Dynamic Programming & Local Computation *

Marc Pouly & Jürg Kohlas
Department of Informatics iUF
University of Fribourg
CH – 1700 Fribourg (Switzerland)
http://diuf.unifr.ch/tcs/

June 13, 2007

Abstract

It has been known for a long time that solving optimization problems by use of local computation techniques reduces to non-serial dynamic programming. Therefore, the valuation algebra axioms constitute in some way also the mathematical justification of dynamic programming. This paper builds up the corresponding theory from the more general perspective of totally ordered semirings with idempotent addition, which, as we will see, induce valuation algebras that model optimization problems. Based on this framework, we present several methods to identify either some or all solution configurations. The first class generalizes the classical approach of (Shenoy, 1996) to covering join trees where partial solution configurations are stored explicitly. The second class represents the solution configuration set (implicitly) by a Boolean function that is likewise constructed by use of local computation techniques. It will then turn out that this Boolean function is representable by a particular PDAG structure (propositional directed acyclic graph) called d-DNNF that, beside model identification, allows to perform efficiently a large number of further interesting queries. In this way, we find a completely new area of applications that qualify for local computation techniques and broaden in this way the horizon of classical dynamic programming.

*Research supported by grant No. 200020-109510 of the Swiss National Foundation for Research.

Contents

1 Introduction

Solving discrete optimization problems is an important and well-studied task in computer science. One particular approach to tackle them is known as dynamic programming (Bertele & Brioschi, 1972) and can be found in almost every handbook about algorithms and programming techniques. From the bird's eye view, dynamic programming is a technique of successive variable elimination, which naturally is very close to local computation techniques. Indeed, (Shenoy, 1996) has shown that the fusion algorithm (Shenoy, 1992) corresponds to non-serial dynamic programming with single variable elimination when applied to valuation algebras with either minimization or maximization as variable elimination operator. In this way, the axiomatic system of a valuation algebra that enables local computation constitutes also the mathematical permission to apply dynamic programming. More concretely, (Shenoy, 1996) proved that the axioms for dynamic programming given by (Mitten, 1964) entail those of the valuation algebra.

Alternatively to fusion, this paper will be based on the collect algorithm (Shenoy & Shafer, 1990; Kohlas, 2003), another local computation technique, which in contrast operates on an existing join tree with marginalization (projection) instead of single variable elimination. In this case, it corresponds to non-serial dynamic programming with block elimination. Furthermore, we will use the more general axiomatic system introduced in (Schneuwly *et al.*, 2004) which completely pass on neutral valuations. The gain is a less restraining system of axioms, and the more efficient structure of a covering join tree, in which nodes that originally do not hold knowledgebase factors are not artificially blown up by the initialization with neutral elements. In order to identify the valuation algebra instances that lead to optimization problems, we focus on valuation algebras that are induced by a particular kind of totally ordered semirings (Kohlas & Wilson, 2006; Kohlas, 2004). The practical importance of these structures will be illustrated by a large collection of applications that ground on such optimization problems.

Computing maximum or minimum values is only half of the problem when dealing with optimization tasks. In fact, one consequentially asks for either one single or all configurations that adopt the computed optimum value. We will refer to such configurations as solution configurations. (Shenoy, 1996) introduced another local computation technique that allows to find single solution configurations as an additional process following the fusion algorithm. In this paper, we will show that this algorithm can be generalized to covering join trees. By imposing then a further restriction on the underlying semiring, we may extend this algorithm such that all solution configurations are found by essentially the same procedure. This comprises the first class of algorithms presented in this paper.

A different approach for the same task leads to a second class of algorithms which aim at a more compact representation of the solution configuration set. Substantially, this method can be summarized as the construction of a Boolean function in parallel to the fusion or collect algorithm, whose set of models corresponds to the solution configurations we are looking for. It turns out that this Boolean function

is representable by a very particular propositional directed acyclic graph (PDAG) called d-DNNF (Darwiche & Marquis, 2002; Wachter & Haenni, 2006), which allows to identify its model set efficiently. In addition, this representation offers the efficient execution of many other queries that go far beyond model enumeration. In this sense, we point out many new fields of application where this alliance of local computation and knowledge representation may become a very helpful tool.

The outline of this paper is as follows. In the first section we will give a short introduction to valuation algebras and show that we can adjoin a unique identity element in order to pass on neutral valuations. This section is followed by an introduction to the semiring theory with a comprehensive catalogue of semiring instances. Section 4 unifies in some sense the two theories and shows how semirings induce valuation algebras. Section 5 discusses the central definition of an optimization problem, which follows very naturally from the underlying semiring properties. The importance of this definition is highlighted by a collection of typical applications that ask to solve such optimization tasks. In Section 6, we show how to solve optimization problems by local computation techniques and establish the relationship with classical dynamic programming. Finally, the last three sections are dedicated to the identification of solution configurations on covering join trees. Thereby, the two different classes of algorithms are discussed and both techniques are illustrated step-wise by concrete examples.

2 Valuation Algebras

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 for 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 power set) of r and Φ a set of valuations with their domains in D. We assume the following operations defined in (Φ, 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. If we interpret valuations as information pieces, the labeling operation tells us to which questions such a piece refers. Combination can be understood as aggregation and marginalization as focusing or extraction of the part we are interested in. 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, $\psi \in \Phi$,

$$d(\phi \otimes \psi) = d(\phi) \cup d(\psi). \tag{2.1}$$

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

$$d(\phi^{\downarrow x}) = x. \tag{2.2}$$

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

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

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

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

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

$$\phi^{\downarrow x} = \phi. \tag{2.5}$$

The axioms require natural properties of a valuation algebra regarding knowledge or information modelling. The first axiom indicates that Φ is a commutative semigroup under combination. If information comes in pieces, the sequence does not influence the overall knowledge. The labeling axiom tells us that the combination of valuations gives knowledge over the union of the domains involved. Neither variables vanish, nor new appear. The marginalization axiom expresses the natural functioning of focusing. Transitivity says that marginalization can be performed in several steps. Explaining the naturalness of the combination axiom is a little more difficult. Assume we have some information over a domain in order to answer a question. It is interesting to see how the answer is affected if a new information piece arrives. The combination axiom tells us that we either combine the new piece to the already given information and focus afterwards to the specified domain, or we cut first the uninteresting parts of the new information out and combine it afterwards. There is no difference between the two approaches. Finally, the domain axiom tells us that information is not influenced by projecting it to the own domain. Without this axiom, this is not always the case (Shafer, 1991).

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

The following lemma describes a few elementary properties derived directly from this set of axioms. Their proofs can be found in (Kohlas, 2003).

Lemma 1

1. *If ϕ, $\psi \in \Phi$ with $d(\phi) = x$ and $d(\psi) = y$, then*
$$(\phi \otimes \psi)^{\downarrow x \cap y} = \phi^{\downarrow x \cap y} \otimes \psi^{\downarrow x \cap y}. \tag{2.6}$$
2. *If ϕ, $\psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$ and $z \subseteq x$, then*
$$(\phi \otimes \psi)^{\downarrow z} = (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow z}. \tag{2.7}$$

Based on the definition of a valuation algebra given here, there are many variations being either more or less restrictive. We confine the following listing on those variations that are reconsidered in the later parts of this paper and refer to (Kohlas, 2003) and (Schneuwly *et al.*, 2004) for a complete listing with considerably more details.

- *Valuation Algebras with Neutral Elements:* Every sub-semigroup Φ_s has a neutral element, i.e. for all $s \in D$ there is an element $e_s \in \Phi_s$ such that for all $\phi \in \Phi$ with $d(\phi) = s$ we get $\phi \otimes e_s = \phi$. For consistency, we add a neutrality axiom to the above system:

  (A7) *Neutrality:* For $x, y \in D$,
$$e_x \otimes e_y = e_{x \cup y}.$$

- *Valuation Algebras with Null Elements:* Every sub-semigroup Φ_s has an absorbing or null element with respect to combination, i.e. $\phi \otimes z_s = z_s$ for all $\phi \in \Phi$ with $d(\phi) = s$. The following axiom must be satisfied for null elements:

  (A8) *Nullity:* For all $\phi \in \Phi$ with $t \subseteq d(\phi)$,

  $$\phi^{\downarrow t} = z_t$$

  implies that $\phi = z_s$.

As we have seen, the existence of neutral elements is not mandatory for a valuation algebra. Indeed, there are concrete instances such as Gaussian potentials (Kohlas, 2003) which do not provide neutral elements. For computational purposes, the existence of a neutral information piece is nevertheless often desirable. Therefore, we adjoin artificially a so-called *identity element* e to a valuation algebra whenever no neutral elements are available. This procedure has been worked out by (Schneuwly *et al.*, 2004) which also discuss the important consequences. We give a very short review of this procedure.

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

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

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

The proof of this and the following lemma can be found in (Schneuwly *et al.*, 2004). 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. Furthermore, the next lemma states that there is no need to artificially adjoin an identity element if the valuation algebra possesses already neutral elements.

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

This concludes our short introduction to the generic framework of valuation algebras and equips us with the needed elements for our further studies. Foremost, Section 3 will introduce another algebraic structure called *semiring*, and we will learn subsequently how the two theories are related. Moreover, these sections contain a large catalogue of examples, including those that lead to optimization problems in the context of semiring induced valuation algebras.

3 Semirings

Semirings are algebraic structures with two binary operations $+$ and $\times$ over a set of values A. We call a tuple $\mathcal{A} = \langle A, +, \times \rangle$ a *semiring* if both operations $+$ and $\times$ are associative and commutative and if $\times$ distributes over $+$. Elsewhere this is also called a *commutative semiring*. Associativity of $+$ and $\times$ allows us to write expressions like $a_1 + a_2 + \cdots + a_n$ and $\sum_i a_i$, or $a_1 \times a_2 \times \cdots \times a_n$ and $\prod_i a_i$ respectively. In particular for an index set $I = I_1 \cup \ldots \cup I_n$, where the I_j are finite and disjoint, commutativity and associativity entail that

$$\sum_{j=1}^{n} \sum_{i \in I_j} a_i = \sum_{i \in I} a_i \quad \text{and} \quad \prod_{j=1}^{n} \prod_{i \in I_j} a_i = \prod_{i \in I} a_i.$$

If there is an element $\mathbf{0} \in A$ such that $\mathbf{0} + a = a + \mathbf{0} = a$ and $\mathbf{0} \times a = a \times \mathbf{0} = \mathbf{0}$ for all $a \in A$, then $\mathcal{A}$ is called a semiring with *zero element*. In this case the zero element $\mathbf{0}$ is clearly unique. A zero element can be adjoined to any semiring by adding an extra element 0 to A and extend $+$ and $\times$ to $A \cup \{0\}$ by $a + 0 = 0 + a = a$ and $a \times 0 = 0 \times a = 0$ for all $a \in A \cup \{0\}$. Then it is easy to verify that $\langle A \cup \{0\}, +, \times \rangle$ is a semiring. Hence, we can assume for our further studies that every semiring contains a zero element. If furthermore $a + b = \mathbf{0}$ implies that $a = b = \mathbf{0}$ for all $a, b \in A$, then the semiring is called *positive*.

A semiring element $\mathbf{1} \in A$ is said to be a *unit element* if $\mathbf{1} \times a = a \times \mathbf{1} = a$ for all $a \in A$. Again, there can be at most one unit element. A semiring is called *idempotent* if $a + a = a$ for all $a \in A$. In this case, the semiring $\mathcal{A} = \langle A, +, \times \rangle$ can be extended to include a unit element as follows: For each $a \in A$ define a new element a_1 such that $a \neq b$ implies $a_1 \neq b_1$. Let then $A' = A \cup A_1$, where $A_1 = \{a_1 : a \in A\}$. Define $+'$ as follows. When a and b are arbitrary elements of A: $a +' b = a + b$, whereas $a +' b_1$, $a_1 +' b$ and $a_1 +' b_1$ are all defined to be $(a + b)_1$. Further, we define $\times'$ as follows: $a \times' b = a \times b$ and $a \times' b_1$ and $a_1 \times' b$ are both defined to be $(a \times b) + a$ and $a_1 \times' b_1$ is defined to be $(a_1 \times' b) +' a_1$. The system $\mathcal{A}' = \langle A', +', \times' \rangle$ is then a semiring with unit element 0_1 which is again idempotent. Hence, we assume subsequently that all idempotent semirings possess a unit element and show next how a partial order can be defined on such semirings.

3.1 Ordered Semirings

On every idempotent semiring A, we can introduce a relation $\leq_{id}$ by:

$$a \leq_{id} b \text{ if and only if } a + b = b.$$

We list some important properties of this relation:

Lemma 4

1. *$\leq_{id}$ is a partial order, i.e. reflexive, transitive and antisymmetric.*

2. $a \leq_{id} b$ *and* $a' \leq_{id} b'$ *imply* $a + a' \leq_{id} b + b'$ *and* $a \times a' \leq_{id} b \times b'$.
3. $\leq_{id}$ *is monotonic, i.e.* $a \leq_{id} b$ *implies* $a + c \leq_{id} b + c$ *and* $a \times c \leq_{id} b \times c$.
4. $\forall a, b \in A$ *we have* $\mathbf{0} \leq_{id} a \leq_{id} a + b$.
5. A *is positive.*

Proof.

1. The relation $\leq_{id}$ is clearly reflexive, because for all $a \in A$ we have $a + a = a$ and therefore $a \leq_{id} a$. Next, suppose that $a \leq_{id} b$ and $b \leq_{id} c$. We have $c = b + c = a + b + c = a + c$ and therefore $a \leq_{id} c$. Finally, antisymmetry follows immediately since $a \leq_{id} b$ and $b \leq_{id} a$ imply that $b = a + b = a$, thus $a = b$.
2. Suppose $a \leq_{id} b$ and $a' \leq_{id} b'$ and therefore $a + b = b$ and $a' + b' = b'$. Then, $(a + a') + (b + b') = (a + b) + (a' + b') = (b + b')$, hence $a + a' \leq_{id} b + b'$. For the second part, we show first that $a \times a' \leq_{id} b \times a'$. This is the case because $(a \times a') + (b \times a') = a' \times (a + b) = a' \times b$. In the same way, we can conclude that $a' \times b \leq_{id} b' \times b$. Hence, we have $a \times a' \leq_{id} b \times a' \leq_{id} b' \times b$ and from transitivity, we obtain $a \times a' \leq_{id} b \times b'$.
3. Apply (2) for $a' = b' = c$.
4. We have $\mathbf{0} + a = a$ as well as $a + (a + b) = a + b$, hence $\mathbf{0} \leq_{id} a \leq_{id} a + b$.
5. Suppose that $a + b = \mathbf{0}$. Applying (4) we obtain $\mathbf{0} \leq_{id} a \leq_{id} a + b = \mathbf{0}$ and by transitivity and antisymmetry we conclude that $a = \mathbf{0}$. Similarly, we can derive $b = \mathbf{0}$.

□

Lemma 5 *In an idempotent semiring, we have* $a + b = \sup\{a, b\}$.

Proof. By Lemma 4 Property (4) we have $a, b \leq_{id} a + b$. Let c be another upper bound of a and b, $a \leq_{id} c$ and $b \leq_{id} c$. Then by property (2) $a + b \leq_{id} c + c = c$. Thus $a + b$ is the least upper bound. □

From the perspective of an optimization task, it is desirable that the above relation is actually a *total order*. Such semirings are also called *addition-is-max semirings* (Wilson, 2004) due to the following refinement of Lemma 5.

Lemma 6 *If* $\leq_{id}$ *is total, we have* $a + b = \max\{a, b\}$.

Proof. Since $\leq_{id}$ is total, we either have $a \leq_{id} b$ or $b \leq_{id} a$ for all $a, b \in A$. Assume that $a \leq_{id} b$ and therefore $a + b = b$. Hence, $a + b = \max\{a, b\}$. □

We have already seen in Lemma 13 that this order behaves monotonic under both semiring operations. For some applications however, a more restrictive version is needed. We define

$$a <_{id} b \text{ if and only if } [a \leq_{id} b \text{ and } a \neq b] .$$

Definition 2 *An idempotent semiring is called strictly monotonic over $\times$ if for $c \neq \mathbf{0}$, $a <_{id} b$ implies that $a \times c <_{id} b \times c$.*

An important consequence of this definition is stated in the following lemma.

Lemma 7 *In a totally ordered, idempotent semiring which is strictly monotonic over $\times$, we have for $a \neq \mathbf{0}$ that $a \times b = a \times c$ if and only if $b = c$.*

Proof. Assume that $b \neq c$, say $b <_{id} c$. Then, because $\times$ behaves strictly monotonic, we have $a \times b <_{id} a \times c$ which contradicts the assumption. Furthermore, the semiring is totally ordered and therefore $b = c$ must hold. □

So far, we studied the properties of the relation $\leq_{id}$ which has been introduced artificially at the beginning of this section. It seems therefore natural to compare this relation with other total orders that may already exist in a particular semiring. This is pointed out in the following theorem.

Theorem 1 *Let A be an idempotent semiring and we assume an arbitrary total, monotonic order $\leq$ defined over A. Then, we have for all $a, b \in A$:*

1. $\mathbf{0} \leq \mathbf{1} \;\Rightarrow\; [a \leq b \;\Leftrightarrow\; a \leq_{id} b]$.
2. $\mathbf{1} \leq \mathbf{0} \;\Rightarrow\; [a \leq b \;\Leftrightarrow\; b \leq_{id} a]$.

Proof. We remark first that $\mathbf{0} \leq \mathbf{1}$ implies $\mathbf{0} = \mathbf{0} \times b \leq \mathbf{1} \times b = b$ and therefore $a = a + \mathbf{0} \leq a + b$ by monotony of the assumed total order. Hence, $a, b \leq a + b$ for all $a, b \in A$. In the same way we derive from $\mathbf{1} \leq \mathbf{0}$ that $a + b \leq a, b$.

1. Assume that $a \leq b$ and hence by idempotency and monotonicity $a + b \leq b + b = b$. Because $\mathbf{0} \leq \mathbf{1}$ we have $b \leq a + b$ and therefore $b \leq a + b \leq b$. By antisymmetry we conclude that $a + b = b$, that is $a \leq_{id} b$. On the other hand if $a \leq_{id} b$, we obtain from $\mathbf{0} \leq \mathbf{1}$ that $a \leq a + b = b$ and therefore $a \leq b$ holds.
2. Assume that $a \leq b$ and hence by monotony and idempotency $a = a + a \leq a + b$. Because $\mathbf{1} \leq \mathbf{0}$ we have $a + b \leq a$ and therefore $a \leq a + b \leq a$. By antisymmetry we conclude that $a + b = a$. On the other hand if $b \leq_{id} a$, we obtain from $\mathbf{1} \leq \mathbf{0}$ that $a = a + b \leq b$ and therefore $a \leq b$ holds.

□

Note that for semirings where $\mathbf{0} = \mathbf{1}$, both consequences of Theorem 1 are trivially satisfied because these semirings consist of only one single element. Furthermore, we should consider the statement of this theorem from the perspective of an optimization problem, although this precedes our discussion. In the case of an idempotent semiring, we can identify every total monotonic order with either the *natural order* $\leq_{id}$ or its inverse. This means that general optimization with respect to any total monotonic order $\leq$ reduces to maximization (or minimization) with respect to $\leq_{id}$. In this spirit, we can restrict our studies to $\leq_{id}$ and in the ongoing part of this paper, we will simply write $\leq$ instead of $\leq_{id}$.

3.2 Semiring Example Catalogue

At this point, we will give a listing of some famous semiring instances which will later serve for illustration purposes.

Example 1: *Arithmetic Semirings.* Let A be the set of non-negative real numbers $\mathbb{R}^+ \cup \{0\}$ with $+$ and $\times$ designating the usual addition and multiplication. This is clearly a positive semiring with the number 0 as zero element and the number 1 as unit element. In the same way, we could also take the field of real or rational numbers, or alternatively only integers or natural numbers. In the former three cases, the semiring would not be positive anymore whereas ordinary addition and multiplication on non-negative integers $\mathbb{N} \cup \{0\}$ yield again a positive semiring. $\ominus$

Example 2: *Bottleneck Algebra.* Another semiring is obtained if we take $a + b = \max\{a, b\}$ and $a \times b = \min\{a, b\}$ over the set of real numbers $\mathbb{R} \cup \{+\infty, -\infty\}$. Then $-\infty$ is the zero element, $+\infty$ the unity, and the semiring is idempotent. It is furthermore totally ordered and the semiring order $\leq_{id}$ coincides with the natural order between real numbers, which is not strictly monotonic over $\times$. $\ominus$

Example 3: *Tropical Semirings.* A very popular semiring is defined over the set of non-negative integers $\mathbb{N} \cup \{0, \infty\}$ with $a + b = \min\{a, b\}$ and the usual addition $+_{\mathbb{N}}$ for $\times$ with the convention that $a +_{\mathbb{N}} \infty = \infty$. This semiring is idempotent, ∞ is the zero element and the integer 0 is the unit element of this semiring. The semiring order $\leq_{id}$ behaves strictly monotonic over $\times$ and conforms to the inverse natural order of non-negative integers. Alternatively, we could also take max for $+$ over the real numbers $\mathbb{R} \cup \{-\infty\}$. Multiplication is again represented by $+_{\mathbb{N}}$ with $a +_{\mathbb{N}} (-\infty) = -\infty$. This semiring is again idempotent, has $-\infty$ as zero element and 0 as unit. $\leq_{id}$ behaves again strictly monotonic over $\times$ but corresponds directly to the natural order of non-negative integers. $\ominus$

Example 4: *Truncation Semirings.* An interesting variation of the Tropical semiring is obtained if we take $A = \{0, \ldots, k\}$ for some integer k. Addition corresponds again to minimization but this time we take the *truncated integer addition* for $\times$, i.e. $a \times b = \min\{a + b, k\}$. This semiring is idempotent, k is the zero element and 0 the unit. The semiring order corresponds again to the inverse integer order but strict monotonicity does not hold anymore. $\ominus$

Example 5: *T-Norms.* Triangular norms were originally introduced in the context of probabilistic metric spaces (Menger, 1942; Schweizer & Sklar, 1960). They are simply binary operations on the unit interval $A = [0, 1]$ which are commutative and associative, are nondecreasing in both arguments, and have the number 1 as unit and 0 as zero element:

1. $\forall a, b, c \in [0, 1]$ we have $T(a, b) = T(b, a)$ and $T(a, T(b, c)) = T(T(a, b), c)$;
2. $a \leq a'$ and $b \leq b'$ imply $T(a, b) \leq T(a', b')$;
3. $\forall a \in [0, 1]$ we have $T(a, 1) = T(1, a) = a$ and $T(a, 0) = T(0, a) = 0$.

In order to obtain a semiring, we define the operation $\times$ on the unit interval by a t-norm and $+$ as max. We obtain an idempotent semiring with 0 as zero element and 1 as unit, where $\leq_{id}$ accords with the natural order of real numbers. The following are typical t-norms:

- *Minimum t-norm:* $T(a, b) = \min\{a, b\}$.
- *Product t-norm:* $T(a, b) = a \cdot b$.
- *Lukasiewicz t-norm:* $T(a, b) = \max\{a + b - 1, 0\}$.
- *Drastic product:* $T(a, 1) = T(1, a) = a$ whereas $T(a, b) = 0$ in all other cases.

Accordingly, we may also take min for addition. Consequently, 1 becomes the zero element, 0 the unit, and the semiring remains idempotent. In this case, the semiring order corresponds to the inverse natural order of real numbers. Note also that strict monotonicity depends on the choice of the t-norm. $\ominus$

Example 6: *Semiring of Boolean Functions.* Consider a set of r propositional variables. Then, Boolean functions $f : \{0, 1\}^r \to \{0, 1\}$ form a semiring with addition $a + b = \max\{a, b\}$ and multiplication $a \times b = \min\{a, b\}$. Both operations max and min are evaluated pointwise for the two Boolean functions a and b. If we adopt the intention that 0 stands for the truth value *false* and 1 for *true*, addition can be regarded as logical disjunction and multiplication as logical conjunction. This semiring is clearly idempotent with the constant mapping $f_0 : \{0, 1\}^r \to 0$ being the zero element and $f_1 : \{0, 1\}^r \to 1$ the unit element of this semiring. Obviously, the semiring order corresponds to the natural order in $\{0, 1\}$ and it is strictly monotonic over $\times$. Of particular interest is the case where $r = 0$. This semiring is referred as the *Boolean semiring.* $\ominus$

Example 7: *Distributive Lattice.* More general, we can say that every distributive lattice is a semiring with join for $+$ and meet for $\times$. These semirings are clearly idempotent. The bottom element $\bot$ of the lattice becomes the semiring's zero element and, if the lattice has a top element $\top$, then it becomes the unit element. This example covers the semiring of Boolean functions as well as the Bottleneck Algebra from Example 2. Another interesting example of a distributive lattice is given by the natural numbers (or all positive divisors of a natural number n) with the greatest common divisor as meet and the least common multiple as join. We refer to (Davey & Priestley, 1990) for a broad listing of further examples of distributive lattices. $\ominus$

After this short excursion into semiring theory, we will now see how semirings induce valuation algebras by a very natural mapping from configurations to semiring values.

4 Semiring Induced Valuation Algebras

For the development of this theory, we consider only variables X that take values from a finite set Ω_X called its *frame*. This term can be extended to non-empty sets of variables $s \in D$ by the Cartesian product of frames Ω_X of each variable $X \in s$,

$$\Omega_s = \prod_{X \in s} \Omega_X. \tag{4.1}$$

The elements $\mathbf{x} \in \Omega_s$ are called *configurations* of s. The frame of the empty variable set is defined by convention as $\Omega_\emptyset = \{\diamond\}$. If $\mathbf{x}$ is a configuration with domain s and $t \subseteq s$, then $\mathbf{x}^{\downarrow t}$ denotes the projection of $\mathbf{x}$ to the subdomain t. In particular, we have $\mathbf{x}^{\downarrow \emptyset} = \diamond$. Sometimes, in order to emphasize the decomposition of a configuration $\mathbf{x}$ into components belonging to two disjoint subsets t and $s - t$ of s, we write $\mathbf{x} = (\mathbf{x}^{\downarrow t}, \mathbf{x}^{\downarrow s-t})$ with the convention that $(\mathbf{x}, \diamond) = (\diamond, \mathbf{x}) = \mathbf{x}$. In the same way, we define the projection of a set of configurations $S \subseteq \Omega_s$ as $S^{\downarrow t} = \{\mathbf{x}^{\downarrow t},\ \mathbf{x} \in S\}$ and for $S_1 \subseteq \Omega_t$ and $S_2 \subseteq \Omega_{s-t}$, $S_1 \times S_2 = \{(\mathbf{x}_1, \mathbf{x}_2),\ \mathbf{x}_1 \in S_1, \mathbf{x}_2 \in S_2\} \subseteq \Omega_s$.

Let us consider a semiring $\mathcal{A} = \langle A, +, \times \rangle$. A *semiring valuation* ϕ with domain s is defined to be a function that associates a value from A with each configuration $\mathbf{x} \in \Omega_s$, formally $\phi : \Omega_s \to A$. We will again denote the set of all semiring valuations with domain s by Φ_s and use Φ for all semiring valuations whose domain belongs to the lattice D. We will now introduce the following operations in (Φ, D):

1. *Labeling:* $\Phi \to D$: $d(\phi) = s$ if $\phi \in \Phi_s$.
2. *Combination:* $\Phi \times \Phi \to \Phi$: for $\phi, \psi \in \Phi$ and $\mathbf{x} \in \Omega_{d(\phi) \cup d(\psi)}$ we define

$$(\phi \otimes \psi)(\mathbf{x}) \quad = \quad \phi(\mathbf{x}^{\downarrow d(\phi)}) \times \psi(\mathbf{x}^{\downarrow d(\psi)}). \tag{4.2}$$

3. *Marginalization:* $\Phi \times D \to \Phi$: for $\phi \in \Phi$, $t \subseteq d(\phi)$ and $\mathbf{x} \in \Omega_t$ we define

$$\phi^{\downarrow t}(\mathbf{x}) \quad = \quad \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}). \tag{4.3}$$

Theorem 2 *A system of semiring valuations (Φ, D) with labeling, combination and marginalization as defined above, satisfies the axioms of a valuation algebra.*

Proof. We will verify the axioms (A1) to (A6) of a valuation algebra. At first glance, we see that the *Labeling* (A2), *Marginalization* (A3) and *Domain* (A6) properties are immediate consequences from the above definitions. For the remaining axioms we have:

(A1) *Commutative Semigroup:* The commutativity of combination follows directly from the commutativity of the $\times$ operation in the semiring A and the definition

of combination. Associativity is proved as follows: assume that ϕ, ψ and η are valuations with domains $d(\phi) = s$, $d(\psi) = t$ and $d(\eta) = u$, then

$$\begin{aligned}(\phi \otimes (\psi \otimes \eta))(\mathbf{x}) &= \phi(\mathbf{x}^{\downarrow s}) \times (\psi \otimes \eta)(\mathbf{x}^{\downarrow t \cup u}) \\ &= \phi(\mathbf{x}^{\downarrow s}) \times \Big(\psi((\mathbf{x}^{\downarrow t \cup u})^{\downarrow t}) \times \eta((\mathbf{x}^{\downarrow t \cup u})^{\downarrow u})\Big) \\ &= \phi(\mathbf{x}^{\downarrow s}) \times \Big(\psi(\mathbf{x}^{\downarrow t}) \times \eta(\mathbf{x}^{\downarrow u})\Big) \\ &= \phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow t}) \times \eta(\mathbf{x}^{\downarrow u}).\end{aligned}$$

The same result is obtained in exactly the same way for $((\phi \otimes \psi) \otimes \eta)(\mathbf{x})$. Thus associativity holds.

(A4) *Transitivity:* Transitivity of projection means simply that we can sum out variables in two steps. That is, if $t \subseteq s \subseteq d(\phi) = u$, then, for all $\mathbf{x} \in \Omega_t$,

$$\begin{aligned}(\phi^{\downarrow s})^{\downarrow t}(\mathbf{x}) &= \sum_{\mathbf{y} \in \Omega_{s-t}} \phi^{\downarrow s}(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y} \in \Omega_{s-t}} \sum_{\mathbf{z} \in \Omega_{u-s}} \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) \\ &= \sum_{(\mathbf{y}, \mathbf{z}) \in \Omega_{u-t}} \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \phi^{\downarrow t}(\mathbf{x}).\end{aligned}$$

(A5) *Combination:* Suppose that ϕ has domain t and ψ domain u and $\mathbf{x} \in \Omega_s$, where $t \subseteq s \subseteq t \cup u$. Then we have for $\mathbf{x} \in \Omega_s$,

$$\begin{aligned}(\phi \otimes \psi)^{\downarrow s}(\mathbf{x}) &= \sum_{\mathbf{y} \in \Omega_{(t \cup u) - s}} (\phi \otimes \psi)(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y} \in \Omega_{u-s}} \Big(\phi(\mathbf{x}^{\downarrow t}) \times \psi(\mathbf{x}^{\downarrow s \cap u}, \mathbf{y})\Big) \\ &= \phi(\mathbf{x}^{\downarrow t}) \times \sum_{\mathbf{y} \in \Omega_{u-s}} \psi(\mathbf{x}^{\downarrow s \cap u}, \mathbf{y}) = \phi(\mathbf{x}^{\downarrow t}) \times \psi^{\downarrow s \cap u}(\mathbf{x}^{\downarrow s \cap u}) \\ &= (\phi \otimes \psi^{\downarrow s \cap u})(\mathbf{x}).\end{aligned}$$

□

If the semiring $\mathcal{A}$ which induces the valuation algebra (Φ, D) has a unit element $\mathbf{1}$, then we have for every domain s a valuation $e_s(\mathbf{x}) = \mathbf{1}$ for all $\mathbf{x} \in \Omega_s$. This is the neutral valuation in the semigroup Φ_s defined by the combination, i.e. for all $\phi \in \Phi_s$ we have $e_s \otimes \phi = \phi \otimes e_s = \phi$. These neutral elements satisfy property (A7):

(A7) *Neutrality:* We have by definition for all $\mathbf{x} \in \Omega_{s \cup t}$:

$$(e_s \otimes e_t)(\mathbf{x}) = e_s(\mathbf{x}^{\downarrow s}) \times e_t(\mathbf{x}^{\downarrow t}) = \mathbf{1} \times \mathbf{1} = \mathbf{1}.$$

It is in general not true that the projection of the neutral valuation e_s to some subdomain $t \subseteq s$ results in the neutral valuation e_t (Example 8 discusses such a case). This property extends the discussion in Section 2 and is referred to as stability. However, it turns out that idempotent addition is a sufficient condition for a semiring to induce a stable valuation algebra.

(A9) *Stability:* Let $\mathbf{x} \in \Omega_t$. If the semiring is idempotent, we have $e_s^{\downarrow t} = e_t$ by

$$e_s^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} e_s(\mathbf{x}, \mathbf{y}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \mathbf{1} \;=\; \mathbf{1}.$$

If the semiring $\mathcal{A}$ inducing the valuation algebra (Φ, D) has a zero element $\mathbf{0}$, we can define null valuations for the sub-semigroups Φ_s as $z_s(\mathbf{x}) = \mathbf{0}$ for all $\mathbf{x} \in \Omega_s$. Then, if ϕ is a valuation with domain s, we clearly have $(\phi \otimes z_s)(\mathbf{x}) = \phi(\mathbf{x}) \times z_s(\mathbf{x}) = \mathbf{0}$ and therefore $\phi \otimes z_s = z_s$ for all $\phi \in \Phi_s$. However, the requirement of axiom (A8) that a valuation that projects to a null element must itself be a null element is only satisfied if the underlying semiring is positive.

(A8) *Nullity:* Let $\mathbf{x} \in \Omega_t$ with $t \subseteq s = d(\phi)$. In a positive semiring, $\phi^{\downarrow t} = z_t$ implies that $\phi = z_s$ by

$$\phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}) \;=\; \mathbf{0}.$$

Note also that due to Lemma 4 property (5), semirings with idempotent addition are positive and thus induce valuation algebras with null elements.

The following theorem summarizes what we have found so far:

Theorem 3

1. *Semirings with unit elements induce valuation algebras with neutral elements.*
2. *Idempotent semirings induce stable valuation algebras.*
3. *Positive semirings induce valuation algebras with null elements.*

Section 3.2 presented a catalogue of different semiring examples and we will now see what kind of valuation algebras these semirings induce.

4.1 Examples of Semiring induced Valuation Algebras

Example 8: *Probability Potentials.* If we consider the arithmetic semiring of Example 1, then the induced valuations are called *probability potentials*. In fact they represent, up to normalization, discrete probability distributions and families of conditional probability distributions, which are used for example in the context of inference in probabilistic networks (Lauritzen & Spiegelhalter, 1988; Shafer, 1996). Combination reduces to point-wise multiplication which models the computation of multidimensional distributions from prior and conditional distributions, as for example $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y}) \cdot p(\mathbf{y})$. Marginalization meets the corresponding standard operation in probability theory. Derived from the properties of the arithmetic

semiring, probability potentials have neutral and null elements. Interestingly, this valuation algebra is not stable despite the presence of neutral elements, because

$$e_s^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y}\in\Omega_{s-t}} e_s(\mathbf{x},\mathbf{y}) \;=\; |\Omega_{s-t}| \;\geq\; 1.$$

This perfectly illustrates the necessity of idempotent addition for stability as it is stated in Theorem 3. ⊖

Example 9: *Relational System.* If we use the Boolean semiring of Example 6, then a valuation ϕ over domain s defines a *relation* over this domain, i.e. a set of tuples $R_\phi = \{\mathbf{x} \in \Omega_s : \phi(\mathbf{x}) = 1\}$. Combination corresponds then to the operation of natural join,

$$R_{\phi\otimes\psi} \;=\; R_\phi \bowtie R_\psi \;=\; \{\mathbf{x} \in \Omega_{s\cup t} : \mathbf{x}^{\downarrow s} \in R_\phi, \mathbf{x}^{\downarrow t} \in R_\psi\}$$

and marginalization to the ordinary projection of relations,

$$R_{\phi^{\downarrow t}} \;=\; \pi_t(R_\phi) \;=\; \{\mathbf{x}^{\downarrow t} : \mathbf{x} \in R_\phi\}.$$

This is a subset of a relational algebra known from classical database theory. ⊖

Example 10: *Constraint Systems.* Finite CSPs (constraint satisfaction problems) are obtained by the Boolean semiring of Example 6. A configuration $\mathbf{x} \in \Omega_s$ is said to satisfy a constraint $\phi \in \Phi_s$, if $\phi(\mathbf{x}) = 1$. Essentially, this is just another interpretation of the relational system in the foregoing example. However, more general CSPs can be derived from other semirings. If we take for example the Tropical semiring from Example 3, we obtain *weighted* CSPs. Another variation of weighted CSPs takes values in the Truncation semiring of Example 4. Finally, *set-based* CSPs are constructed from the subset lattice of Example 7. ⊖

Example 11: *Possibility Potentials.* Possibility potentials have been introduced by (Zadeh, 1978; Zadeh, 1987) as an alternative approach to probability theory, where a valuation $p(\mathbf{x}) \in [0, 1]$ expresses the degree of possibility of some configuration. They are obtained from the t-norm semirings of Example 5 with maximization for addition and any t-norm for multiplication. ⊖

Further interesting valuation algebra instances that are not necessarily based on semirings can be found in (Kohlas, 2003).

5 Optimization Problems

(Schneuwly *et al.*, 2004) summarize the general computational interest in valuation algebras by the notion of the *projection problem* defined as follows:

Definition 3 *A projection problem is given by the task of computing*

$$\phi^{\downarrow t} \quad = \quad (\phi_1 \otimes \phi_2 \otimes \cdots \otimes \phi_n)^{\downarrow t} \tag{5.1}$$

from a set of valuations $\{\phi_1, \ldots, \phi_n\}$ *and a domain* $t \subseteq d(\phi_1) \cup \cdots \cup d(\phi_n)$.

The meaning of this task is essentially to collect all available knowledge and to focus the result afterwards on the given domain of interest t named *query*. This very natural process is indeed central for every formalism that turns out to be a valuation algebra instance. In terms of probability potentials (see Example 8), the projection problem amounts to the computation of a marginal of some factorized probability distribution as it is made with Bayesian networks. And for the relational system of Example 9, the projection problem abstracts standard query answering of database theory.

Projection problems acquire a particular signification in the case of valuation algebras induced by semirings with idempotent addition. Here we have for $\mathbf{x} \in \Omega_t$ and $t \subseteq d(\phi) = s$

$$\phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}) \;=\; \sup\{\phi(\mathbf{x}, \mathbf{y}),\ \mathbf{y} \in \Omega_{s-t}\}. \tag{5.2}$$

It corresponds to the computation of the lowest upper bound of all semiring values that are assigned to those configurations of the compound knowledge ϕ that project to $\mathbf{x}$. This is a consequence of Lemma 5. In particular, if we consider the query to be empty, we obtain for $\mathbf{x} \in \Omega_s$

$$\phi^{\downarrow \emptyset}(\diamond) \quad = \quad \sup\{\phi(\mathbf{x}),\ \mathbf{x} \in \Omega_s\},$$

which amounts to the computation of the lowest upper bound of all semiring values that are assigned to the configurations of ϕ. If we assume furthermore that the underlying semiring is totally ordered, we obtain according to Lemma 6

$$\phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}) \;=\; \max\{\phi(\mathbf{x}, \mathbf{y}),\ \mathbf{y} \in \Omega_{s-t}\}, \tag{5.3}$$

and

$$\phi^{\downarrow \emptyset}(\diamond) \quad = \quad \max\{\phi(\mathbf{x}),\ \mathbf{x} \in \Omega_s\}, \tag{5.4}$$

respectively. This justifies the following definition:

Definition 4 *A projection problem with query* $t = \emptyset$ *over a totally ordered, idempotent semiring is called an optimization problem.*

Optimization problems generally consist in finding a solution in some feasible region that has the minimum (or maximum) value of the objective function. This explanation has been given in (Atallah & Fox, 1998) and agrees perfectly with Definition 4, where ϕ corresponds to the *objective function*. The following section surveys some concrete examples of optimization problems in order to convince the reader of the far reaching application field of this theory.

5.1 Optimization Problems in Practice

Example 12: *Classical Optimization.* Perhaps the most obvious optimization problem is obtained from the Tropical semiring (see Example 3), where $\times$ corresponds to the usual addition of non-negative integers and $+$ either to maximization or minimization. Accordingly, $\phi^{\downarrow\emptyset}(\diamond)$ represents the optimum value over all configurations of the objective function with respect to the semiring order, i.e.

$$\phi^{\downarrow\emptyset}(\diamond) = \max_{\mathbf{x}\in\Omega_{d(\phi)}} \left(\phi_1(\mathbf{x}^{\downarrow d(\phi_1)}) \times \ldots \times \phi_n(\mathbf{x}^{\downarrow d(\phi_n)})\right). \tag{5.5}$$

⊖

Example 13: *Satisfiability Problems.* We have seen in Example 10 that a Boolean semiring induces a valuation algebra which can be used to model constraints over sets of variables. Such a constraint is satisfiable if $R_\phi = \{\mathbf{x} \in \Omega_s : \phi(\mathbf{x}) = 1\}$ contains at least one configuration, and this is the case if $\phi^{\downarrow\emptyset}(\diamond) = 1$, because addition corresponds to maximization in the Boolean semiring. Alternatively, if $\phi^{\downarrow\emptyset}(\diamond) = 0 = z_\emptyset(\diamond)$, the set of constraints is contradictory due to the nullity axiom. Hence, we have found another application of an optimization problem. If we restrict ourselves to binary variables and assume the constraints to be derived from propositional formulae, the above task will solve the classical SAT problem of propositional logic (Garey & Johnson, 1990). ⊖

Example 14: *Maximum Satisfiability Problems.* MAX SAT (Garey & Johnson, 1990) can be regarded as an approximation task of the SAT problem. It asks for the truth value assignment that allows the maximum number of clauses to be true. This number of clauses can be computed by applying the tropical semiring with maximization to constraints derived from clauses of a SAT instance. Doing so, $\phi(\mathbf{x})$ represents the number of true clauses under the truth assignment $\mathbf{x}$ and consequently, $\phi^{\downarrow\emptyset}(\diamond)$ designates the maximum number of clauses which can be made true over all possible truth value assignments. ⊖

Example 15: *Most and Least Probable Configuration.* Example 8 has shown that the arithmetic semiring induces the valuation algebra of probability potentials. However, in many applications one is not directly interested in a concrete probability value of some configuration but rather in the converse question that asks for the configuration with the highest probability value. This value can be determined by use of the product t-norm semiring instead. In this case, addition becomes maximization and the value $\phi^{\downarrow\emptyset}(\diamond) \in [0, 1]$ corresponds to the value of the most probable

configuration. Similarly, if we replace maximization by minimization, the marginal corresponds to the value of the least probable configuration. ⊖

Example 16: *Channel Decoding.* The semiring consisting of the unit interval with minimization for addition and the product t-norm for multiplication can also be used for decoding purposes. Assume an unreliable, memoryless communication channel with $X = (x_1, \ldots, x_n)$ denoting the unknown input of the channel and $Y = (y_1, \ldots, y_n)$ the observed codeword after transmission. In order to deduce the input from the received output, we wish to maximize the probability $P(Y|X)$. We have

$$\max_X P(Y|X) \quad = \quad \max_X \left(\prod_{i=1}^{n} p(y_i|x_i) \cdot p(x_i) \right),$$

where the transmission probabilities $p(y_i|x_i)$ are known from the channel specification. It is convenient for computational purposes that we minimize instead the logarithm of $P(Y|X)$. If we furthermore assume a uniform input distribution, we obtain

$$\min_X \left(-\sum_{i=1}^{n} \log p(y_i|x_i) \right).$$

This is again an optimization problem, known as *maximum likelihood decoding* or *Bayes decoding*, if we pass on the uniform prior distribution. ⊖

All the different scenarios of application presented in this section have in common that we need to solve a projection problem for a given set of valuations $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$. To understand the arduousness of this task, consider a straight forward execution by first computing the combination sequence followed by the final marginalization to the query t as written in Equation (5.1). The intractability of this strategy becomes apparent by glancing at Definition 4.2. We notice that the complexity of the combination operator grows exponentially with the sizes of the involved factors, respectively their domains. To be more concrete, we can give the following estimation for the size of the objective function ϕ:

$$size(\phi) \quad = \quad \prod_{X \in d(\phi)} |\Omega_X|. \tag{5.6}$$

To go around this problem, (Schneuwly *et al.*, 2004) present a framework of different methods which schedule the operations in such a way that the domains do not grow significantly. These methods are called local computation algorithms and the following section outlines briefly how they can be applied. We restrict our attention to the most general algorithm named *collect algorithm* (Shafer & Shenoy, 1990; Kohlas, 2003).

6 Computing Marginals

We have just seen that the straight forward computation of projection problems generally results in severe efficiency problems due to the exponential growth of the factors under combination. A particular way to tame this beast of complexity is known as *local computation* and consists essentially in a clever alternation of combination and marginalization operations in order to avoid a significant increase of the factor domains. The key structure for this re-scheduling of valuation algebra operations is the so-called join tree (Kohlas, 2003).

Definition 5 *A join tree is a labelled tree* (V, E) *whose labeling function* $\lambda : V \to D$ *satisfies the running intersection property, i.e. for two nodes* $v_1, v_2 \in V$*, if* $X \in \lambda(v_1) \cap \lambda(v_2)$*, then* X *is contained in every node label on the unique path between* v_1 *and* v_2*.*

An important objective for our purposes is that the join tree covers the domains of the factors in the projection problem. This means that, for each factor ϕ_i, there exists a node $v \in V$ with $d(\phi_i) \subseteq \lambda(v)$. Such a join tree is called *covering join tree* and we refer to (Lehmann, 2001) for a survey of construction algorithms that build covering join trees efficiently from a given set of input valuations. Furthermore, we pass on unused variables in the join tree by the requirement that

$$\bigcup_{i=1}^{n} d(\phi_i) \quad = \quad \bigcup_{v \in V} \lambda(v). \tag{6.1}$$

Once a covering join tree has been found, we distribute the factors among its nodes. This is done by an assignment mapping $a : \{1, 2, \ldots, n\} \to V$, that assigns to every factor ϕ_i a join tree node $a(i) \in V$ with $d(\phi_i) \subseteq \lambda(a(i))$. This results in a new factorization

$$\psi_1 \otimes \ldots \otimes \psi_m \quad = \quad \phi_1 \otimes \ldots \otimes \phi_n, \tag{6.2}$$

where $m = |V|$ and

$$\psi_i \quad = \quad e \otimes \bigotimes_{j:a(j)=i} \phi_j. \tag{6.3}$$

The factors ψ_i are often called *join tree factors* and each of them corresponds either to a combination of some original projection problem factors or to the identity element if the factor set of the combination in Equation (6.3) is empty. Such a join tree factorization is the starting point of the *collect algorithm* for an efficient computation of projection problems.

6.1 Collect Algorithm

For the description of the collect algorithm, (Shenoy & Shafer, 1990; Kohlas, 2003) regard join tree nodes as virtual processors. Each node can process incoming messages, compute new messages and send them to its neighboring nodes. These messages are scheduled according to a numbering of the nodes which is introduced as follows. We assign number $m = |V|$ to the root node. Then, by directing all edges towards this root node m, it is possible to determine a numbering in such a way, that if j is a node on the path from node i to m, then $j > i$. Note that this numbering is not unique. It is furthermore convenient to introduce the notions of *parent*, *child* and *leaf* nodes in a join tree. The *parents* $pa(i)$ of node i are defined by the set $pa(i) = \{j : j < i \text{ and } (i,j) \in E\}$. Nodes without parents are called *leaves* and the child $ch(i)$ of a node i is the unique node j with $j > i$ and $(i,j) \in E$. Additionally, we introduce the following notation for simplification purposes. Let $\phi \in \Phi$ and $t \in D$, we define

$$\phi^{\Downarrow t} = \phi^{\downarrow t \cap d(\phi)}. \tag{6.4}$$

In the case where $t \subseteq d(\phi)$, it clearly holds that $\phi^{\Downarrow t} = \phi^{\downarrow t}$.

During the collect algorithm, each node waits until it has received a message from all of its parents. Incoming messages are combined to the current node content and then, the node computes a message itself and sends it in turn to its child node. This procedure is repeated up to the root node. We conclude from this first sketch of the collect algorithm that the content of each node changes during the algorithm's run. To incorporate this dynamic behavior, we introduce the following notation:

- $\psi_j^{(1)} = \psi_j$ is the initial content of node j according to Equation (6.3).
- $\psi_j^{(i)}$ is the content of node j before step i of the collect algorithm.

Now, the collect algorithm is specified as follows:

- At step i, node i computes the message

  $$\mu_{i \to ch(i)} = \psi_i^{(i) \Downarrow \lambda(ch(i))}. \tag{6.5}$$

  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

  $$\psi_{ch(i)}^{(i+1)} = \psi_{ch(i)}^{(i)} \otimes \mu_{i \to ch(i)}. \tag{6.6}$$

  The storages of all other nodes do not change at step i, we have

  $$\psi_j^{(i+1)} = \psi_j^{(i)}$$

  for all $j \neq ch(i)$.

According to the introduced node numbering, the collect algorithm traverses the join tree from the leaves upwards to the root node. Leaves can send their message directly and the algorithm stops as soon as the root node has received a message from each of its parents. (Schneuwly *et al.*, 2004) proved the following theorem that specifies the content of the root node at completion of the collect algorithm.

Theorem 4 *At the end of the collect algorithm, the root node m contains the marginal of ϕ relative to its node label,*

$$\psi_m^{(m)} = \phi^{\downarrow\lambda(m)}. \tag{6.7}$$

To get back to our optimization problem, we can easily extract its maximum value by one last marginalization of the root content to the empty domain. Due to the property of Transitivity, we have

$$\phi^{\downarrow\emptyset}(\diamond) \quad = \quad \left(\phi^{\downarrow\lambda(m)}\right)^{\downarrow\emptyset}(\diamond). \tag{6.8}$$

In the foregoing section, we motivated the local computation approach by complexity concerns when dealing with growing factor domains. On this note, Equation (5.6) gave an estimation for the size of the objective function ϕ which, in case of straight forward computation, is calculated explicitly. If we assume r to be the largest variable frame, we have $size(\phi) \in O(r^{|d(\phi)|})$. However, in a message passing algorithm, where intermediate results are stored in join tree nodes, all occurring domains are naturally bounded by the size of the largest node label in the tree, which we denote by w. This measure is often called the *width* of the join tree. Then, the collect algorithm has complexity $O(m \cdot r^w)$ which is a great deal better especially if the width of the join tree is small. We conclude therefore that the optimum arrangement of the computations is achieved if the tree width is as small as possible. Unfortunately, (Arnborg *et al.*, 1987) proved that finding such join trees is NP-complete and consequently, we generally revert to heuristics (Lehmann, 2001). The relevance of the tree width as crucial factor for the complexity of these computations can also be found in classical literature of dynamic programming. In this way, (Bertele & Brioschi, 1972) remarked that "the maximum number of interacting variables is also a reasonable index of computing time".

6.2 Distribute Algorithm

At the end of the collect algorithm, only the root node contains the marginal of ϕ relative to its node label. However, for some applications, it is desirable that this particularity holds also for all other join tree nodes, namely that every node i contains the marginal $\phi^{\downarrow\lambda(i)}$. This can be achieved by another message passing scheme called *distribute algorithm* that proceeds in the reversed order from the root node downwards to the leaves:

- At step $i = m-1, \ldots, 1$, node $ch(i)$ computes the message:

$$\mu_{ch(i)\to i} = \left(\psi_{ch(i)} \otimes \bigotimes_{j \in pa(ch(i)),\ j \neq i} \mu_{j \to ch(i)} \right)^{\Downarrow \lambda(i)} . \tag{6.9}$$

  This message is sent to node i with node label $\lambda(i)$.

- The receiving node i updates its storage by combining the content $\psi_i^{(m)}$ from the collect algorithm with the incoming message.

Again, every node waits until it has received a message from its unique child node. This means that the root node $m = ch(m-1)$ can send its message directly. In order to compute the distribute messages, every node must have access to the messages sent during the collect phase. We therefore assume that these messages were stored in mailboxes and we refer to (Kohlas, 2003) for an illustration of this concept. The algorithm stops when every leaf has obtained a message. (Schneuwly *et al.*, 2004) proved the following theorem that specifies the content of every node after the distribute algorithm.

Theorem 5 *At the end of the distribute algorithm, each node i contains $\phi^{\downarrow \lambda(i)}$.*

As in the collect phase, one message is sent along every join tree edge but this time in the reversed direction. Since these message are of comparable size with respect to Equation 5.6, we conclude that the complexity considerations made for the collect algorithm also apply to the distribute algorithm. We close this section by another lemma that points out to be useful in later parts of this paper. Essentially, it establishes a relationship between the factor domains after the collect algorithm and the according node labels.

Lemma 8 *It holds that*

$$d(\psi_i^{(m)}) - \left(d(\psi_i^{(m)}) \cap \lambda(ch(i)) \right) = \lambda(i) - (\lambda(i) \cap \lambda(ch(i))) . \tag{6.10}$$

Proof. The left-hand part of this equation is clearly contained in the right-hand part, because $d(\psi_i^{(m)}) \subseteq \lambda(i)$. On the other hand, we deduce from Theorem 5 that $\lambda(i) = d(\psi_i^{(m)}) \cup (\lambda(i) \cap \lambda(ch(i)))$. Therefore, if $X \in \lambda(i)$ and $X \notin \lambda(i) \cap \lambda(ch(i)))$, we clearly have $X \in d(\psi_i^{(m)})$, so the right-hand part is contained in the left hand part. This proves the equality. □

7 Solution Configurations & Solution Extensions

We learned in the preceding section how to compute $\phi^{\downarrow\emptyset}(\diamond)$ efficiently by use of the collect algorithm. For the particular case of an optimization problem, we are therefore able to find out the optimum value of the objective function $\phi \in \Phi_s$ over all its configurations $\mathbf{x} \in \Omega_s$. However, if we are concretely looking for a configuration that adopts the computed value, we are again confronted with the computational intractability of building ϕ explicitly. This section is therefore dedicated to the process of identifying single or all existing configurations of ϕ whose value corresponds to the computed optimum $\phi^{\downarrow\emptyset}(\diamond)$. Following (Shenoy, 1996), we refer to such configurations as *solution configurations* for the objective function ϕ.

Definition 6 *Let (Φ, D) be a valuation algebra over a totally ordered, idempotent semiring. For $\phi \in \Phi_s$, we call $\mathbf{x} \in \Omega_s$ a solution configuration if $\phi(\mathbf{x}) = \phi^{\downarrow\emptyset}(\diamond)$.*

The set of all solution configurations for a given valuation ϕ is called *solution configuration set* and defined as:

$$c_\phi = \{\mathbf{x} \in \Omega_s : \phi(\mathbf{x}) = \phi^{\downarrow\emptyset}(\diamond)\}. \tag{7.1}$$

In the further process, we will often deal with *partial solution configuration sets* $c_\phi^{\downarrow t}$ for some $t \subseteq s$. We shall therefore remark that

$$c_\phi^{\downarrow t} = c_{\phi^{\downarrow t}}. \tag{7.2}$$

Another important property follows from $\mathbf{0} \leq \phi(\mathbf{x})$ for all $\mathbf{x} \in \Omega_s$ (see Lemma 4).

Lemma 9 *$\phi^{\downarrow\emptyset}(\diamond) = \mathbf{0}$ implies that $c_\phi = \Omega_s$.*

The existence of at least one solution configuration is an immediate consequence of the semiring's total order, as remarked in Lemma 6. We therefore have $c_\phi \neq \emptyset$ for every $\phi \in \Phi$. More generally, we can always find a *configuration extension* $\mathbf{c} \in \Omega_{s-t}$ such that $\phi(\mathbf{x}, \mathbf{c}) = \phi^{\downarrow t}(\mathbf{x})$ for all $t \subseteq s$ and $\mathbf{x} \in \Omega_t$.

Definition 7 *Let (Φ, D) be a valuation algebra over a totally ordered, idempotent semiring. For $\phi \in \Phi_s$, $t \subseteq s$ and $\mathbf{x} \in \Omega_t$ we define*

$$W_\phi^t(\mathbf{x}) = \{\mathbf{c} \in \Omega_{s-t} : \phi(\mathbf{x}, \mathbf{c}) = \phi^{\downarrow t}(\mathbf{x})\}. \tag{7.3}$$

It is very important to remark the close relationship between this definition and the solution configuration set c_ϕ. For the particular case where $t = \emptyset$ and therefore $\mathbf{x} = \diamond$, we have

$$W_\phi^\emptyset(\diamond) = c_\phi. \tag{7.4}$$

Similarly, if $t = d(\phi)$,

$$W_\phi^t(\mathbf{x}) = \{\diamond\} \tag{7.5}$$

for all $\mathbf{x} \in \Omega_t$. These results are generalized for any $t \subseteq d(\phi)$ in the following lemma, which follows directly from Definition 7.

Lemma 10 *For $\phi \in \Phi$ and $t \subseteq u \subseteq d(\phi)$ we have*

$$c_\phi^{\downarrow u} \;=\; \left\{(\mathbf{x},\mathbf{y}) \;:\; \mathbf{x} \in c_\phi^{\downarrow t} \text{ and } \mathbf{y} \in W^t_{\phi^{\downarrow u}}(\mathbf{x})\right\}.$$

Note that Equation (7.4) follows for $t = \emptyset$ and $u = d(\phi)$.

8 Explicit Identification of Solution Configurations

We now go about the task of identifying solution configurations. Former studies have shown that this must be done without the explicit computation of the objective function ϕ. Since we already know that local computation is a suitable way to avoid complexity problems when dealing with factorizations, it is proximate to embark on a similar strategy. This section presents a first class of methods that store partial solution configuration sets explicitly as lists of configurations.

8.1 Identifying all Solution Configurations with Distribute

Assume a given optimization problem with factor set $\{\phi_1, \ldots, \phi_n\}$. The most proximate approach to identify all solution configurations of $\phi = \phi_1 \otimes \cdots \otimes \phi_n$ follows directly from the two local computation schemes presented in Section 6. After a complete run of the collect and distribute algorithms, every join tree node i contains the marginal of ϕ relative to its node label $\lambda(i)$. From these marginals, we can build non-deterministically all solution configurations by the following recursive procedure. Due to Equations (7.2) and (7.4), we have for the root node m

$$c_\phi^{\downarrow\lambda(m)} \;=\; W^{\emptyset}_{\phi^{\downarrow\lambda(m)}}(\diamond). \tag{8.1}$$

Once such a partial solution configuration relative to some child node $ch(i)$ is known, we can extend each element to parent node label $\lambda(i)$ by use of Lemma 10. We have:

$$\mathbf{c} \in c_\phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \quad\Rightarrow\quad (\mathbf{c}, \mathbf{x}) \in c_\phi^{\downarrow\lambda(i)} \tag{8.2}$$

for $\mathbf{x} \in W^{\lambda(i)\cap\lambda(ch(i))}_{\phi^{\downarrow\lambda(i)}}(\mathbf{c})$. Thus, we obtain c_ϕ by the following procedure:

Algorithm 1:

- Execute collect and distribute on join tree factor set $\{\psi_1, \ldots, \psi_m\}$.
- Identify $c_\phi^{\downarrow\lambda(m)}$ by use of Equation (8.1).
- Build c_ϕ by repeated application of Equation (8.2).

This simple algorithm enumerates all solution configurations of ϕ and operates exclusively on factors whose size is bounded by the width of the underlying join tree. It therefore adopts the complexity of a local computation scheme. However, this approach is devalued from the fact, that it is always necessary to perform a complete run of the distribute algorithm. Thus, we shall study in the following section, under which restrictions one can pass on this full execution of distribute.

8.2 Identifying some Solution Configurations without Distribute

The second approach will essentially be based on the following theorem:

Theorem 6 *Let $\phi \in \Phi_s$, $\psi \in \Phi_t$ and $s \subseteq u \subseteq s \cup t$. For $\mathbf{x} \in \Omega_u$ we have*

$$W_\psi^{t\cap u}(\mathbf{x}^{\downarrow u\cap t}) \;\subseteq\; W^u_{\phi\otimes\psi}(\mathbf{x}).$$

Proof. Assume $\mathbf{x} \in \Omega_u$ and $\mathbf{c} \in W_\psi^{t\cap u}(\mathbf{x}^{\downarrow t\cap u})$. By Definition 7, we have

$$\psi(\mathbf{x}^{\downarrow t\cap u}, \mathbf{c}) \;=\; \psi^{\downarrow t\cap u}(\mathbf{x}^{\downarrow t\cap u}).$$

Hence, we also have

$$\phi(\mathbf{x}^{\downarrow t}) \times \psi(\mathbf{x}^{\downarrow t\cap u}, \mathbf{c}) \;=\; \phi(\mathbf{x}^{\downarrow t}) \times \psi^{\downarrow t\cap u}(\mathbf{x}^{\downarrow t\cap u}).$$

Then, by application of the definition of combination:

$$\begin{aligned}(\phi\otimes\psi)(\mathbf{x},\mathbf{c}) \;=\; \phi(\mathbf{x}^{\downarrow t}) \times \psi(\mathbf{x}^{\downarrow t\cap u}, \mathbf{c}) \;&=\; \phi(\mathbf{x}^{\downarrow t}) \times \psi^{\downarrow t\cap u}(\mathbf{x}^{\downarrow t\cap u}) \\ &=\; \left(\phi\otimes\psi^{\downarrow t\cap u}\right)(\mathbf{x}) \;=\; (\phi\otimes\psi)^{\downarrow u}(\mathbf{x}).\end{aligned}$$

We conclude from $(\phi\otimes\psi)(\mathbf{x},\mathbf{c}) = (\phi\otimes\psi)^{\downarrow u}(\mathbf{x})$ that $\mathbf{c} \in W^u_{\phi\otimes\psi}(\mathbf{x})$ holds. □

The following example proves that indeed only inclusion holds between the two solution extension sets in Theorem 6.

Example 17: We take the Bottleneck semiring from Example 2 with max for $+$ and min for $\times$. Assume the two semiring valuations ϕ and ψ with domains $d(\phi) = \{A\}$ and $d(\psi) = \{A, B\}$. The variable frames are $\Omega_A = \{a, \overline{a}\}$ and $\Omega_B = \{b, \overline{b}\}$.

ϕ =

$\Omega_{\{A\}}$	
a	1
$\overline{a}$	1

ψ =

$\Omega_{\{A,B\}}$		
a	b	6
a	$\overline{b}$	7
$\overline{a}$	b	8
$\overline{a}$	$\overline{b}$	9

$\phi\otimes\psi$ =

$\Omega_{\{A,B\}}$		
a	b	1
a	$\overline{b}$	1
$\overline{a}$	b	1
$\overline{a}$	$\overline{b}$	1

For $u = \{A\} = u \cap t$, we have

$\psi^{\downarrow u\cap t}$ =

$\Omega_{\{A\}}$	
a	7
$\overline{a}$	9

$(\phi\otimes\psi)^{\downarrow u}$ =

$\Omega_{\{A\}}$	
a	1
$\overline{a}$	1

and finally $W_\psi^{u\cap t}(a) = W_\psi^{u\cap t}(\overline{a}) = \{(\overline{b})\} \subset W^u_{\phi\otimes\psi}(a) = W^u_{\phi\otimes\psi}(\overline{a}) = \{(b), (\overline{b})\}$. ⊖

We show next, how single solution configurations can non-deterministically be constructed by use of Theorem 6. The advantage of this method compared with the

procedure developed in Section 8.1 is that we can completely pass on the execution of the distribute algorithm and reconstruct the solution configurations only from the results of the collect phase. On the other hand, this method can only be used for the identification of *some* solution configurations. The algorithm itself will be described as a two-phase local computation scheme and it mainly corresponds to the classical approach given in (Shenoy, 1996). The first phase of the algorithm consists in the execution of collect as presented in Section 6.1 but, in contrast, we equip each join tree node with an additional storage that allows to store backtracking information for the second phase. Once this modified collect algorithm is completed, we propagate from the root node downwards to the leaves of the join tree and build a solution configuration from the previously stored information. The according formula is obtained directly from Lemma 10:

Corollary 1 *It holds that*

$$c_\phi^{\downarrow\lambda(i)} = \left\{ (\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))},\ \mathbf{x}) : \mathbf{c} \in c_\phi^{\downarrow\lambda(ch(i))},\ \mathbf{x} \in W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))}) \right\}.$$

In order to apply this formula recursively in a downward propagation phase, we need to ensure that the root node m can be initialized. Since the root contains $\phi^{\downarrow\lambda(m)}$ at the end of the collect algorithm, this initialization can again be computed by Equation (8.1).

At this point, we need to analyze the statement of Corollary 1 carefully. With the above initialization of the root node and a recursive application of this formula, we would be able to identify the complete set of solution configurations c_ϕ as it has been done in Section 8.1. But this theorem presupposes that $\phi^{\downarrow\lambda(i)}$ is known for all i and this requires again a full execution of distribute. Remember, this is what we try to avoid. The following lemma will be useful for this undertaking. Here, ω_i denotes the domain of the content of node i at completion of the collect algorithm, i.e. $\omega_i = d(\psi_i^{(m)})$.

Lemma 11 *For* $\mathbf{c} \in c_\phi^{\downarrow\lambda(ch(i))}$ *it holds that*

$$W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))}) \ = \ W_{\phi^{\downarrow\omega_i}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}).$$

Proof. For $\mathbf{c} \in c_\phi^{\downarrow\lambda(ch(i))}$ we have $\mathbf{x} \in W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))})$ if and only if

$$\phi^{\downarrow\lambda(i)\cap\lambda(i)}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))}) \ = \ \phi^{\downarrow\lambda(i)}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))},\ \mathbf{x}).$$

Since $\omega_i \subseteq \lambda(i)$ and therefore $\omega_i \cap \lambda(ch(i)) \subseteq \lambda(i) \cap \lambda(ch(i))$, we conclude that

$$\phi^{\downarrow\lambda(i)\cap\lambda(i)}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))}) \ = \ \phi^{\downarrow\omega_i\cap\lambda(i)}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}),$$

and correspondingly

$$\phi^{\downarrow\lambda(i)}(\mathbf{c}^{\downarrow\lambda(i)\cap\lambda(ch(i))},\ \mathbf{x}) \ = \ \phi^{\downarrow\omega_i}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))},\ \mathbf{x}).$$

The fact that $\mathbf{x} \in \Omega_{\lambda(i)-(\lambda(i)\cap\lambda(i))} = \Omega_{\omega_i-(\omega_i\cap\lambda(i))}$ follows from Lemma 8. Thus, we have shown that

$$\phi^{\downarrow\omega_i\cap\lambda(i)}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}) \;=\; \phi^{\downarrow\omega_i}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))},\; \mathbf{x}),$$

which is the case if and only if $\mathbf{x} \in W_{\phi^{\downarrow\omega_i}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))})$. □

We conclude therefore that we can replace the node labels $\lambda(i)$ in Corollary 1 by ω_i. This is an important preparation step for the following transformations. From the distribute algorithm, we know that

$$\phi^{\downarrow\omega_i} \;=\; \left(\phi^{\downarrow\lambda(i)}\right)^{\downarrow\omega_i} \;=\; \left(\psi_i^{(m)} \otimes \mu_{ch(i)\to i}\right)^{\downarrow\omega_i} \;=\; \psi_i^{(m)} \otimes \mu_{ch(i)\to i}^{\downarrow\omega_i\cap\lambda(ch(i))}.$$

Hence, by application of Theorem 6, we find

$$\begin{aligned} W_{\phi^{\downarrow\omega_i}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}) &= W_{\mu_{ch(i)\to i}^{\downarrow\omega_i\cap\lambda(ch(i))}\otimes\psi_i^{(m)}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}) \\ &\supseteq W_{\psi_i^{(m)}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}^{\downarrow\omega_i\cap\lambda(ch(i))}) \supset \emptyset. \end{aligned} \tag{8.3}$$

This means for the statement of Corollary 1 that a possible solution extension for $\phi^{\downarrow\lambda(i)}$ can always be found in the solution extension set relative to $\psi_i^{(m)}$ and since the latter can be computed from the node content at completion of collect, we do not depend on the execution of distribute anymore. However, as we have already seen, the prize we pay for this gain of efficiency is that some solution configurations get lost. Consequently, this algorithm can only be used to find some solution configuration - it is in general impossible to identify c_ϕ completely. We round out this section by putting the different components of the algorithm together:

Algorithm 2:

- Execute collect on join tree factor set $\{\psi_1, \ldots, \psi_m\}$.
- Compute $c_\phi^{\downarrow\lambda(m)}$ by Equation (8.1).
- Choose $\mathbf{c}_m \in c_\phi^{\downarrow\lambda(m)}$ and proceed recursively for $i = m-1, \ldots, 1$:
  - compute $w_i \;=\; W_{\psi_i^{(m)}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{c}_{ch(i)}^{\downarrow\omega_i\cap\lambda(ch(i))})$,
  - choose $\mathbf{x} \in w_i$ and set $\mathbf{c}_i = \left(\mathbf{c}_{ch(i)}^{\downarrow\lambda(i)\cap\lambda(ch(i))},\; \mathbf{x}\right)$.
- Build $\mathbf{c} \in c_\phi$ from $\mathbf{c}_i = \mathbf{c}^{\downarrow\lambda(i)}$.

We illustrate Algorithm 2 by a concrete example:

Example 18: We consider binary variables A, B, C with frames $\Omega_A = \{a, \overline{a}\}$ to $\Omega_C = \{c, \overline{c}\}$. Let ψ_1 to ψ_3 be three join tree factors defined over the Bottleneck semiring from Example 2 with domains $d(\psi_1) = \{A, B\}$, $d(\psi_2) = \{B, C\}$, $\psi_3 = \{B\}$ and the following values:

$\psi_1 =$

$\Omega_{\{A,B\}}$		
a	b	2
a	$\overline{b}$	4
$\overline{a}$	b	3
$\overline{a}$	$\overline{b}$	2

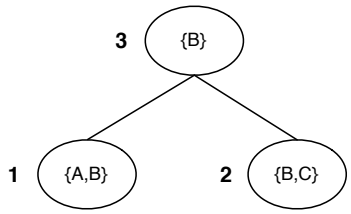
$\psi_2 =$

$\Omega_{\{B,C\}}$		
b	c	5
b	$\overline{c}$	2
$\overline{b}$	c	3
$\overline{b}$	$\overline{c}$	3

$\psi_3 =$

$\Omega_{\{B\}}$	
b	1
$\overline{b}$	6

A join tree that corresponds to this factorization und numbering is shown next:



Let us first compute $\phi = \psi_1 \otimes \psi_2 \otimes \psi_3$ directly such that later results can be verified:

$\phi =$

$\Omega_{\{A,B,C\}}$			
a	b	c	1
a	b	$\overline{c}$	1
a	$\overline{b}$	c	3
a	$\overline{b}$	$\overline{c}$	3
$\overline{a}$	b	c	1
$\overline{a}$	b	$\overline{c}$	2
$\overline{a}$	$\overline{b}$	c	2
$\overline{a}$	$\overline{b}$	$\overline{c}$	2

We see that $c_\phi = \{(a, \overline{b}, c), (a, \overline{b}, \overline{c})\}$. We now start the above algorithm by executing a complete collect run:

$\mu_{1\to 3} =$

$\Omega_{\{B\}}$	
b	3
$\overline{b}$	4

$\psi_3^{(2)} =$

$\Omega_{\{B\}}$	
b	1
$\overline{b}$	4

$\mu_{2\to 3} =$

$\Omega_{\{B\}}$	
b	5
$\overline{b}$	3

$\psi_3^{(3)} =$

$\Omega_{\{B\}}$	
b	1
$\overline{b}$	3

The maximum value of ϕ is $\phi^{\downarrow\emptyset}(\diamond) = \psi_3^{(3)\downarrow\emptyset}(\diamond) = 3$. Next we compute

$$c_\phi^{\downarrow\{B\}} \;=\; W^{\emptyset}_{\phi^{\downarrow\{B\}}}(\diamond) \;=\; W^{\emptyset}_{\psi_3^{(3)}}(\diamond) \;=\; \{(\overline{b})\}.$$

We therefore choose $\mathbf{c}_3 = (\overline{b})$ and proceed for $i = 2$:

$$W^{\{B\}}_{\psi_2^{(3)}}(\overline{b}) = \{(c), (\overline{c})\}, \text{ we choose } (c) \;\rightsquigarrow\; \mathbf{c}_2 = (\overline{b}, c).$$

Finally, for $i = 1$ we obtain:

$$W^{\{B\}}_{\psi_1^{(3)}}(\overline{b}) = \{(a)\} \;\rightsquigarrow\; \mathbf{c}_1 = (a, \overline{b}).$$

We identified the configuration $\mathbf{c} = (a, \overline{b}, c) \in c_\phi$ with $\phi(a, \overline{b}, c) = 3$. Note that we can find the second solution configuration of ϕ by choosing the partial configuration $(\overline{c})$ in step 2. In this special case, it is therefore possible to identify c_ϕ completely. However, as we already know, this is generally not the case. Furthermore, we can not even know without computing ϕ explicitly if all solution configurations have been found or not. The user may convince himself by applying this algorithm to the factor set given in Example 17. $\ominus$

8.3 Identifying all Solution Configurations without Distribute

So far, we have seen that all solution configurations can be found if we agree to perform a complete run of the distribute algorithm. Without this computational effort, only some solution configurations can be identified. However, this section will show that if we impose further restrictions on the underlying semiring, all solution configurations are acquired even with the second method. For this purpose, we remark that equality in Theorem 6 can be achieved if the semiring operator of multiplication is strictly monotonic, according to Definition 2.

Theorem 7 *Let (Φ, D) be a valuation algebra induced by a totally ordered, idempotent semiring whose order behaves strictly monotonic over $\times$. If $\phi \in \Phi_s$, $\psi \in \Phi_t$ and $s \subseteq u \subseteq s \cup t$, we have for all $\mathbf{x} \in \Omega_u$ with $\phi(\mathbf{x}^{\downarrow s}) \neq \mathbf{0}$*

$$W_\psi^{u\cap t}(\mathbf{x}^{\downarrow u\cap t}) \;=\; W^u_{\phi\otimes\psi}(\mathbf{x}).$$

Proof. It remains to prove that

$$W_\psi^{u\cap t}(\mathbf{x}^{\downarrow u\cap t}) \;\supseteq\; W^u_{\phi\otimes\psi}(\mathbf{x}).$$

Assume $\mathbf{x} \in \Omega_u$ with $\phi(\mathbf{x}^{\downarrow s}) \neq \mathbf{0}$ and $\mathbf{c} \in W^u_{\phi\otimes\psi}(\mathbf{x})$. By Definition 7, we have

$$(\phi \otimes \psi)^{\downarrow u}(\mathbf{x}) \;=\; (\phi \otimes \psi)(\mathbf{x}, \mathbf{c}).$$

Applying the definition of combination, we obtain

$$(\phi \otimes \psi)(\mathbf{x}, \mathbf{c}) \;=\; \phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow u\cap t}, \mathbf{c}).$$

Similarly, we deduce from the Combination axiom and the definition of combination

$$(\phi \otimes \psi)^{\downarrow u}(\mathbf{x}) \;=\; (\phi \otimes \psi^{\downarrow u\cap t})(\mathbf{x}) \;=\; \phi(\mathbf{x}^{\downarrow s}) \times \psi^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t}).$$

Therefore,

$$\phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow u\cap t}, \mathbf{c}) \;=\; \phi(\mathbf{x}^{\downarrow s}) \times \psi^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t}).$$

From Lemma 7, we conclude that

$$\psi(\mathbf{x}^{\downarrow u\cap t}, \mathbf{c}) \;=\; \psi^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t})$$

and therefore $\mathbf{c} \in W_{\psi}^{u \cap t}(\mathbf{x}^{\downarrow u \cap t})$. □

This result can now be used in Equation (8.3) if we assume that

$$\mu_{ch(i) \to i}^{\downarrow \omega_i \cap \lambda(ch(i))}(\mathbf{c}^{\downarrow \omega_i \cap \lambda(ch(i))}) \neq \mathbf{0}.$$

Because $\mathbf{c}$ is a solution configuration, we would have

$$\phi^{\downarrow \emptyset}(\diamond) = \phi^{\downarrow \lambda(i)}(\mathbf{c}^{\downarrow \lambda(i)}) = \mu_{ch(i) \to i}^{\downarrow \omega_i \cap \lambda(ch(i))}(\mathbf{c}^{\downarrow \omega_i \cap \lambda(ch(i))}) \times \psi_i^{(m)}(\mathbf{c}^{\downarrow \omega_i}) = \mathbf{0},$$

if this condition was not satisfied, and this in turn implies that all configurations of ϕ are solution configurations due to Lemma 9. In this case, we can omit the construction of the solution configuration set anyway. Hence, we adopt this assumption and obtain

$$W_{\phi^{\downarrow \omega_i}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{c}^{\downarrow \omega_i \cap \lambda(ch(i))}) = W_{\psi_i^{(m)}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{c}^{\downarrow \omega_i \cap \lambda(ch(i))}), \tag{8.4}$$

which allows to build any solution configuration in a similar way to Algorithm 1. Again, we obtain from Lemma 10 and Equation (8.4):

$$\mathbf{c} \in c_{\phi}^{\downarrow \omega_i \cap \lambda(ch(i))} \Rightarrow (\mathbf{c}, \mathbf{x}) \in c_{\phi}^{\downarrow \lambda(i)} \tag{8.5}$$

for $\mathbf{x} \in W_{\psi_i^{(m)}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{c})$. To sum it up, the following algorithm determines c_ϕ in the case of a total order that is strictly monotonic over $\times$.

Algorithm 3:

- Execute collect on join tree factor set $\{\psi_1, \ldots, \psi_m\}$.
- Compute $\phi^{\downarrow \emptyset}(\diamond) = (\phi^{\downarrow \lambda(m)})^{\downarrow \emptyset}(\diamond)$.
- If $\phi^{\downarrow \emptyset}(\diamond) = \mathbf{0}$ return $c_\phi = \Omega_{d(\phi)}$.
- Otherwise, identify $c_m = c_{\phi}^{\downarrow \lambda(m)}$ by Equation (8.1).
- Build c_ϕ by repeated application of Equation (8.5).

9 Implicit Identification of Solution Configurations

The various approaches presented in the foregoing section explicitly store partial solution configurations from which the final solution configuration set c_ϕ is reconstructed. However, for some practical applications we often prefer to have a more compact representation of the solution configuration set that does in no way depend on the number of its elements. We shall give some concrete examples of such scenarios:

- *Solution Counting:* Determining the number of solution configurations $|c_\phi|$ is an important task in many applications and it is clearly needless in such a case to explicitly enumerate all elements in c_ϕ.
- *Counter-Solution Configurations:* Another example where it is unhandy to enumerate c_ϕ is the identification of configurations that are not solution configurations, i.e. the set $\overline{c}_\phi = \Omega_s - c_\phi$.

For the more efficient treatment of these and related *queries*, we propose in this section an alternative method that completely pass on the explicit enumeration of configurations. Instead, we *compile* solution configurations into a Boolean function $f : \{0,1\}^n \to \{0,1\}$, such that they are reflected by the *models* of the latter. By choosing then a suitable representation of the Boolean function that results from this compilation process, we obtain a very compact graphical structure that makes the fast execution of a large number of different queries possible. In order to stress this duality of solution configurations and models of Boolean functions, we restrict ourselves to *propositional variables*, i.e. variables with binary frames. Nevertheless, we accent that a generalization to arbitrary finite variables is possible (Wachter & Haenni, 2007). In order to link frame values with their according variable, we denote the frame values of a variable X as $\Omega_X = \{0_\mathrm{X}, 1_\mathrm{X}\}$, without changing their usual interpretation as truth values. Hence, the task of interest in this section is to construct a Boolean function f such that

$$\mathbf{Models}(f) = \{\mathbf{x} \in \Omega_s : f(\mathbf{x}) = 1\} \;=\; c_\phi. \tag{9.1}$$

We shall concretize this idea by a first example:

Example 19: Consider propositional variables A, B, C and the following valuation defined over a semiring with max for $+$.

$\phi \;=$

$\Omega_{\{\mathrm{A,B,C}\}}$			
0_A	0_B	0_C	5
0_A	0_B	1_C	10
0_A	1_B	0_C	10
0_A	1_B	1_C	3
1_A	0_B	0_C	9
1_A	0_B	1_C	0
1_A	1_B	0_C	7
1_A	1_B	1_C	3

We have $\phi^{\downarrow\emptyset}(\diamond) = 10$ and therefore $c_\phi = \{(0_\mathrm{A}, 0_\mathrm{B}, 1_\mathrm{C}),\ (0_\mathrm{A}, 1_\mathrm{B}, 0_\mathrm{C})\}$. This solution configuration set can be described by the following *propositional formula*:

$$f \ = \ \neg A \wedge ((\neg B \wedge C) \vee (B \wedge \neg C)) \, .$$

As we can see, $f(\mathbf{x}) = 1$ if and only if $\mathbf{x} \in c_\phi$. We have found a propositional formula that represents the solution configuration set and since every formula corresponds to a Boolean function, the keynote of this section is well illustrated. $\ominus$

A similar strategy is embarked in (Mateescu & Dechter, 2006). There, already the knowledgebase factors (constraints) are compiled into a particular Boolean function and graphically represented as *AND/OR multi-valued decision diagrams*. The compilation process consists then in combining these graphs to the final compilation of the solution configuration set, with special focus on the retention of properties that make fast query execution possible. However, in our approach, the compilation process is independent from the representation of semiring valuations. Furthermore, the underlying join tree guarantees that all properties needed for fast query execution are naturally retained during the compilation process.

9.1 Marginalization as Variable Elimination

Compared with former sections, it is a lot more adequate to develop this theory with the operation of marginalization replaced by *variable elimination*

$$\phi^{-X} \quad = \quad \phi^{\downarrow d(\phi) - \{X\}}. \tag{9.2}$$

This leads to the following set of axioms that replaces the system introduced in Section 2. We refer to (Kohlas, 2003) for an equivalence proof of the two systems.

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

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

$$d(\phi \otimes \psi) = d(\phi) \cup d(\psi). \tag{9.3}$$

(A3*) *Variable Elimination:* For $\phi \in \Phi$ and $X \in d(\phi)$,

$$d(\phi^{-X}) = d(\phi) - \{X\}. \tag{9.4}$$

(A4*) *Commutativity of Elimination:* For $\phi \in \Phi$ and $X, Y \in d(\phi)$,

$$(\phi^{-X})^{-Y} = (\phi^{-Y})^{-X}. \tag{9.5}$$

(A5*) *Combination:* For $\phi, \psi \in \Phi$ with $X \notin d(\phi)$, $X \in d(\psi)$,

$$(\phi \otimes \psi)^{-X} = \phi \otimes \psi^{-X}. \tag{9.6}$$

Due to axiom (A4*), variables can be eliminated in any order. Thus we can write the consecutive elimination of a non-empty set of variables $s = \{X_1, \ldots, X_n\}$ as

$$\phi^{-s} = (((\phi^{-X_1})^{-X_2}) \ldots)^{-X_n}.$$

This allows to express the operation of marginalization as variable elimination:

$$\phi^{\downarrow t} \;=\; \begin{cases} \phi^{-(d(\phi)-t)}, & \text{if } t \subset d(\phi) \\ \phi, & \text{otherwise.} \end{cases} \tag{9.7}$$

To sum it up, we are allowed to switch between marginalization and variable elimination according to convenience.

9.2 Memorizing Semiring Valuations

We assume a finite set of propositional variables r and write $P(r)$ for its power set. Let A be a totally ordered, idempotent semiring whose order is strictly monotonic over $\times$. A *memorizing semiring valuation* ϕ with domain $s \subseteq r$ is defined to be a function that associates a two-dimensional vector with each configuration $\mathbf{x} \in \Omega_s$. The first element of this vector $\phi_A(\mathbf{x})$ corresponds again to a semiring value whereas the second element $\phi_F(\mathbf{x})$ constitutes a Boolean function defined over propositional variables in r. More formally, we have

$$\phi : \Omega_s \to A \times F_r \;\text{ and }\; \mathbf{x} \mapsto (\phi_A(\mathbf{x}),\ \phi_F(\mathbf{x})), \tag{9.8}$$

where F_r is the set of Boolean functions over variables in r,

$$F_r \;=\; \{f : \{0,1\}^r \to \{0,1\}\}.$$

Essentially, the construction of a memorizing semiring valuation is similar to the one of usual semiring valuations introduced in Section 4, with the extension that we associate additionally a Boolean function with each configuration. This new component represents the memory part and will be updated whenever a step towards the identification of solution configurations is performed. Note that we assume the Boolean functions to be defined over all variables in r. Since Boolean functions are considered in this context as a part of a configuration's value, they have no meaning with respect to the domain of a memorizing semiring valuation, which is defines as

$$d(\phi) = s \;\text{ if }\; \phi : \Omega_s \to A \times F_r. \tag{9.9}$$

It is proximate to assume the memory originally to be empty. Thus, memorizing semiring valuations are obtained from usual semiring valuations by initializing their memory to the *tautology function* $f_{\mathrm{T}} \in F_r$ with $f_{\mathrm{T}}(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega_r$. In this way, we define for a semiring valuation ψ with $d(\psi) = t$,

$$\tilde{\psi} :\ \mathbf{x} \in \Omega_t \;\;\mapsto\;\; (\psi(\mathbf{x}),\ f_{\mathrm{T}})\,. \tag{9.10}$$

We shall now examine how memorizing semiring valuations are processed and how their memory component is updated. Assume that we eliminate some propositional variable Y from a memorizing semiring valuation ϕ. The semiring components are processed in the usual way as

$$\phi_A^{-Y}(\mathbf{x}) \;=\; \phi_A(\mathbf{x},0) + \phi_A(\mathbf{x},1) \;=\; \max\{\phi_A(\mathbf{x},0),\ \phi_A(\mathbf{x},1)\}.$$

For the memory component, we distinguish the following three cases:

- If $\phi_A(\mathbf{x},0_\mathrm{Y}) < \phi_A(\mathbf{x},1_\mathrm{Y})$, $Y = 1$ must hold in every solution configuration of ϕ. In terms of Boolean functions, this requirement is reflected by the identity function $f_Y \in F_r$ that maps always on the current value of Y. Additionally, we can ignore the memory $\phi_F(\mathbf{x},0_\mathrm{Y})$ since it would lead to a configuration where $Y = 0$, and this cannot be a solution configuration. Hence, we connect the memory $\phi_F(\mathbf{x},1_\mathrm{Y})$ conjunctively with f_Y and obtain $\phi_F(\mathbf{x}) = \min\{f_Y, \phi_F(\mathbf{x},1_\mathrm{Y})\}$. Alternatively, we may also write $f_Y \wedge \phi_F(\mathbf{x},1_\mathrm{Y})$ to accent the conjunctive combination of the two Boolean functions.
- If on the other hand $\phi_A(\mathbf{x},1_\mathrm{Y}) < \phi_A(\mathbf{x},0_\mathrm{Y})$, $Y = 0$ must hold in every solution configuration and this can be achieved by the inverse identity function $f_{\overline{Y}} \in F_r$. Following the same argumentation as above, we obtain $\phi_F(\mathbf{x}) = \min\{f_{\overline{Y}}, \phi_F(\mathbf{x},0_\mathrm{Y})\} = f_{\overline{Y}} \wedge \phi_F(\mathbf{x},0_\mathrm{Y})$.
- Finally, if $\phi_A(\mathbf{x},0_\mathrm{Y}) = \phi_A(\mathbf{x},1_\mathrm{Y})$, both assignments $Y = 0$ and $Y = 1$ lead to solution configurations. Therefore, the memory can either be updated as $\min\{f_Y, \phi_F(\mathbf{x},1_\mathrm{Y})\}$ or as $\min\{f_{\overline{Y}}, \phi_F(\mathbf{x},0_\mathrm{Y})\}$. Since we aim for the identification of all solution configurations, we connect the two updates disjunctively and obtain $\phi_F(\mathbf{x}) = \max\{\min\{f_Y, \phi_F(\mathbf{x},1_\mathrm{Y})\}, \min\{f_{\overline{Y}}, \phi_F(\mathbf{x},0_\mathrm{Y})\}\}$. In the same way, we may also write $(f_Y \wedge \phi_F(\mathbf{x},1_\mathrm{Y})) \vee (f_{\overline{Y}} \wedge \phi_F(\mathbf{x},0_\mathrm{Y}))$.

From this train of thought, we derive the following definition of variable elimination for the set Φ of memorizing semiring valuations over propositional variables in r:

Variable Elimination: $\Phi \times r \to \Phi$: for $\phi \in \Phi$, $Y \in d(\phi)$ and $\mathbf{x} \in \Omega_{d(\phi)-\{Y\}}$

$$\phi^{-Y}(\mathbf{x}) \;=\; (\phi_A^{-Y}(\mathbf{x}),\ \phi_F^{-Y}(\mathbf{x})) \tag{9.11}$$

where

$$\phi_A^{-Y}(\mathbf{x}) \;=\; \phi_A(\mathbf{x},0_\mathrm{Y}) + \phi_A(\mathbf{x},1_\mathrm{Y})$$

and

$$\phi_F^{-Y}(\mathbf{x}) \;=\; \begin{cases} f_{\overline{Y}} \wedge \phi_F(\mathbf{x},0_\mathrm{Y}), & \text{if } \phi_A(\mathbf{x},0_\mathrm{Y}) > \phi_A(\mathbf{x},1_\mathrm{Y}) \\ f_Y \wedge \phi_F(\mathbf{x},1_\mathrm{Y}), & \text{if } \phi_A(\mathbf{x},0_\mathrm{Y}) < \phi_A(\mathbf{x},1_\mathrm{Y}) \\ (f_{\overline{Y}} \wedge \phi_F(\mathbf{x},0_\mathrm{Y})) \vee (f_Y \wedge \phi_F(\mathbf{x},1_\mathrm{Y})) & \text{otherwise.} \end{cases}$$

The following example illustrates the application of this rules:

Example 20: Consider propositional variables A, B and the following valuation defined over a semiring with max for $+$:

ϕ =

$\Omega_{\{A,B\}}$			
0_A	0_B	10	f_T
0_A	1_B	8	f_T
1_A	0_B	3	f_T
1_A	1_B	10	f_T

ϕ^{-A} =

Ω_B		
0_B	10	$f_\mathrm{T} \wedge f_{\overline{A}}$
1_B	10	$f_\mathrm{T} \wedge f_A$

$\phi^{-\{A,B\}}$ =

$\Omega_\emptyset$		
$\diamond$	10	$(f_{\overline{B}} \wedge (f_\mathrm{T} \wedge f_{\overline{A}})) \vee (f_B \wedge (f_\mathrm{T} \wedge f_A))$

We have

$$\begin{aligned} f &= \left(\phi^{\downarrow\emptyset}\right)_F &&= (f_{\overline{B}} \wedge (f_\mathrm{T} \wedge f_{\overline{A}})) \vee (f_B \wedge (f_\mathrm{T} \wedge f_A)) \\ &&&= (f_{\overline{B}} \wedge f_{\overline{A}}) \vee (f_B \wedge f_A), \end{aligned}$$

and finally

$$\mathbf{Models}(f) = \{(0_\mathrm{A}, 0_\mathrm{B}), (1_A, 1_\mathrm{B})\} = c_\phi.$$

$\ominus$

The following lemma links the memory component of memorizing semiring valuations with the solution extension sets introduced in Section 7.

Lemma 12 *For $\mathbf{x} \in \Omega_t$ we have*

$$W_\phi^t(\mathbf{x}) = \left[\mathbf{Models}\left(\phi_F^{\downarrow t}(\mathbf{x})\right)\right]^{\downarrow d(\phi)-t}. \tag{9.12}$$

Proof. For $t = d(\phi)$ and $\mathbf{y} \in \Omega_t$, we have

$$W_\phi^{d(\phi)}(\mathbf{y}) = \{\diamond\} = (\Omega_t)^{\downarrow\emptyset} = [\mathbf{Models}(f_\mathrm{T})]^{\downarrow\emptyset} = [\mathbf{Models}(\phi_F(\mathbf{y}))]^{\downarrow\emptyset}.$$

Next, we assume that the equation holds for $t \subseteq d(\phi)$ and prove its validity for $t - \{X\}$ with $X \in t$. Let $\mathbf{x} \in \Omega_t$, we distinguish the following cases:

1. Assume $\phi(\mathbf{x}, 0_\mathrm{X}) > \phi(\mathbf{x}, 1_\mathrm{X})$. We have

$$\begin{aligned} W_\phi^{t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}}) &= \left\{(\mathbf{c}, 0_\mathrm{X}) : \mathbf{c} \in W_\phi^t(\mathbf{x})\right\} \\ &= \left\{(\mathbf{c}, 0_\mathrm{X}) : \mathbf{c} \in \left[\mathbf{Models}(\phi_F^{\downarrow t}(\mathbf{x}))\right]^{\downarrow d(\phi)-t}\right\} \\ &= \left[\mathbf{Models}\left(f_{\overline{X}} \wedge \phi_F^{\downarrow t}(\mathbf{x})\right)\right]^{\downarrow d(\phi)-(t-\{X\})} \\ &= \left[\mathbf{Models}\left(\phi_F^{\downarrow t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}})\right)\right]^{\downarrow d(\phi)-(t-\{X\})}. \end{aligned}$$

2. Assume $\phi(\mathbf{x}, 0_{\mathrm{X}}) < \phi(\mathbf{x}, 1_{\mathrm{X}})$. We have

$$\begin{aligned} W_{\phi}^{t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}}) &= \big\{(\mathbf{c}, 1_{\mathrm{X}}) : \mathbf{c} \in W_{\phi}^{t}(\mathbf{x})\big\} \\ &= \left\{(\mathbf{c}, 1_{\mathrm{X}}) : \mathbf{c} \in \Big[\mathbf{Models}(\phi_F^{\downarrow t}(\mathbf{x}))\Big]^{\downarrow d(\phi)-t}\right\} \\ &= \Big[\mathbf{Models}\Big(f_X \wedge \phi_F^{\downarrow t}(\mathbf{x})\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})} \\ &= \Big[\mathbf{Models}\Big(\phi_F^{\downarrow t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}})\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})}. \end{aligned}$$

3. Assume $\phi(\mathbf{x}, 0_{\mathrm{X}}) = \phi(\mathbf{x}, 1_{\mathrm{X}})$. We have

$$\begin{aligned} W_{\phi}^{t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}}) &= \big\{(\mathbf{c}, 0_{\mathrm{X}}) : \mathbf{c} \in W_{\phi}^{t}(\mathbf{x})\big\} \cup \big\{(\mathbf{c}, 1_{\mathrm{X}}) : \mathbf{c} \in W_{\phi}^{t}(\mathbf{x})\big\} \\ &= \left\{(\mathbf{c}, 0_{\mathrm{X}}) : \mathbf{c} \in \Big[\mathbf{Models}(\phi_F^{\downarrow t}(\mathbf{x}))\Big]^{\downarrow d(\phi)-t}\right\} \cup \\ &\quad \left\{(\mathbf{c}, 1_{\mathrm{X}}) : \mathbf{c} \in \Big[\mathbf{Models}(\phi_F^{\downarrow t}(\mathbf{x}))\Big]^{\downarrow d(\phi)-t}\right\} \\ &= \Big[\mathbf{Models}\Big(f_{\overline{X}} \wedge \phi_F^{\downarrow t}(\mathbf{x})\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})} \cup \\ &\quad \Big[\mathbf{Models}\Big(f_X \wedge \phi_F^{\downarrow t}(\mathbf{x})\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})} \\ &= \Big[\mathbf{Models}\Big((f_{\overline{X}} \wedge \phi_F^{\downarrow t}(\mathbf{x})) \vee (f_X \wedge \phi_F^{\downarrow t}(\mathbf{x}))\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})} \\ &= \Big[\mathbf{Models}\Big(\phi_F^{\downarrow t-\{X\}}(\mathbf{x}^{\downarrow t-\{X\}})\Big)\Big]^{\downarrow d(\phi)-(t-\{X\})}. \end{aligned}$$

□

The following theorem follows immediately from Lemma 12 and Equation (7.4):

Theorem 8 *It holds that*

$$\mathbf{Models}\Big(\phi_F^{\downarrow \emptyset}(\diamond)\Big) \;=\; W_{\phi}^{\emptyset}(\diamond) \;=\; c_{\phi}. \tag{9.13}$$

Consequently, we found an implicit representation of the solution configuration set of ϕ. Since in our case, ϕ is generally given by a factorization $\phi = \psi_1 \otimes \cdots \otimes \psi_m$, we shall next introduce the combination of memorizing semiring valuations. Again, the semiring components are processed in the usual way,

$$(\phi \otimes \psi)_A(\mathbf{x}) \;=\; \phi_A(\mathbf{x}^{\downarrow d(\phi)}) \times \psi_A(\mathbf{x}^{\downarrow d(\psi)}).$$

Equally simple is the definition for the memory components. Since both memories must be taken into account, they are joined conjunctively as follows:

$$(\phi \otimes \psi)_F(\mathbf{x}) \;=\; \min\{\phi_F(\mathbf{x}^{\downarrow d(\phi)}),\ \psi_F(\mathbf{x}^{\downarrow d(\psi)})\} \;=\; \phi_F(\mathbf{x}^{\downarrow d(\phi)}) \wedge \psi_F(\mathbf{x}^{\downarrow d(\psi)}).$$

Hence, we adopt the following definition of combination for memorizing semiring valuations:

Combination: $\Phi \times \Phi \to \Phi$: for $\phi \in \Phi_s$, $\psi \in \Phi_t$ and $\mathbf{x} \in \Omega_{s \cup t}$

$$(\phi \otimes \psi)(\mathbf{x}) \quad = \quad (\phi_A(\mathbf{x}^{\downarrow s}) \times \psi_A(\mathbf{x}^{\downarrow t}),\ \phi_F(\mathbf{x}^{\downarrow s}) \wedge \psi_F(\mathbf{x}^{\downarrow t})). \tag{9.14}$$

To sum it up, the Boolean function for the solution configuration set of a factorization $\phi = \psi_1 \otimes \cdots \otimes \psi_m$ of semiring valuations can be computed as follows:

1. Construct memorizing semiring valuations $\tilde{\psi}_i$ according to Equation (9.10).
2. Compute $\tilde{\phi}$. We obtain $\tilde{\phi}_A(\mathbf{x}) = (\psi_1 \otimes \cdots \otimes \psi_m)(\mathbf{x})$ and $\tilde{\phi}_F(\mathbf{x}) = f_{\mathrm{T}}$.
3. Finally, build $\tilde{\phi}_F^{\downarrow \emptyset}(\diamond)$ by eliminating all variables in $d(\phi)$.

This sketch of algorithm suffers from the well-known problem that we cannot build $\tilde{\phi}$ explicitly due to the computational intractability addressed in Section 5. For this reason, the following theorem provides an alternative way to tackle this problem.

Theorem 9 *A system of memorizing semiring valuations (Φ, D) with labeling (9.9), variable elimination (9.11) and combination (9.14), satisfies the axioms of a valuation algebra.*

Proof. The axioms (A2*) and (A3*) are immediate consequences of the above definitions. Furthermore, we have also seen that the commutativity and associativity of max and min (written as $\vee$ and $\wedge$) are sufficient conditions for axiom (A1*). It remains to prove that the two axioms (A4*) and (A5*) are satisfied.

(A4*) We will show that for $\mathbf{z} \in \Omega_{d(\phi)-\{X,Y\}}$

$$(\phi_F^{-X})^{-Y}(\mathbf{z}) \quad = \quad (\phi_F^{-Y})^{-X}(\mathbf{z}). \tag{9.15}$$

We use the following short notation for the values that are of interest:

$$\begin{array}{lcllcl} v_1 & = & \phi_A(\mathbf{z}, 0_X, 0_Y), & f_1 & = & \phi_F(\mathbf{z}, 0_X, 0_Y), \\ v_2 & = & \phi_A(\mathbf{z}, 0_X, 1_Y), & f_2 & = & \phi_F(\mathbf{z}, 0_X, 1_Y), \\ v_3 & = & \phi_A(\mathbf{z}, 1_X, 0_Y), & f_3 & = & \phi_F(\mathbf{z}, 1_X, 0_Y), \\ v_4 & = & \phi_A(\mathbf{z}, 1_X, 1_Y), & f_4 & = & \phi_F(\mathbf{z}, 1_X, 1_Y). \end{array}$$

The definition of variable elimination is based on the comparison of the values v_i. For this proof, we remark that we can restrict ourselves to the two cases $>$ and $=$, because all other arrangements including $<$ are symmetric. More concretely, we need to prove the following four cases:

1. $v_1 > v_2 > v_3 > v_4$ (all values are different),
2. $v_1 = v_2 > v_3 > v_4$ (two values are equal),
3. $v_1 = v_2 = v_3 > v_4$ (three values are equal),
4. $v_1 = v_2 = v_3 = v_4$ (all values are equal).

All other cases are covered by these representatives as explained subsequently. We will thus verify Equation (9.15) under the above four situations:

1. Assume that $v_1 > v_2 > v_3 > v_4$. Then we have by application of (9.11)

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_Y) &= \big(v_1,\ f_{\overline{X}} \wedge f_1\big),\\
\phi^{-X}(\mathbf{z}, 1_Y) &= \big(v_2,\ f_{\overline{X}} \wedge f_2\big),\\
(\phi^{-X})^{-Y}(\mathbf{z}) &= \big(v_1,\ f_{\overline{Y}} \wedge f_{\overline{X}} \wedge f_1\big).
\end{aligned}$$

In the same way, we compute

$$\begin{aligned}
\phi^{-Y}(\mathbf{z}, 0_X) &= \big(v_1,\ f_{\overline{Y}} \wedge f_1\big),\\
\phi^{-Y}(\mathbf{z}, 1_X) &= \big(v_3,\ f_{\overline{Y}} \wedge f_3\big),\\
(\phi^{-Y})^{-X}(\mathbf{z}) &= \big(v_1,\ f_{\overline{X}} \wedge f_{\overline{Y}} \wedge f_1\big).
\end{aligned}$$

Thus, Equality (9.15) holds for this first case. Note that it covers also the case where $v_1 > v_2 > v_3 = v_4$ because here only $\phi_F^{-Y}(\mathbf{z}, 0_X)$ changes, which has no influence. In the same way, the cases $v_1 > v_2 = v_3 > v_4$ and $v_1 > v_2 = v_3 = v_4$ are also handled by this argument. To sum it up, we can say that all cases are handled where one value is strictly larger than all others.

2. Assume that $v_1 = v_2 > v_3 > v_4$. Then we have again

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_Y) &= \big(v_1,\ f_{\overline{X}} \wedge f_1\big),\\
\phi^{-X}(\mathbf{z}, 1_Y) &= \big(v_2,\ f_{\overline{X}} \wedge f_2\big),\\
(\phi^{-X})^{-Y}(\mathbf{z}) &= \big(v_1,\ (f_{\overline{Y}} \wedge f_{\overline{X}} \wedge f_1) \vee (f_Y \wedge f_{\overline{X}} \wedge f_2)\big).
\end{aligned}$$

Since this time we have equality between v_1 and v_2, we obtain

$$\begin{aligned}
\phi^{-Y}(\mathbf{z}, 0_X) &= \big(v_1,\ (f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2)\big),\\
\phi^{-Y}(\mathbf{z}, 1_X) &= \big(v_3,\ f_{\overline{Y}} \wedge f_3\big),\\
(\phi^{-Y})^{-X}(\mathbf{z}) &= \big(v_1,\ f_{\overline{X}} \wedge ((f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2))\big).
\end{aligned}$$

As we can see immediately, Equality (9.15) is again satisfied. This covers also the case $v_1 = v_2 > v_3 = v_4$, because here only $\phi_F^{-Y}(\mathbf{z}, 0_X)$ changes which again has no influence. More generally, all cases are handled where two values are equal and strictly larger than the two others.

3. Assume that $v_1 = v_2 = v_3 > v_4$. Then we have

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_Y) &= \big(v_1,\ (f_{\overline{X}} \wedge f_1) \vee (f_X \wedge f_3)\big),\\
\phi^{-X}(\mathbf{z}, 1_Y) &= \big(v_2,\ f_{\overline{X}} \wedge f_2\big),\\
(\phi^{-X})^{-Y}(\mathbf{z}) &= \big(v_1,\ (f_{\overline{Y}} \wedge ((f_{\overline{X}} \wedge f_1) \vee (f_X \wedge f_3))) \vee (f_Y \wedge f_{\overline{X}} \wedge f_2)\big).
\end{aligned}$$

In the same way, we compute

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_X) &= \big(v_1,\ (f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2)\big),\\
\phi^{-Y}(\mathbf{z}, 1_X) &= \big(v_3,\ f_{\overline{Y}} \wedge f_3\big),\\
(\phi^{-Y})^{-X}(\mathbf{z}) &= \big(v_1,\ (f_{\overline{X}} \wedge ((f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2))) \vee (f_X \wedge f_{\overline{Y}} \wedge f_3)\big).
\end{aligned}$$

Because

$$\begin{aligned}
(\phi^{-X})_F^{-Y}(\mathbf{z}) &= (f_{\overline{Y}} \wedge ((f_{\overline{X}} \wedge f_1) \vee (f_X \wedge f_3))) \vee (f_Y \wedge f_{\overline{X}} \wedge f_2)\\
&= (f_{\overline{Y}} \wedge f_{\overline{X}} \wedge f_1) \vee (f_{\overline{Y}} \wedge f_X \wedge f_3) \vee (f_Y \wedge f_{\overline{X}} \wedge f_2)\\
&= (f_{\overline{X}} \wedge ((f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2))) \vee (f_X \wedge f_{\overline{Y}} \wedge f_3)\\
&= (\phi^{-Y})_F^{-X}(\mathbf{z}),
\end{aligned}$$

Equality (9.15) is again satisfied. This covers all cases where three values are equal and strictly larger than the forth value.

4. At last, assume that $v_1 = v_2 = v_3 = v_4$. Then we have

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_Y) &= \big(v_1,\ (f_{\overline{X}} \wedge f_1) \vee (f_X \wedge f_3)\big),\\
\phi^{-X}(\mathbf{z}, 1_Y) &= \big(v_2,\ (f_{\overline{X}} \wedge f_2) \vee (f_X \wedge f_4)\big),\\
(\phi^{-X})^{-Y}(\mathbf{z}) &= (v_1,\ (f_{\overline{Y}} \wedge ((f_{\overline{X}} \wedge f_1) \vee (f_X \wedge f_3))) \vee\\
&\qquad (f_Y \wedge ((f_{\overline{X}} \wedge f_2) \vee (f_X \wedge f_4)))).
\end{aligned}$$

And in the same way

$$\begin{aligned}
\phi^{-X}(\mathbf{z}, 0_X) &= (v_1,\ (f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2)),\\
\phi^{-Y}(\mathbf{z}, 1_X) &= (v_1,\ (f_{\overline{Y}} \wedge f_3) \vee (f_Y \wedge f_4)),\\
(\phi^{-Y})^{-X}(\mathbf{z}) &= (v_1,\ (f_{\overline{X}} \wedge ((f_{\overline{Y}} \wedge f_1) \vee (f_Y \wedge f_2)) \vee\\
&\qquad (f_X \wedge ((f_{\overline{Y}} \wedge f_3) \vee (f_Y \wedge f_4))))).
\end{aligned}$$

As in the former case, $(\phi^{-X})_F^{-Y}(\mathbf{z})$ transforms into $(\phi^{-Y})_F^{-X}(\mathbf{z})$ by successive application of the distributive law. This completes the proof of the Variable Elimination axiom.

(A5*) We will show that for $\mathbf{z} \in \Omega_{d(\phi) \cup d(\psi) - \{X\}}$ and $X \in d(\psi)$, $X \notin d(\phi)$

$$(\phi_F \otimes \psi_F)^{-X}(\mathbf{z}) = \left(\phi_F \otimes \psi_F^{-X}\right)(\mathbf{z}).$$

Let us distinguish two different cases:

1. Assume $(\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) > (\phi_A \otimes \psi_A)(\mathbf{z}, 1_X)$. We obtain:

$$\begin{aligned}(\phi_F \otimes \psi_F)^{-X}(\mathbf{z}) &= f_{\overline{X}} \wedge (\phi_F \otimes \psi_F)(\mathbf{z}, 0_X) \\ &= f_{\overline{X}} \wedge \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) \\ &= \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge (f_{\overline{X}} \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)).\end{aligned}$$

Since $\times$ behaves strictly monotone,

$$\begin{aligned}(\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) &= \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \times \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) \\ &> \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \times \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X) \\ &= (\phi_A \otimes \psi_A)(\mathbf{z}, 1_X)\end{aligned}$$

implies that

$$\psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) > \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X).$$

Therefore, we obtain

$$\begin{aligned}\phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge (X \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)) &= \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \psi_F^{-X}(\mathbf{z}^{\downarrow d(\psi)}) \\ &= (\phi_F \otimes \psi_F^{-X})(\mathbf{z}).\end{aligned}$$

2. Assume $(\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) = (\phi_A \otimes \psi_A)(\mathbf{z}, 1_X)$. We know by the Combination axiom (A5) for semiring valuations, that

$$(\phi_A \otimes \psi_A)^{-X}(\mathbf{z}) = (\phi_A \otimes \psi_A^{-X})(\mathbf{z}).$$

The left-hand part of this equation is written as

$$(\phi_A \otimes \psi_A)^{-X}(\mathbf{z}) \;=\; (\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) + (\phi_A \otimes \psi_A)(\mathbf{z}, 1_X),$$

and the right-hand part as

$$(\phi_A \otimes \psi_A^{-X})(\mathbf{z}) \;=\; \phi_A(\mathbf{z}^{\downarrow d(\phi)}) \times \Big(\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) + \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X)\Big).$$

Remembering that + is idempotent, we obtain the following two equations:

(a) $(\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) \;=\; \phi_A(\mathbf{z}^{\downarrow d(\phi)}) \times \big(\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) + \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X)\big),$

(b) $(\phi_A \otimes \psi_A)(\mathbf{z}, 1_X) \;=\; \phi_A(\mathbf{z}^{\downarrow d(\phi)}) \times \big(\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) + \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X)\big).$

Since $(\phi_A \otimes \psi_A)(\mathbf{z}, 0_X) = \phi_A(\mathbf{z}^{\downarrow d(\phi)}) \times \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)$ and + corresponds to maximization, we conclude from (a) that

$$\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X) \;\leq\; \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)$$

must hold. In the same way, we derive from (b) that

$$\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) \;\leq\; \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X),$$

and consequently

$$\psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X) \;=\; \psi_A(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X).$$

Finally, we obtain:

$$\begin{aligned}
(\phi_F \otimes \psi_F)^{-X}(\mathbf{z}) &= \big(f_{\overline{X}} \wedge (\phi_F \otimes \psi_F)(\mathbf{z}, 0_X)\big) \vee (f_X \wedge (\phi_F \otimes \psi_F)(\mathbf{z}, 1_X)) \\
&= (f_{\overline{X}} \wedge \Big(\phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)\Big)) \vee \\
&\quad (f_X \wedge \Big(\phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X)\Big)) \\
&= \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \Big((f_{\overline{X}} \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 0_X)) \vee (f_X \wedge \psi_F(\mathbf{z}^{\downarrow d(\psi)-\{X\}}, 1_X)\Big) \\
&= \phi_F(\mathbf{z}^{\downarrow d(\phi)}) \wedge \psi_F^{-X}(\mathbf{z}^{\downarrow d(\psi)-\{X\}}) \;=\; (\phi_F \otimes \psi_F^{-X})(\mathbf{z}).
\end{aligned}$$

This proves the Combination axiom.

□

With respect to the former sketch of algorithm, the insight that memorizing semiring valuations form a valuation algebra enables us to compute $\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)$ by a simple run of the collect algorithm (or *fusion* since we deal with variable elimination (Kohlas, 2003)). Thus, we can say that the process of compiling solution configurations into a Boolean function is just another application of local computation. Consequently, this method inherits also the efficiency of the latter.

Algorithm 4:

- Construct memorizing semiring valuations $\tilde{\psi}_i$ according to Equation (9.10).
- Execute collect (fusion) on the factor set $\{\tilde{\psi}_1, \ldots, \tilde{\psi}_m\}$.
- Compute $\tilde{\phi}^{\downarrow\emptyset}(\diamond)$ from the result of the collect algorithm and return $\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)$.

The correctness proof of this algorithm follows from Lemma 12.

9.3 Model Enumeration & further efficient Queries

With the technique described in the foregoing section, we have found a very compact representation of all solution configurations that does not depend on their number. Due to Equation (9.13) we have

$$\mathbf{Models}\left(\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)\right) \;=\; c_\phi.$$

Naturally, an indispensable requirement for the applicability of this method is that we must be able to enumerate the models of the final Boolean function $\widehat{\phi}_F^{\downarrow\emptyset}(\diamond)$ efficiently (i.e. in polynomial time). For this purpose, we should first survey a particular interesting way of representing Boolean functions in general, namely as *Propositional Directed Acyclic Graphs (PDAG)* (Wachter & Haenni, 2006).

Definition 8 *A propositional DAG (PDAG) over a set of propositional variables r is a rooted directed acyclic graph of the following form:*

1. *Leaves are represented by $\bigcirc$ and labeled with $\top$ (true), $\bot$ (false), or $X \in r$.*
2. *Non-leaf nodes are represented by $\triangle$ (logical conjunction), $\triangledown$ (logical disjunction) or $\diamond$ (logical negation).*
3. *$\triangle$- and $\triangledown$-nodes have at least one child and $\diamond$-nodes have exactly one child.*

Leaves labeled with $\top$ ($\bot$) represent the Boolean function that constantly maps to 1 (0). Those labeled with $X \in r$ represent the Boolean function that maps to 1 if and only if $X = 1$. The Boolean function represented by a $\triangle$-node is the one that evaluates to 1 if and only if all its children evaluate to 1. Correspondingly, $\triangledown$-nodes evaluate to 1 if and only if at least one of their children evaluates to 1. Finally, a $\diamond$-node represents the complementary Boolean function of its child.

Figure 9.1 illustrates the PDAG structures that are created from the three variable elimination rules of memorizing semiring valuations. Combination on the other hand just connects two existing PDAGs by a conjunction node to a new PDAG. Thus, because all variables are eliminated, we obtain for $\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)$ a single PDAG structure. Some particularities of this graph are summarized in the following lemma.

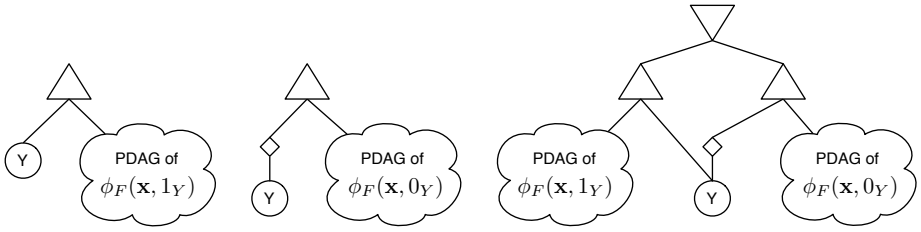


Figure 9.1: PDAG structures that are created from the variable elimination rules of memorizing semiring valuations.

Lemma 13 *The PDAG representation of $\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)$ satisfies the following properties:*

1. *Simple-Negation: Each child of a $\diamond$-node is a leaf.*
2. *Decomposability: The sets of variables that occur in the sub-trees of every $\triangle$-node are disjoint.*
3. *Determinism: The children of every $\triangledown$-node are pairwise logically contradictory., i.e. if α_i and α_j are two children of the same $\triangledown$-node, we have $\alpha_i \wedge \alpha_j \equiv \bot$.*

Proof. The first property follows directly from PDAGs 2 and 3 in Figure 9.1 because these are the only rules that create negation nodes. A variable node is created

whenever the corresponding variable is eliminated. Hence, every variable node occurs exactly once in this PDAG which proves Property 2. Finally, we can conclude from PDAG 3 in Figure 9.1 that in every model of the disjunction node's left child, $Y = 1$ must hold. Similarly, $Y = 0$ must hold in the right child and therefore, the two model sets are disjoint. This is the statement of Property 3. □

A PDAG that satisfies the three properties of Lemma 13 is called *cdn-PDAG* (Wachter & Haenni, 2006) or *d-DNNF* (Darwiche & Marquis, 2002). Before we shall now focus on the computational possibilities that are given by d-DNNF structures, we want to point out an important detail in the proof of Lemma 13. There, d-DNNFs are built from connecting existing d-DNNFs by either a conjunction or a disjunction node. However, we know from (Darwiche & Marquis, 2002) that the d-DNNF language is not closed under these operations. This means concretely that it is in general not possible to reconstruct a d-DNNF structure from the conjunction or disjunction of two d-DNNFs in polynomial time. Fortunately, this does not hold for the case at hand. Since these constructions are performed as valuation algebra operations, we directly obtain the cdn-properties whenever we join two existing d-DNNFs by the rules specified for memorizing semiring valuations.

Theorem 10 *A d-DNNF allows model enumeration in polynomial time.*

(Darwiche, 2001) gives the following recursive procedure for model enumeration in general DNNFs:

Algorithm 5:

1. $\mathbf{Models}(\bigcirc) = \begin{cases} \{\{X = x\}\}, & \text{if } \bigcirc \text{ is labeled with variable } X, \\ \{\{\}\}, & \text{if } \bigcirc \text{ is labeled with } \top, \\ \{\}, & \text{if } \bigcirc \text{ is labeled with } \bot. \end{cases}$

2. $\mathbf{Models}(\diamond) = \{\{X = \overline{x}\}\}$ with X being the label of its $\bigcirc$ child node.

3. $\mathbf{Models}(\bigtriangledown) = \bigcup \mathbf{Models}(\bigtriangledown_i)$ where $\bigtriangledown_i$ are the children of node $\bigtriangledown$.

4. $\mathbf{Models}(\bigtriangleup) = \{\bigcup \mu : \mu \in \mathbf{Models}(\bigtriangleup_i)\}$ where $\bigtriangleup_i$ are the children of node $\bigtriangleup$.

(Darwiche, 2001) estimates the complexity of this procedure as $O(mn^2)$ where m is the size of the d-DNNF and n the number of its models. If we consider the number of models as a constant (which is reasonable for the task of model enumeration), the above algorithm becomes linear in the depth of the d-DNNF.

If we are interested in only one solution configuration, it is sufficient to choose (non-deterministically) a model of an arbitrary child in Rule 4 and ignoring the remaining models. This procedure is called *model selection*. We therefore conclude that the implicit approach of compiling solution configurations into a Boolean function is as powerful as the explicit method introduced in Section 8.3. We therefore extend Algorithm 4 in the following way:

Algorithm 6:

- Construct memorizing semiring valuations $\tilde{\psi}_i$ according to Equation (9.10).
- Execute collect (fusion) on the factor set $\{\tilde{\psi_1}, \ldots, \tilde{\psi_m}\}$.
- Compute $\tilde{\phi}^{\downarrow\emptyset}(\diamond)$ from the result of the collect algorithm.
- If $\tilde{\phi}_A^{\downarrow\emptyset}(\diamond) = \mathbf{0}$, return $c_\phi = \Omega_{d(\phi)}$.
- Identify $\mathbf{Models}\Big(\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)\Big) = c_\phi$ by Algorithm 5.

We already foreshadowed on various occasions that d-DNNFs allow a lot more of *queries* to be performed efficiently. In view of this, the last instruction of Algorithm 6 can be replaced by a collection of further procedures that go far beyond model enumeration and this fact features the implicit approach compared to all explicit methods presented beforehand. We refer to (Darwiche & Marquis, 2002; Wachter & Haenni, 2006) for a complete discussion of efficient queries in d-DNNFs and restrict ourselves to a listing of some particular interesting applications for the purpose of solving optimization problems.

- *Counter-Model Enumeration:* The d-DNNF constructed by Algorithm 6 allows also to enumerate all configurations of ϕ that are not solution configurations, i.e. all configurations with $\phi(\mathbf{x}) < \phi^{\downarrow\emptyset}(\diamond)$.
- *Model Counting:* Counting the number of solution configurations can also be done efficiently.
- *Validity:* This answers the query if $\mathbf{Models}(\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)) = \Omega_{d(\phi)}$, i.e. if all configurations of ϕ adopt the same value.
- *Probability Computation:* If we assume independent marginal probabilities $p(X)$ for all variables $X \in d(\phi)$, we can efficiently evaluate the probability of the Boolean function $\widehat{\phi}_F^{\downarrow\emptyset}(\diamond)$. This allows for example to identify the most probable solution configuration with respect to a given probability distribution over the variables in $d(\phi)$.
- *Probabilistic Equivalence Test*: If two different factorizations over the same set of variables are given, d-DNNFs allow to test probabilistically if the two joint valuations ϕ_1 and ϕ_2 adopt the same solution configurations, i.e. if $c_{\phi_1} = c_{\phi_2}$.

This is only a very small selection from the extensive list of queries that can be performed efficiently on d-DNNFs and therefore on $\widehat{\phi}_F^{\downarrow\emptyset}(\diamond)$. Again, we refer to (Wachter & Haenni, 2006; Darwiche & Marquis, 2002) for further applications and close the discussion by an example that works out the implicit method concretely.

Example 21: We repeat Example 18 with the implicit approach. Since this technique requires a semiring whose total order is monotonic over $\times$, we will use the Tropical semiring from Example 3 with max for $+$ and addition for $\times$. The initialization is done with Equation (9.10), such that we start with the following factor set

$\tilde{\psi}_1 =$

$\Omega_{\{A,B\}}$			
0_A	0_B	2	f_T
0_A	1_B	4	f_T
1_A	0_B	3	f_T
1_A	1_B	2	f_T

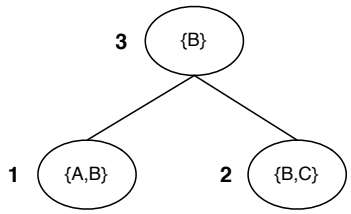
$\tilde{\psi}_2 =$

$\Omega_{\{B,C\}}$			
0_B	0_C	5	f_T
0_B	1_C	2	f_T
1_B	0_C	3	f_T
1_B	1_C	3	f_T

$\tilde{\psi}_3 =$

$\Omega_{\{B\}}$		
0_B	1	f_T
1_B	6	f_T

We can reuse the join tree from Example 18.



We compute first $\tilde{\phi} = \tilde{\psi}_1 \otimes \tilde{\psi}_2 \otimes \tilde{\psi}_3$ directly.

$\tilde{\phi} =$

$\Omega_{\{A,B,C\}}$				
0_A	0_B	0_C	8	f_T
0_A	0_B	1_C	5	f_T
0_A	1_B	0_C	13	f_T
0_A	1_B	1_C	13	f_T
1_A	0_B	0_C	9	f_T
1_A	0_B	1_C	6	f_T
1_A	1_B	0_C	11	f_T
1_A	1_B	1_C	11	f_T

We see that $c_\phi = \{(0_A, 1_B, 0_C), (0_A, 1_B, 1_C)\}$. We now start Algorithm 6.

$\mu_{1\to 3} =$

$\Omega_{\{B\}}$		
0_B	3	f_A
1_B	4	$f_{\overline{A}}$

$\psi_3^{(2)} =$

$\Omega_{\{B\}}$		
0_B	4	f_A
1_B	10	$f_{\overline{A}}$

$\mu_{2\to 3} =$

$\Omega_{\{B\}}$		
0_B	5	$f_{\overline{C}}$
1_B	3	$f_{\overline{C}} \vee f_C$

$\psi_3^{(3)} =$

$\Omega_{\{B\}}$		
0_B	9	$f_A \wedge f_{\overline{C}}$
1_B	13	$f_{\overline{A}} \wedge (f_{\overline{C}} \vee f_C)$

The maximum value of $\tilde{\phi}_A$ is $\tilde{\phi}_A^{\downarrow\emptyset}(\diamond) = \psi_3^{(3)\downarrow\emptyset}(\diamond) = 13$ and we obtain the compilation

$$\tilde{\phi}_F^{\downarrow\emptyset}(\diamond) \;=\; f_B \wedge f_{\overline{A}} \wedge (f_{\overline{C}} \vee f_C).$$

By application of Algorithm 5, we find

$$\mathbf{Models}\left(\tilde{\phi}_F^{\downarrow\emptyset}(\diamond)\right) \;=\; \{(0_\mathrm{A}, 1_\mathrm{B}, 0_\mathrm{C}),\ (0_\mathrm{A}, 1_\mathrm{B}, 1_\mathrm{C})\} \;=\; c_\phi.$$

$\ominus$

10 Conclusion

This paper shows that a totally ordered, idempotent semiring induces a valuation algebra in which the execution of the collect algorithm for a given factorization of some valuation ϕ determines the maximum value over all configurations of ϕ. Hence, the collect algorithm corresponds to non-serial dynamic programming with block elimination. Upon this framework, we present several methods for the task of identifying solution configurations, that are configurations for which ϕ adopts this maximum value computed beforehand. Essentially, two different classes of methods labeled as explicit and implicit are introduced. The explicit methods generalize the procedure introduced in (Shenoy, 1996) by computing the solution configuration set as a local computation scheme on the less restrictive structure of a covering join tree. The implicit method on the other hand approaches the same task in a completely new fashion by the construction of a Boolean function whose models correspond to the solution configurations of ϕ. This results in a very compact representation of the solution configuration set and it turns out that this Boolean function is representable as a d-DNNF. Upon d-DNNFs many interesting queries that go far beyond model enumeration are efficiently computable, which extends the area of application of local computation by a bunch of traditional problems of computer science.

Acknowledgements: Our special thanks go to Rolf Haenni and Michael Wachter whose deep insight into the theory of representing Boolean functions enabled this delightful fusion of our research interests. Additionally, we express our thankfulness to Christian Eichenberger for proof reading this paper with his usual and valuable pertinacity.

References

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

Atallah, Mikhail J., & Fox, Susan (eds). 1998. *Algorithms and Theory of Computation Handbook*. Boca Raton, FL, USA: CRC Press, Inc. Produced By-Suzanne Lassandro.

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

Darwiche, Adnan. 2001. Decomposable negation normal form. *J. ACM*, **48**(4), 608–647.

Darwiche, Adnan, & Marquis, Pierre. 2002. A Knowledge Compilation Map. *J. Artif. Intell. Res. (JAIR)*, **17**, 229–264.

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

Garey, Michael R., & Johnson, David S. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Kohlas, J. 2003. *Information Algebras: Generic Structures for Inference*. Springer-Verlag.

Kohlas, J. 2004. *Valuation Algebras Induced by Semirings*. Tech. rept. 04-03. Department of Informatics, University of Fribourg.

Kohlas, J., & Wilson, N. 2006. *Exact and Approximate Local Computation in Semiring Induced Valuation Algebras*. Tech. rept. 06-06. Department of Informatics, University of Fribourg.

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

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

Mateescu, Robert, & Dechter, Rina. 2006. Compiling Constraint Networks into AND/OR Multi-valued Decision Diagrams (AOMDDs). *Pages 329–343 of: CP*.

Menger, Karl. 1942. Statistical Metrics. *Proceedings of the National Academy of Sciences*, **28**, 535–537.

Mitten, L.G. 1964. Composition principles for the synthesis of optimal multi-stage processes. *Operations Research*, **12**.

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

Schweizer, B., & Sklar, A. 1960. Statistical Metric Spaces. *Pacific J. Math.*, **10**, 313–334.

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.

Shafer, G., & Shenoy, P. 1990. Axioms for Probability and Belief Function Propagation. *In:* Shafer, G., & Pearl, J. (eds), *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers Inc., San Mateo, California.

Shenoy, P. P., & Shafer, G. 1990. Axioms for probability and belief-function proagation. *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. 1992. 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. 1996. Axioms for Dynamic Programming. *Pages 259–275 of:* Gammerman, A. (ed), *Computational Learning and Probabilistic Reasoning*. Wiley, Chichester, UK.

Wachter, M., & Haenni, R. 2006. Propositional DAGs: a New Graph-Based Language for Representing Boolean Functions. *Pages 277–285 of:* Doherty, P., Mylopoulos, J., & Welty, C. (eds), *KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning*. Lake District, U.K.: AAAI Press.

Wachter, M., & Haenni, R. 2007. Multi-State Directed Acyclic Graphs. *Pages 464–475 of:* Kobti, Z., & Wu, D. (eds), *CanAI'07, 20th Canadian Conference on Artificial Intelligence*. LNAI 4509.

Wilson, N. 2004. Decision Diagrams for the Computation of Semiring Valuations. *Proc. 6th International Workshop on Preferences and Soft Constraints*.

Zadeh, L. A. 1987. A Theory of Approximate Reasoning. *Pages 367–412 of:* Yager, R. R., Ovchinnikov, S., Tong, R. M., & Nguyen, H. T. (eds), *Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh*. New York: John Wiley & Sons, Inc.

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