works. Then, at the end of the computations in the HUGIN Architecture, each node $i \in V$ stores $\phi^{\downarrow\lambda(i)}$ and every separator $j \in S$ the marginal $\phi^{\downarrow\sigma(j)}$.*

Proof. The proof is based on the correctness of the LSA. First, let us introduce the separators between the nodes in the given join tree as real nodes. Let V' denote the set containing the newly introduced nodes. Putting identity elements e on V' gives a new factorization whose value is still ϕ. We then execute the LSA on the altered tree.

Fix a node $i \notin V'$. It sends a message $\mu_{i\to j}$ in the inward propagation phase of the LSA and divides it afterwards out of its actual content which we abbreviate with η_i. By construction, the receiving node $j = ch(i)$ is an element of V'. The node i contains $\eta_i \otimes (\mu_{i\to j})^{-1}$ and j stores $e \otimes \mu_{i\to j} = \mu_{i\to j}$ after this step. Then, the node j is ready to send a message towards the node $k = ch(j)$. But we clearly have $\mu_{i\to j} = \mu_{j\to k}$. Since every emitted message is divided out of the store of the sending node, we get the updated content

$$\mu_{i\to j} \otimes (\mu_{j\to k})^{-1} \quad = \quad f_{\gamma(\mu_{j\to k})}$$

at j. The LSA is then continued and we assume that the node k is now ready to send a message during the outward propagation phase towards j. We know by the correctness of the LSA that this message equals $\phi^{\downarrow\lambda(k)\cap\lambda(j)}$. This is also the new store of j, see Equation (5.3) and apply Lemma 6. The message sent from j towards i is finally $\phi^{\downarrow\lambda(j)\cap\lambda(i)} = \phi^{\downarrow\lambda(k)\cap\lambda(i)}$ such that we get there $\eta_i \otimes (\mu_{i\to j})^{-1} \otimes \phi^{\downarrow\lambda(i)\cap\lambda(k)} = \phi^{\downarrow\lambda(i)}$. The last equality follows from the correctness of LSA, see Theorem 4. But

$$(\mu_{i\to j})^{-1} \otimes \phi^{\downarrow\lambda(i)\cap\lambda(k)}$$

is the message from k towards i in the HUGIN architecture on the original tree which passed already through the separator j. □

As LSA, the HUGIN Architecture works thus in regular valuation algebras. It works in particular in information algebras, where it can be considerably simplified, see Section 5.3. It works in separative algebras, provided a working collect algorithm for some root node, i.e. provided all marginals necessary for this method exist. Then, it works with all root nodes. A sufficient condition for this, based on a generalization of Shafer's concept of construction sequences (Shafer, 1996) is given in (Kohlas, 2003).

5.3 Local Computation in Information Algebras

In Section 4.2 idempotent valuation algebras, called information algebras, were identified as regular algebras. Idempotency implies that each element is its own inverse, $\phi^{-1} = \phi$. Therefore, it is possible to apply the LSA and the HUGIN architectures. As usual we consider a factorization (5.1) and a corresponding covering join tree.

The messages sent during the inward propagation phase do not change. However, division has not to be carried out, because every valuation is its own inverse. This means for the LSA that the message is combined to the current content of the sending node, since it is a marginal of the current content. By idempotency this has no effect. If $\eta_i = \psi_i^{(i)}$ is the store content at node i at step i of the collect algorithm, the message sent to $j = ch(i)$ is

$$\mu_{i\to j} \quad = \quad \eta_i^{\downarrow\omega_i^{(i)}\cap\lambda(j)}.$$

In the outward phase node j sends the usual LSA message to its parents i,

$$\mu_{j\to i} \quad = \quad \eta_i^{\downarrow\lambda(j)\cap\lambda(i)}.$$

This is a very simple, uniform and symmetric procedure.

A similar argument is valid for the HUGIN Architecture, where every message emitted from j to i during the outward propagation phase equals $\phi^{\downarrow\lambda(i)\cap\lambda(j)}$ and meets $\mu_{i\to j}$ in the separator. But by idempotency we have $\mu_{i\to j} \otimes \phi^{\downarrow\lambda(i)\cap\lambda(j)} = \phi^{\downarrow\lambda(i)\cap\lambda(j)}$, see Equation (5.3). The valuations in the separators have no effect to the messages passing through them. So, essentially HUGIN is like LSA, except that separator nodes have been added to store the inward messages.

This demonstrates that idempotency is a very strong condition. Besides simplifying local computation, it has other profound consequences, essentially because idempotency allows to define a partial order between elements (which represent pieces of information), such that the semigroup of the valuation algebra becomes in fact a semilattice, see (Kohlas, 2003).

6 Conclusion

This paper shows that local computation can be performed on covering join trees without filling the nodes artificially with neutral elements. Not only extends this the applicability of local computation architectures to valuation algebras without neutral elements, but increases also the efficiency of the architectures, even if neutral elements do exist. The basic idea is to adjoin an identity element, if it does not yet exists. This allows then to modify all known local computation architectures for covering join trees of a factorization. It is also shown that local computation architectures proposed for discrete probability potentials, especially Bayesian networks, can be generalized to separative valuation algebras. The result is a truly generic theory of local computation. This theory can and has been implemented in a fully *generic software system* called NENOK (Pouly, 2008; Pouly, 2010), see also `http://marcpouly.ch/nenok`. It offers generic implementations of the local computation architectures SSA, LSA, HUGIN and idempotent architectures together with many other tools for local computation purposes. User can instantiate their own valuation algebra and access the generic local computation library of NENOK. This is thought as a research platform for rapid prototyping of local computation procedures for concrete valuation structures.

Acknowledgments

This research was supported by the grants No. 20-67996.02 and PBFRP2-124190 of the Swiss National Foundation for Research.

References

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

Cano, J. E., Moral, S., & Verdegay, J. F. 1992. Propagation of convex sets of probabilities in directed acyclic networks. *Pages 289–292 of: Proceedings of the 4th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '92).*

Chekuri, Ch., & Rajaraman, A. 1997. Conjunctive Query Containment Revisited. *Pages 56–70 of: ICDT '97: Proceedings of the 6th International Conference on Database Theory.* London, UK: Springer-Verlag.

Clifford, A. H., & Preston, G. B. 1967. *Algebraic Theory of Semigroups.* Providence, Rhode Island: American Mathematical Society.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. 1999. *Probabilistic Networks and Expert Systems.* Information Sci. and Stats. Springer, New York.

Croisot, R. 1953. Demi-groupes inversifs et demi-groupes réunions de demi-groupes simples. *Ann. Sci. Ecole norm. Sup.*, **79**(3), 361–379.

Davey, B.A., & Priestley, H.A. 1990. *Introduction to Lattices and Order.* Cambridge University Press.

Dechter, R. 1999. Bucket Elimination: A Unifying Framework for Reasoning. *Artificial Intelligence*, **113**, 41–85.

Gottlob, G., Leone, N., & Scarcello, F. 1999a. A Comparison of Structural CSP Decomposition Methods. *Pages 394–399 of: Proceedings of the 16th International Joint Conference on Artificial Intelligence IJCAI.* Morgan Kaufmann.

Gottlob, G., Leone, N., & Scarcello, F. 1999b. Hypertree decompositions and tractable queries. *Pages 21–32 of: PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA: ACM Press.

Gottlob, G., Leone, N., & Scarcello, F. 2001. The complexity of acyclic conjunctive queries. *J. ACM*, **48**(3), 431–498.

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. 1990. Bayesian Updating in Causal Probabilistic Networks by Local Computation. *Computational Statistics Quarterly*, **4**, 269–282.

Kohlas, J. 2002. *Information in Context.* Tech. rept. 02–15. Department of Informatics, University of Fribourg.

Kohlas, J. 2003. *Information Algebras: Generic Structures for Inference*. Discrete Mathematics and Theoretical Computer Science. London, Berlin, Heidelberg: Springer-Verlag.

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., & Shenoy, P.P. 2000. Computation in Valuation Algebras. *Pages 5–39 of:* Kohlas, J., & Moral, S. (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning*. Kluwer, Dordrecht.

Kohlas, J., & Wilson, N. 2008. Semiring induced Valuation Algebras: Exact and approximate Local Computation algorithms. *Artif. Intell.*, **172**(11), 1360–1399.

Kohlas, J., Haenni, R., & Moral, S. 1999. Propositional Information Systems. *Journal of Logic and Computation*, **9**(5), 651–681.

Lauritzen, S. L., & Jensen, F. V. 1997. Local Computation with Valuations from a Commutative Semigroup. *Ann. Math. Artif. Intell.*, **21**(1), 51–69.

Lauritzen, S. L., & Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statis. Soc. B*, **50**, 157–224.

Lehmann, N. 2001. *Argumentation System and Belief Functions*. Ph.D. thesis, Department of Informatics, University of Fribourg.

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*. Lecture Notes in Artif. Intell. Springer.

Pouly, M. 2008. *A Generic Architecture for Local Computation*. Ph.D. thesis, Department of Informatics, University of Fribourg.

Pouly, M. 2010. NENOK - A Software Architecture for Generic Inference. *Int. J. on Artif. Intel. Tools*, **19**.

Pouly, M., & Kohlas, J. 2011. *Generic Inference - A unifying Theory for Automated Reasoning*. John Wiley & Sons, Inc.

Schneuwly, C. 2007. *Computing in Valuation Algebras*. Ph.D. thesis, Department of Informatics, University of Fribourg.

Schneuwly, C., Pouly, M., & Kohlas, J. 2004. *Local Computation in Covering Join Trees*. Tech. rept. 04-16. Department of Informatics, University of Fribourg.

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

Shafer, G. 1996. *Probabilistic Expert Systems.* CBMS-NSF Regional Conference Series in Applied Mathematics, no. 67. Philadelphia, PA: SIAM.

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

Shenoy, P. P. 1997. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, **17**, 239–263.

Shenoy, P. P., & Shafer, G. 1990. Axioms for probability and belief-function propagation. *Pages 169–198 of:* Shachter, Ross D., Levitt, Tod S., Kanal, Laveen N., & Lemmer, John F. (eds), *Uncertainty in Artificial Intelligence 4.* Machine intelligence and pattern recognition, vol. 9. Amsterdam: Elsevier.

Shenoy, P.P. 1992a. 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. 1992b. Valuation-Based Systems for Bayesian Decision Analysis. *Operations Research*, **40**(3), 463–484.

Tamura, T., & Kimura, N. 1954. On decompositions of a commutative semigroup. *Kodai Math. Sem. Rep.*, 109–112.